Learning Class Disjointness Axioms Using Grammatical Evolution

Thu Huong Nguyen, Andrea Tettamanzi

HAL Id: hal-02100230
https://hal.inria.fr/hal-02100230

Submitted on 15 Apr 2019

Learning Class Disjointness Axioms Using Grammatical Evolution

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. Today, with the development of the Semantic Web, Linked
Open Data (LOD), expressed using the Resource Description Frame-
work (RDF), has reached the status of "big data" and can be considered
as a giant data resource from which knowledge can be discovered. The
process of learning knowledge defined in terms of OWL 2 axioms from
the RDF datasets can be viewed as a special case of knowledge discov-
ery from data or "data mining", which can be called "*RDF mining*".
The approaches to automated generation of the axioms from recorded
RDF facts on the Web may be regarded as a case of inductive reasoning
and ontology learning. The instances, represented by RDF triples, play
the role of specific observations, from which axioms can be extracted by
generalization. Based on the insight that discovering new knowledge is
essentially an evolutionary process, whereby hypotheses are generated
by some heuristic mechanism and then tested against the available evi-
dence, so that only the best hypotheses survive, we propose the use of
Grammatical Evolution, one type of evolutionary algorithm, for mining
disjointness OWL 2 axioms from an RDF data repository such as DBpe-
dia. For the evaluation of candidate axioms against the DBpedia dataset,
we adopt an approach based on possibility theory.

Keywords: Ontology learning · OWL 2 axiom · Grammatical Evolu-
tion.

1 Introduction

The manual acquisition of formal conceptualizations within domains of knowl-
edge, i.e. ontologies [1] is an expensive and time-consuming task because of the
requirement of involving domain specialists and knowledge engineers. This is
known as the "*knowledge acquisition bottleneck*". Ontology learning, which com-
prises the set of methods and techniques used for building an ontology from
scratch, enriching, or adapting an existing ontology in a semi-automatic fashion,
using several knowledge and information sources [2, 3], is a potential approach
to overcome this obstacle. An overall classification of ontology learning methods
can be found in [4, 3, 5]. 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 and knowledge can consist of concepts, relations, or axioms from a domain-specific application.

Linked Open Data (LOD) being Linked Data[1] published in the form of an Open Data Source can be considered as a giant real-world knowledge base. Such a huge knowledge base opens up exciting opportunities for learning new knowledge in the context of an open world. Based on URIs, HTTP, and RDF, Linked Data is a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web. Some approaches to ontology learning from linked data can be found in [6–8]. The advantages of LOD with respect to learning described in [8] is that it is publicly available, highly structured, relational, and large compared with other resources. 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.

In ontology learning, one of the critical tasks is to increase the expressiveness and semantic richness of a knowledge base (KB), which is called *ontology enrichment*. Meanwhile, exploiting ontological axioms in the form of logical assertions to be added to an existing ontology can provide some tight constraints to it or support the inference of implicit information. Adding axioms to a KB can yield several benefits, as indicated in [9]. 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 [10], 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 *Person, City* and asserting that *Sydney* is both a *Person* and a *City* would be logically consistent, without any errors being recognized by a reasoner. However, if a constraint of disjointness between classes *Person* and *City* 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 November 11, 2018 returned only 25 solutions, whereas the realistic number of class disjointness axioms generated from hundreds of classes in DBpedia (432 classes in DBpedia 2015-04, 753 classes in DBpedia 2016-04) is expected to be much more (in the 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. Recent methods [11, 12] apply top-down or intensional approaches to learning disjointness which rely on schema-level information, i.e., logical and lexical decriptions of the classes. The contributions based on bottom-

---

[1] http://linkeddata.org/

up or extensional approaches [9, 10], 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: some related works are described briefly in Section 2. In Section 3, some background is provided. OWL 2 classes disjointness axioms discovery with a GE approach is presented in Section 4. An axiom evaluation method based on possibility theory is also presented in this section. Section 5 provides experimental evaluation and comparison. Conclusions and directions for future research are given in Section 6.

2   Related work

The most prominent related work relevant to learning disjointness axioms consists of the contributions by Johanna Völker and her collaborators [12, 10, 13]. In early work, Völker developed supervised classifiers from LOD incorporated in the *LeDA* tool [12]. 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 associative rule mining [10] 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 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 [9], 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. That approach is very similar to ours in that it uses an evolutionary algorithm for learning concepts. Bühmann and Lehmann also developed methods for generating more complex axiom types [14] 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 $A \sqsubseteq B \sqcap \exists r.C$.

Our solution is based on an evolutionary approach deriving from previous work, but concentrating on a specific algorithm, namely Grammatical Evolution (GE) [15] to generate class disjointness axioms from an existing RDF repository which is different from the use of Genertic Algorithm as in the approach of Bühmann and Lehmann [14]. GE aims at overcoming one shortcoming of GP, which is the growth of redundant code, also known as *bloat*. Furthermore, instead of using probability theory, we applied a possibilistic approach to assess the fitness of axioms.

3 Background

This section provides a few background notions required to understand the application domain of our contribution.

3.1 RDF Datasets

The Semantic Web[2] (SW) is an extension of the World Wide Web and it can be considered as the Web of data, which aims to make Web contents machine-readable. The Linked Open Data[3] (LOD) is a collection of linked RDF data. The LOD covers the data layer of the SW, where RDF plays the roles of its standard data model.

RDF uses as statements triples of the form (Subject, Predicate, Object). According to the World Wide Web Consortium (W3C), RDF[4] has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. RDF data may be viewed as an oriented, labeled multigraph. The query language for RDF is SPARQL,[5] 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.

One of the prominent examples of LOD is DBpedia,[6] 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.

---

[2] https://www.w3.org/standards/semanticweb/
[3] https://www.w3.org/egov/wiki/Linked_Open_Data
[4] https://www.w3.org/RDF/
[5] https://www.w3.org/TR/rdf-sparql-query/
[6] https://wiki.dbpedia.org/

3.2 OWL 2 Axioms

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 2[7] is an ontology language for the Semantic Web with formally defined meaning which strikes a good compromise between these two objectives. In addition, OWL 2 is standardized and promotes interoperability with different applications. Furthermore, depending on the applications, it will be possible to select an appropriate *profile* (corresponding to a different language fragment) exhibiting the desired trade-off between expressiveness and computational complexity.

4 A Grammatical Evolution Approach to Discovering OWL 2 Axioms

This section introduces a method based on Grammatical Evolution (GE) to mine an RDF repository for class disjointness axioms. GE is similar to GP in automatically generating variable-length programs or expressions in any language. In the context of OWL 2 axiom discovery, the "programs" are axioms. A population of individual axioms is maintained by the algorithm and iteratively refined to find the axioms with the highest level of credibility (one key measure of quality for discovered knowledge). In each iteration, known as a *generation*, the fitness of each individual in the population is evaluated using a possibilistic approach and is the base for the parent selection mechanism. The offspring of each generation is bred by applying genetic operators on the selected parents. The overall flow of such GE algorithm is shown in Algorithm 1.

4.1 Representation

As in O'Neill et al [15] and unlike GP, GE applies the evolutionary process on variable length binary strings instead of on the actual programs. GE has a clear distinction in representation between genotype and phenotype. The genotype to phenotype mapping is employed to generate axioms considered as phenotypic programs by using the Backus-Naur form (BNF) grammar [15–17].

---

[7] https://www.w3.org/TR/owl2-overview/

Algorithm 1 - GE for discovering axioms from a set of 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 by performing $popSize$ times **$CreateNewAxiom()$**
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
     $Lr \leftarrow Pop \backslash listElites$
10:   Eliminate the duplicates in Lr
     $Lr \leftarrow$ 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:     $parent1 \leftarrow listCrossover$[i]
13:     $parent2 \leftarrow listCrossover$[i+1]
14:     $child1, child2 \leftarrow CROSSOVER(parent1,parent2)$ 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
      $w1,w2 \leftarrow CROWDING((parent1, parent2, child1, child2)$
18:     Add $w1$, w2 to new population $newPop$
19:   $Pop = newPop$
20:   Increase the number of generation $curGeneration$ by 1
21: return Pop

Structure of BNF Grammar We applied the extended BNF grammar consisting of the production rules extracted from the normative grammar[8] of OWL 2 in the format used in W3C documentation for constructing different types of OWL 2 axioms. 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, resources, literals, properties, and other elements of the language that actually occur in the RDF repository could be generated. We organized our BNF grammar in two main parts (namely static and dynamic) as follows:

– the static part contains production rules defining the structure of the axioms loaded from the text file. Different grammars will generate different kinds of axioms.

---

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

– the dynamic part contains production rules for the low-level non-terminals, which we will call *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 BNF grammar ensures the changes in the contents of RDF repositories will not require to rewrite the grammar.

In the following, we will refer to generating class disjointness axioms containing atomic expression such as DisjointClasses(Film, WrittenWork) or complex expression in the cases of relational operators, i.e., intersection and union, such as DisjointClasses(Film, ObjectIntersectionOf(Book,ObjectUnionOf(Comics, MusicalWork))). We built the following pattern of the grammar structured for generating class disjointness axioms:

% 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' 'ClassExpression ')'
```

% Dynamic part - Primitives

```
(r7) Class := % production rules are constructed by using SPARQL queries
```

This produces rules of the primitive `Class`, which will be filled by using SPARQL queries to extract the IRI of a class mentioned in the RDF store. 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 nonterminal `Class` are represented as follows:

```
(r7) Class := dbo:Film          (0)
            |  dbo:MusicalWork   (1)
            |  dbo:WrittenWork   (2)
            |  dbo:Book          (3)
            |  dbo:Comic         (4)
```

Encoding and Decoding Individual axioms are encoded as variable-length binary strings with numerical chromosomes. The binary string consists of a sequence of 8-bit words referred to as *codons*.

According to the structure of the above BNF grammar, chromosomes are then decoded into OWL 2 axioms in different OWL syntaxes through the mapping process according to the function:

$$\text{Rule} = \text{Codon value } \textbf{modulo} \text{ Number of Rules for the current terminal} \qquad (1)$$

In the advantageous cases, axioms are generated before the end of the genome is reached; otherwise, a *wrapping* operator [15, 16] 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. The production rule for `ClassExpression` is recursive and may lead to a large fan-out; to alleviate this problem and promote "reasonable" axioms, we increase the probability of obtaining a successful mapping to complex axiom expressions, we double the appearance probability of non-terminal `ClassExpression`. Rule (r4) in the grammar is modified to

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

4.2 Initialization

In the beginning of the evolutionary process, a set of chromosomes, i.e., genotypic individuals, are randomly initialized once and for all. Each chromosome is a set of integers with the initialized length *initlenChrom*. Its length can be extended to *maxlenChrom* in the scope of the threshold of *maxWrap* in the wrapping process. The next step is the transformation of genotypes into phenotypic individuals, i.e., axioms according to grammar *Gr*, by means of the mapping process based on the input grammar called *CreateNewAxiom()* operator. The population of *popSize* class disjointness axioms is created by iterating *popSize* times *CreateNewAxiom()* operator described in Algorithm 2.

Algorithm 2 - CreateNewAxiom()

Input: *Chr*: Chromosome - a set of integers; *Gr*: BNF grammar
Output: *A*: a new axiom individual

1: $maxlenChrom \leftarrow initlenChrom * maxWrap$
2: $ValCodon \leftarrow random(maxValCodon)$.
3: Set up *Chr* as input genotype *gp* used in mapping proccess to axiom individual *A*
4: **while** (*Chr.length* $<maxlenChrom$) && (incomplete mapping) **do**
5:     mapping from input genotype *gp* to output phenotype of axiom individual *A*
       according to grammar *Gr*
6: return *A*

4.3 Parent selection

Before executing the selection operator, the axioms in the populations are evaluated and ranked in descending order of their fitness. To combat the loss of fittest axioms as a result of the application of the variation operators, elitism selection is applied to copy the small proportion *pElite* of the best axioms into the next generation (line 7-8 of Algorithm 1). In the remaining part of the population, the elimination of duplicates is carried out to ensure only distinct individuals will be included in the *candidate list* for parent selection. The parent selection mechanism amounts to choosing the fittest individuals from this list for reproduction. Fig. 1. illustrates the process of selecting potential candidate solutions for recombination, i.e., a list of parents. The top proportion *pselectSize* of distinct individuals in the *candidate list* is selected and it is replicated to maintain the size *popSize* of population. The list of parents is shuffled and the individuals are paired in order from the beginning to the end of the list.
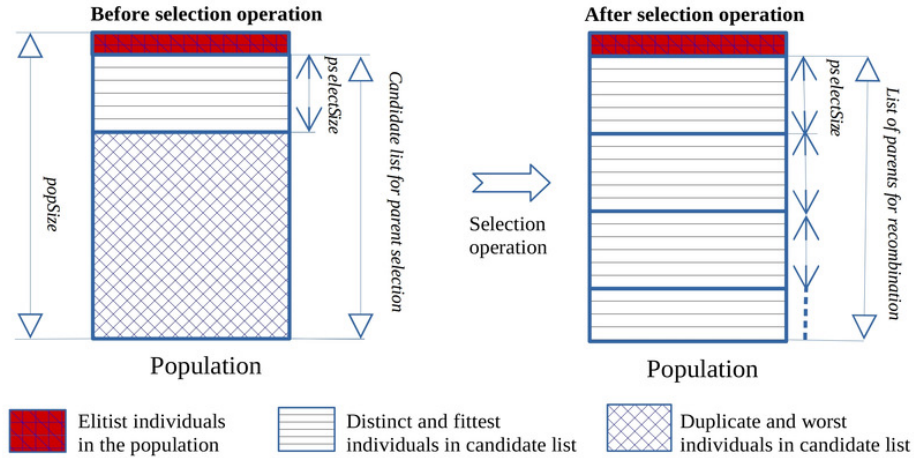

Fig. 1. An illustration of the parent selection mechanism.

4.4 Variation Operators

The purpose of these operators is to create new axioms from old ones. The standard genetic operators of crossover and mutation in the Evolutionary Algorithms (EA) are applied in the search space of genotypes. Well-formed individuals will then be generated syntactically from the new genotypes in the genotype-to-phenotype mapping process.

Crossover A standard one-point crossover is employed whereby a single crossover point on the chromosomes of both parents is chosen randomly. The sets of codons

---

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

---

beyond those points are exchanged between the two parents with probability
pCross. The result of this exchange is two offspring genotypes. The mapping of
these genotype into phenotypic axioms is performed by executing the *Create-NewAxiom()* operator (Algorithm 2) again with the offspring chromosomes as
input.

Mutation The mutation is applied to the offspring genotypes of crossover with
probability *pMut*. A selected individual undergoes single-point mutation, i.e. a
codon is selected at random, and this codon is replaced with a new randomly
generated codon.

4.5   Survival selection

In order to preserve population diversity, we used the *Deterministic Crowding*
approach developed by Mahfoud [18]. Each offspring competes with its most
similar peers, based on a genotypic comparison, to be selected for inclusion in
the population of the next generation. Algorithm 3 describes this approach in
detail. Even though we are aware that computing distance at the phenotypic level
would yield more accurate results, we chose to use genotypic distance because it
is much faster and easier to compute.

4.6   Fitness Evaluation

As a consequence of the heterogeneous and collaborative character of the linked
open data, some facts (instances) in the RDF repository may be missing or erro-
neous. This incompleteness and noise determines a sort of *epistemic* uncertainty
in the evaluation of the quality of a candidate axiom. In order to properly cap-
ture this type of uncertainty, typical of an open world, which contrasts with
the *ontic* uncertainty typical of random processes, we adopt an axiom scoring

heuristics based on possibility theory, proposed in [19], which is suitable for dealing with incomplete knowledge. This is a justified choice for assessing knowledge extracted from an RDF repository. We now provide a summary of this scoring heuristics.

Possibility theory [20] 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.

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 set of logical consequences

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

obtained through the instatiation of ϕ to the vocabulary of the RDF repository; the cardinality of content(ϕ) is finite and every formula $\psi \in$ content(ϕ) 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 [19]: for $u_\phi > 0$,

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

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

The cardinality of the sets of the facts in the RDF repository reflects the generality 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. These numbers of confirmations and counterexamples are counted by executing corresponding SPARQL queries via an accessible SPARQL endpoint.

In principle, the fitness of axiom ϕ should be directly proportional to its necessity $N(\phi)$, its possibility $\Pi(\phi)$, and its support u_ϕ, which is an indicator of its generality. In other words, what we are looking for is not only credible axioms, but also general ones. A definition of the fitness function that satisfies such requirement is

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

which we adopted for our method.

5 Experiment & Result

We applied our approach to mine the classes disjointness axioms relevant to the topic *Work* in DBpedia. The statistical data of classes and instances about this topic in DBpedia 2015-04 is given in Table 1.
All data used in this experiment is represented in the form of RDF triples, as explained in Section 3. In order to assess its ability to discover axioms, we ran

Table 1. Statistical data in the topic Work in DBpedia

Total number of classes	62
Total number of classes having instances	53
Total number of instances	519,5019

the GE indicated in Section 4 by repeating the sample procedure of Algorithm 1 for each run with the same parameters indicated in Table 2. The chart in Fig. 2 illustrates the average diversity of the population of axioms over the generations of the evolutionary process. It shows how many different "species" of axioms are contained in the population, i.e., axioms that cover different aspects of the known facts. One of the remarkable points here is that there is a more rapid loss of diversity in the phenotype axioms compared with this decrease in the genotype ones. The use of *Crowding method* on genotypes instead of phenotypes can be the reason of this difference. Likewise, a set of codons of two parent chromosomes which are used for the mapping to phenotypes can fail to be swapped in the single-point crossover operator.

From the chart in Fig. 3, we can observe a gradual increase in the quality of discovered axioms over generations.

In order to evaluate the effectiveness of our method in discovering disjointness class axioms of the *Work* on RDF datasets of DBpedia, a benchmark of

Average axioms diversity over generations

(20 runs)
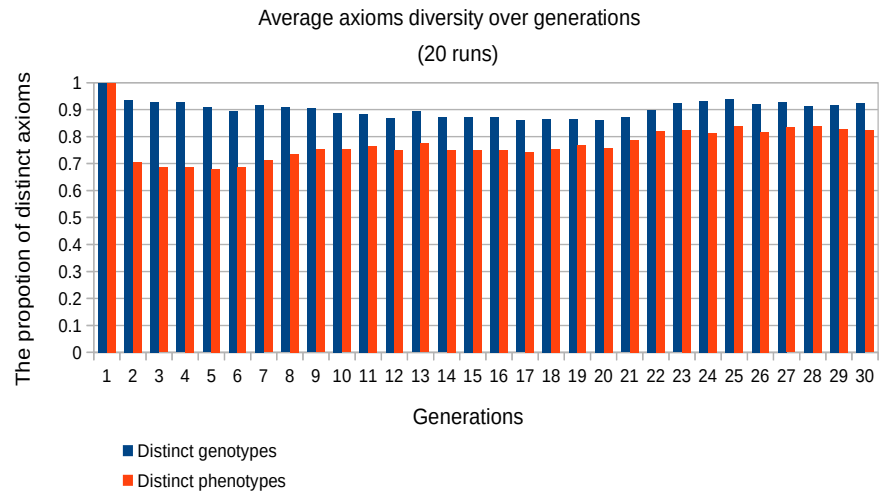
Fig. 2. The diversity of axioms over generations
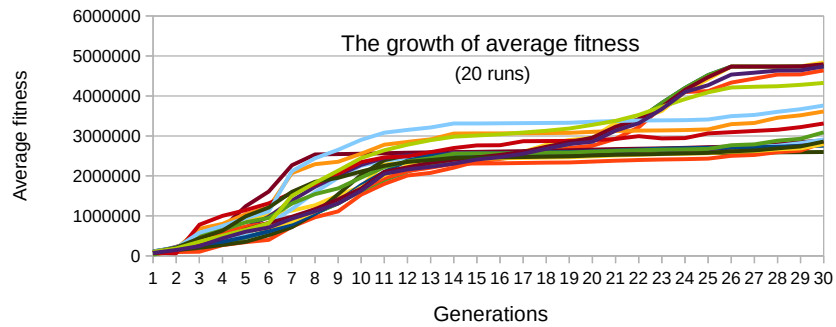
Fig. 3. The growth of average fitness over generations

Nguyen and Tettamanzi

Table 2. Input parameter values for GE

Parameter	Value
popSize	500
numGenerations	30
initlenChrom	20
maxWrap	2
pCross	80%
pMut	1%
pselectSize	70%
pElite	2%

Table 3. Experimental results

	Our approach		GoldMiner
	Complex axioms	Atomic axioms	Atomic axioms
Precision (per run)	0.867 ± 0.03	0.95 ± 0.02	0.95
Recall (per run)	N/A	0.15 ± 0.017	0.38
Recall (over 20 runs)	N/A	0.54	0.38

class disjointness axioms about this topic was manually created, which we called *Gold Standard*. The process of creating the *Gold Standard* was carried out by knowledge engineers and consisted of two phases. In the first phase, the disjointness of the top-most classes to their siblings was assessed manually. Therefrom, two sibling classes being disjoint will infer automatically the disjointness of their corresponding pair of subclasses. This process is repeated in the same way on the next level of concepts. The second phase of *Gold Standard* creation consisted in manually annotating the disjointness for the not yet noted 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 the disjointness evaluation between 1,891 pairs of distinct classes relevant to the chosen topic. Table 3 summarizes the performance of our approach in discovering axioms with the parameters setting in Table 2 over 20 runs. The precision and recall are computed by comparison to the *Gold Standard*. Although the main purpose in our research is to focus on exploring the more complex disjointness axioms which contain logical relationship—intersection and union expression, we also performed experiments to generate axioms involving atomic classes only, for comparison purpose. We carry out the comparison with the results of *GoldMiner* [10] in generating class disjointness axioms about the topic *Work*. The precision and recall are computed by comparison to the *Gold Standard*. The results in Table 3 confirm the high accuracy of our approach in discovering class disjointness axioms in the case of atomic expressions (Precision $= 0.95 \pm 0.02$). Also, the recall value is higher than the value in *GoldMiner*. There are a number of class disjointness axioms generated in our experiments which are absent in the result of *GoldMiner*. For example, there are no any axioms relevant to class Archive in the axioms generated by *GoldMiner*. In the case of more complex axioms, there is

a smaller degree of precision (Precision $= 0.867 \pm 0.03$). The reason may stem from the complexity in the content of generated axioms which is relevant to more different classes. We do not present the recall for the case of complex axioms, since the discovery process of this type of axioms cannot define how many of the complex axioms should have been generated. After 20 runs, from 10,000 candidate individual axioms, we got 5,728 qualified distinct complex axioms. We performed an analysis of the discovered axioms and found some noticeable points. Almost all generated axioms have high fitness values with millions of support instances from the DBpedia dataset, which witness the generality of the discovered axioms. However, we found some deficiencies in determining the disjointness of classes. As in the case of axiom DisjointClasses(MovingImage, ObjectUnionOf(Article,ObjectUnionOf(Image, MusicalWork))), 4,839,992 triples in DBpedia confirm that this class disjointness axiom is valid. However, according to the *Gold Standard*, these two classes should not be disjoint *a priori*. Indeed, the class MovingImage can be assessed as a subclass of Image, which makes the disjointness between class MovingImage and any complex class expression involving relational operator union of class Image altogether impossible. Another similar case is the axiom DisjointClasses(ObjectUnionOf(Article, ObjectUnionOf(ObjectUnionOf(ObjectUnionOf (ObjectUnionOf(TelevisionShow, WrittenWork), MusicalWork), Image), Film)), UndergroundJournal), having 5,037,468 triples in its support. However, according to the *Gold Standard*, these two classes should not be disjoint.

From the above examples we can infer that the main reason for such erroneous axioms may lie in the inconsistencies and errors in the DBpedia dataset. Therefore, a necessary direction to improve the quality of the knowledge base is to use the results of our mining algorithms to pinpoint errors and inconsistencies and thus aim at tighter constraints by excluding problematic triples from the RDF repository.

6 Conclusion and Future Work

We proposed an algorithm based on GE to discover class disjointness axioms from the DBpedia dataset relevant to the topic "Work". The experiment results have allowed us to evaluate the effectiveness of the model and analyze some of its shortcomings.

In future work, we will focus on three main directions:

1. improving the diversity of generated axioms by applying the crowding method at the level of phenotypes;
2. mining different types of axioms like identity axioms, equivalence axioms and relevant to broader topics;
3. enhancing the performance of the algorithm on parallel hardware in order to be able to carry out bigger data analytics.

References

1. Guarino, N., Oberle, D., Staab, S.: What Is an Ontology? Handbook on Ontologies. International Handbooks on Information Systems, Springer (2009)
2. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent Systems 16(2), 72–79 (March 2001)
3. Lehmann, J., Völker, J.: Perspectives on Ontology Learning, Studies on the Semantic Web, vol. 18. IOS Press (2014)
4. Drumond, L., Girardi, R.: A survey of ontology learning procedures. In: WONTO. CEUR Workshop Proceedings, vol. 427. CEUR-WS.org (2008)
5. Hazman, M., El-Beltagy, S.R., Rafea, A.: Article: A survey of ontology learning approaches. International Journal of Computer Applications 22(8), 36–43 (May 2011)
6. Zhao, L., Ichise, R.: Mid-ontology learning from linked data. In: JIST. Lecture Notes in Computer Science, vol. 7185, pp. 112–127. Springer (2011)
7. Tiddi, I., Mustapha, N.B., Vanrompay, Y., Aufaure, M.: Ontology learning from open linked data and web snippets. In: OTM Workshops. Lecture Notes in Computer Science, vol. 7567, pp. 434–443. Springer (2012)
8. 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)
9. Bühmann, L., Lehmann, J.: Universal OWL axiom enrichment for large knowledge bases. In: EKAW. Lecture Notes in Computer Science, vol. 7603, pp. 57–71. Springer (2012)
10. Völker, J., Fleischhacker, D., Stuckenschmidt, H.: Automatic acquisition of class disjointness. J. Web Sem. 35, 124–139 (2015)
11. Lehmann, J.: Dl-learner: Learning concepts in description logics. Journal of Machine Learning Research 10, 2639–2642 (2009)
12. Völker, J., Vrandecic, D., Sure, Y., Hotho, A.: Learning disjointness. In: ESWC. Lecture Notes in Computer Science, vol. 4519, pp. 175–189. Springer (2007)
13. Fleischhacker, D., Völker, J.: Inductive learning of disjointness axioms. In: OTM Conferences (2). Lecture Notes in Computer Science, vol. 7045, pp. 680–697. Springer (2011)
14. Bühmann, L., Lehmann, J.: Pattern based knowledge base enrichment. In: International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 8218, pp. 33–48. Springer (2013)
15. O'Neill, M., Ryan, C.: Grammatical evolution. Trans. Evol. Comp 5(4), 349–358 (Aug 2001), `http://dx.doi.org/10.1109/4235.942529`
16. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments - Chapter 2 Grammatical Evolution, Studies in Computational Intelligence, vol. 194. Springer (2009)
17. Ryan, C., Collins, J.J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: EuroGP. Lecture Notes in Computer Science, vol. 1391, pp. 83–96. Springer (1998)
18. Mahfoud, S.W.: Crowding and preselection revisited. In: PPSN. pp. 27–36. Elsevier (1992)
19. Tettamanzi, A.G.B., Faron-Zucker, C., Gandon, F.L.: Testing OWL axioms against RDF facts: A possibilistic approach. In: EKAW. Lecture Notes in Computer Science, vol. 8876, pp. 519–530. Springer (2014)
20. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 1, 3–28 (1978)