Predicting the Acceptability of Atomic Candidate OWL Class Axioms

Ali Ballout, Célia da Costa Pereira, Andrea G. B. Tettamanzi

Predicting the Acceptability of Atomic Candidate OWL Class Axioms

Ali Ballout
Université Côte d'Azur, I3S, Inria
Sophia Antipolis, France
ali.ballout@inria.fr

Célia da Costa Pereira
Université Côte d'Azur, I3S, CNRS
Sophia Antipolis, France
celia.pereira@unice.fr

Andrea G. B. Tettamanzi
Université Côte d'Azur, I3S, Inria
Sophia Antipolis, France
andrea.tettamanzi@unice.fr

***Abstract*—The task of evaluating the fitness of a candidate axiom against known facts or data is known as candidate axiom scoring. Being able to accurately score candidate axioms is a prerequisite for automatic schema or ontology induction, but can also be useful for ontology and/or knowledge graph validation. Accurate axiom scoring heuristics are often heavy to compute, which is a big problem if one wants to exploit them in iterative search methods like level-wise generate-and-test or evolutionary algorithms, where large numbers of candidate axioms need to be scored. We tackle the challenge of learning a predictive model as a surrogate to reasoning, that predicts the acceptability of candidate class axioms, that is fast to execute yet accurate enough to be used in such settings. For this purpose, we leverage a semantic similarity measure extracted from the subsumption hierarchy of an ontology. We prove that the method proposed in this paper is able to learn the acceptability labels of candidate OWL class axioms with high accuracy and that it can do so for multiple types of OWL class axioms.**

***Index Terms*—Ontology Learning, OWL Class Axioms, Concept Similarity, Reasoning**

I. Introduction

Ontologies play a crucial role in organizing digital information by categorizing entities and defining their relationships [1]. Despite their importance, creating and maintaining ontologies is resource-intensive, often requiring domain experts. Ontology learning aims to mitigate this by automating ontology construction and enrichment [2].

Ontology learning is a multidisciplinary field, incorporating techniques from:

- Natural Language Processing (NLP) for data preprocessing and term extraction [3].
- Data-driven methods like data mining for identifying terms and their relationships [4].
- Inductive Logic Programming (ILP) to generate hypotheses based on existing knowledge [5].

These techniques contribute to various stages of ontology learning, including preprocessing, term extraction, and relation identification [3]. However, they come with limitations like high computational cost and reliance on potentially inaccurate data.

Our work introduces a new approach that overcomes these limitations by reducing computational costs and improving accuracy. Specifically, our method predicts labels for a range of axiom types without relying on error-prone statistics.

The remainder of this paper is organized as follows: Section II discusses related work, Section III provides essential background, Section IV describes our methodology, and Section V presents our experiments and findings. We conclude with notes and observations.

II. Related Work

Data-driven ontology learning methods such as the possibilistic heuristic by Tettamanzi et al. [6] focus primarily on statistical analyses using RDF data. Although these methods achieve high accuracy, they are time-consuming and susceptible to errors. An attempt to mitigate the time issue through time-capping resulted in a slight increase in the error rate [7].

Inductive Logic Programming (ILP) offers alternative approaches, exemplified by the `SoftFOIL` method [8]. While effective in automating the induction of fuzzy axioms, it suffers from limitations such as a greedy search algorithm and the risk of entering infinite loops.

Emerging hybrid methods aim to combine the benefits of both ILP and statistical learning. One example is the work by Malchiodi et al. [9], which modified the support vector clustering algorithm to predict the possibilistic scores for OWL axioms. These methods show promise but are resource-intensive.

For OWL class axiom labeling, frameworks like that developed by Ballout et al. [10] employ semantic similarity for training models focused on the binary classification of axioms. These approaches have demonstrated efficacy but are not standalone solutions [11].

In our current work, we aim for efficient and accurate binary classification of candidate axioms. Our method is designed to function either as a standalone labeler or as an extension to existing methods like DL-Learner [12]. We benchmark our approach against state-of-the-art description logic reasoners using DBpedia to validate its efficacy and efficiency.

III. Background

Our primary goal is to create a model for predicting the acceptability of axioms, independent of potentially error-prone instance data. This necessitates a labeled training set of axioms and a method for determining their semantic relationships as model features.

Semantic similarity serves as a measure of the semantic closeness between ontological concepts, often based on the subsumption (rdfs:subClassOf) relation [13]. This is distinct from semantic relatedness; for instance, a *train* and an *airplane* are semantically similar as both are vehicles [13], [14]. Such similarity measures are often based on path lengths and node depths in a subsumption hierarchy, or on information content derived from ontology and corpus statistics [15].

Corby et al. proposed a semantic similarity measure based on the ontological distance between concepts in an inheritance hierarchy [14]. The ontological distance is calculated using the lengths of subsumption paths and the depths of concepts in the hierarchy. This measure has been implemented in the `Corese` software platform [16], which we employ to compute concept similarity in our method.

$$\begin{aligned} D_H(t_1,t_2) &= \min_t(l_H(\langle t_1,t\rangle)+l_H(\langle t_2,t\rangle)) \\ &= \min_t\left(\sum_{\{x\in<t_1,t>,x\neq t_1\}} 1/2^{d_H(x)} \right. \\ &\quad \left. + \sum_{\{x\in<t_2,t>,x\neq t_2\}} 1/2^{d_H(x)}\right). \end{aligned} \quad (1)$$

In summary, the ontological distance $D_H(t_1,t_2)$ in hierarchy H is the minimum sum of subsumption path lengths between t_1 and t_2 and a common supertype. The path length from a type t_1 to its direct supertype t is $1/2^{d_H(t)}$, where $d_H(t)$ is the depth of t in H. This measure is implemented in the `Corese`[1] software platform [16], which we use for computing concept similarity in our approach.

IV. Method

In an **OWL** ontology containing an inheritance hierarchy of concepts formed by the subsumption axiom rdfs:subClassOf, our aim is to predict if a candidate atomic axiom (consisting of a single named class on each side) is accepted or not. We do this by training a model on a set of previously labeled axioms of the same type (one of: subsumption, disjointness, equivalence) and their similarity weights. In the absence of a scored set our method creates one using explicit axioms available in the ontology. To measure the similarity between (candidate) axioms, we construct a similarity measure by extending the *ontological distance* discussed in Section III, which is defined among classes, not axioms. This enables the model to predict the label of any new atomic candidate axiom of the same type. To this end, we consider the following steps:

1) **OWL ontology closure reasoning**: This step involves using DL-Reasoners the likes of HermiT [17] and Pellet [18] to infer all the axioms (binary relations between named classes) that can be inferred from the knowledge available in the ontology. In this paper, we use it when testing the ability of our model to accept axioms that are not entailed by a reasoner, or to reject ones that are entailed by a reasoner.
2) **Axiom extraction and labeling**: This step constitutes the creation of the set of accepted and rejected labeled axioms of a certain type to be learned. One approach can be querying existing axioms and labeling them as accepted/rejected. Another is to use a scorer to label a set of generated candidate axioms to learn.
3) **Semantic measure retrieval and assignment**: This step involves the retrieval of concepts used in our set of axioms, and their ontological distance from the ontology, followed by extending that similarity to those axioms. This was done by calculating a single value that represents the similarity between each pair of axioms, by applying a function such as *Average* to the ontological distances of concepts in those axioms.
4) **Axiom base vector space modeling**: This step focuses on using axiom similarity measures as weights, each axiom can be represented as a vector in an axiom based vector space.
5) **Binary Classification**: This step is dedicated to training a Machine Learning model with the data set (vector space model in addition to the extracted labels) and predicting if new candidate axioms are accepted or rejected.

A. OWL Ontology Closure Reasoning

The optional first step in our methodology involves inferring all entailed axioms from a target OWL ontology, particularly useful when the ontology is rich in explicit axioms. This serves two purposes:

1) Enlarging the training set of axioms, enhancing the experimental accuracy and facilitating performance comparison with DL-reasoners.
2) Enabling the filtering of new labeled or scored axioms against our ontology closure, allowing us to assess the model's performance on axioms not entailed by reasoners.

B. Axiom Extraction and Labeling

In this work, we used two strategies to create a set of labeled axioms. The first approach utilizes existing scoring methods or previously scored axioms from literature. The second, a faster method, is what we term the "type and counter-type technique."

1) Scoring Method: We used multiple sets of scored axioms for the ontology DBpedia, one set[2] of axioms generated and scored by Nguyen *et al.* [19] using the possibilistic heuristic [6]. The rest of teh sets we scored ourselves using the same heuristic. The scorer though accurate, is extremely slow limiting the size of the sets scored. The scorer currently supports DBpedia and is publicly available[3]. We utilized the version detailed in the work of Felin *et al.* [20]. These scored data sets are what we use for our evaluation and comparison with reasoners in the case of DBpedia. The possibilistic heuristic scores are considered our ground truth and baseline. For experiments using other ontologies we employed the second approach.

2) Type and Counter Type Technique: In this technique, we query an ontology for existing axioms of a certain type, thereby obtaining a set of axioms labeled as *accepted*. We then query axioms of the counter type. For instance, subClassOf and disjointWith can be considered counter types to each other. Axioms of the counter type are assigned the label *rejected*. Since existing disjointWith axioms can be considered

[1]https://project.inria.fr/corese/

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

[3]https://github.com/RemiFELIN/RDFMining

false or *rejected* subClassOf, we can construct a sample containing both labels to train a model. We consider disjointWith to be a counter-type for subClassOf and equivalenntClass. While subClassOf and equivalenntClass are counter-types of disjointWith. However, before querying an ontology for existing axioms, it is advisable to apply a reasoner to obtain the closure of the ontology. The reasoner can be any suitable choice, such as HermiT or Pellet, and not necessarily the reasoner included in the engine used. By applying a reasoner to obtain the closure of an OWL ontology, we ensure that we obtain a complete and consistent set of labeled axioms for our model.

The axiom type designated as *accepted* corresponds to the type that the model will be labeling. For instance, if we assign the *accepted* label to disjointWith axioms, it implies that disjointWith is the axiom type that the model is designed to address. While this may initially appear as a limitation, the efficiency of the process mitigates this concern. As demonstrated in Table II, which details the time consumption, the speed of dataset preparation and model training is sufficiently rapid. This allows us to learn a model for each type of axiom within a relatively short time frame.

We employ Query 1 to extract and label existing axioms of both the type and counter-type. Following the extraction, we ensure an equal representation of both type and counter-type axioms. The SPARQL endpoint utilized in this process is Corese [16]. We have selected DBpedia as our test case to illustrate a real-world application. While other ontologies could be employed, DBpedia offers several advantages. It has been widely used in related work and aligns well with the scoring heuristic we employ for evaluating our candidate axioms.

```
SELECT ?class1 ?class2 ?label WHERE {
{ ?class1 a owl:Class . ?class2 a owl:Class . ?class1
    rdfs:subClassOf ?class2
filter (!isBlank(?class1) && !isBlank(?class2))
filter (?class1 != ?class2)
bind(1.0 as ?label) }
Union{ ?class1 a owl:Class . ?class2 a owl:Class .
    ?class1 owl:disjointWith ?class2
filter (!isBlank(?class1) && !isBlank(?class2))
filter (?class1 != ?class2)
bind(0.0 as ?label) }}
```

Query 1: Axiom extraction

C. Semantic Measure Retrieval and Assignment

To be able to assign similarity measures between axioms, we need to retrieve the ontological distances between all classes a construct the concept similarity matrix **(CSM)**. sing *Corese* in which the ontological distance metric is implemented, this translates into a function added to the *SPARQL* query. Query 2 retrieves three columns, the first two contain the combination of all classes with the third containing the ontological distance denoted by similarity. Blank nodes are ignored.

After retrieving the table of similarities it is pivoted to construct a symmetric $n \times n$ matrix where the first column and the first row are the classes and the cells are the similarities between them with a diagonal of only *1's* since as we mentioned the similarity between a class C and itself is *1*.

Labels	*Axioms*	A_0	A_1	...	A_m
Accepted	A_0	1	$S_{0,1}$	...	$S_{0,m}$
Rejected	A_1	$S_{1,0}$	1	...	$S_{1,m}$
⋮	⋮	⋮	⋮	⋱	⋮
Rejected	A_{m-1}	$S_{m-1,0}$	$S_{m-1,1}$	...	$S_{m-1,m}$
Accepted	A_m	$S_{m,0}$	$S_{m,1}$	...	1

Matrix 1: Axiom similarity matrix with labels

```
SELECT * (kg:similarity(?class1, ?class2) as ?s) WHERE {
?class1 a owl:Class . ?class2 a owl:Class
filter (!isBlank(?class1) && !isBlank(?class2)) }
```

Query 2: Class ontological distance retrieval

D. Axiom Base Vector Space Modeling

From the CSM, we can derive the axiom similarity matrix **(ASM)** with labels illustrated in Matrix 1, as explained in the work of Ballout *et al.* [11] which also highlights that the function used to calculate the similarity S between a pair of axioms can be either *Average* or *Minimum*.

We define then a vector-space model to represent axioms as vectors. Indeed, we represent each axiom as a vector whose elements are the kernels representing the similarity value between the considered axiom and the other axioms present in the labeled dataset T_A. We have been inspired by the Kernel trick in Support Vector Machine (SVM) which is used to deal with nonlinear classification [21].

The number of dimensions m of our vector space corresponds to the number of axioms we have in T_A. Each axiom is then represented as a vector V in this m-dimensional space, and corresponds to a row in the axiom similarity Matrix 1.

The axiom similarity matrix is updated whenever an axiom is generated or a new candidate axiom is suggested.

E. Binary Classification

It is now possible to apply classification machine learning methods using our labeled dataset of axioms represented as vectors. We used a range of methods throughout our experiments, including but not limited to trees, random forests, kNN, Support vector classifier, gradient boosting, and neural networks. Trees and random forests were used to check that there was no bias during the classification. The reason for this choice is that such methods allow us to interpret our predictive model easily and visually analyse the decisions. kNN instead was used to test the effect of the similarity measure as a distance (the metric used was Manhattan, due to the large number of dimensions and the weight was distances). To avoid information leakage since the matrix is symmetric, considering the size of the matrix is $m \times m$, all predicting models were trained using an $m' \times m'$ sub-matrix, with $m' < m$, of the axiom similarity matrix with the labels, and then tested on the $(m - m')$ remaining axioms. Our goal is then to predict the

TABLE I: Ontology statistics. Reasoner used: Corese built in reasoner: ^{C}, Pellet: ^{P}, Hermit: ^{H}.

Ontology	Classes	subClassOf	disjointWith	equivalentClass
NTNames^{C}	47	278	10	50
Pizza^{C}	101	651	5	101
MatOnto^{C}	847	853	158	9
DBpedia^{HPC}	866	7334	70669	268

TABLE II: Time cost per step for investigated ontology in seconds for subClassOf.

Ontology	CSM	Axioms processed	ASM	Single Axiom vector encoding	Model training
NTNames	0.02	556	22	0.03	0.2
Pizza	0.04	1302	105	0.05	0.4
MatOnto	2.84	1706	212	0.09	0.6
DBpedia	2.17	8600	2497	0.08	3.9

label of those axioms which have not been used during the training period. The labels are binary, where 1 represents the label *Accepted* and 0 represents the label *Rejected*.

V. Experiments and Results

For the following experiments, we used this configuration:

- Dual CPUs: 2 × Intel(R) Xeon(R) CPU E5-2689 0 @ 2.60GHz total of 16 cores and 32 threads.
- A total of 32 GB of RAM memory @ 1600 MHz.
- Code and ontologies used available in our repository.[4]

A. Dataset Preparation

As outlined in Section IV-E, our experiments spanned various classification methods, ontologies, and axiom types. For illustrative purposes, we will focus on disjointness axioms to demonstrate the absence of leakage or bias from using the subClassOf hierarchy. Our initial step involved constructing the datasets as follows:

1) Ontology Selection: We selected ontologies of varying sizes and domains for our experiments, with details summarized in Table I. The chosen ontologies include:

- **NTNames**[5]: A small, hand-crafted ontology focusing on New Testament names.
- **Pizza**[6]: A medium-sized ontology from the University of Manchester, widely recognized in the field.
- **MatOnto**[7]: A material science ontology used by the Ontology Alignment Evaluation Initiative[8].
- **DBpedia**[9]: A large, real-world knowledge base derived from Wikipedia, selected for its complexity and size. This ontology was also used for DL-reasoner application and further experimentation.

2) Axiom Extraction and Labeling: Each ontology was loaded into Corese. Since in some of the ontologies the explicit number of axioms was not big enough for any meaningful experiment, we applied different reasoning methods to obtain the maximum number of deduced axioms. For all ontologies, we used Corese reasoner, which is a rule-based engine that can handle RDF(S) and OWL 2 RL profiles. For DBpedia specifically, we additionaly used Hermit and Pellet reasoners to compute the closure, a step which took two hours for each run using Protegé.[10] Additionally, we used the disjointness axioms generated and scored by Nguyen *et al.* [19] for DBpedia, as explained in Section IV-B. After that, we extracted a balanced set of *Accepted* and *Rejected* axioms from each ontology using SPARQL queries and the technique of type and counter type explained in Section IV-B2. For instance, for subsumption in DBpedia, we selected 4300 *Accepted* and 4300 disjointWith axioms representing *Rejected* subClassOf axioms, resulting in a total of 8600 axioms for T_A, which is our axiom dataset for this scenario.

Explicit axiom extraction by querying an ontology using a SPARQL endpoint needs minimal time, as for generating a set of labeled axioms as in [19] the time cost would depend on the scorer and the method used to generate candidate axioms.

3) Concept Similarity Matrix: After preparing the set of axioms, we have to produce the concept similarity matrix **(CSM)**, as explained in Section IV-C. The processing time for creating the **CSM** depends on the number of concepts. In our tested dataset, MatOnto had 847 concepts and creating its **CSM** took 2.84 seconds. In Table II we present the timings for the worst-case scenario, which consists of applying our method to the most dense axiom type that is subsumption for comparison reasons. Other axiom types are naturally much faster due to their lesser population.

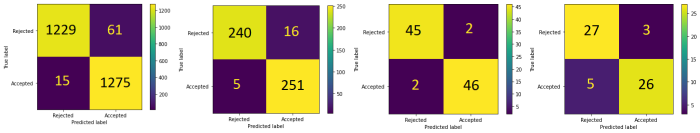
4) Axiom Similarity Matrix and Vector Space: The next step is constructing the axiom similarity Matrix 1 **(ASM)**, and encoding each of the axioms as vectors V in our vector space.

Table II details the time required to complete these operations, all the values presented are the mean average of *five* consecutive runs. As observed the time needed to complete the task of building the *ASM* significantly increases as the number of axioms processed increases, but we also know that it depends on the size of the *CSM* as well from the time needed to encode a single axiom into a vector. As the number of concept increases the *CSM* being searched increases in size as it has the shape $n_{concepts} \times n_{concepts}$ and the number of cells n^2. We would like to note that the test scenario presented in Table II is extreme, especially the case of DBpedia, as the number of axioms needed for training a model does not need to be as large to achieve peak accuracy/results.

B. Training and Testing

Using the dataset obtained as described in the previous section, the classifiers we tested were Decision Trees (DT), Random Forests (RF), kNN, support vector classifier (SVC), Neural Networks (NN) and Gradient boosting (GB). Experiments were performed over every class axiom type, where the axiom population was enough, for every ontology in our dataset some results are shown in Figure 1. We also experimented with both Average and Minimum functions to

[4]https://github.com/ali-ballout/axiom-acceptability
[5]https://semanticbible.com/index.html
[6]https://protege.stanford.edu/ontologies/pizza/
[7]https://github.com/inovexcorp/MatOnto-Ontologies
[8]http://oaei.ontologymatching.org/
[9]http://downloads.dbpedia.org/
[10]https://protege.stanford.edu/



(a) subClassOf DBpedia (b) subClassOf MatOnto (c) disjointWith MatOnto (d) equivalentClass Pizza

Figure 1: Performance by axiom type for investigated ontologies

TABLE III: F1 scores using DBpedia owl:disjointWith, average and minimum as similarity functions.

Function	GB	kNN	NN	RF	SVC	DT
F1 (AVG)	**0.988**	0.976	0.953	0.985	0.976	0.982
F1 (MIN)	**0.977**	0.974	0.962	0.976	0.962	0.968

get the similarity between axioms S. The results of such experimentation can be found in Table III where models were trained using 813 axioms split into 406 *Rejected* and 407 *Accepted* axioms, and tested on 349 axioms split into 175 *Rejected* and 174 *Accepted*.

We would also like to note that the model training times mentioned in Table II, are the average for the above mentioned methods except for the extreme case of DBpedia subsumption using gradient Boosting, where it may take up to 136 $seconds$ to train the model on a set of $6,000$ axioms.

In order to prove that the proposed kernel-based representation of axioms is advantageous for addressing the task of predicting the acceptability of candidate axioms, we performed an additional experiment. For this experiment we used a simple naive kNN method using our axiom similarity measure *directly* to check the n nearest axioms to a candidate axiom and then assigning that candidate axiom the label that occurs most in the n neighbors. To make things fair, we compare the performance of this naive approach to our proposed kernel-based vector-space representation of axioms, using kNN. We perform the comparison using DBpedia's subClassOf axioms as a dataset. We used as a training set 200 axioms, and for the test set 4000 axioms; both sets are balanced in terms of labels.

Table IV presents the F1 scores of each of the methods as a function of the number n of neighbors. From the results we can see that the naive method performs best when the number of neighbors is 1, this is because the method does not learn anything and simply retrieves the closest axioms based on the similarity measure. Our proposed method on the other hand maintains performance since it uses the similarity matrix as a kernel to map the neighbors to a plane of dimensions equal to the number of axioms used in training. Our method outperforms the naive one by a significant margin every time. This corroborates the hypothesis that the proposed kernel-based vector-space representation of axioms is capable of capturing useful features of the semantics of candidate axioms, this offering a clear advantage over the simple and direct use of the axiom similarity matrix.

TABLE IV: F1 scores for the naive nearest neighbor method and our proposed method using K-nearest neighbor as the machine learning model, as a function of number n of neighbors.

Model	n_1	n_3	n_{15}
Naive	0.59	0.49	0.33
kNN	0.86	0.87	0.86

C. Proof of Concept

In order to prove that the proposed kernel-based representation of axioms is advantageous for addressing the task of predicting the acceptability of candidate axioms, we performed an additional experiment. For this experiment we used a simple naive kNN method using our axiom similarity measure *directly* to check the n nearest axioms to a candidate axiom and then assigning that candidate axiom the label that occurs most in the n neighbors. To make things fair, we compare the performance of this naive approach to our proposed kernel-based vector-space representation of axioms, using kNN. We perform the comparison using DBpedia's subClassOf axioms as a dataset. We used as a training set 200 axioms, and for the test set 4000 axioms; both sets are balanced in terms of labels.

Table IV presents the F1 scores of each of the methods as a function of the number n of neighbors. From the results we can see that the naive method performs best when the number of neighbors is 1, this is because the method does not learn anything and simply retrieves the closest axioms based on the similarity measure. Our proposed method on the other hand maintains performance since it uses the similarity matrix as a kernel to map the neighbors to a plane of dimensions equal to the number of axioms used in training. Our method outperforms the naive one by a significant margin every time. This corroborates the hypothesis that the proposed kernel-based vector-space representation of axioms is capable of capturing useful features of the semantics of candidate axioms, this offering a clear advantage over the simple and direct use of the axiom similarity matrix.

D. Comparing With Reasoners

In this experiment, we compare the performance of our model with HermiT, Pellet, and Corese in accepting or rejecting candidate OWL class axioms. For this experiment, we prepare a dataset generated and scored by the possibilistic heuristic using DBpedia ontology which is widely used and represents our real-world scenario. The generation and scoring required seven days of processing time. The dataset contains: 811 accepted subClassOf axioms and 745 rejected ones, as well as 1276 accepted disjointWith axioms and 823 rejected ones. We split the dataset into training and test sets. The split was as follows: the subClassOf training set contains 399 accepted and 398 rejected, while the test set contains 412 accepted and 347 rejected. As for disjointWith, the training set contains 524 accepted and 596 rejected axioms, while the test set contains 752 accepted and 227 rejected.

We evaluated our model and the reasoners on test sets, employing two training scenarios for our model. In the first scenario, we trained our model using a set extracted from

TABLE V: Performance of our proposed model when trained using a scored training set and a training set extracted from the closure of DBpedia as a product of HermiT, Pellet and Corese, compared to reasoners in accepting or rejecting axioms of different types.

		Model scored training set			Model closure training set			Reasoners		
Axiom type	Label	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
subClassOf	Accepted	**1.00**	**0.91**	**0.95**	0.66	1.00	0.79	**1.00**	**1.00**	**1.00**
	Rejected	**0.90**	**1.00**	**0.95**	1.00	0.48	0.65	**1.00**	**1.00**	**1.00**
disjointWith	Accepted	**0.97**	**0.94**	**0.96**	0.27	0.99	0.43	0.00	0.00	0.00
	Rejected	**0.83**	**0.92**	**0.87**	0.99	0.20	0.34	0.23	1.00	0.37

TABLE VI: subClassOf axioms rejected by the scorer and entailed by the reasoners

```
SubClassOf(Tax Genre)
SubClassOf(Organ Aircraft)
SubClassOf(Guitar Locomotive)
SubClassOf(Treadmill ArchitecturalStructure)
SubClassOf(Quote Work)
SubClassOf(Instrument MilitaryVehicle)
SubClassOf(Guitar Aircraft)
```

the closure of DBpedia, produced by the type counter type method. In the second scenario, we used a scored training set independent of the reasoners. We evaluated the reasoners by checking if they could entail the candidate axioms from DBpedia. If a candidate is entailed then it is labeled as accepted otherwise it is labeled rejected. We assessed precision, recall, and F1-score with our ground truth being the labels produced by the possibilistic scorer. The results presented in Table V.

We noticed that the results of our model were worse when training it using the closure, and almost perfect when using the scored training set. Therefore, we hypothesized that the labels of the training set from the closure (reasoners product) could have wrong labels, while the labels from the scorer were more accurate. To test this hypothesis, we ran the scored training set through the reasoners to see how they would would label it. Indeed, the reasoners were entailing axioms labeled as rejected by the scorer, which appear to have a great impact on how the model labeled the test set. A sample of these axioms can be seen in Table VI.

E. Key Findings and Observations

In our experiments, various models trained on the same dataset showed similar results, consistently achieving high accuracy levels averaging between $90-95\%$. Key observations from our experiments are outlined below:

1) Axiom Similarity Function: Throughout our experiments, we have observed a consistent trend where most *ASMs* that were constructed using the *Average* function to obtain S gave better results than those that were built using the *Minimum* function. This can be observed in Table III.

The models are scored based on their performance on the test set (i.e., when classifying axiom candidates never seen before).

We note that Neural Networks were the only model to break this trend. It performed better when the *Minimum* function was used as seen in Table III. However, despite that, it had the worst performance out of all the models presented using both functions, even though different configurations were tested.[11]

2) Classifier Of Choice: Tree-based models consistently achieved the highest scores. This fact is very evident in Table III, where the leading models when using the better similarity function *Average* were, in order: *Gradient Boosting*, *Random Forest*, and *Tree*. For this reason, and since all models score close to each other with an accuracy greater than 95%, we find *Random Forests* combined with the *Average* function for S to be the classifier of choice when creating the model. We chose *Random Forest* over *Gradient Boost* since the time for training *Random Forest* models is faster while giving up at most 0.1% accuracy.

3) Model Perfomance Dealing With Variables : We chose the confusion matrix metric to summarize the results while showing the *support*, and how the method performs with different sizes of datasets and ontologies as a whole. Figure 1 presents the performance of the method on three different ontologies with different sizes as well as different types of axioms and populations. It is very clear how well the method performs across all those variables. The figure shows results only from the test set, i.e., when the model predicts labels for candidate axioms it has never seen before. This brings us to the conclusion that the method is performing as expected with F1 score between 95% to 99%.

VI. Conclusion

We presented a method for accurately predicting the acceptability of various OWL class axioms, using a semantic similarity measure based on ontological distance. The method's robustness was confirmed through extensive testing across different ontologies.

Our approach achieved high accuracy for all OWL axiom types, making it suitable for integration with existing methods like DL-Learner [12] or Grammatical evolution approaches [22]. The effectiveness is attributed to the semantic similarity measure's ability to capture the axioms' model-theoretic semantics, suggesting avenues for further research into the relationship between similarity measures and axiom truthfulness.

Future research directions include:

- Extending the method to complex and property axioms.
- Exploring ensemble approaches incorporating active learning.

[11] Can be found in `keratest.py` in the repository

REFERENCES

[1] F. Hawthorne, S. Mills, F. Hatert, and M. Rumsey, "Ontology, archetypes and the definition of 'mineral species'," *Mineralogical Magazine*, vol. 85, no. 2, pp. 125–133, 2021. [Online]. Available: https://doi.org/10.1180/mgm.2021.21

[2] C. Ramesh, C. K. Rao, and A. Govardhan, "Web mining based framework for ontology learning," *Computer Science & Information Technology*, vol. 5, no. 13, pp. 43–56, 2015.

[3] M. N. Asim, M. Wasim, M. U. G. Khan, W. Mahmood, and H. M. Abbasi, "A survey of ontology learning techniques and applications," *Database*, vol. 2018, no. 2018, pp. 1–24, 2018.

[4] D. Fleischhacker and J. Völker, "Inductive learning of disjointness axioms," in *On the Move to Meaningful Internet Systems: OTM 2011 - Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Hersonissos, Crete, Greece, October 17-21, 2011, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 7045. Springer, 2011, pp. 680–697.

[5] N. Fanizzi, C. d'Amato, and F. Esposito, "DL-FOIL concept learning in description logics," in *Inductive Logic Programming, 18th International Conference, ILP 2008, Prague, Czech Republic, September 10-12, 2008, Proceedings*, ser. Lecture Notes in Computer Science, F. Zelezný and N. Lavrac, Eds., vol. 5194. Springer, 2008, pp. 107–121.

[6] A. G. Tettamanzi, C. Faron-Zucker, and F. Gandon, "Possibilistic testing of OWL axioms against RDF data," *International Journal of Approximate Reasoning*, 2017.

[7] A. G. Tettamanzi, C. F. Zucker, and F. Gandon, "Dynamically time-capped possibilistic testing of SubClassOf axioms against RDF data to enrich schemas," *Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015*, 2015.

[8] F. A. Lisi and U. Straccia, "A logic-based computational method for the automated induction of fuzzy ontology axioms," *Fundamenta Informaticae*, vol. 124, no. 4, pp. 503–519, 2013.

[9] D. Malchiodi and A. G. Tettamanzi, "Predicting the possibilistic score of OWL axioms through modified support vector clustering," *Proceedings of the ACM Symposium on Applied Computing*, pp. 1984–1991, 2018.

[10] A. Ballout, C. da Costa Pereira, and A. G. B. Tettamanzi, "Learning to classify logical formulas based on their semantic similarity," in *PRIMA 2022: Principles and Practice of Multi-Agent Systems - 24th International Conference, Valencia, Spain, November 16-18, 2022, Proceedings*, ser. Lecture Notes in Computer Science, R. Aydogan, N. Criado, J. Lang, V. Sánchez-Anguix, and M. Serramia, Eds., vol. 13753. Springer, 2022, pp. 364–380.

[11] A. Ballout, A. G. B. Tettamanzi, and C. Da Costa Pereira, "Predicting the score of atomic candidate owl class axioms," in *2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2022, pp. 72–79.

[12] L. Bühmann, J. Lehmann, and P. Westphal, "Dl-learner—a framework for inductive learning on the semantic web," *Journal of Web Semantics*, vol. 39, pp. 15–24, 2016.

[13] A. Ballatore, M. Bertolotto, and D. C. Wilson, "An evaluative baseline for geo-semantic relatedness and similarity," *GeoInformatica*, vol. 18, no. 4, pp. 747–767, 2014.

[14] O. Corby, R. Dieng-Kuntz, C. Faron-Zucker, and F. Gandon, "Searching the semantic web: Approximate query processing based on ontologies," *IEEE Intell. Syst.*, vol. 21, no. 1, pp. 20–27, 2006.

[15] H. Al-Mubaid and H. A. Nguyen, "A cluster-based approach for semantic similarity in the biomedical domain," in *IEEE Engineering in Medicine and Biology*, 2006, pp. 2713–2717.

[16] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker, "Querying the semantic web with corese search engine," in *ECAI*. IOS Press, 2004, pp. 705–709.

[17] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang, "HermiT: An OWL 2 reasoner," vol. 53, no. 3, pp. 245–269.

[18] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51–53, 2007, software Engineering and the Semantic Web.

[19] T. H. Nguyen and A. G. Tettamanzi, "An evolutionary approach to class disjointness axiom discovery," in *Proceedings - 2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019*, 2019.

[20] R. Felin, O. Corby, C. Faron, and A. G. B. Tettamanzi, "Optimizing the computation of a possibilistic heuristic to test owl subclassof axioms against rdf data," in *2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2022, pp. 80–87.

[21] M. N. Murty and R. Raghava, "Kernel-based svm," in *Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks*. Cham: Springer International Publishing, 2016, pp. 57–67.

[22] T. H. Nguyen and A. G. B. Tettamanzi, "Learning class disjointness axioms using grammatical evolution," in *Genetic Programming - 22nd European Conference, EuroGP 2019, Held as Part of EvoStar 2019, Leipzig, Germany, April 24-26, 2019, Proceedings*, ser. Lecture Notes in Computer Science, L. Sekanina, T. Hu, N. Lourenço, H. Richter, and P. García-Sánchez, Eds., vol. 11451. Springer, 2019, pp. 278–294.