Scalable Prediction of Atomic Candidate OWL Class Axioms Using a Vector-Space Dimension Reduced Approach

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

Scalable Prediction of Atomic Candidate OWL Class Axioms Using a Vector-Space Dimension Reduced Approach

Ali Ballout[1], Célia da Costa Pereira[1] and Andrea G. B. Tettamanzi[2]
[1]*Université Côte d'Azur, I3S, Inria, Sophia Antipolis, France*
[2]*Université Côte d'Azur, I3S, CNRS, Sophia Antipolis, France*
ali.ballout@inria.fr, {andrea.tettamanzi, celia.pereira}@unice.fr

Keywords: Ontology Learning, OWL Axioms, Concept Similarity, Vector-Space Modeling.

Abstract: Scoring candidate axioms or assessing their acceptability against known evidence is essential for automated schema induction and can also be valuable for knowledge graph validation. However, traditional methods for accurately scoring candidate axioms are often computationally and storage expensive, making them impractical for use with large knowledge graphs. In this work, we propose a scalable method to predict the scores of atomic candidate OWL class axioms of different types. The method relies on a semantic similarity measure derived from the ontological distance between concepts in a subsumption hierarchy, as well as feature ranking and selection for vector-space dimension reduction. We train a machine learning model using our reduced vector-space, encode new candidates as a vector, and predict their scores. Extensive tests that cover a range of ontologies of various sizes and multiple parameters and settings are carried out to investigate the effectiveness and scalability of the method.

1 INTRODUCTION AND MOTIVATION

Ontologies play a critical role in artificial intelligence (AI) systems by providing structured and formal representations of knowledge in a specific domain (Chandrasekaran et al., 1999). In the semantic Web, ontologies can be expressed using the Web Ontology Language (OWL) (OWL Working Group, 2012). These ontologies consist of classes that represent concepts in the domain and relationships that define how they are related (Khadir et al., 2021). They also include a set of axioms that provide a logical basis for reasoning about the domain and making inferences based on the knowledge represented in the ontology. Class axioms, in particular, are important for defining the properties and characteristics of classes in the ontology. For example, a class axiom might specify that a certain class is a subclass of another class, or that it has certain attributes or relationships with other classes. This structured knowledge representation allows AI systems to reason about the domain and make predictions or recommendations based on that knowledge.

However, creating ontologies can be a time-consuming and error-prone process, particularly for large and complex domains. This challenge is known as the knowledge acquisition bottleneck (Cullen and Bryman, 1988). As a solution for this, the field of ontology learning (Maedche and Staab, 2004) emerges, which is the process of automatically constructing an ontology from a given set of data (Lehmann and Völker, 2014). This process involves identifying the classes and relationships that exist within the data and encoding this knowledge in a structured and formal representation. Linguistic and statistical approaches are traditionally utilized for the ontology learning process, where machine learning techniques are often combined with such approaches to complement and improve their results (Khadir et al., 2021).

Candidate axiom scoring involves evaluating the suitability of a candidate axiom based on the available evidence from known facts or data. This task is crucial for automated schema or ontology induction and can also aid in ontology and knowledge graph validation. Essentially, candidate axiom scoring is a technique used in ontology learning to assess the quality of candidate axioms. Its significance lies in its ability to identify the most credible axioms that can be incorporated into an ontology (Ballout et al., 2022b).

Nevertheless, machine learning techniques that tackle the task of candidate axiom scoring, face a scalability problem when dealing with large ontolo-

gies and the number of facts and entities they include (Nickel et al., 2012). This is because the process can be intensive in terms of storage and computation, particularly for large and complex datasets. For ontology learning, this would require techniques to step away from instance data when possible and rely more on what has already been established in an ontology's structure. As a result, scalable techniques and models with the ability of addressing ontologies of different sizes while maintaining satisfactory performance without incurring excessive computational and storage costs become a necessity.

In this paper, we present the issue of dealing with large ontologies and its effect in terms of storage and computation when attempting to score candidate class axioms. We highlight the importance of the challenge at hand by experimenting with a state-of-the-art (SOTA) approach using large-size ontologies. In addition, we propose an approach that scores atomic candidate OWL class axioms of different types for ontologies of different sizes. We do so by utilizing an ontological semantic similarity (Corby et al., 2006) between concepts and extending it to axioms, we incorporate feature selection technique on our dataset to pick the most impactful axioms to act as our dimensions in an axiom-based vector space. We encode candidate axioms into this vector space without the need for any instance data. We experiment using DBpedia, Gene ontology (GO), and Cell ontology (CL) to test the scalability of the approach, as well as the effect of feature selection on performance, storage cost and computation time.

This work is structured as follows: in Sect. 2 we give an overview of some related work and SOTA approaches; Sect. 3 provides some background about ontological semantic similarity, the possibilistic axiom scorer and feature selection. As for Sect. 4 it lays out the methodology explaining how the axioms are extracted and scored, how we build the semantic measure, and also how we model an axiom based vector-space leading to the prediction of a candidate axiom's score. We detail our experiments in Sect. 5 then present and analyze the results in Sect. 6. We end the paper with some notes and conclusions.

2 RELATED WORK

Since the current study aims at developing a novel approach to scalable prediction of candidate class axiom scores, it is relevant to provide an overview of previous research on the topic of predicting the score of candidate OWL class axioms.

Some of the works addressing the challenge of predicting the fitness or score of candidate OWL class axioms build on an idea presented in (Ballout et al., 2022a), where a number of truth-labeled formulas (in this case axioms) are modeled into a vector-space with a semantic similarity measure used as the weights, then a model is trained using these formulas to enable it to predict a label/score for new candidate formulas. This approach proves to be accurate even when the number of available formulas with known scores/labels is sparse. This kind of approach has been validated in works related to ontology learning, such as (Hassanpour et al., 2014), where the authors deal with semantic web rule language (SWRL) rules.

One such method that applies the above approach to OWL class axioms is (Malchiodi and Tettamanzi, 2018). It uses a similarity measure based on instance data counting, which is reminiscent of the Jaccard index. It shows the ability of such a technique to approximate a score for atomic candidate OWL subsumption axioms, but much remains to be improved in terms of performance and accuracy. The same similarity is used in (Malchiodi et al., 2020), which uses methods such as principle component analysis (PCA) to map axioms into a lower-dimension space. This form of dimensionality reduction, which is unrelated to ours in methodology or goal,[1] combined with instance-based similarity measures, resulted in less than satisfactory performance. Indeed, the authors expected to see a clear separation between accepted and rejected axioms, which would have made it possible for unsupervised method to perform the task of labeling candidate axioms, but their results did not support this hypothesis.

However, an instance-based similarity measure fully relies on an ontology's instance data, and any lack of such data results in ignorance, while an excessive amount of data overwhelms the method. Some follow a different path, such as (Chen et al., 2022), which takes the embedding approach utilizing an ontology's class subClassOf hierarchy, also known as the *is-a* hierarchy, to predict subsumers. This approach, like the ones we mentioned before, only addresses subsumption. It also proves to be computationally complex as it follows a breadth-first algorithm when embedding a subsumption relation. This algorithm keeps extracting the subsumers of each of the classes till reaching a leaf, or the superclasses till reaching the root in order to generate a sentence. The authors mention limiting the length of their sentences for the evaluation, highlighting a trade-off between

[1] PCA is a technique used to map data into lower dimensional planes, where as feature selection ranks your dimensions in terms of how useful they are to predict the target value and allows you to drop the low ranking dimensions.

having complete sentence context and redundancy.

This is answered by (Ballout et al., 2022b), which addresses the problem of instance data dependency and provides an alternative axiom-based vector-space modeling technique. This technique uses a semantic similarity based on the subsumption hierarchy of an ontology. This work, as ours, extends its scope to include disjointWith class axioms as well. It highlights a reduction in error rate and computation time compared to other works. In terms of performance, it outperforms (Malchiodi et al., 2020) without any attempt to reduce dimensionality. The authors only experiment using DBpedia, which includes 762 concepts, positioning it as a smaller ontology when compared with ontologies with tens of thousands of concepts such as GO and CL.

In general, previous research on the prediction of candidate class axiom scores has provided valuable insights into this problem and has laid the foundation for the current study. Our work aims to address the shortcomings of the mentioned works, especially with respect to computational complexity, storage cost, and scalability, by developing a novel approach for scalable prediction of atomic candidate OWL class axiom scores. We do this by incorporating feature selection and utilizing the ontological concept semantic similarity extended to axioms detailed in (Ballout et al., 2022b). Our approach, as all the ones mentioned, addresses atomic OWL class axioms, which are axioms containing one concept on each side and, like (Ballout et al., 2022b), it can deal with subsumption, disjointness and equivalence. In this paper, we will compare our work with (Ballout et al., 2022b).

3 BACKGROUND

3.1 Ontological Axiom Semantic Similarity

To achieve scalability, one of the challenges we want to overcome is the reliance on instance data, which can be overwhelming when dealing with larger ontologies. For this reason we seek a similarity measure that is independent of instance data. In this case it is a measure dependent on the **subsumption** hierarchy. In our work, we use a semantic similarity measure between axioms developed in (Ballout et al., 2022b). Here we summarize how that measure is calculated between the axioms and how it is extended from an ontological concept similarity measure.

The concept similarity measure acting as a foundation for the axiom similarity is detailed in (Corby et al., 2006) under the subsection titled *Ontological Approximation*, it is a distance calculated between two concepts by using the subsumption path length with the general definition:

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

Formula 1 translates to: for all type pairs t_1 and t_2 in an inheritance hierarchy H, the *ontological distance* between t_1 and t_2 in the inheritance hierarchy H is the minimum of the sum of the lengths of the subsumption paths between each of them and a common super type. And the length of the subsumption path between a type t_1 and its direct supertype t is equal to $1/2^{d_H(t)}$ with $d_H(t)$ being the depth of t in H.

The authors of (Ballout et al., 2022b) then extend this measure to axioms by performing the following steps:

1. Extract the distances between all concepts in the ontology and store them in a concept similarity matrix 1.
2. Compare each axiom with all other axioms in the dataset.
3. When comparing two axioms, retrieve from the concept similarity matrix the similarity/distance between the concepts on the left side of the axiom.
4. Repeat the previous step for the right side.
5. In case of symmetric axiom types (disjointness/equivalence) repeat the comparison between the left concept from the first axiom and the right concept of the second axiom, and then between the right concept from the first axiom and the left concept from the second one. Keep the higher values between both comparisons.
6. Take the average of the two values that you have as a result of the previous step.
7. Store that value in an axiom similarity matrix 1.

This process is used for the construction of an axiom-based vector-space, where each axiom or candidate can be represented by a vector of its similarity to all other axioms.

3.2 Axiom Scoring via Possibility Theory

Many of the works mentioned in Sect. 2 like (Ballout et al., 2022b; Malchiodi et al., 2020; Malchiodi

and Tettamanzi, 2018), use a dataset of subclassOf axioms scored by a possibilistic heuristic (Tettamanzi et al., 2017). This heuristic uses possibility theory (Dubois and Prade, 1988), a mathematical theory of epistemic uncertainty, whose central notion is that of a possibility distribution that assigns to each elementary event a degree of possibility ranging from 0 (impossible, excluded) to 1 (completely possible, normal). A possibility distribution π induces a *possibility measure* Π, corresponding to the greatest of the possibilities associated to an event and the dual *necessity measure* N, equivalent to the impossibility of the negation of an event.

Since we compare our work with (Ballout et al., 2022b) we consider the same scorer for the DBpedia subClassOf dataset. Here, we provide a brief explanation of the theory behind the scoring.

If we denote by u_ϕ the *support* of ϕ, which is the cardinality of its content, by u_ϕ^+ the number of confirmations of ϕ and by u_ϕ^- the number counterexamples of ϕ, the possibility and the necessity of candidate axiom ϕ may be defined as follows:

- if $u_\phi > 0$,

$$\Pi(\phi) = 1 - \sqrt{1 - \left(\frac{u_\phi - u_\phi^-}{u_\phi}\right)^2}; \quad N(\phi) = \begin{cases} \sqrt{1 - \left(\frac{u_\phi - u_\phi^+}{u_\phi}\right)^2}, & \text{if } u_\phi^- = 0, \\ 0, & \text{if } u_\phi^- > 0; \end{cases} \tag{2}$$

- if $u_\phi = 0$, $\Pi(\phi) = 1$ and $N(\phi) = 0$, we are in a state of maximum ignorance, given that no evidence is available in the RDF dataset to assess the credibility of ϕ.

The possibility and necessity of an axiom can then be combined into a single handy acceptance/rejection index.

$$\begin{aligned} \mathrm{ARI}(\phi) &= N(\phi) + \Pi(\phi) - 1 = N(\phi) - N(\neg\phi) \\ &= \Pi(\phi) - \Pi(\neg\phi) \in [-1, 1], \end{aligned} \tag{3}$$

because $N(\phi) = 1 - \Pi(\neg\phi)$ and $\Pi(\phi) = 1 - N(\neg\phi)$ (duality of possibility and necessity). A negative $\mathrm{ARI}(\phi)$ suggests rejection of ϕ ($\Pi(\phi) < 1$), whilst a positive $\mathrm{ARI}(\phi)$ suggests acceptance ($N(\phi) > 0$), with a strength proportional to its absolute value. A value close to zero reflects ignorance.

Given a candidate axiom ϕ, expressing a *hypothesis* about the relations holding among some entities of a domain, we wish to evaluate its credibility, in terms of possibility and necessity, based on the *evidence*[2] available in the form of a set of facts contained in an RDF dataset.

[2]Instance data in the RDF set that either confirms or contradicts a candidate axiom.

Concepts	C_0	C_1	...	C_n
C_0	1	$S_{0,1}$	...	$S_{0,n}$
C_1	$S_{1,0}$	1	...	$S_{1,n}$
⋮	⋮	⋮	⋱	⋮
C_{n-1}	$S_{n-1,0}$	$S_{n-1,1}$	...	$S_{n-1,n}$
C_n	$S_{n,0}$	$S_{n,1}$	...	1

(a) Concept similarity matrix.

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

(b) Axiom similarity matrix.

Matrix 1: Structure of the concept similarity and axiom similarity matrices.

3.3 Feature Ranking and Selection

In machine learning, feature selection is referred to as the process of obtaining a subset from an original feature set according to certain feature selection criterion, which selects the relevant features of the dataset. It plays a role in compressing the data processing scale, where the redundant and irrelevant features are removed (Cai et al., 2018). It is particularly useful in the case of high-dimensional datasets. It does not involve dimension aggregation, nor attempts to map higher-dimensional spaces to lower ones as done in (Malchiodi et al., 2020).

According to their relationship with learning methods, feature selection methods can be classified into filter, wrapper, and embedded models (Cai et al., 2018). In our work we use the filter model, which has a lesser computational cost than the others (Cai et al., 2018). A good feature selection method should have high learning accuracy but less computational overhead (time complexity and space complexity).

Filter feature selection methods typically utilize evaluation criteria to increase the correlation between the feature and the class label and decrease the correlation among features. In addition, the correlation among features is often replaced by either redundancy or diversity (distance). These measures of relevance, redundancy, and diversity may be identical or distinct. Filter methods involve selecting features based on their individual statistical properties. This can be done using techniques such as correlation analysis or mutual information, which measure the strength of the relationship between a given feature and the tar-

get variable. At the end, features are ranked based on their effect on the class label and then a percentage or number of the can be kept while the rest is discarded.

For example, the mutual information gain (also known as mutual information or MI) between two variables x and y can be calculated as follows:

$$MI(x,y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \quad (4)$$

where $p(x)$ and $p(y)$ are the marginal probability distributions of x and y, respectively, and $p(x,y)$ is the joint probability distribution of x and y. Mutual information measures the amount of mutual dependence between the two variables, with higher values indicating a stronger relationship. It is often used in feature selection to identify the most relevant features for a given task.

4 METHODOLOGY

Our objective is to develop a scalable method to predict a score for atomic candidate OWL class axioms by learning from a set of previously scored axioms of the same type. To this aim, we exploit the hierarchy of concepts formed by the subsumption rdfs:SubClassOf axioms, combined with feature selection. A separate model is required for each type of axiom addressed. Following are the steps of our methodology:

1. **Axiom Extraction and Scoring:** This step describes the creation of the set of scored axioms of a certain type to be learned. One approach would be to use a scorer to label a set of generated candidate axioms to learn as done in (Ballout et al., 2022b; Malchiodi et al., 2020; Malchiodi and Tettamanzi, 2018). Another approach can be to query existing axioms and label them as accepted, then generating some random axioms and checking that they are not explicitly available or entailed in the ontology, and labeling them as rejected as done in (Chen et al., 2022).
2. **Axiom Similarity Calculation:** This step details how we extract the concepts used in our set of axioms and retrieve the ontological distances pertaining to these concepts only. Unlike the SOTA (Ballout et al., 2022b), we only query concepts present in our set of axioms instead of all concepts present in the ontology. We then calculate the axiom similarity measure using the algorithm detailed in (Ballout et al., 2022b) and briefly explained in Sect. 3.1, but we enhance performance by leveraging the power of multiprocessing.
3. **Axiom-Based Vector-Space Modeling:** This step focuses on using the axiom similarity measures as weights; each axiom can be represented as a vector in an axiom-based vector-space.
4. **Vector-Space Dimensionality Reduction:** This step consists of applying feature selection on the axiom similarity matrix to reduce the number of axioms being used as dimensions to a certain predefined number. This results in reduced computation and processing when encoding axioms into the vector-space as well as smaller concept and axiom similarity matrices. Unlike (Ballout et al., 2022b), which does not acknowledge the challenge of dealing with a large or rich ontology, we added this step to ensure that the method does not time out or run out of storage no matter what the size of the ontology is.
5. **Candidate Axiom Encoding:** This step describes how a new candidate axiom is introduced into the vector-space, including the case where the axiom consists of concepts not available in the concept similarity matrix. The SOTA (Ballout et al., 2022b) does not include such a step since it naively queries all available concepts. On the other hand, since we added a step that limits our method to query only concepts found in the set of axioms used, we had to add this new step to query any new concept that might be introduced by a candidate axiom.
6. **Prediction:** This step is dedicated to training a machine learning model with the dataset (vector-space model and scores) and predicting the scores of new candidate axioms.

We begin by preparing the set of scored axioms that we want to use to train our model. Then we extract the concepts which our axiom set consists of. After that we query the ontology to retrieve the ontological distances between only the concepts we extracted. We then calculate the axiom similarity between our axioms and use it to model our axiom-based vector-space. We then utilize feature selection to reduce the size of our vector-space by reducing the number of axioms acting as dimensions to those that are most impactful, which leads to a reduction in the size of the concept and axiom similarity matrices. We train a machine learning model using our new reduced vector-space, encode new candidate axioms as a vector, and predict their scores.

4.1 Axiom Extraction and Scoring

We adopt two approaches to create our set of axioms. The first approach is discussed in (Ballout et al.,

2022b; Chen et al., 2022). It dictates that we first query an ontology for existing axioms of a certain type and by that we obtain a set of axioms which would have positive scores. Following that, we generate rejected axioms. We do this by constructing an axiom with a pair of random concepts, the axiom is of the form subClassOf/disjointWith(C_1C_2), with $C_1 \neq C_2$. We then check if the axiom exists in or is entailed by the ontology; if so, the generated axiom is discarded, otherwise it is kept. The resulting generated set of axioms will have a negative score. The rationale is that a randomly generated axiom can be expected to be false with a very high probability. In our work, we add a limit to the number of axioms being selected and generated, while in the SOTA (Ballout et al., 2022b) every possible combination is generated: this is a critical point when dealing with large ontologies.

Query 1 is used to extract a given number of existing axioms and generate a given number of random ones, followed by removing the existing from the generated. We ignore the blank nodes as well as instances where both classes are the same. This produces a balanced set of positive and negative axioms. The server used is Corese (Corby et al., 2004) which applies reasoning to check entailed axioms.

```
0 select DISTINCT ?class1 ?class2 ?label where {
1       {select ?random ?class1 ?class2 ?label where {
2       ?class1 a owl:Class
3       ?class2 a owl:Class
4       ?class1 rdfs:subClassOf ?class2
5       filter (!isBlank(?class1) && !isBlank(?class2) &&
              (?class1 != ?class2))
6       bind(1.0 as ?label)
7       BIND(RAND() AS ?random) .
8       }ORDER BY ?random
9       limit 500}
10      UNION
11      {select ?random ?class1 ?class2 ?label where
12      {{?class1 a owl:Class
13      ?class2 a owl:Class
14      filter (!isBlank(?class1) && !isBlank(?class2) &&
              (?class1 != ?class2))
15      bind(0 as ?label)
16      BIND(RAND() AS ?random) .}
17      minus
18      {?class1 a owl:Class
19      ?class2 a owl:Class
20      ?class1 rdfs:subClassOf ?class2
21      filter (!isBlank(?class1) && !isBlank(?class2))
22      bind(0 as ?label)
23      BIND(RAND() AS ?random) .}
24      } ORDER BY ?random
25      limit 500}}
```

Query 1: Extraction of an rdfs:subClassOf axiom balanced set with a size of 1000 axioms using random generation.

A second, more judicious approach adopted in (Ballout et al., 2022b) is to only query existing axioms. For example, if we want to train a model to predict subClassOf axioms, we would query for n subClassOf axioms and consider that as the set of positive subClassOf axioms. We would then query n disjointWith axioms and consider them as the set of negative subClassOf axioms. If one query retrieves a number of axioms lesser than the limit n, we can drop the excess axioms from the other set to maintain balance. SOTA (Ballout et al., 2022b) applies no limit and extracts all available axioms. We again provide a limit, as methods that implement the work of (Ballout et al., 2022a) perform well even with small datasets. Query 2 shows our implementation.

```
0SELECT ?class1 ?class2 ?label WHERE {
1       ?class1 a owl:Class
2       ?class2 a owl:Class
3       ?class1 rdfs:subClassOf ?class2
4       filter (!isBlank(?class1) && !isBlank(?class2)
              && (?class1 != ?class2))
5       bind(1.0 as ?label)
6       BIND(RAND() AS ?random) .} ORDER BY ?random
7       LIMIT 500
8SELECT ?class1 ?class2 ?label WHERE {
9       ?class1 a owl:Class
10      ?class2 a owl:Class
11      ?class1 owl:disjointWith ?class2
12      filter (!isBlank(?class1) && !isBlank(?class2)
              && (?class1 != ?class2))
13      bind(0 as ?label)
14      BIND(RAND() AS ?random) .} ORDER BY ?random
15      LIMIT 500
```

Query 2: Extraction of an rdfs:subClassOf axiom balanced set with a size of 1000 axioms using a negated axiom type.

The following step is to score the axioms. This is done by inputting the extracted axioms into the possibilistic heuristic (Tettamanzi et al., 2017), and receiving an output file containing the scores (ARI). We note that the process is very slow; (Malchiodi and Tettamanzi, 2018) mentions that it took a little less than a year to score 722 axioms, whence the need for a method such as ours.

4.2 Axiom Similarity Matrix

Like in (Ballout et al., 2022b), the axiom similarity is derived from the concept similarity. This means that we first need to query *Corese*, where the ontological distance metric is implemented as a function, to retrieve this similarity.

Query 3 provides our implementation of the retrieval of the concept similarity measure. In contrast with (Ballout et al., 2022b) our approach takes advantage of the ability to run multiple queries at a time

and considers only concepts found in our set of axioms instead of all concepts in the ontology. This results in an initial reduction of computational cost and matrix size. We divide our set of concepts into k subsets, then query Corese with k queries, each including one of those subsets and the main set concepts for the distances between them. This allows us to speed up the process of creating the concept similarity matrix 1 by k times. This also drastically reduces the storage space needed to store the matrices, by reducing the values to exactly what is used.

```
0select * (kg:similarity(?class1, ?class2) as
      ?similarity) where {
1          ?class1 a owl:Class
2          ?class2 a owl:Class
3          filter (!isBlank(?class1) &&
                !isBlank(?class2) &&
4          str(?class1) IN (subset) && str(?class2) IN
                (concepts))}
```

Query 3: Concept ontological distance retrieval.

Next, we use the algorithm explained in Sect. 3.1 and detailed in (Ballout et al., 2022b) (*Algorithm 1* under subsection '*Semantic Measure Construction and Assignment*') to calculate the similarity between axioms. We end up with an $m \times m$ axiom similarity matrix 1. The diagonal of this matrix will contain only 1s as the similarity between an axiom A and itself is 1.

4.3 Axiom Base Vector-Space Modeling

We model our vector space to encode axioms into vectors. The initial number of dimensions d of this vector space is equal to the number of axioms in our scored axiom set. Each axiom can be represented as a vector V in this d-dimensional space. Considering the weights of our vectors are the similarities between axioms, it would be intuitive to view our axiom similarity matrix as a representation of our axiom-based vector space.

4.4 Vector-Space Dimension Reduction

We now shift our focus to ranking and reducing the dimensions of our vector space. By doing so we achieve the following:

- A reduction in the error rate due to the reduction in noise and redundancy.
- A reduction in the size of our vector-space and storage space for the axiom similarity matrix.
- A reduction in the size of our concept similarity matrix. The matrix will only include concepts that constitute axioms acting as dimensions in our vector-space.
- A reduction in the computational complexity when encoding new candidate axioms into the vector-space. We will be comparing the new axiom to a subset of the initial axioms acting as dimensions d.
- A reduction in the look-up time when retrieving the concept similarity value from the new smaller concept similarity matrix.
- A better dataset for our machine learning model to train on with regards to redundancy.

To this aim, we consider our dimensions as features and apply a supervised filter-type feature selection method such as mutual information. This works by taking as input the axiom similarity matrix along with the scores of the axioms and returning a ranking of the dimensions from the most to the least impactful . We then keep a percentage of these dimensions according to their ranks and discard the rest.

Our new axiom similarity matrix is of size $m \times z$, z being the number of dimensions selected from the original dimensions d. This in turn affects the concept similarity matrix, which will become of size $n \times f$, f being the number of concepts included in the selected axioms acting as dimensions in our reduced vector space. New candidate axioms will be encoded into the vector space with the reduced number of dimensions. This means lower processing complexity in terms of computation cost and storage cost, which are the keys for scalability.

4.5 Candidate Axiom Encoding

The candidate axiom encoding process includes two cases. In the first case, the candidate axiom is made up of two concepts already found in our concept similarity matrix. If so, the candidate axiom goes straight through the algorithm mentioned in Sect. 3.1, as we did with the training set. The candidate then becomes a new vector in our vector-space, ready for score prediction.

The second case covers candidate axioms that contain concepts not found in our concept similarity matrix. Such candidates invoke a new similarity measure retrieval query 3. In this query the *subset* variable includes the new concepts and the *concepts* variable includes the concepts found in the new reduced set of axioms acting as dimensions. This produces at most two new rows to be added to the matrix, if none of the candidate's concepts are in the concept similarity matrix. After that, the candidate proceeds normally through the vector encoding algorithm.

4.6 Prediction

Now that we have our reduced vector space, we can apply machine learning methods. A simplistic method such as k-NN can be used to highlight the strength of our similarity measure, so we use it along with more sophisticated methods such as random-forest regressors. We choose this method to be able to compare with (Ballout et al., 2022b), since they achieve their best results using it.

5 EXPERIMENTATION PROTOCOL

We use the following hardware configuration for our experiments:

- CPU: Intel(R) Xeon(R) CPU W-11955M @ 2.60GHz base and 4.5 GHz all core boost. With 8 cores and 16 threads.
- A total of 128 GB of RAM memory with frequency 3200 MHZ.
- 1 TB of NVME SSD storage with read and write speeds of up to 2000 MB per second.

In addition, the code[3] uses the Python multiprocessing package to distribute the dataset-building and querying tasks over all available cores.

We searched for ontologies of different sizes and domains to use in our experiments and selected the following:

- DBpedia,[4] a project aiming to extract structured content from the information created in the Wikipedia project. This was designated as the real-life scenario having a large number of concepts as well as being prone to errors and not being hand-crafted by engineers. Number of concepts: 762.
- GO, [5] which is the Gene ontology, the world's largest source of information on the functions of genes. Number of concepts: 29,575.
- CL, [6] which is the Cell ontology, designed as a structured controlled vocabulary for Cell types,t his ontology was constructed for use by the model organism and other bioinformatics databases. Number of concepts: 78,835.

[3]https://github.com/ali-ballout/Scalable-Prediction-of-Atomic-Candidate-OWL-Class-Axioms-Using-a-Vector-Space-Dimension-Reduced-Appr

[4]https://www.dbpedia.org/resources/ontology/

[5]http://geneontology.org/docs/download-ontology/

[6]https://www.ebi.ac.uk/ols/ontologies/cl

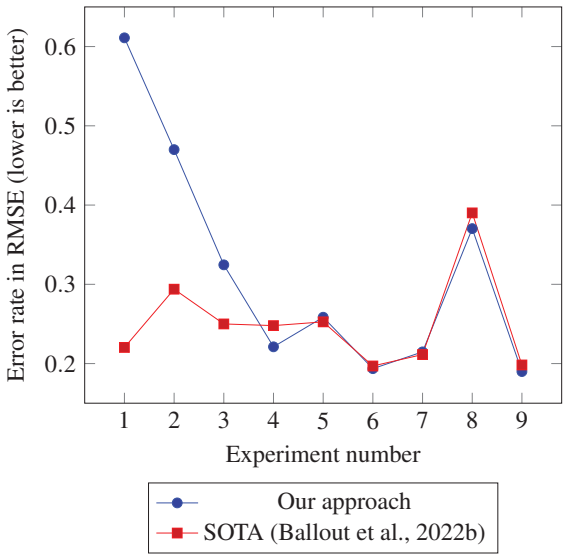

Figure 1: A graph comparing the performance of our proposed approach and the SOTA in RMSE using the subClassOf dataset containing 722 axioms through out 9 experiments (train/test splits), in each experiment the SOTA is trained using 100% of the dimensions, while our model is trained using a selected percentage of the dimensions equivalent to the experiment's number $\times$ 10% so from 10% to 90%

For DBpedia, and to be able to compare with (Ballout et al., 2022b), we use the same scored subClassOf dataset consisting of 722 axioms. The dataset is scored using the possibilistic heuristic detailed in (Tettamanzi et al., 2017) and briefly explained in Sect. 3.2. As for GO and CL, and since (Ballout et al., 2022b) does not experiment using these ontologies, we create our own disjointWith datasets using the process described in Sect. 4.1. For each ontology, we create a balanced set of 600 axioms. For the sake of experimentation and since scoring 722 axioms in our smallest tested ontology took little under a year (Malchiodi and Tettamanzi, 2018), we gave the score of 1 to all positive axioms and 0 to all negative axioms. This would turn the task in those cases to classification, which is not an issue, as long as the approach proves scalable and with good accuracy. After all, the work that is the foundation for all these vector-space approaches (Ballout et al., 2022a) is presented as a formula classifier.

Following the preparation of our dataset, we are now able to train a regressor, in the case of DBpedia, for performance experimentation in terms of error rate. To compare with (Ballout et al., 2022b), we use a random forest regressor, with which they achieve their best results. In our experiments we consider **Processing time** as the time needed in seconds to construct the concept similarity matrix (CSM) and the axiom

similarity matrix (ASM) and **Encoding time** the time needed to encode cone new candidate axiom. And in regards to storage cost, we consider the size in mega bytes (MB) for the ASM and in number of values stored for the CSM.

With the smallest ontology, DBpedia, we perform experiments to analyse the effect of feature selection on prediction accuracy. Since the DBpedia datasets are correctly scored using (Tettamanzi et al., 2017), they are the most suitable to use for this type of analysis. The results for these experiments are presented in Table 1 and Figure 1.

Using GO and CL we perform experiments to analyse the impact of our approach on storage usage and computational cost. For these experiments we set our feature selection percentage at 40% based on the results from Figure 1. We also note that in these experiments we do not include the performance metric for lack of space and since its not our main concern, but would like to highlight that an average F1 score of 0.86 was achieved in both experiments. We compare the feasibility of both our approach and (Ballout et al., 2022b) when applied to these ontologies. The results are presented in Table 2 for GO and Table 3 for CL.

6 RESULTS AND ANALYSIS

Figure 1 depicts the performance of both our proposed method and the SOTA (Ballout et al., 2022b). It compares the error rate in terms of RMSE for both methods through out a series of 9 experiments where the percentage of dimensions selected in our method is incremented by 10% each run. This plot shows the effect of feature selection on the prediction accuracy of the model. We can see that when a very small number of dimensions is selected ($< 30\%$) the method cannot make accurate predictions. The error rate decreases as we increase the number of selected dimensions until we reach 40%. Here we can see that our approach performs better with fewer dimensions that the SOTA. After the 40% mark, we get a similar or slightly better performance. We can conclude from this that 40% to 50% can be considered an optimal percentage of dimensions to remove redundancy and improve performance. This is exactly the reason why we set the feature selection percentage parameter to 40% for the GO and CL experiments.

Table 1 highlights the effects of our approach on computational cost and storage cost. Due to our method querying only selected concepts instead of all concepts, we can see that the initial size of the concept similarity matrix is almost seven times smaller than that in the SOTA while processing with the same set of axioms. This smaller size results in faster lookups to calculate the axiom similarity measure, which leads to a reduction in time cost from 13.7 seconds to 3.8 seconds. Also, fewer dimensions in our vector-space lead to a faster encoding time for a candidate axiom as we can see a reduction from 0.019 seconds to 0.0053 seconds.

In Table 2, we see from the size of the CSM and the concepts queried, that our method has scaled well from a small-size ontology such as DBpedia to a larger one such as GO. Even though the number of concepts in the ontology addressed changes from 762 to $78,835$ our method is able to maintain almost the same size of the CSM by dealing with a rela-

Table 1: Comparison of computational cost in seconds as well as storage cost in number of values for CSM using using the DBpedia scored subClassOf dataset.

Approach	Number of axioms processed	Initial CSM size	Concepts queried	Processing time	Encoding time
SOTA	722	580,644	762	13.72	0.019
proposed approach	722	**85,264**	**292**	**3.86**	**0.005**

Table 2: Comparison of computational cost in seconds as well as storage cost in MB for ASM and number of values for CSM using the GO disjointWith dataset. Time out error: TO.

Approach	Number of axioms processed	ASM size	Initial CSM size	Concepts queried	Processing time	Encoding time
SOTA	600	62,000	6,214,957,225	78,835	TO	TO
proposed approach	600	**5.1**	**95,481**	**309**	**312.54**	**0.034**

Table 3: Comparison of computational cost in seconds as well as storage cost in MB for ASM and number of values for CSM using the CL disjointWith dataset. Time out error: TO.

Approach	Number of axioms processed	ASM size	Initial CSM size	Concepts queried	Processing time	Encoding time
SOTA	600	8,000	874,680,625	29,575	TO	TO
proposed approach	600	**2.05**	**216,225**	**465**	**103.7**	**0.015**

tively small number of concepts (309). When compared to (Ballout et al., 2022b), we notice that it is unfeasible to apply the method to the ontology. Concerning computational cost, the SOTA times out and crashes without completing the task. As for storage cost, the size of the CSM for the SOTA would be approximately 62 Gbytes compared to 5.1 Mbytes for our method.

We notice that our approach consumes an increased amount of processing time up to 312 seconds but maintains a very short axiom encoding time of 0.034 seconds. This increase in processing time is attributed to the querying of the semantic similarity measure in such a large ontology. It is dependant on the capability of the SPARQL endpoint and the size of the ontology. However, this is well within acceptable time.

Similarly, when dealing with the medium-size CL, having 29,575 concepts, our approach displays consistency and stability in terms of storage and computation. Processing time is 103 s, which falls within expectation when compared to the processing time of GO, the same can be said for the encoding time. Again, approach (Ballout et al., 2022b) times out and crashes proving neither feasible nor scalable.

7 CONCLUSION

We have proposed a scalable approach for the score prediction of atomic candidate *OWL* class axioms of different types. The method relies on a semantic similarity measure derived from the ontological distance between concepts in a subsumption hierarchy, as well as feature selection for vector-space dimension reduction. Extensive tests that covered a range of ontologies of different sizes as well as multiple parameters and settings were carried out to investigate the effectiveness and scalability of the method.

The results obtained support the effectiveness of the proposed method in predicting the scores of the considered OWL axiom types with lower error rates than the SOTA. More importantly, it does so while being scalable, consistent and stable when dealing with ontologies of different sizes. This allows us to confidently say that our proposed method is feasible and able to address large real-world ontologies.

Based on our findings, it is clear that some research paths emerge, including:

- Developing the method to be able to predict the scores of complex candidate axioms.
- Incorporating active learning (Settles, 2009) for scalability before reaching the stage of applying feature selection.

REFERENCES

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

Ballout, A., Tettamanzi, A. G. B., and Da Costa Pereira, C. (2022b). 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)*, pages 72–79.

Cai, J., Luo, J., Wang, S., and Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79.

Chandrasekaran, B., Josephson, J. R., and Benjamins, V. R. (1999). What aro ontologies, and why do we need them? *IEEE Intelligent Systems and Their Applications*, 14:20–26.

Chen, J., He, Y., Geng, Y., Jiménez-Ruiz, E., Dong, H., and Horrocks, I. (2022). Contextual semantic embeddings for ontology subsumption prediction.

Corby, O., Dieng-Kuntz, R., and Faron-Zucker, C. (2004). Querying the semantic web with corese search engine. In *ECAI*, pages 705–709. IOS Press.

Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., and Gandon, F. (2006). Searching the semantic web: Approximate query processing based on ontologies. *IEEE Intell. Syst.*, 21(1):20–27.

Cullen, J. and Bryman, A. (1988). The knowledge acquisition bottleneck: Time for reassessment? *Expert Systems*, 5:216–225.

Dubois, D. and Prade, H. (1988). *Possibility Theory—An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York.

Hassanpour, S., O'Connor, M. J., and Das, A. K. (2014). Clustering rule bases using ontology-based similarity measures. *Journal of Web Semantics*, 25:1–8.

Khadir, A. C., Aliane, H., and Guessoum, A. (2021). Ontology learning: Grand tour and challenges. *Computer Science Review*, 39:100339.

Lehmann, J. and Völker, J. (2014). An introduction to ontology learning.

Maedche, A. and Staab, S. (2004). *Ontology Learning*. Springer, Berlin, Heidelberg.

Malchiodi, D., Da, C., Pereira, C., and Tettamanzi, A. G. B. (2020). Classifying candidate axioms via dimensionality reduction techniques. pages 179–191.

Malchiodi, D. and Tettamanzi, A. G. (2018). Predicting the possibilistic score of OWL axioms through modified support vector clustering. *Proceedings of the ACM Symposium on Applied Computing*, pages 1984–1991.

Nickel, M., Tresp, V., and Kriegel, H. P. (2012). Factorizing yago : Scalable machine learning for linked data. *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web*, pages 271–280.

OWL Working Group (2012). OWL—web ontology language. https://www.w3.org/OWL/.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

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