CEO: Counterfactual Explanations for Ontologies

Matthieu Bellucci [a,*], Nicolas Delestre [a], Nicolas Malandain [a] and Cecilia Zanni-Merk [a]
[a] *Normandie Université, INSA Rouen Normandie, LITIS, MIND, Rouen, Normandie, France*
E-mail: matthieu.bellucci@insa-rouen.fr

Abstract. Debugging and repairing Web Ontology Language (OWL) ontologies has been a key field of research since OWL became a W3C recommendation. One way to understand errors and fix them is done through explanations. These explanations are usually extracted from the reasoner and displayed to the ontology authors as is. In the meantime, there has been a recent call in the eXplainable AI (XAI) field to use expert knowledge in the form of knowledge graphs and ontologies. In this paper, a parallel between explanations for machine learning and for ontologies is drawn. This link enables the adaptation of XAI methods to explain ontologies and their entailments. Counterfactual explanations have been identified as a good candidate to solve the explainability problem in machine learning. The CEO (Counterfactual Explanations for Ontologies) method is thus proposed to explain inconsistent ontologies using counterfactual explanations. A preliminary user-study is conducted to ensure that using XAI methods for ontologies is relevant and worth pursuing.

Keywords: Counterfactual explanations, Explainability, Ontology, Knowledge graph, Artificial Intelligence

1. Introduction

Explainability has recently become a critical factor to develop artificial intelligence (AI) algorithms because of the increasing use of AI in sensitive industry applications as well as our day-to-day life. The field of explainable AI (XAI) has started to address this desire for explainability but has heavily focused on explaining machine learning algorithms. In order to create new explainable systems, scholars are exploring the of use knowledge representations [1], as illustrated by the design of neurosymbolic AI which combine neural networks with symbolic approaches [2]. Moreover, F. Lecue [3] discusses how Semantic Web and Knowledge Graphs technologies could be applied to most AI areas to increase their level of explainability. He argues that every subdomain of the AI field could benefit from explanations, including the Knowledge Representation and Reasoning (KRR) domain. However, explainability methods for the KRR domain are not well studied.

In particular, the Web Ontology language (OWL) is a popular language for the design of ontologies [4]. It is widely used in medicine and the web, as demonstrated by ongoing projects such as the Gene Ontology [5] and DBPedia [6]. A large variety of tools have been developed for OWL ontologies: the Protégé editor [7], its numerous plugins and reasoners facilitate the edition of ontologies. Several APIs also exist that allow developers to use OWL ontologies inside their applications. Among these tools, some are dedicated to debugging and repairing OWL ontologies. These tools explain to the ontology designer why some entailments were made by the reasoner. Nevertheless, we identified a lack of variety in the types of explanations.

*Corresponding author. E-mail: matthieu.bellucci@insa-rouen.fr.

1.1. Explaining OWL ontologies

Explanations in OWL ontologies are necessary to help a designer or a user understand entailments, debug and repair an ontology [8]. Since OWL ontologies are based on description logics, it is possible to extract some explanations of these entailments by using a reasoner. Methods to generate explanations are divided into two types: black-box and glass-box methods [4]. According to [9], *glass-box* methods introduce significant modifications to description logic reasoners with the goal to use available internal information for a fast computation of diagnoses. *Black-box* methods use a reasoner as an oracle to check if some set of axioms is consistent.

The simplest type of explanations that can be extracted from the reasoner is logical proofs. They display each step of the reasoning process that resulted in a specific entailment. The main issues with such explanations is that they become difficult to understand when they get very large [10]. Justifications are another popular form of explanation [11]. They are also called MUPS for Minimal Unsatisfiability-Preserving sub-TBoxes. They consist in finding the smallest sets of axioms necessary for a given entailment to hold. However, as Alrabbaa et al. [12] mention, justifications can still be very large and thus suffer from the same issue as proofs.

In order to overcome these issues, interactive debugging tools have been proposed. OntoDebug [9] implements this idea. Its goal is to ask the user for additional knowledge that can reduce the length of proofs and justifications. However, Coetzer and Britz [13] have shown that the debugging approach of OntoDebug can lead to unintuitive results. They use defeasible description logics to prevent these unintuitive results. Despite all the efforts in the development of debugging tools, ontology authors still struggle to debug and repair their ontologies [14].

The explainability of OWL ontologies is confronted with the same issues as XAI, with a similar goal. Both seek to provide understandable explanations of decisions made by an algorithm. In the XAI field, such algorithms are machine learning algorithms whereas for OWL ontologies, they are reasoners. Interestingly, there is some shared terminology, e.g. glass-box and black-box, that also share the same notions. Finally, as F. Lecue [3] advocates, ontologies could benefit from the advances in XAI in the same manner as the XAI field benefits from the KRR domain. Indeed, in the majority of the reviewed literature on OWL explanations, the explanations are made only for domain experts and ontology authors. Providing explanations to laymen could help ontologies gain popularity and be used as trustworthy decision-support systems.

Finally, Bellucci et al. [15] proposed a system that combines a machine learning model with an ontology in order to gain explainability. In this system, the ontology decides whether the model's prediction is valid. They argue that counterfactual explanations may be especially adapted to ontologies. That is why in the following, counterfactual explanations for ontologies are further explored.

1.2. Counterfactual explanations

Counterfactual thinking is a well-known reasoning method for humans. In a psychology bulletin, Roese [16] defines it as "mental representations of alternatives to the past". It consists in altering some factual antecedent to an event and assessing the consequences of that alteration. In the following, the terms counterfactual explanations, counterfactuals and CF are used interchangeably.

Stepin et al. [17] compare counterfactual explanations to contrastive explanations. Contrastive explanations explain an event P by answering the question "Why did P happen rather than Q ?". Given multiple contrastive explanations, the explainee will have sufficient information to make abductive inferences about the event P. Counterfactuals are contrastive explanations about a fact or event that occurred. According to Stepin et al. [17], counterfactuals provide explanatory alternatives to how things would stand if a different decision had been made at some points.

Verma et al. [18] briefly compiled works on counterfactuals in fields like philosophy, psychology and social sciences. These articles tend to say that counterfactuals are an ideal method of explanation. Keane et al. [19] also surveyed the literature and drew the same conclusions. They collected evidences from the literature that counterfactuals are technically feasible, psychologically relevant and GDPR-compliant. Because of these findings, counterfactual explanations have recently been identified as a good candidate to solve the XAI problem. However, Keane et al. also pointed out important flaws in the current applications of counterfactuals for machine learning, in particular the lack of user-studies to validate the proposed methods.

1.3. Contributions

From our observations, the literature on counterfactual explanations has mostly focused on explaining machine learning models ([17, 19]). To the best of our knowledge, counterfactual explanations for ontologies have not been explored yet. We have shown that explainability of ontologies is closely related to the XAI field. However, XAI researchers are currently focused on how to use ontologies to improve the explainability of machine learning models. We believe that adapting XAI methods to ontologies could help both fields. That is why a method to generate counterfactual explanations based solely on an OWL ontology is introduced. The main contribution of this paper is the Counterfactual Explanations for Ontologies (CEO) method based on existing explainability methodologies for machine learning. A minor contribution necessary to the design of the CEO method is the definition of desirable properties of counterfactual explanations specific to OWL ontologies.

The rest of this paper is structured as follows: Section 2 introduces counterfactual generation methods developed for machine learning and defines equivalent concepts for OWL ontologies. Section 3 presents the CEO method. A user-study is presented in Section 4 to validate the method on a musical instrument ontology. To conclude this article, the contributions, results and future directions are discussed in Section 5.

2. Preliminary work

In order to generate counterfactual explanations for ontologies, similar methods for other domains are to be reviewed. However, the latest research on generating counterfactual explanations is heavily focused on machine learning. Therefore a review of counterfactual explanations for machine learning is conducted. Then, the problem of generating counterfactual explanations is defined for OWL ontologies, along with other concepts inspired by this review. Beforehand, the classic loan approval example is described, based on the one proposed by Verma et al. [18] which will be used in the remainder of this paper to facilitate comprehension.

2.1. An example

Suppose a bank's customer seeks a loan. The loan approval system uses a classifier, which studies the customer's file. This file contains information on the customer's identity and financial situation. The feature vector is $(Income, CreditScore, Education, Age)$. When the customer is denied the loan by this system, they may ask for some explanations about this decision: "Why was the loan denied ?" and "What can I do differently so that the loan will be approved in the future ?". The first question can be answered using current explainability methods. A probable answer to that first question might be "Your *Income* is too low". The second question clearly requires a counterfactual explanation: what are the smallest changes that the customer can do in order to change the outcome i.e. get the loan. A possible counterfactual may be: "Your *Income* should be of \$40K instead of \$30K". Another could be: "Your *Education* should be master's degree instead of a bachelor's and your *Income* should increase by \$4K." These formulations allow the customer to choose between different paths in order to get the loan. This also allows them to understand which variables in their file are the most important for the model.

2.2. Counterfactuals for machine learning

Wachter et al. [20] propose a machine learning oriented definition for counterfactuals: "Score p was returned because variables V had values $(v_1, v_2, \dots)$ associated with them. If V instead had values $(v'_1, v'_2, \dots)$, and all other variables had remained constant, score p' would have been returned". In the same article, they propose the first counterfactuals generation method for machine learning, according to [18, 19]. It relies on solving an optimization problem under specific constraints that are motivated by psychological studies. Wachter et al. describe the following desired properties of counterfactuals, that can later be translated into mathematical constraints:

- Counterfactuals should be in **close possible worlds**, or in other words, alter values as little as possible. Going back to the example, this means that the CF "Your *Income* should increase by \$3K" is preferred to any CF that recommends an *Income* increase greater than \$3K.
- **Diverse** counterfactuals should be provided. They argue that it is more informative to provide a diverse set of counterfactual explanations. It gives the user multiple different sets of actions to change the model's outcome. Regarding the example, it would be more informative to have counterfactuals that describe modifications for each variable. For instance, a diverse set of CF might be: "Increase *Income* by \$10K", "Increase *Education* to a master's degree", or "Increase $CreditScore$ by 10 points and *Income* by \$3K". The customer will be able to choose the set of modifications that seems the easiest to achieve.
- Counterfactuals should be **sparse**. Wachter et al. say that sparsity is highly desirable to create human-understandable counterfactuals. Sparsity of CF ensures that only a small amount of variables are changed while the others remain constant. For the loan example, this means that the following CF "Increase *Income* by \$10K" is preferable to "Increase *Income* by \$2K, increase *Age* by 2 years, increase $CreditScore$ by 2 points and increase *Education* to a master's degree". Indeed, it seems intuitive that the customer will have more difficulties to understand the reasons justifying the last set of actions than the former one.

Most methods that followed Wachter et al.'s work have adopted the same idea to generate counterfactuals, i.e. solve an optimization problem under specific constraints. A variety of constraints have been proposed to enhance the quality of counterfactuals. Verma et al. [18] reviewed the literature on counterfactual explanations for machine learning. They enumerate recurring desired properties, their related mathematical constraints and show how to include them in an optimization problem.

Let x be an input vector and f a machine learning model. The main goal of a counterfactual explanation is to find modifications on x in order to get the desired outcome $\widehat{y}$ with $\widehat{y} \neq f(x)$ i.e. find every counterfactual $\widehat{x}$ where $f(\widehat{x}) = \widehat{y}$. This can be formulated as an optimization problem (see Equation 1). Constraints are added to this problem by a simple addition operation.

$$arg\min_{\widehat{x}} \left(f(\widehat{x}) - \widehat{y}\right)^2 \tag{1}$$

2.2.1. Validity

Validity of a counterfactual corresponds to whether the solution $\widehat{x}$ returned by the optimization problem has the expected outcome ([18, 20]), i.e. a counterfactual $\widehat{x}$ is valid iff $f(\widehat{x}) = \widehat{y}$. In [21], Mothilal et al. consider a CF valid if $f(\widehat{x}) \neq f(x)$. The choice of validity definition is made based on the user's expectations. In this paper, the first definition is adopted.

2.2.2. Proximity

Proximity is a measure of the distance between the original input x and a CF $\widehat{x}$. This property is used in every CF method. Verma et al. [18] associate proximity with validity, which emphasizes the importance of this property.

$$arg\min_{\widehat{x}} \left(f(\widehat{x}) - \widehat{y}\right)^2 + d(x, \widehat{x}) \tag{2}$$

Equation 2 plugs a distance constraint into the main optimization problem to ensure the proximity property. The function d is a distance metric. Many different distances have been proposed for counterfactuals. The feature vector x may contain numerical and categorical features which need to be processed differently. Let $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_J$ be the feature space of the input x. To deal with these different feature types, the distance metric d aggregates the distances between each feature as such: $d(x, \widehat{x}) = agg(\delta)$ where $\delta = (\delta_1, \ldots, \delta_J)$ and agg is a function such that $agg : \mathcal{X} \rightarrow \mathbb{R}^+$ Each component δ_j represents the distance between x_j and $\widehat{x}_j$. Finally, each δ_j needs to be computed based on the type of x_j.

The MACE method [22] imposes that $\delta_j \in [0, 1]$. Among all the categorical features, the MACE method distinguishes a subset of features called ordinal that represents ordered categories. Equation 3 shows how each δ_j is

computed. As a reminder, $\mathbb{I}[x_j \neq \widehat{x}_j]$ returns 1 if $x_j \neq \widehat{x}_j$ and 0 otherwise.

$$\delta_j = \begin{cases} |x_j - \widehat{x}_j| / R_j & \text{if } x_j \text{ numerical or ordinal. } R_j \text{ is the range of the feature } x_j. \\ \mathbb{I}[x_j \neq \widehat{x}_j] & \text{if } x_j \text{ is categorical.} \end{cases} \tag{3}$$

Once δ is computed, MACE uses the distance function defined by Equation 4.

$$d(x, \widehat{x}) = \alpha \|\delta\|_0 + \beta \|\delta\|_1 + \gamma \|\delta\|_\infty \tag{4}$$

The authors of MACE use these different norms to ensure desirable properties. The 0-norm restricts the number of features that change, therefore ensures sparsity of the CF. To ensure proximity, they use the 1-norm to restrict the average change distance and the ∞-norm to restrict the maximum change. This proximity metric is also used by the GeCo method [23]. Slightly different proximity metrics are discussed in [21, 23].

2.2.3. Sparsity

Sparsity corresponds to the number of features that have been modified to create the counterfactual from the original input. In several papers, researchers advocate for short explanations, i.e. sparse CF ([18, 20, 21]). However, Keane et al. [19] argue that this condition is based on intuition and some loosely related psychological studies on working memory limitations. An appropriate level of sparsity is yet to be defined. This is apparent in a user-study by Förster et al. [24], which demonstrated that users sometimes prefer longer explanations rather than shorter ones. They explain this finding with the fact that too short explanations may fail to incorporate the most important aspects and leave too many causes unexplained. They also mention that the ideal length depends on the context and the user. Humans generally prefer short explanations. However, in some cases such as scientific explanations, longer explanations are preferred.

Verma et al. [18] add a penalty function to encourage sparsity in the optimization problem, as shown in Equation 5.

$$arg \min_{\widehat{x}} (f(\widehat{x}) - \widehat{y})^2 + d(x, \widehat{x}) + g(\widehat{x} - x) \tag{5}$$

In this equation, g is the penalty function that encourages sparsity. The 0-norm or 1-norm is traditionally used ([18, 20]). Many scholars integrate the sparsity constraint directly into the proximity metric, as MACE [22] demonstrates. Indeed, in Equation 4, the 0-norm is already included. Adding another constraint with the same effect may be redundant.

The DiCe method [21] does not include sparsity in the optimization problem because their formulation of this property is not convex. Instead, they encourage sparsity in the generated counterfactuals by conducting a post-hoc operation where they restore the value of continuous numerical features back to their original values greedily, until the predicted class changes.

2.2.4. Feasibility

Feasibility and plausibility are properties that appear in many papers. Schleich et al. [23] describe feasibility as a measure of whether a user could realistically achieve the changes made in the counterfactual. For instance, the *Age* variable can only increase in a feasible CF, since it is not possible to become younger. They define plausibility as a measure of whether the counterfactual makes sense in the real world. For instance, recommending an *Age* of 200 years old is not plausible.

Keane et al. [19] criticize these notions as they are not clearly defined in the literature. The main goal of these notions is to make sure that counterfactuals are realistic and fair. There are many existing methods to ensure plausibility and feasibility, the following appear the most in current works:

- Some scholars argue that a counterfactual is feasible and/or plausible if it follows the training data distribution ([18, 22]) or stays within the range of a given feature ([25]). The idea is that a counterfactual would be unrealistic if it resulted in a combination that is far from the training data distribution. Looking back to the example presented in Section 2.1, a customer of 18 years old with a high school diploma is denied a loan. A

counterfactual requiring this customer to get a doctorate's degree and be 20 years old is unrealistic. Firstly, because it is not feasible for this particular customer since getting a doctorate's degree would necessitate more than two years. Secondly, it is highly improbable in general to have a doctorate's degree at this age. Therefore, such datapoint is not in the training data distribution and the CF is neither plausible nor feasible.

- The notion of actionability of a feature is discussed in the literature ([17, 21, 22]). Non actionable features are features that should not be modified for various reasons. An example of non actionable feature is the birthplace of a person. The birthplace cannot be modified. Moreover, if a CF modifies it, this shows that the model is discriminatory. Hence, a feasible CF is a CF that does not modify any non actionable feature.
- User-defined constraints are implemented in the latest methods ([21–23, 26]). The user or a domain expert can determine specific constraints to generate plausible and feasible explanations. The main issue is that those constraints need to be enunciated in a way that can be utilized by a given method. GeCo [23] defines a plausibility and feasibility constraint language (PLAF) to enable experts to easily insert those constraints in the optimization problem. DiCe [21] allows the user to define ranges for specific features. They also propose to incorporate domain knowledge in the form of pairs of features and their relation. For example, it is possible to say that when *Education* increases in a CF, then the *Age* should also increase. These constraints must be respected for a CF to be feasible.

We argue, that feasibility and plausibility are not always desirable. Indeed, avoiding these properties may reveal unexpected yet pertinent information about the model. Especially concerning fairness. Mothilal et al. [21] show counterfactuals on the COMPAS dataset that have not been filtered to be feasible or plausible. It reveals that changing only the race of the individual changes the outcome, which in turn lets the users understand that the model is not fair.

2.2.5. *Diversity*

Diversity refers to the distance between two counterfactuals. Verma et al. [18] highlight that most proposed algorithms return a single counterfactual for a given input. This implies that the optimization problem needs to be solved multiple times to get multiple counterfactuals. Moreover, providing multiple different counterfactuals is beneficial for the user since it helps further understand what the model observed as well as provides multiple paths to modify the outcome ([20, 21, 23]). To do so, the DiCe method [21] adds a diversity term to maximize in the optimization problem and generates a set of counterfactuals instead of a single one. Another way to generate multiple diverse counterfactuals is proposed in the GeCo [23] method, which uses a genetic algorithm to solve the optimization problem, that outputs a set of good counterfactuals. The most diverse counterfactuals are then selected, based on a particular diversity metric. Diversity being the distance between counterfactuals, it is analogous to the proximity metric. Any proximity measure can be utilized by replacing the original input with a counterfactual.

2.2.6. *Evaluation*

Finally, these methods must be evaluated on real datasets to assess their quality. The most used method of evaluation uses objective metrics as proxies of quality of a counterfactual. Förster et al. [24] deplore the lack of user studies in XAI. Keane et al. [19] declare that only 31% of the papers on CF they reviewed contained user evaluations. Stepin et al. [17] make the same observation, only 3 papers out of 31 propose a user study to evaluate their method, while 21 of them use an objective method. Unfortunately, conducting valuable user studies is complex and costly, which is why most papers use objective metrics as proxies of quality of CF. These metrics are usually validity, proximity, sparsity, feasibility and diversity ([18, 22, 23]. Authors set a particular proximity metric to evaluate their method against others, which may advantage their own method but allows for a normalized comparison. However, some methods cannot use different proximity metrics. When trying to compare their GeCo method, Schleich et al. [23] could not use their own proximity metric because the MACE and DiCE methods do not support combinations of L_p-norms. The other evaluation metrics can be computed post hoc and do not pose this issue.

2.3. *Building counterfactuals for ontologies*

In this paper, we set to create counterfactual explanations for OWL ontologies based the current works in machine learning. The first task is to map the definitions of desired properties seen in Section 2.2 to the OWL ontologies

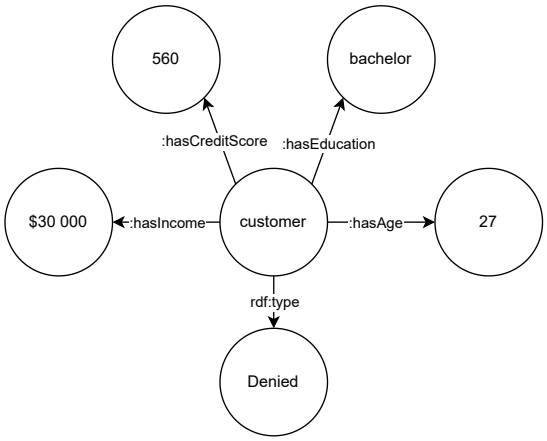

Fig. 1. An IKG representing a customer classified as `Denied` with $Income = \$30K$, $CreditScore = 560$, $Education =$ bachelor and $Age = 27$

domain. This mapping paves the way to our method described in Section 3. We use the OWL2 language as defined in [27] and its mapping to RDF Graphs given in [28]. For ease of reading, an ontology's class defined in its T-Box is written as `Class`. Any term defined in the OWL2 specifications [27], the RDF specification [29] or the mapping from OWL2 to RDF [28] is written in *italics*.

First, the problem is reformulated, then the desired properties of a counterfactual, i.e. validity, proximity, sparsity, feasibility and diversity, are defined for OWL ontologies.

2.3.1. Problem formulation

The CF problem in machine learning is the following. Given a model f and an input vector x, a counterfactual of x is a vector $\widehat{x}$ such as $f(x) \neq f(\widehat{x})$. Specifically, only the supervised classification problem is studied. Let C_o be the original class predicted by $f(x)$ and C_d be the desired modified outcome. In the example described Section 2.1, C_o corresponds to the class `Denied`, the desired outcome C_d is the class `Approved`. The classifier f becomes an OWL ontology $\mathcal{O}$. Indeed, both the classifier's trained parameters and the ontology contain knowledge under different forms.

Any input vector can be seen as an individual defined in the ontology's A-Box. Each feature of this input vector corresponds to an assertion. Therefore, the input vector actually corresponds to the set of assertions that share the same *sourceIndividual*. With the RDF Graphs representation, each assertion is a triple of the form $(subject - predicate - object)$ [29]. Since every assertion of this set shares the same *sourceIndividual*, these assertions can be rewritten as RDF triples where their *Subject* is the same. Finally, this enables the representation of this set of assertions as a star graph, where nodes are the objects of these assertions and edges are their *predicates*. The center node is the shared individual. This RDF graph corresponds to the input vector x. In the remainder of this paper, this type of graph is named an Individual Knowledge Graph or IKG . Let I be an IKG, the individual at the center of this IKG is noted I_c. Assertion, RDF triple or triple will also be used interchangeably from now on. Let I be an IKG containing the triple $(I_c - \text{rdf:type} - C_o)$. The IKG $\widehat{I}$ is a counterfactual of the IKG I if it contains the triple $\left(\widehat{I}_c - \text{rdf:type} - C_d\right)$ and not the triple $\left(\widehat{I}_c - \text{rdf:type} - C_o\right)$.

A graphical representation of an IKG is proposed in Figure 1. The *subject* of this IKG is the customer of the bank. Every assertion in the ontology that has this particular customer as *subject* is represented in the IKG, in the form of RDF triples.

2.3.2. Properties definition

Now that the definition of a counterfactual for an OWL ontology has been given, the different properties seen in Section 2.2 can be applied. In this section, I is the original IKG of class C_o, that is to say I contains the triple $(I_c - \text{rdf:type} - C_o)$. It is assumed that if an IKG contains the triple $(I_c - \text{rdf:type} - C)$, then it also contains every triple where the object is a parent of C. Equivalently, if an IKG does not contain the triple $(I_c - \text{rdf:type} - C)$, then it does not contain any triple where the object is a subclass of C.

Validity Based on the definition for machine learning, a counterfactual is valid if the outcome is the one expected. Therefore, we consider a counterfactual $\widehat{I}$ valid if:

- It contains the triple $\left(\widehat{I}_c - \text{rdf:type} - C_d\right)$.
- It does not contain the triple $\left(\widehat{I}_c - \text{rdf:type} - C_o\right)$.
- The ontology is consistent after adding the IKG $\widehat{I}$ to its A-Box.

Proximity Proximity is a measure of the distance between the original input and its counterfactual. In the literature, the problem of measuring the distance between two semantic entities has already been explored. Similarity functions are preferred to distances because they do not need to verify the triangle inequality. Similarities based on description logics are discussed in [30]. Euzenat et al. [31] present the OLA similarity which first encodes an ontology as a labelled graph; the similarity between two nodes of this graph depends on the similarity of the terms (labels, names...), the similarity of the neighbours and the similarity of other local descriptive features. Hu et al. [32] introduce the notion of signature vectors. The idea is to decompose a concept C into a set of primitive concepts and attribute a weight to each of these primitive concepts based on their number of occurrences in C. This concept C can now be represented as a vector where each feature is a primitive concept and the value of the feature is the importance of this primitive concept for C. Computing the similarity between two concepts is equivalent to computing the similarity between their signature vectors. The principle challenge of this method is to define those primitive concepts. For each ontology, primitive concepts must be defined which can be an arduous task for large ontologies. Finally, S. Ontañón [30] discusses the "edge counting" distance, first introduced by Rada et al. [33]. This distance is applicable to taxonomies and hierarchies. The taxonomy is seen as a tree where the parent relation defines the edge between the elements in the tree. The distance is simply the number of edges that must be traversed to go from one element of the tree to another. Overall, there exists a large choice of semantic similarity metrics, each adapted to specific use-cases.

Sparsity Sparsity is the amount of features modified on the original input to get the counterfactual. For IKGs, the definition remains the same, it is the number of assertions of the original IKG that have been modified to get the CF.

Feasibility Feasibility or plausibility is the measure of how realistic and achievable a counterfactual is. Because ontologies use domain knowledge, an unrealistic counterfactual should not be consistent with the ontology. Thus the consistency of the ontology reflects the plausibility of the counterfactual. Nevertheless, the issue of actionability persists. A CF is not feasible if it modifies non actionable features. That is why we propose to flag some predicates as non actionable in the ontology. For instance, a triple $(I_c - \text{:hasRace} - \textit{Caucasian})$ should not be modified since race is a protected feature. Any triple with this predicate is not actionable, therefore, the predicate definition should reflect this non-actionability. One way to do so that is used in the CEO method is to consider non actionable predicates as subclasses of an abstract class `NonActionable`

Diversity Diversity being the distance between two counterfactuals, the similarity function used for proximity can be applied to two counterfactuals to compute the diversity.

3. Counterfactual Explanations for OWL ontologies

The CEO method takes an IKG I as input and seeks a set of valid counterfactuals, as defined in Section 2.3. In the current state of our method, only *ClassAssertions* and *ObjectPropertyAssertions* are observed. A new set of *ClassAssertions* is decided by the user. The set of *ClassAssertions* of I is then replaced by the user-defined set. The new IKG resulting from this replacement is called $\widehat{I}$ and is considered the closest counterfactual from I. However, it is likely that these changes render the ontology inconsistent. The method seeks appropriate modifications of the *ObjectPropertyAssertions* in $\widehat{I}$ to make the ontology consistent with the changes in *ClassAssertion*. The CEO method is a black-box method according to the definition given for OWL explainability methods in Section 1.1. Indeed, it uses the reasoner as an oracle to check whether the addition of an IKG renders the ontology inconsistent.

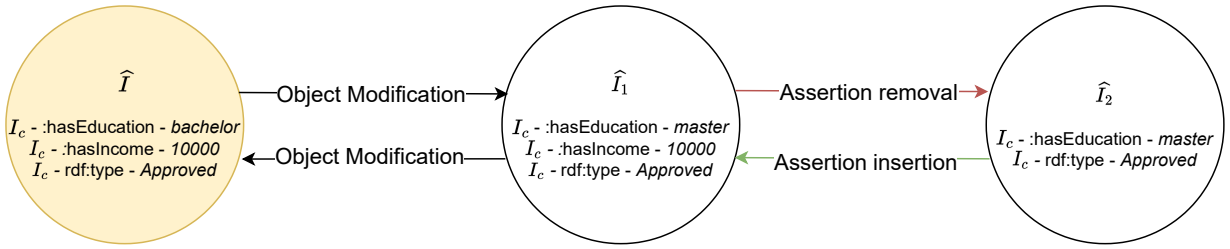

Fig. 2. Representation of a subgraph of Ω

A major difference between supervised classification and OWL ontologies reasoning is that supervised classification requires an input with a fixed set of features. An OWL ontology reasoner does not impose a specific set of assertions to make inferences. To do so, reasoners use the open world assumption which considers that non specified assertions are unknown. Thus, the open world assumption should be taken into account when generating counterfactuals for OWL ontologies. Removing or adding assertions may provide additional information to the user and should be included in the method. Different sets of modifications, insertions and removals are found by the method, sorted based on a set of metrics and proposed to the user as counterfactual explanations. Specifically, the CEO method is decomposed into four parts:

1. Generate a set of candidate counterfactuals.
2. Remove non valid and non feasible candidates.
3. Compute proximity and sparsity on the set of valid candidates.
4. Display the valid candidates to the user, sorted by proximity and sparsity.

The implementation of each step is further discussed in this section. First, a simplification is made about the *subject* of every triple. In the definition of an IKG in Section 2.3, the *subject* of any triple in an IKG I is an individual noted I_c. It can be argued that for any counterfactual $\widehat{I}$ of I, $I_c = \widehat{I}_c$. Indeed, this individual is the only constant thing between the original IKG and its counterfactual. For instance, let I_c be the bank's customer from the example in Section 2.1. The counterfactual provides possible modifications of assertions to obtain the loan. The customer may be told to modify their *Income* or their *Education* but it does not make sense to modify the nature of the customer, i.e. the *subject* of these assertions. This is why, from now on, the individual *subject* of an IKG is considered constant and is noted I_c in every case.

3.1. Counterfactuals search space

Some counterfactual methods propose to define the search space for counterfactuals ([22, 23]) and then compute the different metrics for the counterfactuals in this search space. In the case of OWL ontologies, this space can be represented as a directed graph.

Let $\Omega = (V, E)$ the graph representing the search space. V is a set of IKGs that are the nodes of this graph. E is the set of directed edges that link one IKG to another. Three elementary operations are defined on an IKG:

Object modification This operation consists in modifying the *ClassAssertions* of an object in a triple.
Assertion removal This operation consists in removing an assertion from an IKG.
Assertion insertion This operation consists in adding an assertion to an IKG.

Two IKGs are connected in the graph if and only if a single elementary operation is needed to go from one IKG to the other. For instance, let $\widehat{I} = \{(I_c - \text{:hasEducation} - \textit{bachelor}), (I_c - \text{:hasIncome} - 10000)\}$ and $\widehat{I_1} = \{(I_c - \text{:hasEducation} - \textit{master}), (I_c - \text{:hasIncome} - 10000)\}$. This graph is represented in Figure 2. $\widehat{I}$ and $\widehat{I_1}$ are connected in Ω because a single object modification needs to be applied to go from $\widehat{I}$ to $\widehat{I_1}$, namely changing *bachelor* to *master*. Let $\widehat{I_2} = \{(I_c - \text{:hasEducation} - \textit{master})\}$. In this case, $\widehat{I}$ and $\widehat{I_2}$ are not connected because more than one elementary operation should be applied. But $\widehat{I_1}$ and $\widehat{I_2}$ are connected because only an assertion removal connects them, namely removing the *:hasIncome* assertion. Each elementary operation has an inverse operation as can be

seen in Figure 2. The inverse operation of object modification is itself. The inverse of assertion removal is assertion insertion and vice versa.

3.2. Exploring the search space

The graph Ω contains all possible IKGs, making it complex to compute. Moreover, only a fraction of the IKGs in this graph are valid. Therefore, the graph can be generated in an optimized manner, by avoiding non valid IKGs. As a reminder, a counterfactual of the IKG I is an IKG that contains the triple $(I_c - \text{rdf:type} - C_d)$ and not the triple $(I_c - \text{rdf:type} - C_o)$ where C_o is the class of the original IKG and C_d is the desired class of the counterfactuals. Let $CA(I)$ be the set of *ClassAssertions* of the IKG I, $OA(I)$ the set of *ObjectPropertyAssertions* of I and $DA(I)$ the set of *DataPropertyAssertions* of I. The simplest counterfactual of an IKG I is $\widehat{I}$ and is defined in Equation 6. It is unlikely that this counterfactual is consistent therefore valid, but it is the closest to the original IKG I. Thus, this IKG is an ideal starting point to explore the graph Ω.

$$\begin{cases} CA(\widehat{I}) = CA(I) \setminus \{(I_c - \text{rdf:type} - C_o)\} \cup \{(I_c - \text{rdf:type} - C_d)\} \\ OA(\widehat{I}) = OA(I) \\ DA(\widehat{I}) = DA(I) \end{cases} \tag{6}$$

Algorithm 1 is proposed to generate Ω. The graph is created with $\widehat{I}$ as the starting node. Nodes that are connected with $\widehat{I}$ by assertion removal operation are explored and added to Ω until consistent ones are found, with the *findRemovalNeighbors* function. Once these consistent nodes are known, the *generateAncestors* and *generateDescendants* functions are invoked on every explored node. These functions generate new nodes and edges by inserting back the removed assertions and modifying the class of their objects. The class of the objects must remain within the range of the predicate of the assertions.

This algorithm guarantees to obtain at least a consistent CF. In the worst case, this CF is an IKG with no assertion. Intuitively, the algorithm searches for faulty assertions that provoke inconsistencies, by removing them. When a consistent IKG is found, this means that every faulty assertions have been removed. These faulty assertions are added back one by one with modification on their objects to fix the inconsistency.

However, the number of candidate counterfactuals explored by Algorithm 1 in the worst case is exponential with the number of assertions in the original IKG. Therefore, this algorithm does not scale up well with large ontologies. Moreover, a reasoner is called for every node generated. When the number of entities in the ontology increases, the reasoning time and the number of candidate counterfactuals to explore also increase.

3.3. Computing metrics in the graph

This graph representation enables the use of a graph edit distance to measure proximity.

$$GED\,(I_1, I_2) = \min_{(e_1,\ldots,e_k)\in\mathcal{P}(I_1,I_2)} \sum_{i=1}^{k} c\,(e_i) \tag{7}$$

The graph edit distance written as $GED(I_1, I_2)$ is defined in Equation 7, where $\mathcal{P}\,(I_1, I_2)$ is the set of edit paths transforming I_1 to I_2 and $c(e)$ is the cost of each elementary operation. The cost for each elementary operation must be defined. Let $c_m(e)$ be the cost for a class modification operation, $c_r(e)$ for assertion removal operation and $c_i(e)$ for assertion insertion. We argue that for a given assertion, the cost of removing this assertion should be greater than modifying it. Likewise, inserting this assertion should cost more than removing the given assertion. This choice is justified by the feasibility and comprehensibility of an operation. The original IKG directly represents the real world therefore it is feasible.

Modifying the original assertions lowers the feasibility as it strays further away from the real world example. This decrease is proportional to the similarity of the modified assertion with the original one. It represents

Algorithm 1 Algorithm to generate a relevant sub-graph of Ω

function GENERATESUBGRAPH($\widehat{I}$)
  $\Omega \leftarrow createEmptyGraph()$ ▷ $\Omega = (V, E)$
  $addNode(\Omega, \widehat{I})$
  $\Omega \leftarrow \Omega \cup findRemovalNeighbors(\widehat{I})$
  for $I_i \in V$ **do** ▷ For all IKG in Ω
    $\Omega \leftarrow compose\,(\Omega, generateAncestors(I_i, \Omega))$ ▷ *compose* combines the nodes and edges of two graphs into a single graph.
  end for
  for $I_i \in V$ **do** ▷ For all IKG in Ω
    $\Omega \leftarrow compose\,(\Omega, generateDescendants(I_i, \Omega))$
  end for
  return Ω
end function
function FINDREMOVALNEIGHBORS(I)
  $neighbors_V \leftarrow \{I\}$
  $neighbors_E \leftarrow \{\}$
  $neighbors \leftarrow (neighbors_V, neighbors_E)$ ▷ *neighbors* is a graph.
  if $isConsistent(I)$ **then** ▷ If the IKG I is consistent with the studied ontology, stop search for neighbors.
    return *neighbors*
  end if
  for $A_i \in I$ **do** ▷ For all assertions in the IKG I
    $I' \leftarrow assertionRemoval(A_i, I)$ ▷ Remove the assertion A_i from I
    $neighbors_V \leftarrow neighbors_V \cup \{I'\}$
    $neighbors_E \leftarrow neighbors_E \cup \{(I, I', assertionRemoval)\}$ ▷ I and I' are connected with an assertion removal operation
    $neighbors \leftarrow compose\,(neighbors, findRemovalNeighbors(I'))$
  end for
  return *neighbors*
end function

the effort that the user has to make to go from their starting point to the new class. Removing an assertion removes any information about the changes to achieve, the user loses information about what and how they should change concerning the assertion. Finally, inserting an assertion requires the user to achieve something that was absent from their starting point, which may lead to unfeasible or incomprehensible requests. For instance, the assertion $(customer - \text{:hasEducation} - bachelor)$ is studied. Modifying the class of the object means that the customer should change its education level, the greater the change is, the less feasible it is. Removing this assertion removes information about the change they must achieve. They should perhaps lose their degree, which is not feasible, or change their degree in an unknown way. Removing an assertion leads to incomprehensible but not necessarily unfeasible changes. Finally, inserting an education level is difficult to comprehend. The education level was missing from the original input, adding it may not make sense depending on the reason of its original absence. Moreover, achieving this change might not be feasible based on the customer status.

Arguably, adding an insertion should be costlier than removing an assertion, which should be costlier than modifying an assertion. The following costs are proposed:

Class modification The cost is the similarity measure between the original class and the modified class.

Assertion removal The cost is greater than the cost of any class modification on this assertion.

Assertion insertion The cost is greater than the cost of removing this assertion.

Equation 8 is the proposed cost function, where *sim* is a similarity measure for two assertions and $\mathcal{A}$ is the space of all possible assertions.

$$c(e) = \begin{cases} sim(a_1, a_2) & \text{if } e \text{ is a class modification operation from assertions } a_1 \text{ to } a_2 \\ \max_{a \in \mathcal{A}} sim(a_1, a) + 1 & \text{if } e \text{ is an assertion removal operation where } a_1 \text{ is the studied assertion.} \\ \max_{a \in \mathcal{A}} sim(a_1, a) + 2 & \text{if } e \text{ is an assertion insertion operation where } a_1 \text{ is the studied assertion.} \end{cases} \tag{8}$$

Proximity is the similarity between any counterfactual and the original input I. Since the GED only allows to compute similarity between nodes in the graph, the original input is replaced by $\widehat{I}$ which is considered the closest counterfactual from I. Thus, proximity of a counterfactual $\widehat{I}_i$ is $GED(\widehat{I}, \widehat{I}_i)$, with the cost function defined in Equation 8.

It is also possible to compute sparsity in the same manner, with a different cost function. Indeed, an edge of the graph is an elementary operation on one assertion. Therefore, the shortest path between two IKGs contains one edge per modified assertion. The length of this path is equal to the number of modified assertions. Thus, the GED with $\forall e, c(e) = 1$ as cost function is a measure of sparsity. Finally, a similarity measure should be defined to compare two objects of an assertion.

3.4. Assertions similarity

In Section 2.3.1, an IKG is defined as a set of assertions or sharing the same *sourceIndividual*. The proximity metric for machine learning computes the distance between each feature and then aggregates these distances to get the proximity. The same idea can be applied to an IKG, by comparing assertions together. This raises the issue of comparing two assertions. The methodology of proximity for machine learning could be applied, i.e. decompose the triple, measure the distance between each element of the triple and aggregate these distances. However, in the case of counterfactuals, we will show that the distance between triples can be simplified into the distance between the objects.

Problem simplification The predicate of an RDF triple represents the nature of the relation between the subject and the object. Can a distance between two predicates be defined and is it sensible to compare two triples of different predicates ? For instance, what is the distance between the predicates *rdf:type* and *:hasAge* ? In machine learning, only the same features were compared. Hence, only similarity between triples with the same predicate is defined.

In the introduction of Section 3, it was argued that the *sourceIndividual* remains the same for counterfactuals. Likewise, only triples with the same predicate should be compared. Therefore, measuring the similarity between two triples can be simplified into measuring the similarity between the respective objects of these triples. The subject of any triple in an IKG is an individual. According to the mapping of OWL2 to RDF [28], assertions that have an individual as subject can be of the following types: *SameIndividual*, *DifferentIndividual*, *ClassAssertion*, *ObjectPropertyAssertion* and *DataPropertyAssertion*. Therefore, the possible types of objects are classes, individuals and literals. The *SameIndividual* and *DifferentIndividual* assertions are not of interest for counterfactuals and are therefore removed from IKGs. *DataPropertyAssertions* have literals as objects. Similarity measures between literals such as strings or numbers already exist. *ObjectPropertyAssertions* require a similarity between two individuals. However, an *ObjectProperty* is defined with a *Domain* and a *Range* which are classes. The individual must belong to this *Range*, thus it must have a *ClassAssertion*. That is why the same similarity will be used for classes and individuals.

Edge-counting similarity The edge-counting similarity measure proposed by Rada et al. [33] is a good choice for computing similarities between classes because of its simplicity to understand and visualize. This similarity measure is based on hierarchical is-a relations. In OWL ontologies, such hierarchy is obtained with the *SubClassOf* relation. A hierarchy tree is created where the edges are the *SubClassOf* relation and the nodes are classes. The similarity between two classes is the length of the shortest path from one class to the other in this tree.

Concerning individuals, three cases are possible:

- If the individual is not subject of any *ClassAssertion*, then the class used is *owl:Thing* which is the root of the tree.
- If the individual is subject of a single *ClassAssertion*, this class is used to measure the similarity with another individual.
- If the individual is subject of multiple *ClassAssertions*, then the similarity between each class is calculated and the smallest one is kept.

This particular similarity measure is used to compute the weight of the edges associated to class modification operations for the graph edit distance described in Section 3.3.

In conclusion, the CEO method takes an IKG as input and a set of desired classes given by the user. A subgraph of all possible counterfactuals is generated in a way that guarantees at least one valid counterfactual. Invalid and unfeasible counterfactuals are removed from the set of candidate counterfactuals. Specifically, non actionable assertions are assertions that have an *ObjectProperty* which is a *SubClassOf* an abstract property named `NonActionable`. If an assertion with this predicate is modified, the CF is considered unfeasible. Then, the proximity and sparsity is computed for every valid and feasible counterfactual. Finally, the counterfactuals are sorted by proximity and sparsity and displayed to the user. Since proximity and sparsity have the same order of magnitude, they are added together to get the ranking of counterfactuals.

4. Experiments

In order to validate the CEO method and identify its weaknesses, a small scale user study was conducted. The goal of this study is to verify the relevance and comprehensibility of the explanations. It uses the experimental context described in [15]. Code, ontology and evaluation data of this experiment are available at this link: https://github.com/matt-bellucci/CEO. An ontology is created, based on a simple hierarchy of musical instruments families. There are 17 classes that each represent a musical instrument. 5 object properties are used in the class definition of these instruments: the texture, the mechanism, the type of mouthpiece for wind instruments, the shape and the presence of visible strings. Several convolutional neural networks with the ResNet50 architecture, pretrained on the ImageNet dataset have been finetuned on 4000 pictures of musical instruments. One of those models detects the type of instrument while the others detect each object property defined in the ontology of musical instruments. The result of these models are added to the ontology as an IKG. If the ontology is inconsistent after these additions, that means that the instrument detected is not consistent with the properties detected. The study follows up on this idea and proposes counterfactual explanations to render the prediction consistent. For instance, a harpsichord is detected as well as pedals and a wooden texture. This is not consistent with the ontology because a harpsichord does not have pedals. The goal of the CEO method is to propose modifications for these inconsistent assertions so that the user understands which properties were wrong and how to change them.

At the beginning of the survey, the users are first presented a description of this system, what its goal is and how it works. Then, an image is presented with the output of the machine learning models. Figure 3 shows the description of the output of the models for the first case of the survey and the explanations, with Figure 4 as the input image. It is explained that the results are not consistent with expert knowledge and finally a question is formulated: “What changes on the properties detected should be made so that the ontology is consistent ?”. A set of valid and feasible counterfactuals is generated and sorted by proximity and sparsity. The first ten counterfactuals are proposed to the user, hiding the rest of the generated set. A counterfactual is presented as a text telling the user which changes to make in no particular order. An example of a counterfactual given to a user is the following: “Replace brass metal texture with wood AND remove pedals mechanism”.

Then, the user answers two questions:

1. Which explanation did the user prefer ?
2. Which explanations seemed relevant to the user ?

These questions enable the evaluation of the quality of explanations and the quality of the proximity metric. Ideally, the preferred explanations are within the first explanations because they are sorted by proximity and sparsity. The

What the AI algorithm detected

The AI algorithm determined that the instrument in the image is a **Harpsichord**.
It detected the following properties:

- Wood texture
- Brass texture
- A keyboard mechanism
- Pedals mechanism

An inconsistency is detected, these properties do not match a harpsichord.
The system proposes different explanations that answer the question "What changes on the properties detected should be made to make them consistent with a harpsichord ?"

1. **Replace brass texture with another texture** AND **replace pedals mechanism with another mechanism**.
2. **Replace brass texture with wood** AND replace pedals mechanism with another mechanism.
3. **Replace brass texture with another texture** AND **replace pedals mechanism with another string instrument mechanism**.
4. Replace brass texture with another texture AND **replace pedals mechanism with a keyboard**.

Fig. 3. Description and the first four explanations of the first case of the survey

Fig. 4. Input image of the first case of the survey.

Table 1
Results of the user study for each case presented to six domain experts.

Case	Preferred explanations (% who chose it)	% of relevant explanations	Average AUC ROC
1	2nd (16.7%), 3rd (16.7%), **7th** (50%), 8th (16.7%)	70 %	0.77
2	5th (16.7%), **10th** (83.3%)	90%	0.55
3	1st (33.3%), **3rd** (50%), None (16.7%)	22%	0.93
4	6th (33.3%), 7th (16.7 %), 8th (33.3%), 10th (16.7%)	80%	0.55
5	**2nd** (83.3%), None (16.7 %)	20%	0.96
6	3rd (16.7%), 8th (16.7%), **9th** (50%), None (16.7%)	50%	0.76
7	1st (16.7%), **3rd** (50%), 9th (16.7%), None (16.7%)	40%	0.87
8	7th (33.3%), **8th** (66.7%)	80%	0.65
9	1st (16.7%), **3rd** (66.7%), 9th (16.7%)	20%	0.78

last question permits to plot the ROC curve (Receiver Operating Characteristics) which represents the amount of relevant explanations on the y-axis and the amount of non relevant explanations on the x-axis. The Area Under the Curve (AUC) for the ROC curve is used in information retrieval to measure the capacity of a system to rank the relevance of documents [34]. In this study, the AUC ROC is a measure of the quality of the ranking of explanations, therefore is a proxy measure of the quality of the proximity and sparsity metrics.

The user-study was conducted on a sample of six domain experts, i.e. experienced musicians. Nine different examples were presented. To compensate for this small sample size, a discussion took place with each expert after the survey in order to get their feedback, which will help in the analysis of the results.

4.1. Results

Results are combined in Tables 1 and 2. Table 1 shows the rank of the preferred explanations and the percentage of experts that chose them, the percentage of relevant explanations and finally the average AUC ROC for each case. The AUC ROC is calculated based on the answer of each expert, then the average of these scores is displayed in the table. Table 2 shows key statistics for each rank of explanation. A consensus is reached when 50% or more agree on the preferred explanation.

The first observation from Table 1 is that the quality of the explanations is highly dependant on the case. Indeed, the amount of relevant explanations varies from 20% up to 90%. Cases 3, 5 and 9 are the only ones to include an assertion about the shape of the instrument. Experts have identified a flaw in the ontology concerning the shape property based on the generated explanations. The poor relevance of these cases can therefore be attributed to this

Table 2
Results of the user study for each rank of explanation

Explanation ranking	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Proportion of relevance	9/9 (100%)	7/9 (78%)	7/9 (78%)	4/9 (44%)	3/9 (33%)	4/9 (44%)	4/9 (44%)	4/9 (44%)	3/9 (33%)	2/9 (22%)
Number of consensus	0/9 (0%)	1/9 (11%)	3/9 (33%)	0/9 (0%)	0/9 (0%)	0/9 (0%)	1/9 (11%)	1/9 (11%)	1/9 (11%)	1/9 (11%)
Occurrences of preference	4	6	12	0	1	2	6	8	5	6

Table 3
Results of the user study for each rank of explanation after removing cases 3, 5 and 9

Explanation ranking	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Proportion of relevance	6/6 (100%)	6/6 (100%)	5/6 (83%)	4/6 (67%)	3/6 (50%)	4/6 (67%)	4/6 (67%)	4/6 (67%)	3/6 (50%)	2/6 (33%)
Number of consensus	0/6 (0%)	0/6 (0%)	1/6 (16.7%)	0/6 (0%)	0/6 (0%)	0/6 (0%)	1/6 (16.7%)	1/6 (16.7%)	1/6 (16.7%)	1/6 (16.7%)
Occurrences of preference	1	1	5	0	1	2	6	8	4	6

error. Table 3 is the same as Table 2 without cases 3, 5 and 9 that may bias the analysis of preferences since only the first three explanations were relevant.

The AUC ROC is always greater than 0.5 which indicates that the relevant explanations are not packed at the bottom of the ranking. Table 3 also shows that the proposed explanations are relevant in more than 50 % cases, except for the tenth explanation. However, the top explanations are not preferred by the experts. Because of the small sample size, the occurrences of preference for each rank is also displayed in Tables 2 and 3. This measure is the number of times it was preferred by any expert in every cases. It does not take into account when experts did not have a preferred explanations, which happened in cases 3, 5, 6 and 7. The last four explanations stand out as being the most preferred, by cumulating a total of 24 preferences over 34 votes.

Concerning the content of the explanations, the first explanations are the most abstract. They successfully identify the faulty assertions but give a generic change, such as “replace wood by another texture” or “replace the strings by another mechanism”. Preferred explanations are rarely in the top ranks because experts sought more specific changes that have the same level of abstraction as the original assertion. The other explanations are grouped by possible modifications on one assertion. For instance, in case 1, the 3rd to 6th explanations only explore modifications on the mechanism without modifying the other assertions. Finally, removal operations are usually positioned at the bottom of the ranking because of the way the cost of these operations is computed. However, as Table 3 demonstrates, experts prefer explanations at the bottom of the ranking.

Overall, the experts complained that the survey necessitated an important cognitive effort and struggled to interpret the explanations. They argued that the way of presenting the explanations was problematic and suggested an interactive interface rather than a simple text. Nevertheless, they unanimously found that counterfactual explanations are useful to understand a decision process. They also pointed out some issues in several explanations that come from flaws in the design of the ontology. Finally, they deplored the absence of assertion insertions that could have been highly relevant in some cases. For instance, wind instruments always have a mouthpiece. When a mouthpiece was not present in the original IKG, it was not present in the counterfactuals either. Experts expected the addition of a mouthpiece in these cases.

4.2. Analysis

The main point of this study was to assess whether the proposed explanations are relevant. Four cases had less than 50% relevant explanations and three among them were justified because of a faulty axiom in the ontology. Ontologies are designed with domain experts, therefore using the CEO method during the design phase can prove useful to detect errors. Indeed, the automatic exploration of candidate counterfactuals may create unexpected individuals that a human would not have tested. However, the current exploration algorithm is computationally costly since the number of counterfactuals generated scales exponentially with the number of assertions in the original IKG.

The valid candidates generated and presented to the users are mostly relevant and the AUC ROC scores show good results. This would indicate that the proximity and sparsity metrics functioned as desired. However, the preferred explanations are mostly in the bottom of the ranking. This is an issue because experts complained that there were too many explanations for each case. If 5 explanations were retained instead of 10, the best explanations would

have been missed. Likewise, the best explanations may not be present in the 10 explanations shown, since more than 10 valid candidates were generated for most cases. Therefore, the proximity metric does not accurately sort the explanations.

Two problems on the proximity metric have been identified. First, explanations with the lowest proximity are the most abstract which is not what experts expect. They prefer to be shown classes that have the same depth as the original class. The similarity metric that compares two assertions should be modified to walk the hierarchical tree breadth-first instead of depth-first. Secondly, explanations with assertion removals are ranked in the bottom because of the design of the cost function. Yet it was shown that the bottom explanations are the most preferred. In some cases the removal of an assertion was the expected explanation by the experts. The cost function for assertion removal should be modified to prevent relevant removals from being placed in the bottom explanations. These changes are relevant for this particular ontology. For other ontologies and applications, different similarities may be better suited. Finding a similarity suited for an application is a challenge that needs to be addressed to apply the CEO method.

Another issue of the current sorting system is that there are groups of counterfactuals that modify the same assertion. In Section 2, the notion of diversity was introduced. Adding the diversity in the sorting system may alleviate this problem. Still, diversity is analogous to proximity and is exposed to the same issue that is the choice of a similarity metric. Furthermore, there is a risk that a high quality explanation will not be displayed because it is too similar to a lower quality explanation that got a better ranking. Before adding a diversity constraint, a satisfying proximity must be found. That is why diversity was not included in the current version of CEO.

As was pointed out by the experts, the lack of assertion insertion is problematic. Some explanations could have greatly benefited from such operations. The generation of the subgraph Ω must be reworked to include assertion insertions in an efficient way. Likewise, some counterfactuals are not explored and thus the best explanations may be missing from the set of valid and feasible counterfactuals. The impact of this issue is hard to measure since it is not possible to know if the best explanations have been explored. A cause of confusion for the experts was the meaning of removing an assertion. For them, removing an assertion meant that it is absent from the instrument. But with the open-world assumption, removing an assertion means that it is unknown, not missing. A way to render removal assertions more intuitive is to add *NegativeObjectPropertyAssertions* to the IKG when a removal operation is done. Thus removing an assertion will have the same meaning for the user and the OWL ontology. Similarly, the CEO method does not handle *DataPropertyAssertions* which may be needed in some applications such as the loan approval example described in Section 2.1.

To address the complexity of understanding the counterfactual explanations, their presentation should be reworked. An interactive interface that allows the user to see and visualize the explanations in the graph may decrease their cognitive effort to understand the explanations. Representing the graph of possible counterfactuals may also solve the problem of diversity, by visually identifying clusters of nodes that all modify the same assertion in different ways. This interface may also allow the user to modify key elements of the CEO method such as the similarity metric. A new user study should be conducted to verify this hypothesis.

5. Conclusion

In this paper, counterfactual explanations for ontologies were studied, inspired from the literature on counterfactuals for machine learning. As a result, the Counterfactual Explanations for Ontologies (CEO) method was presented. It is divided into 4 steps. First, a graph of candidate counterfactuals is generated, then these counterfactuals are filtered to keep only the valid and feasible ones. Afterwards, proximity and sparsity metrics are computed with a graph edit distance to finally present the explanations sorted based on these metrics. Each step is independent from one another. This renders the CEO method highly modular and adaptable. Choices for each step must be made based on the application. For instance, a tradeoff between computation time and the number of explanations generated must be made for the graph generation method. Likewise, concerning the proximity, an adapted similarity must be chosen. This modularity is both an advantage and a drawback. Indeed, the CEO method can be tailored to each user which is encouraged to improve explainability. However, it requires to make informed choices for each step which makes it complex to implement.

Finally, a small scale user study was conducted on a musical instrument classification task to verify the relevance and quality of generated explanations. Six experts filled a survey to assess the quality and relevance of explanations. The results of this method should not be used to compare CEO with different methods as the small sample size may skew the results. To better understand and analyse the results, a discussion with each expert was held afterwards to get their feedback that could not be gathered through a simple survey. It showed that relevant explanations are generated and experts found the explanations useful to understand the results of the presented system.

Three problems were clearly identified. The first problem is the proximity measure and more specifically the similarity chosen. It is not suited for this application and should be modified to improve the ranking of the explanations. The second problem is the exploration of the graph of all possible counterfactuals. It does not explore assertion insertions and does not guarantee to explore the best counterfactuals. However, there is a tradeoff to be made between computation time and the proportion of the graph to explore. Lastly, some types of assertions are not yet handled by the CEO method, namely *DataPropertyAssertions* and *NegativeObjectPropertyAssertions*. Handling these assertions may improve the compatibility of this method with a larger set of applications and improve intuitiveness of some operations.

In future work, problems highlighted by the user study will be addressed. Particularly, different algorithms for the graph exploration will be evaluated. In the current algorithm, only hierarchical relations are exploited, the rest of the T-Box is not used to generate the graph. Exploiting the T-Box may reduce computation time without hindering the quality of explanations. Doing so may also facilitate the exploration of assertion insertions which is lacking in the current version. Other similarity metrics will also be investigated. Particularly, a variation of the edge-counting similarity that favors classes with the same depth will be researched, as well as other classes of similarity metrics that were seen in the literature. Finally, *NegativeObjectPropertyAssertions* and *DataPropertyAssertions* support will be implemented in the next versions of this method.

References

[1] I. Tiddi, F. Lécué and P. Hitzler, *Knowledge Graphs for EXplainable Artificial Intelligence*, Studies on the Semantic Web, Vol. 47, IOS Press, Incorporated, 2020. ISBN 9781643680804.

[2] A. d'Avila Garcez and L.C. Lamb, Neurosymbolic AI: The 3rd Wave (2020).

[3] F. Lecue, On the role of knowledge graphs in explainable AI, *Semantic Web* **11**(1) (2020), 41–51. doi:10.3233/sw-190374.

[4] A. Kalyanpur, B. Parsia, E. Sirin and J. Hendler, Debugging unsatisfiable classes in OWL ontologies, *Journal of Web Semantics* **3**(4) (2005), 268–293. doi:10.1016/j.websem.2005.09.005.

[5] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin and G. Sherlock, Gene Ontology: tool for the unification of biology, *Nature Genetics* **25**(1) (2000), 25–29. doi:10.1038/75556.

[6] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer and C. Bizer, DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6**(2) (2015), 167–195. doi:10.3233/sw-140134.

[7] M.A. Musen, The protégé project, *AI Matters* **1**(4) (2015), 4–12. doi:10.1145/2757001.2757003.

[8] M. Horridge, Justification based explanation in ontologies, PhD thesis, The University of Manchester (United Kingdom), 2011.

[9] K. Schekotihin, P. Rodler and W. Schmid, OntoDebug: Interactive Ontology Debugging Plug-in for Protégé, in: *Lecture Notes in Computer Science*, Springer International Publishing, 2018, pp. 340–359.

[10] C. Alrabbaa, F. Baader, S. Borgwardt, P. Koopmann and A. Kovtunova, Finding Small Proofs for Description Logic Entailments: Theory and Practice (Extended Technical Report), *LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, vol 73, 2020, pages 32–67* (2020). doi:10.29007/nhpp.

[11] P. Lambrix, Completing and Debugging Ontologies: state of the art and challenges (2019).

[12] C. Alrabbaa, S. Borgwardt, T. Friese, P. Koopmann, J. Méndez and A. Popovič, On the eve of true explainability for OWL ontologies: Description logic proofs with Evee and Evonne, *Proc. DL* **22** (2022).

[13] S. Coetzer and K. Britz, Debugging Classical Ontologies Using Defeasible Reasoning Tools, in: *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021. doi:10.3233/faia210374.

[14] M. Vigo, S. Bail, C. Jay and R. Stevens, Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design, *International Journal of Human-Computer Studies* **72**(12) (2014), 835–845. doi:10.1016/j.ijhcs.2014.07.005.

[15] M. Bellucci, N. Delestre, N. Malandain and C. Zanni-Merk, Combining an explainable model based on ontologies with an explanation interface to classify images, *Procedia Computer Science* **207** (2022), 2395–2403. doi:10.1016/j.procs.2022.09.298.

[16] N.J. Roese, Counterfactual thinking., *Psychological Bulletin* **121**(1) (1997), 133–148. doi:10.1037/0033-2909.121.1.133.

[17] I. Stepin, J.M. Alonso, A. Catala and M. Pereira-Farina, A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence, *IEEE Access* **9** (2021), 11974–12001. doi:10.1109/access.2021.3051315.
[18] S. Verma, J. Dickerson and K. Hines, Counterfactual Explanations for Machine Learning: A Review (2020).
[19] M.T. Keane, E.M. Kenny, E. Delaney and B. Smyth, If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques, *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21), August, 2021* (2021).
[20] S. Wachter, B. Mittelstadt and C. Russell, Counterfactual explanations without opening the black box: Automated decisions and the GDPR, *Harv. JL & Tech.* **31** (2017), 841.
[21] R.K. Mothilal, A. Sharma and C. Tan, Explaining machine learning classifiers through diverse counterfactual explanations, in: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ACM, 2020. doi:10.1145/3351095.3372850.
[22] A.-H. Karimi, G. Barthe, B. Balle and I. Valera, Model-Agnostic Counterfactual Explanations for Consequential Decisions, in: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, S. Chiappa and R. Calandra, eds, Proceedings of Machine Learning Research, Vol. 108, PMLR, 2020, pp. 895–905. https://proceedings.mlr.press/v108/karimi20a.html.
[23] M. Schleich, Z. Geng, Y. Zhang and D. Suciu, GeCo: Quality Counterfactual Explanations in Real Time (2021).
[24] M. Förster, M. Klier, K. Kluge and I. Sigler, Evaluating explainable Artifical intelligence - What users really appreciate, in: *In Proceedings of the 28th European Conference on Information Systems (ECIS)*, 2020.
[25] A. White and A. d'Avila Garcez, Measurable Counterfactual Local Explanations for Any Classifier (2019).
[26] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T.D. Bie and P. Flach, FACE, in: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, ACM, 2020. doi:10.1145/3375627.3375850.
[27] W3C, OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, 2012, Online; accessed on 11-November-2022.
[28] W3C, OWL 2 Web Ontology Language Mapping to RDF Graphs, 2012, Online; accessed on 11-November-2022.
[29] W3C, Resource Description Framework (RDF) Model and Syntax Specification, 1999, Online; accessed on 11-November-2022.
[30] S. Ontañón, An overview of distance and similarity functions for structured data, *Artificial Intelligence Review* **53**(7) (2020), 5309–5351. doi:10.1007/s10462-020-09821-w.
[31] J. Euzenat, C. Allocca, J. David, M. d'Aquin, C. Le Duc and O. Sváb-Zamazal, Ontology distances for contextualisation, Contract, INRIA, 2009, euzenat2009b. https://hal.inria.fr/hal-00793450.
[32] B. Hu, Y. Kalfoglou, H. Alani, D. Dupplaw, P. Lewis and N. Shadbolt, Semantic Metrics, in: *Managing Knowledge in a World of Networks*, Springer Berlin Heidelberg, 2006, pp. 166–181. doi:10.1007/11891451_17.
[33] R. Rada, H. Mili, E. Bicknell and M. Blettner, Development and application of a metric on semantic nets, *IEEE Transactions on Systems, Man, and Cybernetics* **19**(1) (1989), 17–30. doi:10.1109/21.24528.
[34] M.-R. Amini and E. Gaussier, *Recherche d'information: Applications, modèles et algorithmes-Fouille de données, décisionnel et big data*, Editions Eyrolles, 2013.
[35] F. van Harmelen and A. ten Teije, A Boxology of Design Patterns for Hybrid Learning and Reasoning Systems, *Journal of Web Engineering, Vol. 18 1-3, pgs. 97-124, 2019* (2019). doi:10.13052/jwe1540-9589.18133.
[36] I. Tiddi and S. Schlobach, Knowledge graphs as tools for explainable machine learning: A survey, *Artificial Intelligence* **302** (2022), 103627. doi:10.1016/j.artint.2021.103627.
[37] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viegas and J. Wilson, The What-If Tool: Interactive Probing of Machine Learning Models, *IEEE Transactions on Visualization and Computer Graphics* (2019), 1–1. doi:10.1109/tvcg.2019.2934619.
[38] P.A. González-Calero, B. Díaz-Agudo, M. Gómez-Albarrán et al., Applying DLs for retrieval in case-based reasoning, in: *In Procs. of the 1999 Description Logics Workshop (Dl'99). Linkopings universitet*, Citeseer, 1999.
[39] M. Kulmanov, F.Z. Smaili, X. Gao and R. Hoehndorf, Semantic similarity and machine learning with ontologies, *Briefings in Bioinformatics* **22**(4) (2020). doi:10.1093/bib/bbaa199.
[40] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature Machine Intelligence* **1**(5) (2019), 206–215. doi:10.1038/s42256-019-0048-x.
[41] M. Bellucci, N. Delestre, N. Malandain and C. Zanni-Merk, Towards a terminology for a fully contextualized XAI, *Procedia Computer Science* **192** (2021), 241–250. doi:10.1016/j.procs.2021.08.025.