TU WIEN Informatics

Verbesserung von Entscheidungsbäumen mit Domänen-Ontologien

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Logic and Computation

eingereicht von

Majlinda Llugiqi, B.Sc
Matrikelnummer 11931216

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Prof. Asoc. Magdalena Ortiz
CoBetreuung: Priv.-Doz. Dr.Nysret Musliu

Wien, 19. April 2022

Majlinda Llugiqi, B.Sc	Prof. Asoc. Magdalena Ortiz

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Improving decision trees with domain ontologies

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Logic and Computation

by

Majlinda Llugiqi, B.Sc
Registration Number 11931216

to the Faculty of Informatics

at the TU Wien

Advisor: Prof. Asoc. Magdalena Ortiz
CoAdvisor: Priv.-Doz. Dr.Nysret Musliu

Vienna, 19th April, 2022

Majlinda Llugiqi, B.Sc	Prof. Asoc. Magdalena Ortiz

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Majlinda Llugiqi, B.Sc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 19. April 2022

Majlinda Llugiqi, B.Sc

Acknowledgements

First and foremost, I am extremely grateful to my supervisors Prof. Asoc. Magdalena Ortiz and Priv.-Doz. Dr. Nysret Musliu, for their invaluable advice, unwavering support, and patience throughout my master's thesis. Their vast knowledge and lot of experience have helped me throughout my master thesis.

I'd like to thank my parents and family for their incredible patience, encouragement, and unconditional support, to whom I dedicate this work.

Finally, I'd like to express my gratitude to Gent for his unwavering emotional support, love, encouragement, and belief in my qualities.

This work was financially supported by the University of Technology Vienna.

Kurzfassung

Systeme der künstlichen Intelligenz (KI) basieren häufig auf maschinellen Lernverfahren, die allein aus einem Datensatz lernen, ohne sich auf Allgemeinwissen oder andere domänenspezifische Kenntnisse zu stützen, als die, die ausdrücklich als "Merkmaleïn den Daten enthalten sind. Trotz der enormen Menge an Expertenwissen, das für den Bereich existiert, und der Tatsache, dass ein Großteil davon in bestehenden Ontologien leicht nutzbar ist, wird normalerweise nicht einmal ein kleiner Teil davon verwendet.

In dieser Arbeit untersuchen wir die Möglichkeiten der Nutzung von Domänenwissen, insbesondere von medizinischen Ontologien, um das Lernen von Entscheidungsbäumen für solche Domänen zu verbessern. Insbesondere bewerten wir Techniken sowohl für die Konstruktion von Entscheidungsbäumen direkt aus Daten, als auch für die Konstruktion von Entscheidungsbäumen als Approximation eines neuronalen Netzes aus dem Trepan-Algorithmus, die mit einigen aus Ontologien berechneten Maßen erweitert werden können. Wir wählen einige bestehende Ansätze aus der Literatur aus und analysieren sie, und wir schlagen auch andere Varianten vor. Darüber hinaus bewerten wir die Auswirkungen solcher ontologischen Maße sowohl auf die *Verständlichkeit* als auch auf die *Genauigkeit* der resultierenden Bäume. Um die Verständlichkeit zu bewerten, werden neben zwei syntaktischen Komplexitätsmaßen, die wir berechnen, führen wir auch drei Benutzerbefragungen mit vier verschiedenen Aufgaben durch und bewerten die Ergebnisse in Bezug auf Antwortzeit, Korrektheit, Vertrauen in die Antworten sowie die Wahrnehmung der Verständlichkeit der Bäume durch die Benutzer. Für den Vergleich der Ansätze erstellen wir einen Testsatz aus sieben medizinischen Datensätzen, gepaart mit themenspezifischen Ontologien, die aus zuverlässigen, echten medizinischen Ontologie-Repositorien stammen. Angesichts des Mangels an Benchmarks, die Datensätze des maschinellen Lernens mit Ontologien verknüpfen, ist die Erstellung eines solchen Testsatzes wichtig für dieses Forschungsgebiet.

Die Ergebnisse unserer Experimente zeigen, dass die Einbeziehung von Heuristiken aus Ontologien in den Prozess der Entscheidungsbaumerstellung die Genauigkeit von Entscheidungsbäumen für die meisten der von uns verwendeten Domänen mäßig verbessert, insbesondere für Entscheidungsbäume, die aus einem neuronalen Netz unter Verwendung des Trepan-Algorithmus extrahiert wurden.

Darüber hinaus stellen wir anhand der Ergebnisse der benutzerbasierten Umfragen fest, dass die Benutzer den Baum, der mit unserer modifizierten Version von hubscore, genannt

relevance-score, aus der SNOMED-CT-Ontologie erstellt wurde, im Durchschnitt etwas einfacher zu verstehen finden und dass die Benutzer mehr Vertrauen in ihre Antworten zu diesen Bäumen haben sowie einen höheren Anteil an richtigen Antworten geben.

Abstract

Artificial Intelligence (AI) systems often build on machine learning techniques that learn from a dataset in isolation, without relying on any common sense knowledge, or any knowledge of the domain other than what is explicitly reflected as 'features' in the data. Despite the vast amount of expert knowledge that exists for the domain, and the fact that much of it is readily usable in existing ontologies, usually not even a small fragment of it is used.

In this thesis, we investigate how domain knowledge, especially medical ontologies, might be used to improve decision tree learning. In particular, we assess techniques both for constructing decision trees directly from data, as well as constructing decision trees as approximation of a neural network extracted with the Trepan algorithm, that can be enhanced with some measures computed from ontologies. We select and analyze some existing approaches from the literature, and also propose some variations. Moreover, we asses the impact of such ontological measures on both the *understandability* and the *accuracy* of the resulting trees. For evaluating the understandability, beside two syntactic complexity measures that we calculate, we also perform three user questionnaires depending on the domain, with four different tasks and evaluate the results in terms of time response, correctness, confidence on the answers, as well as the users' perception of the understandability of the trees.

For the comparison of the approaches we create a test set of seven medical datasets paired with topic-specific ontologies extracted from reliable real-life medical ontology repositories. Given the lack of benchmarks linking machine learning datasets and ontologies, the construction of such a test set is important on its own.

The results of our experiments show that incorporating heuristics from ontologies into the decision tree building process moderately improves the accuracy of decision trees for most of the domains we use, particularly for decision trees extracted from a neural network using Trepan.

Furthermore, based on the findings of the user-based surveys, we observe that users, on average, find the tree built with our modified version of hubscore called relevance-score from the SNOMED-CT ontology slightly easier to understand; users are more confident in their answers concerning these trees, and give a higher proportion of correct answers.

Contents

CHAPTER 1

Introduction

1.1 Motivation

The advancement of Artificial Intelligence (AI) during the last decade has been remarkable. However, despite having overcome issues that looked unachievable only a few years ago, AI is frequently allowed levels of autonomy that are out of proportion to its capabilities. Machines have highly effective techniques of learning from data and discovering patterns that humans cannot, but while doing it, they usually do not integrate such patterns with other information and world models. AI systems are frequently built on Machine learning (ML) approaches that learn from a dataset in isolation. The techniques for relying on common sense knowledge or domain knowledge other than what is clearly reflected in the data as 'features' are still lacking.

To add to the challenges, many ML approaches are are often blackbox approaches and understanding how they work is very difficult. Considering that ML is supposed to make the life of people easier, this drawback presents a big hurdle in the integration of ML in everyday life. The explainablity of ML models has become a very important field of study, especially in critical sectors like medicine, where understanding why the particular decisions are made is crucial [TJMG19, HLD+19, ZLFC18].There has been a growing interest in explainable classifiers such as decision trees, which may be used as classifier or as a technique to approximate other classifiers in order to comprehend them [XUD+19].

There are several techniques for explaining ML models, but very few of them seek to include pre-existing domain knowledge into the process. A possible source for the domain knowledge can be ontologies. Ontologies provide a logic-based data model for knowledge representation by using description logics as its representational basis. This may be used in a variety of real-world applications like for enhancing machine learning approaches, because of their domain knowledge reasoning. There are several ontologies that have been used for a variety of purposes and include high-quality knowledge. Given the

huge amount of medical knowledge in existing ontologies, it would make sense to try to enhance such decision trees using that knowledge. But this calls for ontologies that precisely match the features of the dataset of interest and such ontologies are not readily available. This knowledge is frequently embedded in larger ontologies. In any case, extracting topic-specific parts of such ontologies is an important topic that has been widely investigated and we want to investigate what methods are being used in this direction and whether any of them can be leveraged to improve decision trees.

There has been some work in the literature in recent years that uses knowledge to obtain better explainable classifiers [CWBdPM21, PPP20]. However, as we discuss below, these works are limited, not easy to apply and to compare in different domains. There are only preliminary experimental results using handcrafted ontologies that are not realistic medical ontologies and they are quite sensitive to the syntax of the ontology.

1.2 Aims of the thesis

The general aim of the thesis is to investigate possible applications of domain knowledge, particularly medical ontologies, to increase the accuracy and understandability of decision trees. We tackle the following research questions:

- (RQ1) How can we more systematically generate or extract meaningful medical ontologies that contain domain knowledge about a set of 'features' from given datasets?
- (RQ2) Do the existing approaches that use domain knowledge obtain better and more explainable classifiers?
- (RQ3) Are there other ways to harness knowledge from an ontology to obtain better and more explainable classifiers?

1.3 Contributions

The main contributions of this thesis are:

- We build a test set of seven medical datasets linked with topic-specific ontologies collected from existing reliable high-quality medical ontology containing the same concepts as datasets' features in a semantic level. Due to the scarcity of benchmarks connecting machine learning datasets and ontologies, creating such a test set is a noteworthy contribution in its own. Moreover, we handcraft two ontologies for heart and breast cancer domains using corresponding datasets' features as concepts and roles, the same way the authors of Trepan Reloaded [CWBdPM21] did. In addition, for these two domains we extend two existing ontologies with the corresponding datatsets' features.

- We study some specific heuristics to incorporate the knowledge from the ontologies we created and extracted to decision tree construction algorithms. Specifically we consider *information content* from Trepan Reloaded [CWBdPM21] and propose another heuristic called *relevance-score* which works in cases when information content gives is not applicable, which is the case in particular for ontologies extracted from SNOMED-CT.

- We evaluate these two heuristics from different ontologies on two different ways of obtaining decision trees, for direct decision tree construction and for decision trees extracted with Trepan algorithm. We do this for seven domains. To compare between the heuristics we use *accuracy* of the trees.

- To measure the understandability of the trees, we look into the syntactic complexity of the decision trees using two different measures from the literature. Moreover, we conduct three user-based studies analogous to Trepan Reloaded, with 234 participants, to see whether the reported improvements in understandability from Trepan Reloaded paper are reproduced. Moreover, from the user-questionnaires we find a correlation of syntactic complexities of the trees with user subjective understandability, correctness of the responses and time response.

- From results we observe that the accuracy of decision trees when the heuristics from ontologies are incorporated in the direct tree building process increases for most of the considered domains, albeit modestly. Moreover, for the decision trees extracted from Trepan algorithm, using domain knowledge from a high-quality ontology either increases the accuracy or the accuracy stays the same for different domains. Furthermore, from user-based questionnaires, we observe that users find the tree built with our heuristic called relevance-score from the SNOMED-CT ontology slightly simpler to understand on average, and users are more confidence in their replies about these trees, as well as give a larger proportion of correct answers.

1.4 Organization

The remainder of the thesis is structured as follows. In Chapter 2, we begin with some preliminary material, we discuss the notion of ontologies, description logics, the SNOMED-CT ontology, then we continue with explainability of machine learning and decision trees. Then, in Chapter 3, we explore modularization and forgetting as two methods for obtaining topic specific ontologies, and we focus on a method that combines modularity and forgetting for ontology extraction from large ontologies. We cover several attempts to incorporate domain knowledge at various stages of the machine learning pipeline in Chapter 4, with an emphasis on the Trepan Reloaded algorithm. Our work on different approaches to the improvement of learned decision trees using knowledge from ontologies is then presented in Chapter 5. In Chapter 6, we present experimental

evaluation methods, followed by the results of such experiments. Finally, Chapter 7 concludes this thesis and gives an outlook on possible future work.

CHAPTER 2

Preliminaries

In this chapter we introduce some preliminaries that will be used throughout the rest of this master thesis. In the first section we present the basics about ontologies, discuss components and advantages of ontologies, the basics about description logics as a formal language commonly used for defining ontologies, as well as the SNOMED-CT ontology. In Section 2.2 we discuss machine learning and present explainability of machine learning, with a focus on decision trees.

2.1 Ontologies

Ontology is a philosophical term that refers to the study of existence and the essence of being. From the computer science perspective, Gruber [Gru93] in 1993 defined the ontology notion as "explicit specifications of conceptualizations", whereas the notion of conceptualization is defined as a simplified, abstract perspective of the world that we want to portray for whatever reason.

An ontology establishes a standard terminology for sharing knowledge in a certain domain. It comprises machine-interpretable definitions of key concepts in the domain as well as relationships between them.

The types of knowledge that may be represented by ontologies varies. However, Corcho et al. [CFLGP06] describe a core set of components that all ontologies have in common:

- **Concepts** (classes), represent types of objects in the domain, which may be abstract or concrete. Ontology classes are usually organized into hierarchical taxonomies, which may be used to apply inheritance methods by assessing if an object is necessarily an instance of another class if it is an instance of one.
- **Relations** (roles) represent relations between concepts in the domain. Ontologies usually contain binary relations, with the first argument being the domain of the

relation and the second argument being the range. Relations, like classes, may be arranged into taxonomies.

- **Instances** represent instantiations of concepts, i.e Person(Lisa) represents that Lisa is an instance of concept Person.
- **Axioms** model statements in a domain that are always true and are usually utilized to express information that can't be formally specified by the other components. Axioms are also used to ensure that the ontology is consistent.

Some of the advantages of utilizing an ontology include: humans or software agents having a shared understanding of the structure of information, reusing and sharing domain knowledge, making domain assumptions explicit, assessing domain knowledge, and distinguishing domain knowledge from operational knowledge, or linking technical language with human understandability [NM+01]. Due to these benefits, ontologies have been used in different areas such as: artificial intelligence [BCM05, **?**], software engineering [HS06], computer security [RAA+14, KKK13] and biomedicine [SAMK05, RLM+06].

There are different categories based on the expressiveness of the knowledge representation formalism used to build ontologies. Lassila and McGuinnes [LM01] categorized ontologies into four groups from the least expressive to very expressive ontologies:

- Glossaries represent the most basic forms of ontologies, which are just a list of concepts, a regulated vocabulary is an example of this type of ontology.
- Thesauri also known as taxonomies are ontologies as lists of concepts with a fixed set of relationships between them.
- Ontologies represented using metadata, XML, schemas and data models. This category of ontologies can specify concept hierarchies, attributes, relations, and axioms.
- Ontologies represented using logical languages are the most expressive types of ontologies. The formal languages include syntax, well-defined semantics, and reasoning techniques like as consistency testing. A formal language commonly used for defining ontologies is *description logics*. SNOMED-CT which we will discuss into more details in Section 2.1.2 falls into this group of ontologies.

2.1.1 Desctiption Logics

Description Logics is a formal language for describing knowledge and reasoning about it. In other words, description logics is a term for a family of knowledge representation formalisms that express an application domain's knowledge by first defining the domain's important concepts and then utilizing these concepts to specify attributes of objects and individuals that appear in the domain [BN03].

In description logic languages, there are three fundamental building [BCM+03] components:

- Atomic concepts or unary predicates which represent types of objects in the domain, e.g. *Woman*, *Man*, *Person*.
- Atomic roles or binary predicates which represent binary relations between objects in the domain, e.g. *hasChild*, *hasParent*, *livesIn*.
- Individuals or constants represent actual objects in the domain, e.g. *lisa*, *vienna*, *smith*.

Complex concept and role descriptions may be created by combining these components using logical constructors such as conjunction, disjunction, existential quantification, and so on. There are different variants of description logics depending on the kind of logical constructors they allow. The $\mathcal{EL}$ family, in which the ontologies in this thesis are written, allows only conjunctions and existential restrictions.

The semantics of concepts are specified in terms of *interpretations* I that consists of a non-empty domain Δ^I and an interpretation function $.^I$, which maps every individual name to an element of the domain Δ^I, each atomic concept to a subset of the domain, each atomic role to a binary relation on the domain, and to each individual name an object of the domain.

A *knowledge base* in description logics is divided into two parts: a terminological component (TBox) and an assertional component (ABox). A TBox is made up of a finite number of terminological *axioms* which are statements describing the relationships between concepts and roles. On the other hand, an ABox consists of assertional knowledge which are statements regarding individual membership to concepts (concept assertions) and relations between individuals (role assertions).

Subsumption, commonly represented as $C \sqsubseteq D$, is the most fundamental inference on concept expressions in Description Logics. The task of determining subsumption consists on deciding if the concept indicated by D (the subsumer) is more general than the one denoted by C (the subsumee). Subsumption, in other words, determines whether the first concept always refers to a subset of the set signified by the second [NB03].

In this master's thesis, we will capture the degree of generality or specificity of a concept and utilize it as information content to mix with information gain while creating decision trees, as well as the distance to a general concept. In chapter Chapter 5, we will go through how to use information from ontologies in detail.

2.1.2 SNOMED-CT Ontology

Systematized Nomenclature of Medicine — Clinical Terms (SNOMED-CT)[1] is a medical terminology that is used to standardize the storage, retrieval, and interchange of electronic

[1] https://www.snomed.org

health data. SNOMED-CT's purpose is to develop a taxonomy of terms referring to entities in a specific medical environment, as well as a set of criteria to ensure that each term is used with exactly one meaning.

The core component types in SNOMED-CT are: concepts, descriptions and relationships [Bha15].

- *Concepts*: every concept has its distinct identifier and represents a distinct clinical meaning.
- *Descriptions*: every concept has two sorts of descriptions: a fully specified name (FSN), which is unique and displays the meaning of the concept, and a synonym, which may be used to display or pick a concept. There may be multiple synonyms for a notion, but only one FSN.
- *Relationships*: show a connection between two concepts. Within SNOMED-CT, many forms of relationships are supported.

SNOMED-CT is used in over 50 countries and its 31st of July 2021 release, contains more than 350,000 concepts. [Bha15] summarizes the benefits of SNOMED-CT as:

- Benefits to individuals: allowing for uniform recording of clinical data, providing support systems with the ability to examine the record and offer real-time guidance, supporting the exchange of relevant information with those involved in the delivery of care, allowing all providers to comprehend the information in the same way. Moreover, SNOMED-CT allows multilingual use, removing language barriers.
- Benefits to populations: early detection of emergent health risks, population health monitoring, and reactions to changing healthcare practices are all made easier, providing precise and focused access to important data, eliminating costly duplications and mistakes, providing access to pertinent data to enable clinical research and provide evidence for future treatment improvements and adding possibilities for extensive study of clinical information to evaluate anomalies and exceptions to care delivery audits.

SNOMED-CT has been utilized in a variety of clinical documentation applications because it allows for the representation of comprehensive clinical data in a manner that can be processed automatically [Bha15]. In this thesis, we will use SNOMED-CT which to the best of our knowledge is the best high-quality ontology that contains a vast amount of knowledge about the specific domains that we want to include.

2.2 Explainability of machine learning

The advancement of Artificial Intelligence (AI) during the last decade has been remarkable. However, despite having overcome issues that looked far beyond reach only a few years ago, AI is often granted levels of autonomy that are disproportionate to its capabilities. AI systems often do not understand the world, and we do not understand them. Machines have highly effective techniques of learning from data and discovering patterns that humans cannot, but they are often unable to integrate such patterns with other information and world models. Machine learning is a subfield of AI that aims to analyze the structure of data and fit that data into models. It focuses on algorithms that evolve as they analyze data, allowing computers to develop without having to be explicitly programmed.

For humans, insights into decision making are usually opaque. Understanding decision making is especially crucial in highly sensitive fields such as healthcare, the legal system, finance, bio-informatics, and the automobile industry, where the wrong decision can have serious consequences [BH21]. In machine learning, explainability involves being able to explain what happens in your model from input to output. Another definition of explainability is by Lepri et al. [LOL+18] as a way of improving the transparency of machine learning models. Transparency encompasses a wide range of attempts to give useful information about how a model reaches a decision, frequently seeking to increase trustworthiness to end users [BXS+20]. Three degrees of transparency are considered by Lipton [Lip18]: the training procedure (algorithmic transparency), specific components such as parameters (decomposability), and the overall model (simulatability).

The value of explainability as a notion has been reflected in legal and ethical principles for data and machine learning. Articles 13-15 of the European General Data Protection Regulation (GDPR) require that data subjects are provided with some easily understandable information regarding the logic involved, as well as the importance and expected implications of such processing for the data subject [SP18].

Various reasons why explainability of machine learning models is essential have been investigated in literature. Arrieta et al. [ARS+20], summarized the arguments that we found in the literature advocating for explainability:

goals pursued from literature toward reaching explainability:

- Trustworthiness: for the majority of the authors trustworthiness is the most significant reason why machine learning explainability is important particularly in high-risk industries like as healthcare and finance. Before machine learning solutions are deployed and trusted, all stakeholders must properly understand how the model works [Lip18]. Although trustworthiness should surely be a component of any explainable model, this does not imply that every explainable model is trustworthy on its own.

- Casuality: discovering causality between data variables is another reason for explainability. A machine learning model only finds correlations in the data it learns from, which may not be enough to reveal a cause-and-effect link [CPC19]. Causation, on the other hand, indicated that the change in one event is the result of the change of the other event, therefore an explainable machine learning model might validate the results produced by causality inference techniques or provide a first impression of possible causal linkages within the available data.
- Transferability: models are constantly constrained, which should allow for smooth transferability. In order to apply the prediction model with unseen data, the prediction model must transmit a knowledge of future behavior to a human decision-maker [Lip18].
- Informativeness: a considerable lot of information is required to be able to tie the user's decision to the model's solution and avoid falling into misinterpretation errors. Explainable machine learning models should provide information about the problem being addressed for this purpose [ARS+20].
- Confidence: confidence should always be measured against a model that is expected to be reliable [DVK17]. Models that are unstable should not give trustworthy interpretations. As a result, an explainable model should include information regarding the operating regime's confidence.
- Fairness: knowing the reasoning for a particular choice is a societal requirement. In order to perceive conformity to ethical norms, decision-makers must convey their results in an intelligible manner. This right to explanation is available to anybody who is impacted by an automated decision.
- Accessibility: some authors advocate for explainability as a quality that allows end users to become more active in the process of upgrading and developing a particular machine learning model [ARS+20].
- Interactivity: One of the aims of an explainable machine learning model is to incorporate the capacity of a model to engage with the user. This aim is connected to industries where end users are extremely important, and their capacity to alter and interact with models is what assures success[DVK17, Lip18].
- Privacy awareness: Inability to comprehend what the model has recorded and stored in its internal representation may constitute a breach of privacy. In contrast, the capacity of non-authorized third parties to explain the inner relations of a trained model may jeopardize the differential privacy of the data origin.

There are several techniques for describing the models and/or their decisions. There are explanations for both global model behavior and local explanations for the model's decision about each occurrence in the data. Arrieta et al. [ARS+20] distinguish different types of explanations such as:

- Text explanations: include natural language text or propositional symbols to describe the model's behavior by defining abstract concepts that represent high-level operations.
- Visual explanation: include visuals that assist in the comprehension of a model. Most visualizations are utilized as supplementary tools, particularly when appealing to a non-expert audience.
- Local explanations: try to explain how a model works in a specific context. As a result, the provided explanations may not always extend to a global scale, accurately expressing the model's overall behavior.
- Explanations by example: in this group there are explanations that show representative examples from the training dataset. In this case, to be understood the training data must be in a human-readable format.
- Explanations by simplification: relate to methods for approximating an opaque model with a simpler, more interpretable one. The major problem is that the basic model must be flexible enough to correctly represent the complicated one. The trepan algorithm [CS95], which we will discuss in Section 4.1 falls into this group.
- Feature relevance explanations: Attempt to quantify the impact of each input variable on a model's decision. This yields a ranking of significance scores, with higher scores indicating that the related variable was more significant to the model.

With increasingly advanced machine learning approaches like Neural Network, understanding why the model reaches a conclusion is difficult. Neural Network models' complexity and their multi-layered neural structure makes decision-making transparency even more difficult. Thus, in our thesis we use decision trees to interpret such models.

2.2.1 Decision Trees

A few machine learning models have the attribute of explainability, which includes transparency, simplicity of understanding, and the ability to query. One of those are decision trees. Decision trees is a type of a visual interpretation that depicts the decision-making process by visualizing several courses of action and their potential outcomes. Decision trees are arranged in a hierarchical manner and contain three types of nodes: leaf nodes, inner nodes and a root node (the most upper node). Inner nodes are assigned with a logical test based on the feature in the domain. Depending on whether the logical test is over binary, nominal, or real values attributes, split nodes may have several branches. In the most basic case, this test takes into account one feature, and the result is decided by the value of that feature in the given sample.

In the case of classification, leaf nodes are given a class label, whereas in the case of regression, a continuous quantity is assigned. The algorithm starts at the root node (topmost node) and descends until it reaches a leaf node. A specific feature value is

compared to the splitting value at each inner node. Depending on the conclusion of this comparison, the traversal proceeds on the left or right path [BH21]. In Figure 2.1 is an example of a decision tree and its components.

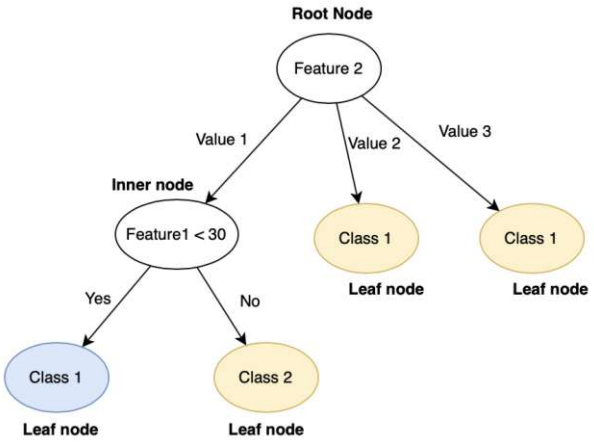


Figure 2.1: Example of a decision tree

Decision trees are typically used in situations where understandability is critical for the application at hand, hence simple trees are preferable [BP21].

Different algorithms are used to build decision trees. The simplest and among the first decision tree algorithms is ID3 algorithm introduced by Quinlan [Qui83, Qui86]. ID3 algorithm lack on handling numeric and missing values, as well as data may be over-fitted [SG14]. To tackle this issues, the same author Quinlan introduced C4.5 [Qui14], a successor of ID3, which we will discuss later in this section. Other known decision tree algorithms are: Classification And Regression Tree (CART) [Lew00], CHi-squared Automatic Interaction Detector (CHAID) [RB10] and Multivariate Adaptive Regression Splines (MARS) [Fri91]. They differ in terms of the split at each node, the measure used to gather input data, pruning, supporting categorical and/or continuous variables, being able to perform regression tasks and so on.

In our work we will use the C4.5 algorithm. The advantages for using this decision tree algorithm are: it supports categorical and continuous variables, has multiple splits at each node, does pruning after construction and uses entropy and information gain as measures for splitting nodes [CA21]. Information gain, as the name indicates, is the amount of information obtained by an attribute, and it shows the significance of the attribute. At each stage of the tree's construction, information gain is utilized to choose attribute that distinguishes the best the training samples based on their classification objective, for this entropy is calculated. Entropy is the degree of impurity in the given dataset. Entropy is calculated by Equation (2.1), where $p(x)$ is the probability mass function of the random variable X. In Algorithm 1 is shown the pseudocode of the C4.5 algorithm.

The base cases are [GGB12]:

Algorithm 1 The pseudocode of the C4.5 algorithm [LP10].

1. Check for **base cases**.
2. For each attribute ***a***.
   find the **normalized information gain** from splitting on ***a***.
3. Let ***a_best*** be the attribute with the **highest normalized information gain**
4. Create a **decision node** that splits on ***a_best***
5. Recur on the sublists obtained by splitting on ***a_best***, and add those nodes as children of **node**.

- The training set is empty, then a tree leaf labeled 'failure' is returned.
- The attribute list is empty, then a leaf with the most common class or the disjunction of all classes is returned.
- All the training set's examples belong to the same class, then a leaf labeled with that class is returned.

$$H(X) = -E[log_P(X)] = -\sum_{x \in X} p(x)\ log\ p(x) \tag{2.1}$$

The greater the entropy, the more difficult it is to come at any conclusion. When the sample is completely homogenous, the entropy is zero.

After the calculation of the entropy, the information gain is calculated by the formula Equation (2.2), which compares the entropy before and after the split. The best split may be determined by selecting the one that maximizes information gain based on entropy values.

$$IG(T, X) = H(T) - H(T, X) \tag{2.2}$$

Despite the benefits, there are significant drawbacks to using decision trees as explainable models. Such disadvantages include: minor change in the data might result in a significant change in the structure of the decision tree, the training period for a decision tree is frequently long, decision trees generally perform well when there are only a few highly significant attributes, but less effectively when there are numerous complex relationships, and most decision tree algorithms are insufficient on performing regression [RM15, LB02, Qui96].

CHAPTER 3

Obtaining topic specific ontologies

In this chapter, we discuss how to extract the knowledge we want from a given ontology. One of our goals is to reuse existing knowledge. However, there are no particular ontologies for the domains that we require, and that this information is frequently contained in larger ontologies like SNOMED-CT, which is very accurate but also quite large, making it not trivial to get the right knowledge. Extracting topic specific parts of such ontologies is an important topic and it has been widely investigated. In this chapter, we want to investigate what strategies can be employed in this direction, with a particular focus on the two main ones, forgetting and modularity, and see whether any of them work for us.

The manual construction of ontologies is very expensive, taking a lot of time and numerous resources. Several tools exist in computer science to assist end users and system developers in developing high quality ontologies. Various tools, in particular, assist individuals in creating categories, partonomies, taxonomies, and other organization levels of ontologies, either manually or semi-automatically [CC05]. The following are some of the most important ontology editors and managers: Protégé[1], SWOOP[2], OWL-S Editor[3], OntoManager[4].

It's generally always worthwhile to investigate what others have done and see if we can improve and enhance current sources for our domain and goal. Many ontologies are currently accessible in electronic form, and you may import them into your ontology-development environment. However, many ontologies are large and detailed, such as SNOMED-CT, in which we are interested. Working with and maintaining such vast ontologies is difficult. Usually only a very small part of such ontologies is of importance for our domain. One of the most essential success factors nowadays is the ability to

[1] http://protege.stanford. edu
[2] http://www.mindswap.org/2004/SWOOP
[3] http://owlseditor.semwebcentral.org
[4] http://ontoware.org/details/ontomanager

eliminate any knowledge that is irrelevant to the task at hand, hence reducing the amount of knowledge to be considered to a reasonable level. As a result, rather than starting from scratch, one may build ontologies from existing ones, which saves time and reduces the risk of errors.

In the literature one may find two approaches with the aim of building ontologies from existing ones:

- *Forgetting*: The goal of forgetting is reducing an existing ontology by deleting some concepts and roles while keeping all logical consequences up to the remaining symbols. Forgetting allows one to focus on the information they need. Forgetting may be described in two ways: syntactically as the dual of uniform interpolation [KWW09], and model-theoretically as semantic forgetting [WWT+14]. Up to a specific number of names, uniform interpolation keeps all logical consequences, whereas semantic forgetting preserves equivalence up to a certain number of names. As a result, semantic solutions are usually more powerful than uniform interpolants; nonetheless, they typically require the target language to be extended to express them. There are several algorithms and tools for forgetting. Some of the tools are: LETHE, FAME, NUI and SCAN. We will go into further details regarding forgetting tools in section 3.1.
- *Modularity*: The goal of modularization is to generate an ontology that includes all of the ontology's axioms that are relevant to the input signature, since there are parts of a large ontology that may be extracted and utilised outside of the context of the entire ontology. No consequences in the signature of a logical module are lost when it is extracted from its original context, and no new consequences are acquired. Thus, logical modules are self-contained components inside an ontology that may be securely removed without adding or deleting entailments in the signature of other modules, from a model-theoretic perspective [GPSK06]. Some approaches for extracting sections of an ontology are semantic modules, minimal subsumption modules and locality-based modules. We will go into further details regarding these approaches in section 3.1.

The most essential use of both forgetting and modularity approaches is reusing parts of larger ontologies, since it is easier for domain experts to use the terminology as well as subontologies contain only terms relevant and familiar to a specified sub-domain.

The difference between forgetting and modularity is that unlike modularity in forgetting approach the new ontology must be constructed without the 'forgotten' symbols. Another difference between the two methods is that forgetting is not constrained by how the original ontology is expressed so it will be defined irrespective of the ontology's axiomatisation, whereas modules are subsets of the original ontology [KWZ10]. Eventhough it may seem as a benefit, in some cases it may be perceived as a disadvantage of forgetting over modularity due to the fact that ontology engineer is not familiar with the new axioms, which might be difficult to understand and process [KWZ10].

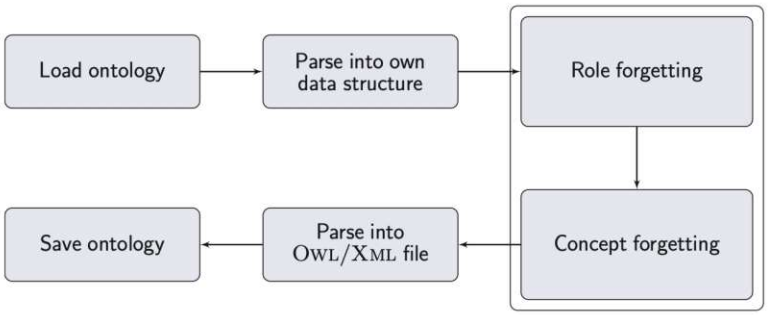


Figure 3.1: The top-level design of FAME[ZS18].

3.1 Approaches for Forgetting and Modularity

In the literature, there are several forgetting tools implemented such as FAME [ZS18], LETHE [KSD+15], SCAN [GSS08] and NUI [KWW09].

FAME is among the first tools for semantic forgetting in description logics that is automated [ZS18]. It is a Java implementation of an Ackermann-based approach for forgetting concept and role names from ontologies expressed using the $\mathcal{ALCOIH}$ description logic [WDL+20]. The forgetting process in FAME is shown in fig. 3.1, where we distinguish four main phases: clausification, which is converting ontology O into a collection of clauses, the role forgetting phase, the concept forgetting phase and declausification, which is converting the resultant clause set into an ontology O'.

Because concept definer names may be introduced during the role forgetting process, FAME removes role names first (to facilitate the normalisation of the input ontology). As a result, these definer names, which are treated as standard concept names, can be deleted as part of the concept forgetting process. After this step FAME parses the result as an ontology either in OWL or XML format and stores it.

LETHE [KSD+15] is another forgetting tool that aims to project a given ontology to a new smaller ontology that utilizes the concepts selected by the user while deleting others, yet retaining any logical entailments that can be represented with those concepts. LETHE uses a resolution-based technique to compute its forgetting solution for description logic-based ontologies.

LETHE is available as a command-line tool, a graphical user interface, and a Java library [Koo20]. LETHE, as a command line or console interface, allows users to choose a time out after which the partial uniform interpolant is stored even if the calculation has not yet finished. Using a graphical user interface, the user uploads the ontologies in OWL syntax and it is shown in description logic syntax. Then the user chooses the target signature as well as the forgetting method to be used. During computation, the user is shown a progressbar in which the current name is being forgotten. Furthermore, the user can stop the operation at any time and view the current calculated uniform inerpolant. The third form of using LETHE and the most relevant for practical application is the

ability of using LETHE as a Java library. LETHE offers a facade that is compatible with the OWL API 5.1.7 and supports conventional Java data structures.

Another forgetting tool is SCAN. It uses a resolution-based technique, notably the SCAN algorithm [GO92], to compute forgetting solutions for knowledge-bases represented in first-order logic. SCAN is not guaranteed to provide a solution since forgetting in first-order logic is not frequently solvable. SCAN applies Skolemization to delete existing quantifiers, hence Skolem expressions may occur in SCAN's solutions [AS17].

The last forgetting tool that we are going to discuss is NUI [KWW09]. NUI is a framework based on resolution, for forgetting concept and role names from TBoxes in $\mathcal{ELH}$, a light-weight description logic that is less expressive than $\mathcal{ALC}$, extended with role inclusions and domain and range restrictions [WDL+20]. This motivation on choosing this description logic is the fact that forgetting appears to be of particular importance for large-scale and comprehensive ontologies, and that many of these ontologies are provided in this language.

The main difference between the tools is the forgetting methodology used and the expressivity of the logics being supported. SCAN, LETHE and NUI use a resolution-based approach to compute their forgetting solutions, whereas FAME uses an Ackermann's Lemma-based method. Another distinction is that LETHE and FAME guarantee that they will terminate, whereas SCAN does not guarantee. Overall, NUI and LETHE work with less expressive description logics, whereas FAME can handle more expressive ontologies like $\mathcal{ALCHOI}$. These tools perform well on medium-sized ontologies, but are not very suitable to use when dealing with very large ontologies such as SNOMED-CT, for which the tools do not seem to guarantee a solution.

The other approach that we are interesting in is modularization. We will discuss three kinds of modules: semantic modules, minimal subsumption modules and locality-based modules.

Semantic modules are logic-based modules [PJC09]. To formalize such modules the model-theoretic inseparability relation [KLWW13] is used. Inseparability relation means to be indistinguishable from the original ontology. Formally we say that two general TBoxes T_1 and T_2 are Σ-inseparable, $T_1 \equiv_\Sigma T_2$, if $\{I_\Sigma | I \models T_1\} = \{I_\Sigma | I \models T_2\}$ [CAS+19b]. Based on inseparability relation, several module notions have been proposed including plain, self-contained and depleting modules [KLWW13, KWZ10]. From ontologies defined as $\mathcal{ELI}$-terminologies, the MEX system [5] extracts minimum depleting and self-contained semantic modules.

The second modules that we discuss are minimal modules. Minimal modules can serve as explanations for the complete set of entailments over a signature. Minimal modules can help us grasp the internal structure of vast and complicated ontologies in this manner. Furthermore, knowing how to compute all minimum modules helps us to choose the smallest one. Minimal subsumption modules are often smaller than semantic modules

[5] https://cgi.csc.liv.ac.uk/ konev/software/

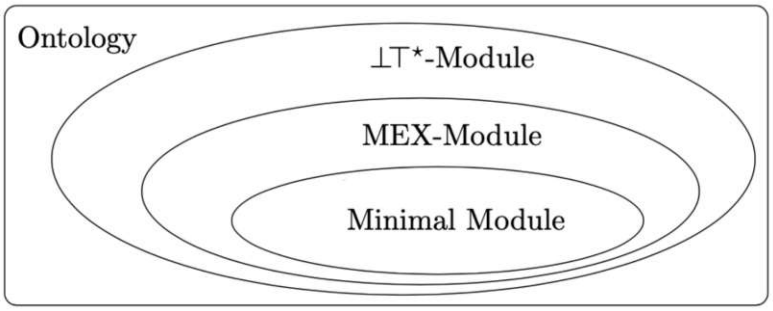


Figure 3.2: Zooming in on an ontology. Modification from [CLMW17] figure.

[CLW18]. However, deciding whether or not to keep subsumption queries might be costly [CAS+19b]. Extracting minimum modules is inherently difficult, which is why efficient extractable approximations of the (union of all) minimal modules were created. The family of syntactic locality-based modules is one of these approximations.

Another approach to modularization is locality-based modularization, which is one of the most common ways of extracting parts of an ontology. Modularization methods based on the notion of locality have been presented in [GHKS07], with the goal of identifying a collection of axioms $a \in A$ that do not modify the meaning of the terms in a signature Σ. The authors in [GHKS07] define six different types of locality, the most commonly used of which are:

- *top locality* ($\top$-locality), if an axiom does not define new sub-concepts for a given concept C, it is considered $\top$-local for C
- *bottom locality* ($\bot$-locality) if it does not define new super-concepts, it is considered $\bot$-local for C
- *star modules* integrates top and bottom locality notions by iterative and exhaustive application [CAS+19a].

Locality-based modularization is used for more localized, and often easier, processing using other tools like reasoning, querying, retrieval, and ontology mapping [CAS+19b]. Moreover, locality based modularization is available in the OWL API.

Based on the size of the extracted ontology, locality-based modules give larger ontologies, followed by semantic modules and then minimal modules, as shown also in the Figure 3.2.

3.2 Ontology extraction for large ontologies via modularity and forgetting

As we discussed in section Chapter 3, a module is a subset of an ontology's axioms, whereas forgetting is a compact representation of the ontology restricted to a subset of

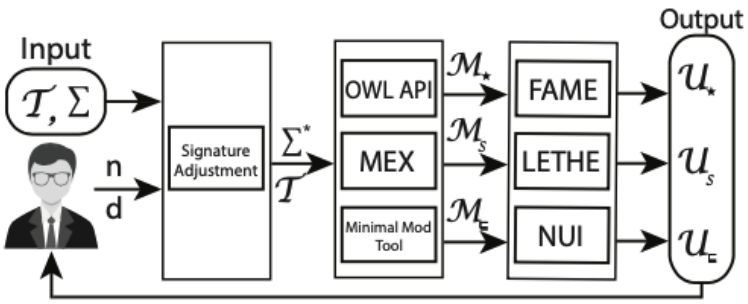


Figure 3.3: Workflow for computing uniform interpolants for the adjustment Σ_* of the signature Σ [CAS+19b].

the signature while removing any other concept names and roles. The issue with ontology modules is that they contain a high number of symbols that are not part of the intended signature specified as input. Current forgetting methods, on the other hand, suffer when applied directly to large ontologies with small signatures. Due to the fact that we will use SNOMED-CT in this thesis, neither of these approaches is good for us. Thus, we will focus on another work that tackles these problems.

Chen et al. [CAS+19b] found a way to overcome this issues by proposing a method that combines forgetting and modularity, and it is demonstrated to be a viable technique to compute subontologies that only include terms of relevance to the developer or end-user.

The authors used three types of modules for modularization: locality-based, semantic, and minimum subsumption modules, as well as three tools for forgetting: NUI, LETHE, and FAME. For evaluations SNOMED-CT and NCIt [6] ontologies are used. We have already discussed SNOMED-CT in Section 2.1.2, whereas NCIt (National Cancer Institute Thesaurus) is a reference terminology that covers a wide range of cancer-related disorders, findings, and anomalies.

An evaluation on standard concept name lists showed that precompiling ontology modules minimizes the number of concepts that must be forgotten by the forgetting tools, which allows then the tools to be feasible due to the size of the ontology being significantly smaller.

The technique presented in this work uses an iterative process consisting of four stages: extension of the given signature and, if necessary, partitioning, modularization, forgetting, and domain expert evaluation. The domain expert input is used for signature extension and according to their experiments, there is no need for the iterative process of the domain expert giving feedback in the SNOMED-CT setting. The workflow for computing subontologies from a bigger ontology is shown in Figure 3.3.

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

To begin, a signature Σ is provided as well as a large ontology τ, from which we want to extract a smaller ontology. Signatures often include particular concept names of relevance to the user as well as a list of concept names that already exist in the domain. For adjusting these two types of signatures, signature extension and signature partition are used. When given a small random signature, minimal subsumption modules, star modules, and semantic modules typically show empty modules; hence, signature expansion with roles and their target concepts is required. Furthermore, if there is a signature with a high number of concepts, a smaller signature is required. Following signature adjustment, the second phase is modularization, which is followed by the use of the three previously described forgetting tools in such modules. Finally, the results are presented to the domain expert, who provides feedback.

In this thesis, for extracting our real-life domain ontologies we will use a tool from github, which is based on the ideas presented in the paper discussed above, due to its benefits over modularization and forgetting tools on their own.

CHAPTER 4

Using domain knowledge in Machine Learning

In the rest of this chapter we discuss several attempts to incorporate domain knowledge at different stages of the machine learning pipeline such as pre-processing, feature extraction and selection phase, in order to improve the performance or reduce the cost of machine learning methods. Moreover, we discuss the literature that uses ontologies to explain black box models, with a particular focus on Trepan Reloaded, which will be discussed in details in Section 4.2.

4.1 Different ways of using ontologies to enhance machine learning process

For many years, researchers have been looking for ways to incorporate domain knowledge from ontologies into machine learning steps. It is not yet a well established field and we could not find a systematic discussion of these approaches, nevertheless, we mention some existing approaches in the literature that try to improve performance or lower training costs of machine learning process by incorporating domain knowledge.

In the paper [SMDE19] authors suggested a way for automatically generating features for the Machine Learning algorithm using ontologies. The features are created by integrating the existing concepts and relationships in the knowledge base, which are expressed as an ontology. The technique can create new definitions that are subsequently used as features by using "Expansion of features" and "Generation of derived class expansions" algorithms shown in the paper. Authors compared this method with traditional approach using several classifiers in Activities of Daily Living (ADLs) Recognition Using Binary Sensors Data Set [OdTS13] and it was shown that this method improved performance by 1.9 percent on average.

Another work in this direction is the system called SemML by Svetashova et al. [SZP+20]. SemML is a system that uses ontologies to enhance machine learning processes. Ontology Extender, Domain Knowledge Annotator, Machine Learning Annotator, and Ontology Interpreter are four semantic components proposed by the authors to improve the welding quality monitoring process. The Bosch use-case of electric resistance welding was studied, and it was found that this system helps in overcoming communication and data integration problems.

Zhang et al. [ZSH02] presented a decision tree method modification that may leverage user-supplied ontologies to infer classification rules at higher levels of abstraction. The authors looked into decision tree learning, in which each attribute is connected with a single hierarchically structured ontology and each instance is labeled with one of m disjoint class labels. Whereas a standard decision tree algorithm recursively selects an attribute from a set of candidate attributes based on an information gain criterion at each step, in an ontology-driven decision tree algorithm, since each attribute has a hierarchically structured taxonomy over possible values of the attributes associated with it, the learning algorithm must choose not only a specific attribute, but also an appropriate level of abstraction in the taxonomy. The authors offer some preliminary findings to show that the suggested method is feasible.

Knowledge can be also used in pre-processing, feature extraction and selection phases as was done in the paper [Van17]. The authors proposed a hybrid method (data-driven and knowledge-based) for white box machine learning in order to obtain a more effective and less expensive training phase. In order to accomplish so, domain knowledge was included into the pre-processing, feature extraction, and selection phases. They suggested a mechanism for class balance as pre-processing step. The authors combined SMOTE or any other sampling technique's beneficial properties with a knowledge-based sampling method. For automatically feature discovery, the authors mapped the entities as URIs, which correspond to nodes in a graph of linked data, and then tried to find new features by doing a breadth first search in a graph of linked data, with the condition that the new candidate must be informative, which means it must be correlated with the target and have a low number of missing values. And for feature selection, the authors presented a technique for describing knowledge as a graph, with nodes representing features and edges expressing relationships between them. After sorting the features using a ranking algorithm, the top k features are selected. This technique has a lower computational cost than previous feature selection algorithms since it is dependent solely on the number of features rather than the number of data samples, which can get very large in many cases. Moreover the authors proposed a hybrid method for primary headache diagnosis, in which they developed a mobile headache journal, which is a mobile application that allows patients to record their headache attacks and the medications they've taken, and then use this information, along with background knowledge, to generate a decision tree that can help an expert make the right diagnosis. No evaluation is done in the paper.

In addition to improving performance or reducing the cost of machine learning operations, ontologies may be used to explain black box models. Below are some of the contributions

made in this direction.

Panigutti et al. [PPP20] introduced Doctor XAI, a model-agnostic explainability technique that can cope with multi-labeled, ontology-linked data. The idea is to predict the patient's next time of visit, given the clinical history of the patient.Usually to do so a deep learning model is employed, however due to the lack of understandability of such models, the use and its acceptance is low. Doctor XAI focuses on providing local explanations for a classification without using any internal parameter of the black box model to generate the explanation. It begins by selecting neighbor patients while looking at semantically similar patients using ontology-based similarity metrics, then they generate some synthetic patients using the ontology, and label the synthetic patients using black box models. These synthetic neighbors are then used to train a decision tree. The explanation is shown as decision rules. Moreover, the authors demonstrated that incorporating an ontology into the explanation process improves the fidelity of the interpretable model, which measures how well the interpretable model mimics the behavior of the black box.

4.2 Trepan Reloaded

Confalonieria et al. [CWBdPM21] made a contribution in the direction of using ontologies to explain black box models. They introduced a version of Trepan [CS95] called Trepan Reloaded. Trepan algorithm was introduced in 1995 and it is an approach that extracts concept description in the form of decision trees from a trained network as an inductive learning problem. It varies from other inductive learning problems in that an *oracle*, which may be the model or the network itself, can be used to answer questions during the learning process. The function represented by the network is the target concept in Trepan, and the hypothesis created by the learning method is a decision tree that approximates the network [CS95].

Trepan Reloaded is an extension of Trepan that includes ontologies that model domain knowledge in the process of extraction explanation to improve their understandability.

In their experiments, the authors used two datasets: Loan Prediction Problem Dataset [1] and Cleveland Heart Disease Data Set [2]. Moreover, the authors hand crafted two ontologies for the corresponding domains. The workflow of the Trepan Reloaded algorithm is shown in Figure 4.1

The datasets are first preprocessed, with missing values being handled and the datasets being normalized and encoded. The feed-forward model is then trained using the train-test split method. Then each feature of the dataset is manually mapped to a concept or role name from the ontology. Then the information content is calculated for every concept or role name.

The authors hypothesized that features would be easier to understand if they were linked to more general concepts in the ontology, therefore they looked into the concept of

[1] https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset
[2] https://archive.ics.uci.edu/ml/datasets/heart+disease

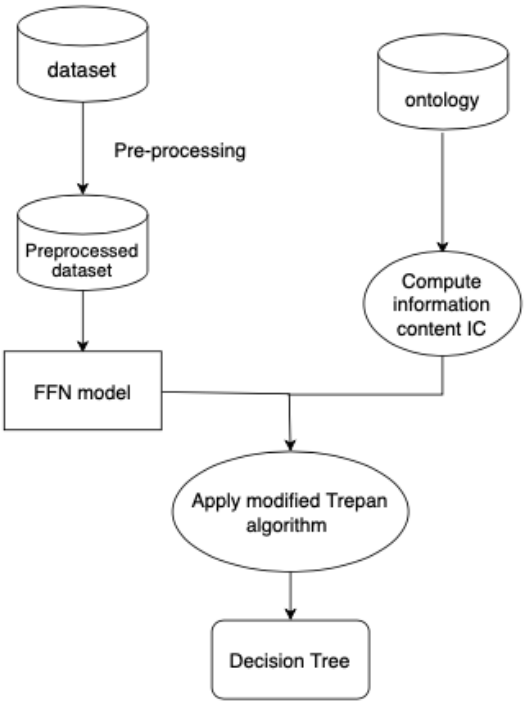


Figure 4.1: Trepan Reloaded algorithm workflow.

information content, which evaluates a concept's semantic generality or specificity based on *refinement operators*.

Refinement operators are well-known in Inductive Logic Programming. In this setting specialisation refinement operators are used, which take a concept C as input and return a set of descriptions that are more specific than C by taking into account a TBox $\mathcal{T}$.

The formula 4.1 is used to calculate the information content.

$$\mathrm{IC}(X_i) := \begin{cases} 1 - \frac{\log(|\mathrm{subConcepts}(X_i)|)}{\log(\mathrm{sub}(\mathcal{T})|)} & \text{if } X_i \in \mathrm{sub}(\mathcal{T}) \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Where $subConconcept(X_i)$ is the set of specialisations for X_i and $sub(\mathcal{T})$ is the set of subconcepts that may be constructed from axioms. After the information content is calculated for each concept and role name, the Trepan Reloaded algorithm is applied to extract the decision trees. The only difference from Trepan algorithm, which is shown in Algorithm 2, is that when training the decision trees, instead of information gain, the split is done based on the modified information gain which is calculated using the information content by the Equation (4.2). Finally from this algorithm the decision tree is extracted from the trained network.

$$\text{IG'}(X_i, S|\text{IC}) := \begin{cases} (1 - \text{IC}(X_i))\text{IG}(X_i, S) & \text{if } 0 < \text{IC}(X_i) < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

In this thesis for the sake of simplicity we will use Equation (4.3)

$$\text{IG'}(X_i, S|\text{IC}) := \begin{cases} (1 - \text{IC}(X_i)) \cdot \text{IG}(X_i, S) = \frac{\log(|\text{subConcepts}(X_i)|)}{\log(\text{sub}(\mathcal{T})|)} \cdot \text{IG}(X_i, S) & \text{if } 0 < \text{IC}(X_i) < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

And we will use IC' instead of IC:

$$\text{IC'}(X_i) := \begin{cases} \frac{\log(|\text{subConcepts}(X_i)|)}{\log(\text{sub}(\mathcal{T})|)}) & \text{if } X_i \in \text{sub}(\mathcal{T}) \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Algorithm 2 Trepan (Oracle,Training,Features)[CWBdPM21].

```
Priority queue Q ← ∅
Tree T ← ∅
use Oracle to label examples in Training
enqueue root node into Q
while nr_internal_nodes < size_limit do
    pop node n from Q draw and store examples_n for n
    store constants_n for n
    use features to build set of candidate_splits
    use examples_n and Oracle(constraints_n) to decide Best_split (from gain ratio)
    add n to T
    for element c ∈ Best_split do
        add c as child of n
        if c is not a leaf according to Oracle(constraints_n) then
            enqueue node c into Q with negative information gain as priority
        end
    end
end
Return T
```

For evaluation of the resulting decision trees, the authors measured accuracy and fidelity. Based on their results, they showed that in both heart and loan domains, there is a small drop on fidelity, as well as accuracy, of trees extracted from Trepan Reloaded compared to trees extracted from Trepan. Nevertheless, the authors showed that even that Trepan Reloaded algorithm compromises little on the accuracy, it improves the understandability of the trees. For understandability the authors measured a syntactic complexity, and they performed user-based questionnaires on both domains. Based on the results that

are achieved in [PLGM16], the authors defined the syntactic complexity of decision trees as in Equation (4.5).

$$U(n, b) = \alpha \frac{n}{k} + (1 - \alpha) \frac{b}{k^2} \tag{4.5}$$

where b is number of branches, n is the number of leaves, $k = 5$ coefficient of the linear regression and $\alpha \in [0, 1]$ is a tuning factor that adjust weight of n and b.

Whereas from user-based questionnaires, the authors measured the correct responses, the response time for each questions, confidence and user understandability.

Based on the results from user-based questionnaires and the syntactic complexity, the authors fitted a mixed-effect logistic regression model [BDB08], where they showed that for trees with bigger syntactic complexity, are harder to answer correctly. Moreover, they showed that people answered correctly and in less time to questions where the ontology was present in the tree-building process. Another noteworthy finding, is that users find decision trees build with the presence of the ontology more understandable.

In our work, we will test Trepan Reloaded algorithm with a variety of datasets in the medical domain, as well as with ontologies extracted from existing high-quality sources. We will also test the Trepan Algorithm with several heuristics from ontologies. Furthermore, we will evaluate the improvement using a more accurate set of metrics.

CHAPTER 5

Domain knowledge for better decision trees

In this chapter we introduce the main contributions of this thesis. We start by describing the ontology extraction workflow for seven domains that we have extracted. Then, we discuss the improvements to the Trepan Reloaded approach as well as the improvements to the decision tree algorithm. In particular, we introduce a variant of hubscore with distance that we have used as a heuristic from the ontologies.

5.1 Extracting Ontologies

As we mentioned in Chapter 3 our motivation is to rely on existing ontologies, to extract knowledge from them and use it to possibly improve some learning algorithms. One of the challenges is that there are no specific ontologies for the domains that we consider, and that this knowledge is often contained in larger ontologies. In our thesis to overcome this problem we used the methodology explained in Section 3.2 for extracting the specific ontologies for the domains that we used. We extracted seven ontologies from SNOMED-CT with the corresponding signatures, which we choose to be the corresponding names of the features from the datasets.

The seven datasets that we found for the domains that we analyzed are: Heart Disease dataset (https://archive.ics.uci.-edu/ml/datasets/heart+disease), Breast Cancer dataset (https://archive.ics.uci.edu/ml-/datasets/breast+cancer), Chronic Kidney Disease dataset (https://www.kaggle.com/-mansoordaku/ckdisease), Hepatitis dataset (https://www.kaggle.com/codebreaker619/-hepatitis-data), Diabetes dataset (https://www.kaggle.com/mathchi/diabetes-data-set), Lung Cancer dataset (https://www.kaggle.com/mysarahmadbhat/lung-cancer) and Stroke Prediction dataset (https://www.kaggle.com/fedesoriano/stroke-prediction-dataset). The details of the dataset are shown in Table 5.1.

Domains	Attributes	Instances	Attribute Charact.	Missing values
Heart Disease	14	303	Categorical, Numeric	Yes
Breast Cancer	9	286	Categorical	Yes
Diabetes	9	768	Numeric	No
Hepatitis	20	155	Categorical, Numeric	Yes
Stroke Prediction	12	5109	Categorical, Numeric	Yes
Chronic Kidney Disease	26	400	Categorical, Numeric	Yes
Lung Cancer	16	309	Categorical, Numeric	No

Table 5.1: Details of the datasets used

The workflow of the ontology extraction from SNOMED-CT having the datasets is shown in Figure 5.1. We wanted the ontologies to represent the same features as the dataset has, therefore we began by looking into the SNOMED-CT for the concepts that represent corresponding features or the concepts that best explains our dataset's features.To find the corresponding concept we searched for SNOMED-CT concepts in bioportal [1] as shown in Figure 5.2. Since every concept or relationship in SNOMED-CT has a unique integer identifier, we found these identifiers and defined the signature list. In cases where we could not find some feature from the dataset in SNOMED-CT we looked for other concepts or roles with closely related meaning.

We did this for each of the seven domains and provided these signatures as well as the SNOMED-CT ontology to the tool explained in Section 3.2, in order to extract ontologies that are as similar to the datasets' features as possible. Table 5.2 shows the sizes of the signatures used to extract the ontologies for each domain, as well as the details of the generated ontologies.

Moreover, besides for the extractions of these ontologies, for the breast cancer and heart disease domains, we additionally obtained ontologies in two other ways:

- Hand-crafted, with we will refer to as *small* ontologies, similarly as the authors have done it in Trepan Reloaded [CWBdPM21], starting from scratch and adding features of the datasets as concepts or roles without the use of any domain expert. An example of extended ontology is shown in Figure 5.3.
- Extended two existing ontologies, Heart Failure Ontology [2] and Breast Cancer Grading Ontology [3] with the datasets' features as concepts or roles that do not exist in the existing ontology. We will refer to these extended ontologies as *extended*. An example of extended ontology is shown in Figure 5.4, where the classes, object

[1] https://bioportal.bioontology.org/ontologies/SNOMEDCT?p=classesconceptid=429740004
[2] https://bioportal.bioontology.org/ontologies/HFO
[3] https://bioportal.bioontology.org/ontologies/BCGO

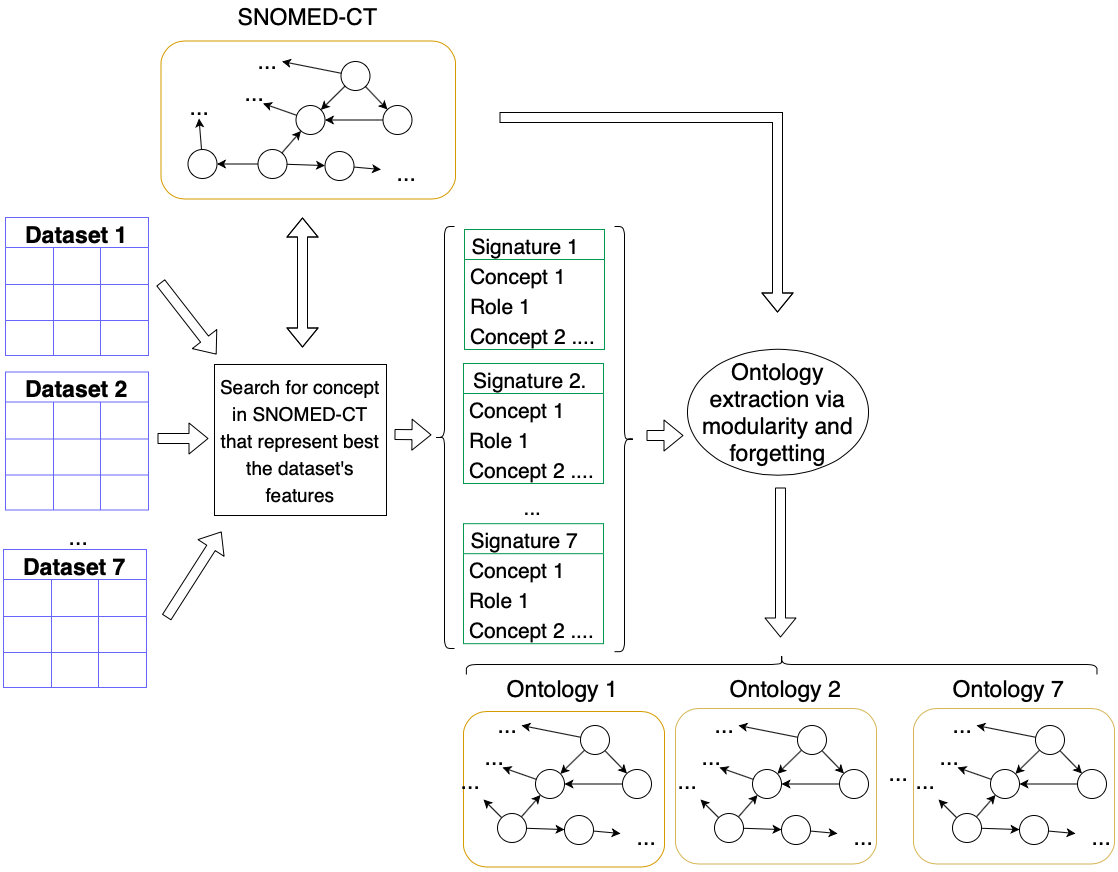


Figure 5.1: The workflow of the ontology extraction process.

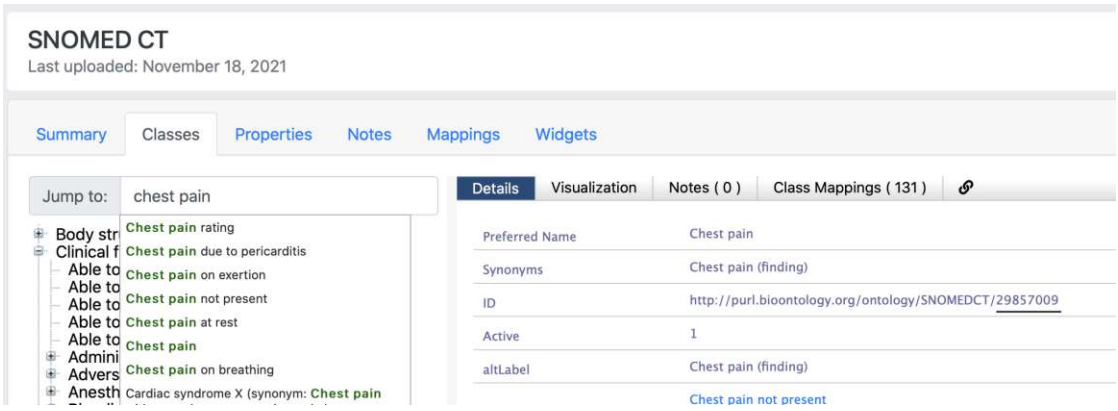


Figure 5.2: An example of searching for concepts in SNOMED-CT

	Input Signature	Extracted ontology		
Domains	**Size**	**Logical axioms**	**Classes**	**Object properties**
Heart	14	69	53	15
Breast	9	26	27	6
Diabetes	9	34	35	10
Hepatitis	18	94	92	11
Stroke	12	42	40	12
Kidney	25	90	87	20
Lung	17	53	51	9

Table 5.2: Sizes of the signatures used for ontology extraction from SNOMED-CT and details of the generated ontologies

Ontologies	Details	Logical Axioms	Classes	Object prop.
Small	Heart Disease	66	29	3
	Breast Cancer	49	23	2
Extended	Heart Disease	2107	1664	3
	Breast Cancer	126	69	17

Table 5.3: Details of the ontologies created and extended for heart and breast cancer.

properties and data properties in red are added to the existing Breast Cancer Grading Ontology.

The reason of using domain knowledge from different ontologies is to compare the results with those in Trepan Reloaded paper, and also compare between hand-crafted and extended ontologies that are done without the use of domain experts' knowledge with high-quality existing ontologies like SNOMED-CT. The details of these ontologies for heart and breast cancer domains are shown in Table 5.3.

Because there are few benchmarks connecting machine learning datasets and ontologies, developing such a test set with real-life medical ontologies is a significant contribution. It allows us to evaluate the incorporation of knowledge from domain ontologies in the decision tree building process from classical decision tree algorithms, as well as the trees extracted using the Trepan method.

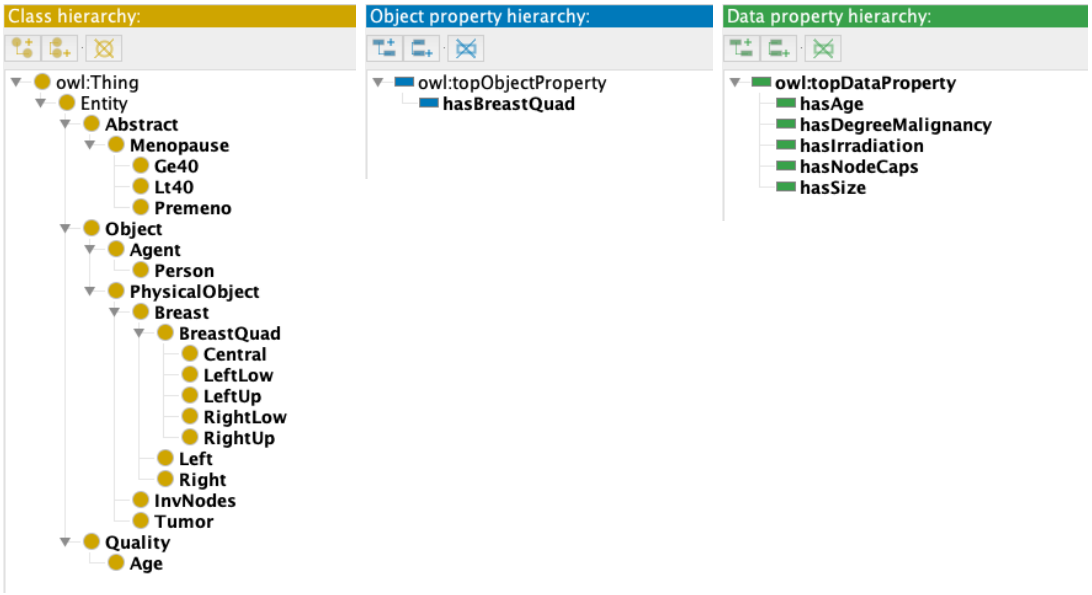


Figure 5.3: An example of an hand-crafted ontology with concepts, object properties and data properties.

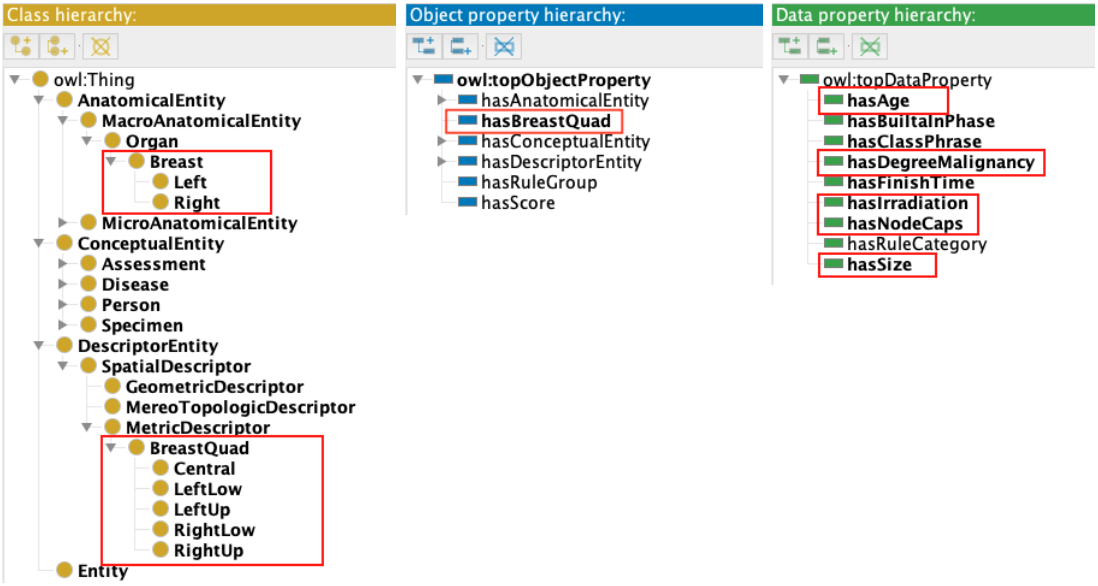


Figure 5.4: An example of extending an ontology with concepts, object properties and data properties, where red boxes are the extensions of Breast Cancer Grading ontology.

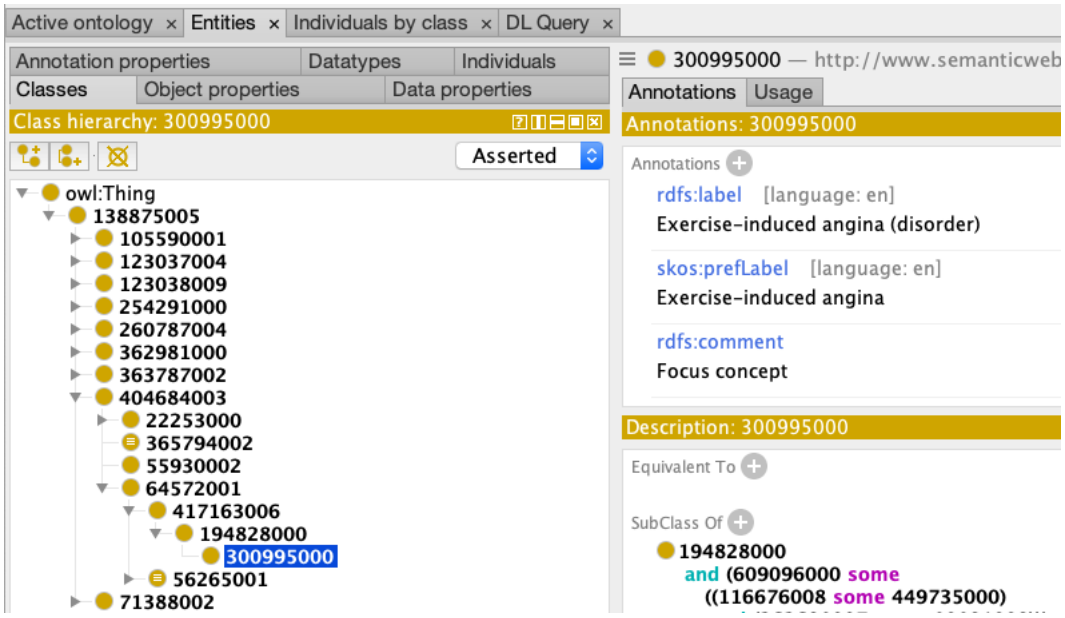


Figure 5.5: Example when the information content of a feature is zero.

5.2 Improvement to Trepan Reloaded

In this thesis, we used Trepan algorithm to generate decision trees from a neural network. In addition to [CWBdPM21], beside the heart domain that is used in the paper, we evaluated six additional medical domains with real-life ontologies, in order to investigate if their hypothesis that decision trees are more understandable if they link to more general concepts existing in an ontology is valid for other domains as well.

Due to the way the information content is calculated from the ontology in Equation (4.1), for the ontologies extracted from SNOMED-CT, the information content won't work, because the concept from signature would be placed in the lowest hierarchy in the resulting ontology from SNOMED-CT, thus the formula for calculating the information content would give us zero for every concept. An example is shown in Figure 5.5, where we can see that the concept "Exercise-induced angina" has no sub-concepts thus the information content would be 0.

Therefore, in addition to information content, we propose and modify another heuristic called *hubscore*, in a way that preserve the spirit of the ideas of the information content and can still give you a decent result if you use an off-the-shelf ontology. Furthermore, we used various combinations of these two strategies. We will go into further details about hubscore in Section 5.4.

Moreover, for two of the domains, in addition we evaluated trees extracted from Trepan algorithm, when using the heuristics mentioned above from hand-crafted ontologies that we created as well from the existing ontologies that we extended.

5.3 Improvement to Decision Trees algorithm

In addition to using domain knowledge to generate decision trees from a neural network, we used domain knowledge directly when creating decision trees from a classical decision tree algorithm. We used the C4.5 algorithm as explained in Section 2.2.1. In our work when calculating the information gain to find the best split for building the decision tree, we combined it with domain knowledge. We evaluated seven domains with the ontologies that we extracted. As with the trees extracted with Trepan algorithm, to generate decision trees, we used information content from [CWBdPM21], a modification of hubscore (more on hubscore in Section 5.4), as well as combinations of these heuristics.

Moreover, for two of the domains, same as for trees extracted with Trepan algorithm, in addition we evaluated the decision trees when using the heuristics mentioned above from hand-crafted ontologies that we created as well as from the existing ontologies that we extended.

5.4 Hubscore and Relevance-score

Hubscore was introduced by Butt et al. [BHX16] as one of the features of a concept in an ontology, together with the *AuthorityScore*, to determine its rank, known as *DWRank*. The goal was to identify ontologies to represent their data by searching and ranking ontologies based on a specific query term. The authors defined hubscore as a measure of the centrality of a notion within an ontology, which is distinguished by two characteristics:

- Connectivity: A concept is more central to an ontology, if there are more intra-ontology relationships originating from the concept.
- Neighbourhood: A concept is more central to an ontology, if there is an intra-ontology relationship connecting it to another central concept.

An intra-ontology relationship $I_a = ((v, u), O)$ is a directed edge (v, u), where $(v, u) \in E(O)$ for $v \in V(O)$, $u \in V(O)$ and $O = (V, E)$, V being a finite set of vertices, E being a set of edges and O being an ontology as graph based formalisation.

To give a formal definition of hubscore the author firstly define forward link concepts and back link concepts as:

- Forward link concepts $C_{FLink}(v, O)$ is a set of concepts V' in an ontology O, where $V' \subset V(O)$ and $\forall v_i \in V', \exists (v, v_i) \in E(O)$
- Back link concepts $C_{BLink}(v, O)$ is a set of concepts V'' in an ontology O, where $V'' \subset V(O)$ and $\forall v_j \in V'', \exists (v_j, v) \in E(O)$

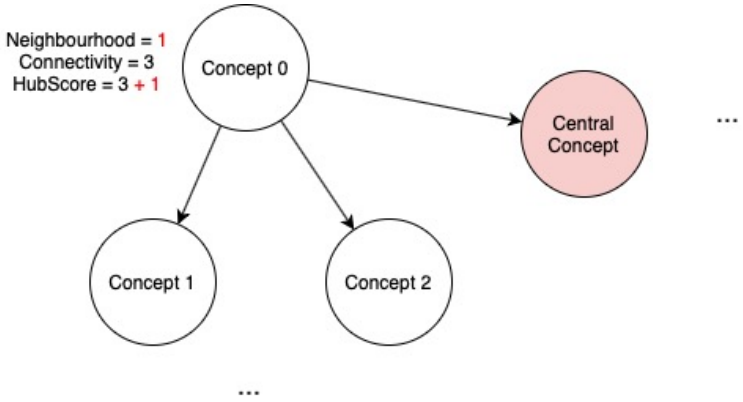


Figure 5.6: Example of hubscore calculation

The hubscore function formally is defined iterative function at any iteration k as:

$$h_k(v, O) = \sum_{v_i \in C_{FLink}(v,O)} \frac{h_{k-1}(v_i, O)}{|C_{BLink}(v_i, O|} \tag{5.1}$$

An example of hubscore calculation is given in Figure 5.6, where we want to calculate the hubscore of concept 0. We can observe from Figure that concept 0 is related to three other concepts, so the connectivity is three, and the neighbourhood is one since it has a relation to another central concept, therefore the hubscore of concept 0 is four.

Algorithm 3 The pseudocode of calculating hubscore with distance (relevance-score)

1. For each concept or role i in the ontology
   Calculate hubscore(C_i) as the sum of connectivity(C_i) and neighbourhood(C_i).
2. Define central concepts as concepts or roles with non minimal hubscore
3. Scale every concept's hubscore from 0 to 1
4. For every C_i concept or role
   if some hubscore_ancestor$_j$(C$_i$) is bigger then hubscore(C_i) then
   let distance(C_i,ancestor(C_i)) be the distance of the concept C_i to that ancestor in the hierarchy
   assign C_i with the hubscore of hubscore_ancestor$_j$(C$_i$) divided by the distance(C_i,ancestor(C_i))

We used hubscore to measure the degree of centrality of a concept. In addition to the calculations from [BHX16] we added one more constraint to the measurement and defined it as *relevance-score*. We calculate the hubscore for each concept and role in the ontology that is represented as a feature in the domain dataset. Then, as a second phase, we select central concepts with the highest hubscore or with a non-minimal hubscore. Furthermore, we scale each concept's hubscore from, and then for those with the lowest hubscore, we calculate the distance to the closest central concept in the hierarchy, and then assign the

hubscore to that concept or role as the hubscore of the closest central concept divided by the distance to that concept in the hierarchy. The reason for dividing hubscore by the distance is that hubscore alone indicates centrality of the concept and usually in ontologies extracted from real-life the concepts from the signature show up in the end of the hierarchy and most of these concepts are subclasses of central concepts. The pseudocode of calculating relevance-score is shown in Algorithm 3.

From now on, we will use *IG* for information gain, *RS* for relevance-score, and *IC* for information content.

CHAPTER 6

Experimental evaluation

In this part, we cover our experimental evaluation. We begin by discussing the methods used in our experiments, as well as the metrics that we used to evaluate our experiments, then we go into more details about the two methods that we have used to evaluate the understandability of learned decision trees. Finally, we discuss the results of our experiments.

6.1 Methods

In our experiments, in order to extract decision trees, we trained a Feed Forward Network implemented in *PyTorch*. We generated three trees for every domain using the domain knowledge extracted from SNOMED-CT ontology, one Trepan tree that does not use any domain knowledge, one tree where the $IG' = IG * RS$ and one tree where the $IG' = RS$. Beside those three trees, for heart disease and breast cancer domains, we constructed six more trees when using small ontology and six more when using extended ontology. Those six trees differ from each other the way information gain is calculated, we distinguish: $IG' = IG * IC$, $IG' = IG * RS$, $IG' = IC$, $IG' = RS$, $IG' = RS * IC$, $IG' = IG * IC * RS$. For a better understanding we have shown in Table 6.1 for every domain which trees we have constructed with different heuristics from different ontologies.

In addition, we used these heuristics from ontologies to build the decision trees from the C4.5 decision tree algorithm, not only the trees extracted from Trepan algorithm.

For evaluating and comparing trees we used accuracy and fidelity. Accuracy in classification tasks is the proportion of correct predictions made by our model compared to the total number of predictions as shown in the Equation (6.1), whereas fidelity is defined as the percentage of examples on which the classification by the surrogate, in our case the build decision tree, agrees with that by the initial model, which in our case is the neural network.

	Trep.	Small						Extended						Snomed	
Domains	IG'=IG	IG'=IG*IC	IG'=IG*RS	IG'=IC	IG'=RS	IG'=RS*IC	IG'=IG*IC*RS	IG'=IG*IC	IG'=IG*RS	IG'=IC	IG'=RS	IG'=RS*IC	IG'=IG*IC*RS	IG'=RS	IG'=RS*IC
Heart	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Breast	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Diabetes	✓													✓	✓
Hepatitis	✓													✓	✓
Stroke	✓													✓	✓
Kidney	✓													✓	✓
Lung	✓													✓	✓

Table 6.1: Trees constructed for each domain with different heuristics from various ontologies.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{6.1}$$

$$\text{Fidelity} = \frac{\text{Number of surrogate predictions that agree with initial model's predictions}}{\text{Total number of predictions}} \tag{6.2}$$

In order to evaluate the understandability of learned decision trees, we used two methods based on:

- Syntactic complexity of a decision tree.
- User-based questionnaires, in which participants rate the trees based on their understandability.

6.1.1 Syntactic Complexity

There has been a lot of research towards analyzing which syntactic properties of decision trees correlate with and may act as proxies for their interpretability. Some of these properties are the number of nodes, maximum/average depth, number of leaves, branching factor and so on. For evaluating the syntactic complexity of our resulting trees we used two syntactic measures.

The first measure that we used is by Gaines et al. [Gai96]. The authors suggested a method for measuring syntactic complexity in order to evaluate the understandability of

production rules, decision trees, and exception rules. The authors believed that properties such as the number of nodes N, number of leaves (final nodes) F, number of arcs A and number of clauses C, are the most important properties that influences the complexity of the structure of the decision tree. As a result, the authors devised the complexity measure as:

$$complexity := \frac{N + 2E + 3C}{5} \tag{6.3}$$

where E is the number of the so-called *excesses* calculated as follows:

$$E := A + F - N \tag{6.4}$$

The second measure that we used for evaluating understandability of decision trees using syntactic complexity is shown in [CWBdPM21]. Based on the results in [PLGM16], the authors defined the syntactic complexity of decision trees as:

$$U(n, b) := \alpha \frac{n}{k} + (1 - \alpha)\frac{b}{k^2} \tag{6.5}$$

where b is number of branches, n is the number of leaves, $k = 5$ and $\alpha = 0.5$. The value k is a coefficient of the linear regression from [PLGM16] and $\alpha \in [0, 1]$ is a tuning factor that adjust the weight of n and b and by taking $\alpha = 0.5$ we give the same weight to n and b, as the authors of Trepan Reloaded did.

6.1.2 User-based questionnaires

The second method that we used for evaluating the understandability of the decision trees are user-based questionnaires. We created three online questionnaires using an online software called soscisurvey [1]. The questionnaires contain questions about four out of the seven domains we analyzed. Two questionnaires are in the heart disease and breast cancer domains, whereas the third questionnaire contains questions regarding chronic kidney disease as well as hepatitis disease. The three questionnaires follow the same structure as the one showed by Confalonieri et al. [CWBdPM21].

Every questionnaire begins with a video that explains what the users should do in the questionnaire. The participants are then asked some questions regarding their age, gender, education, and their familiarity with decision trees. Furthermore, each questionnaire contains four tasks: classification, inspection, comparison and empowerment as it was done first by Piltaver et al. [PLGM16] and then modified by Confalonieri et al. [CWBdPM21].

- In the *classification* task, participants were provided with an attribute-value table and a decision tree, and were asked three questions: firstly to classify the instance shown in the table using the decision tree provided, then to select how confident

[1] https://www.soscisurvey.de

	Trep.	Small	Extended		Snomed
Domains	IG'=IG	IG'=IG*IC	IG'=IG*IC	IG'=IG*RS	IG'=RS*IC
Heart	✓	✓			✓
Breast	✓	✓	✓	✓	✓
Kidney-Hepatitis	✓				✓

Table 6.2: Decision Trees shown in each questionnaire.

you are with your answer and the third question is to answer how understandable you found that decision tree. An example of a classification task with its questions is shown in Figure 6.1.

- In the *inspection* task, participants were presented with a sentence and a decision tree, and were asked to determine whether the sentence is true. Moreover, the same two questions as in the classification task about the confidence on the answer and to rank the understandability of the decision tree were asked. An example of an inspection task with its questions is shown in Figure 6.2.
- In the *comparison* task, participants were provided with two trees per question, one tree extracted by Trepan without any ontology knowledge, and the other one using any of the heuristics calculated from SNOMED-CT, Small or Extended ontology. Participants were asked to select which tree they find more understandable and rate it on a five-point scale. An example of a comparison task with its questions is shown in Figure 6.3.
- In the *empowerment* task, participants were provided with a decision tree and a specific statement, and they had to decide if there is any action that they could take to change the decision outcome, and if there is, they had to specify it. An example of a empowerment task with its questions is shown in Figure 6.4.

For the sake of simplicity for each task we showed at most six trees. We showed trees extracted using Trepan (without domain knowledge) and trees build using modified information gain $IG' = IG * RS$, where the husbscore is calculated from the ontology extracted from SNOMED-CT. In questionnaires with the heart and breast cancer domain, in addition, questions regarding trees extracted from small ontologies when the information gain is modified with information content as *IG'=IG*IC* were asked. In addition to these trees, for the breast cancer questionnaire we also asked questions for the trees extracted with the modified information gain $IG' = IG*IC$ and $IG' = IG*RS$ where the information content is calculated from the extended ontology. For a better understanding Table 6.2 shows for which decision trees the participants are questioned about in each questionnaire.

For the first two tasks, classification and inspection, we measured:

- Correctness of the response

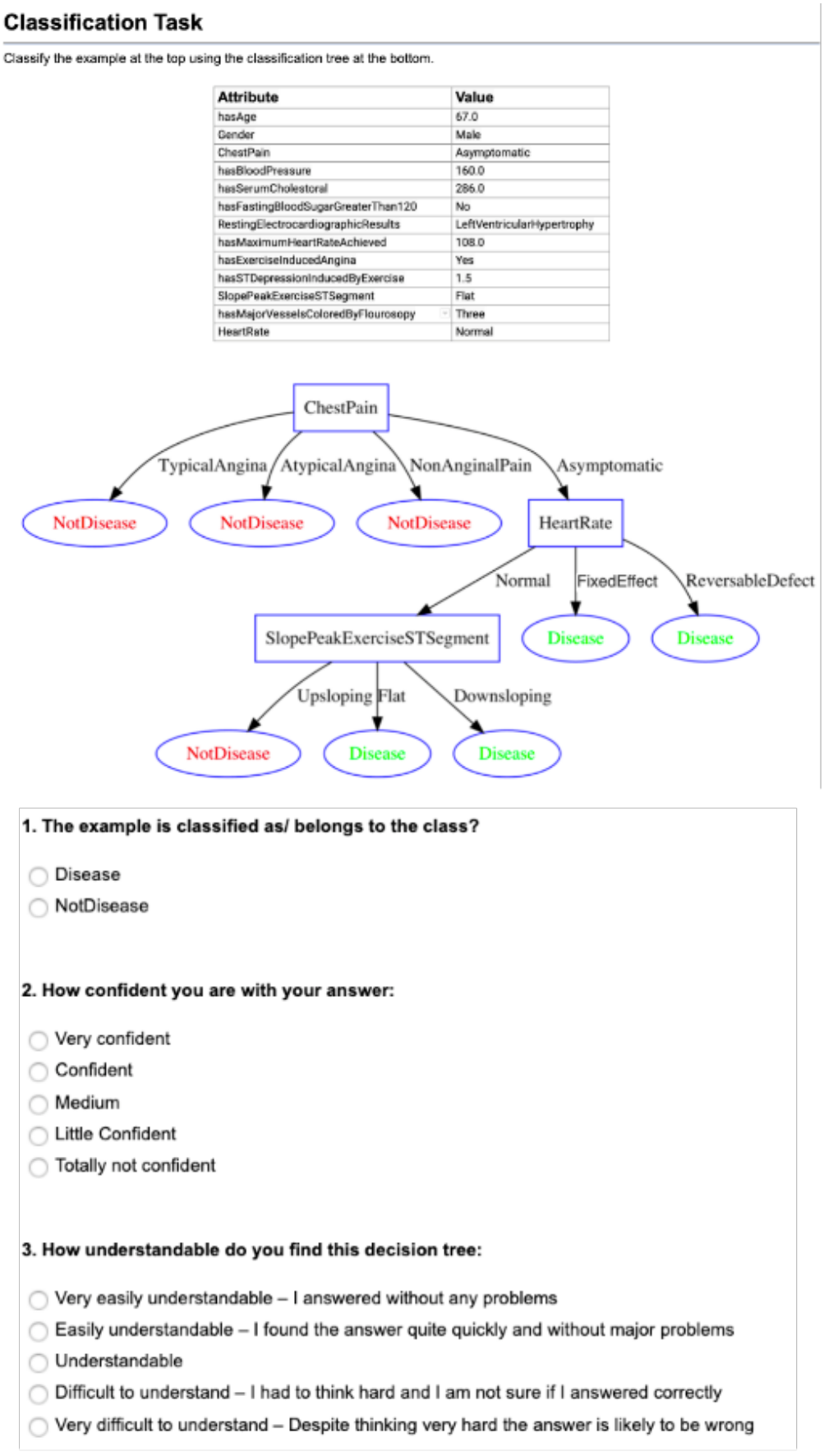
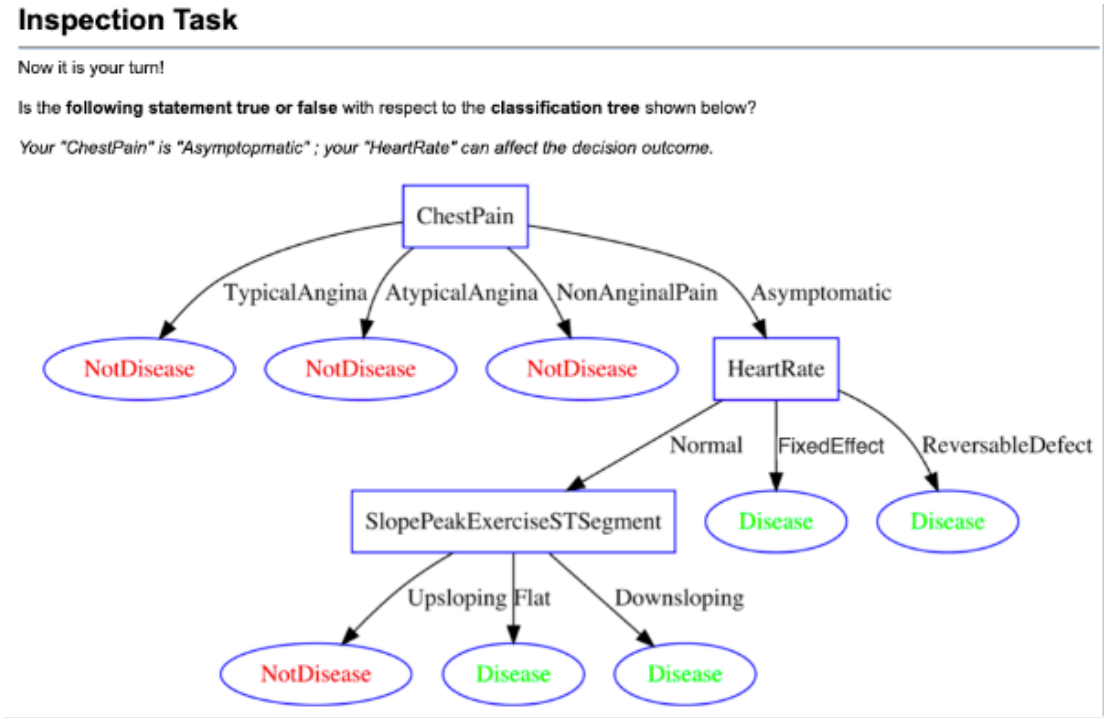


Figure 6.1: Example of classification task.



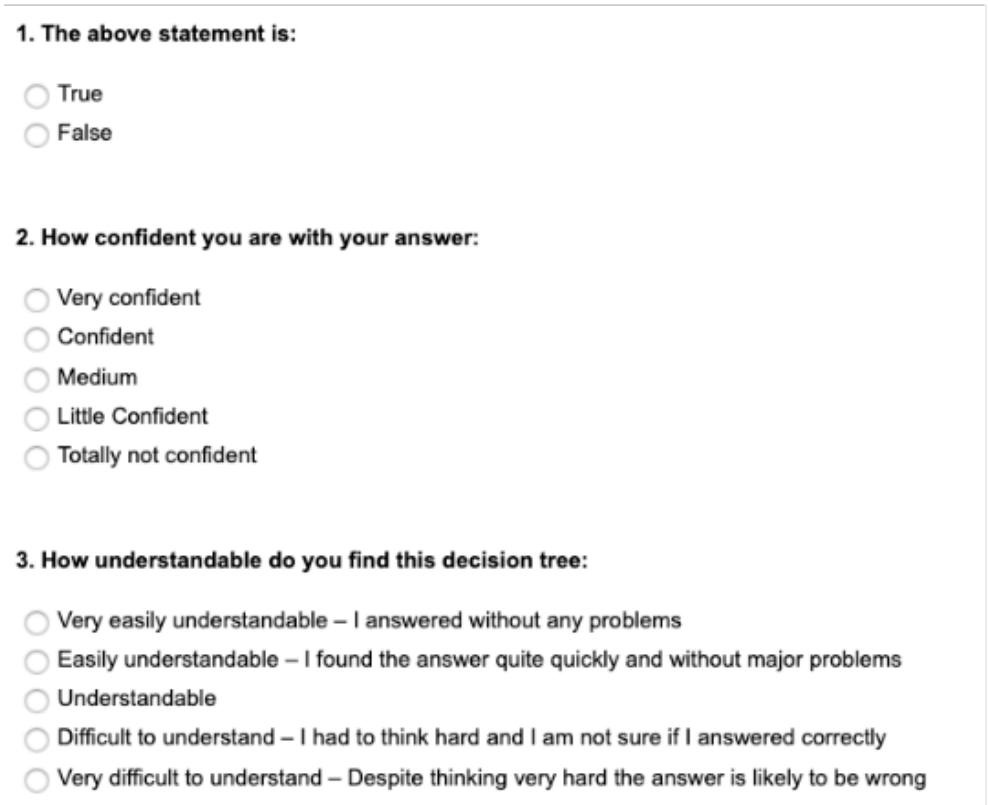


Figure 6.2: Example of inspection task.

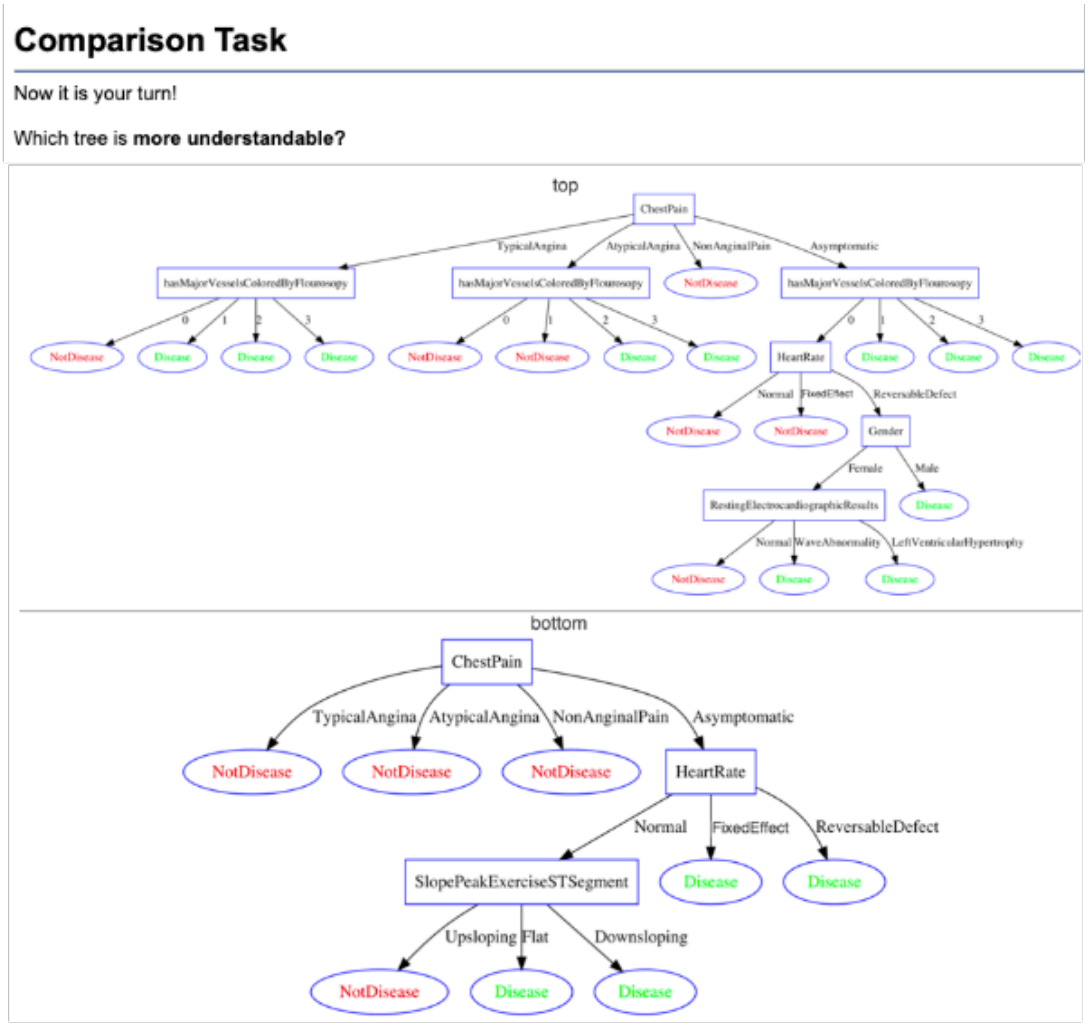


Figure 6.3: Example of comparison task.

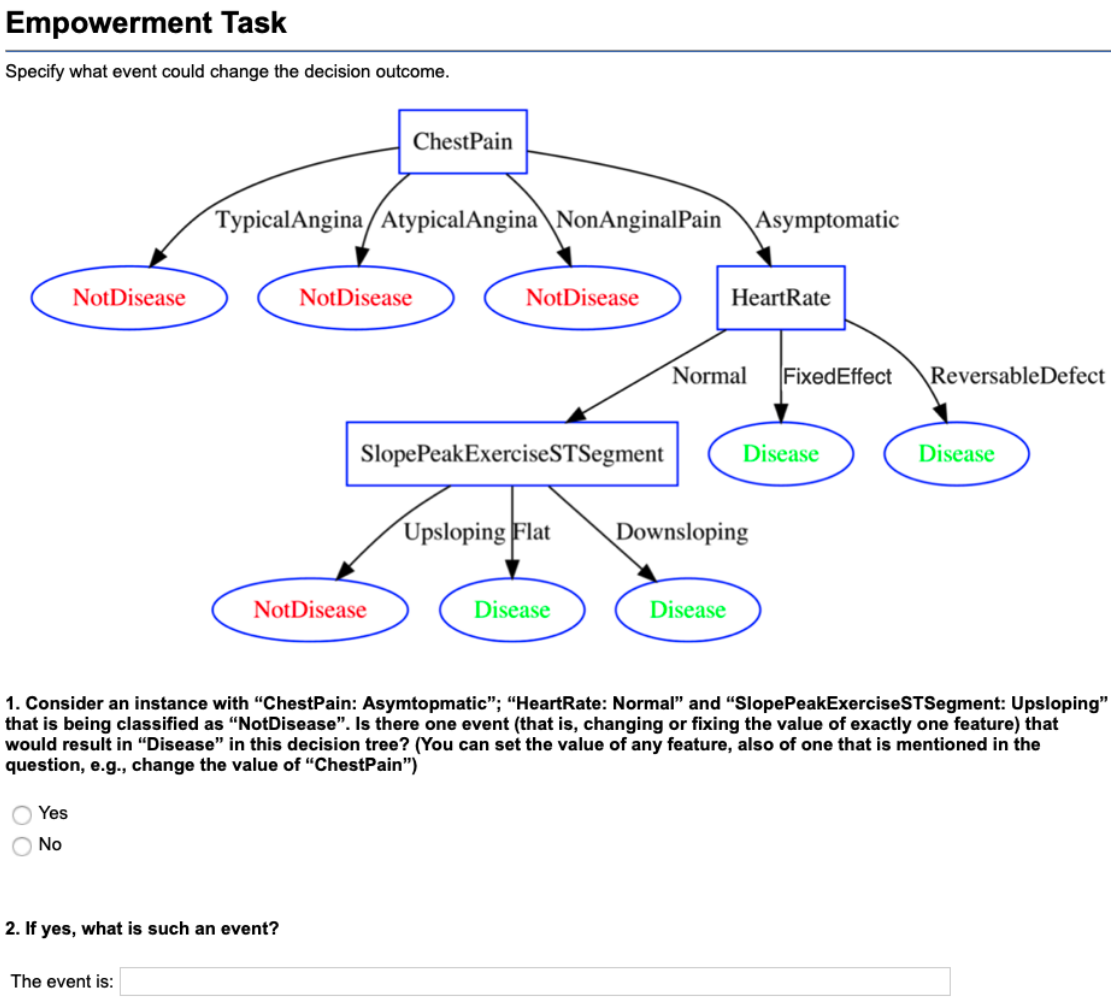


Figure 6.4: Example of empowerment task.

- Confidence in the response
- Response time
- Understandability of the tree

For comparison task, we look at which tree the participants find more understandable. Whereas for the empowerment task we measure correctness of the response as well as the response time.

The questionnaire about heart disease domain had:

- 104 participants in total, 62 females, 38 males and 4 did not give a gender,

- the average age among the participants was 21 years old, the range was from 18 to 51 years old,
- 2 of them have a Ph.D. level of education, 11 a Master degree, 64 a Bachelor degree and the rest had a high-school diploma and
- 57% of the participants were familiar with decision trees.

The questionnaire about breast cancer domain had:

- 75 participants in total, 41 females, 33 males and 1 did not give a gender,
- the average age among the participants was 23 years old, the range was from 18 to 44 years old,
- 1 of them has a Ph.D. level of education, 17 a Master degree, 48 a Bachelor degree and the rest had a high-school diploma and
- 74% of the participants were familiar with decision trees.

The questionnaire about hepatitis and kidney disease had:

- 54 participants in total, 27 females, 27 males and 1 did not give a gender,
- the average age among the participants was 26.6 years old, the range was from 18 to 52 years old,
- 5 of them have a Ph.D. level of education, 21 a Master degree, 25 a Bachelor degree and the rest had a high-school diploma and
- 94.4% of the participants were familiar with decision trees.

Overall, for all three questionnaires we had:

- 234 participants in total, 130 females, 98 males and 6 did not give a gender,
- the average age among the participants was 23.14 years old, the range was from 18 to 52 years old,
- 8 of them have a Ph.D. level of education, 49 a Master degree, 137 a Bachelor degree and the rest had a high-school diploma and
- 70.94% of the participants were familiar with decision trees.

6.2 Results

In this section, we present the results of our evaluation. We evaluated two techniques for building decision trees:

1. directly from classical decision tree algorithm
2. extracted from Trepan algorithm

For each of the two, we considered the accuracy of the trees. For the decision trees extracted with Trepan algorithm we also considered fidelity. Furthermore, as indicated in Section 6.1.1, we provide the findings for syntactic complexity calculated in two different ways. Furthermore, we provide the results from our performed user-based questionnaires, comparing several matrices such as correct responses, time response and understandability, when the ontology is not used and when different heuristics from the ontology are applied, as well as the syntactic complexity of the trees.

6.2.1 Results for direct construction of Decision Trees

In Table 6.3 we show the test accuracy of decision trees for different domains. Moreover, we present the accuracy of such decision trees when the C4.5 algorithm is programmed from scratch [2] and when relevance-score is used to modify the information gain. We used relevance-score calculated by the ontologies that are extracted from SNOMED-CT as we discussed in Section 5.1. We show two forms of using relevance-score from ontologies to modify information gain, the first one is when information gain is multiplied with relevance-score, and the other one is when relevance-score is used instead of information gain in the decision tree construction process.

From Table 6.3 we cannot conclude that using domain knowledge in the decision tree construction helps in increasing the accuracy. For some of the domains such as breast cancer, hepatitis, stroke, kidney and lung disease using relevance-score as information gain outperforms the other methods shown in the table. For these domains using relevance-score instead of information gain increases the accuracy by 7.6%, 15.25%, 0.5%, 9.52% and 13.65% respectively when compared to using C4.5 with default information gain. For the other domains such as heart and diabetes, using information from domain knowledge did not help on increasing the accuracy, where the accuracy drop is significant from 74.6 and 68.5 when using C4.5 algorithm without ontology knowledge to 57.1 and 59.2 respectively, when modifying information gain with ontology knowledge.

In Table 6.4 we show the results for heart and breast cancer domains, when using three different corresponding ontologies: small, extended and SNOMED-CT. From small and extended ontologies we calculated the relevance-score and the information content, whereas for SNOMED-CT using information content has no benefit as shown

[2] https://github.com/dpkravi/DecisionTreeClassifier

Method	Heart	Breast	Diabetes	Hepatitis	Stroke	Kidney	Lung
IG = IG	74.6022	63.0952	68.4809	73.7500	91.56	68.2500	80.8602
IG' = IG*RS	57.0968	**67.8836**	58.7133	77.5000	**92.0336**	**74.7500**	**91.9032**
IG' = RS	57.0968	**67.8836**	59.2310	**85.0**	91.7399	**74.7500**	**91.9032**

Table 6.3: Test accuracy in (%) using decision tree algorithm with and without relevance-score from SNOMED-CT ontologies.

in Section 5.2, thus we use only relevance-score. For small and extended ontologies we also use information content, relevance-score, and the combinations of these heuristics with information gain. From the results on the Table 6.4 we can observe that for heart disease using both relevance-score and information contents alone or multiplying with information gain calculated from extended ontology, gives highest accuracy among other methods for building trees when using heuristics from ontologies, and it increases the accuracy compared to no domain knowledge from ontology is used. On the other hand, for breast cancer domain, except when using information content alone or multiplying with information gain from extended dataset, on any other case accuracy is increased when trees are build with the heuristics from ontologies.

6.2.2 Results for Decision Trees build with Trepan Algorithm

In Table 6.5 we show the test accuracy of decision trees extracted from Trepan algorithm with and without domain knowledge for different domains. We used relevance-score calculated by corresponding ontologies that are extracted from SNOMED-CT as we discussed in Section 5.1. We used relevance-score alone as well as multiplied with the information gain. From Table 6.5 we observe that for heart, breast cancer, stroke and kidney disease using relevance-score during tree construction increases the accuracy by 5.5, 18.2, 0.1 and 2.5 (or 7%, 23.2%, 0.1% and 2.6% increase) respectively. On the other hand for diabetes, hepatitis and lung disease domain, the accuracy stays the same after incorporating relevance-score into the decision tree construction.

In Table 6.6 we can observe that same as for the accuracy, for heart, breast, stroke and kidney using relevance-score calculated by corresponding ontologies extracted from SNOMED-CT increases the fidelity by 13.8, 1.9, 0.1 and 2.5 (or 18.8%, 2.4%, 0.1% and 2.6%) respectively. For diabetes and lung domains the fidelity is the same with and without using relevance-score, whereas for hepatitis disease compared to decision trees constructed without domain knowledge, the fidelity is decreased by 10 when multiplying information gain with relevance-score and by 26.7 when using relevance-score instead of information gain.

In Table 6.7 we show test accuracy and test fidelity for heart and breast cancer domains. The experiments are done using relevance-score, information content and their combinations from small, extended and snomed ontologies.

Ontology	Method	Heart	Breast
None	IG = IG	74.6022	63.0952
Small	IG' = IG * IC	73.9462	68.2275
	IG' = IG * RS	76.9570	70.1058
	IG' = IC	73.9462	68.2275
	IG' = RS	76.9570	70.1058
	IG' = IC * RS	76.9570	70.1058
	IG' = IG * IC * RS	76.9570	70.1058
Extended	IG' = IG * IC	71.6129	61.2302
	IG' = IG * RS	76.6237	70.1058
	IG' = IC	71.6129	61.2302
	IG' = RS	76.6237	70.1058
	IG' = IC * RS	78.2688	70.1058
	IG' = IG * IC * RS	78.2688	70.1058
Snomed	IG' = IG*RS	57.0968	67.8836
	IG' = RS	57.0968	67.8836

Table 6.4: Test accuracy in (%) using decision tree algorithm for breast and heart domains, without any ontology, as well as with heuristics from three different ontologies.

For heart domain we can observe that the best accuracy is achieved when building the trees using the product of information gain and relevance-score from both small and extended ontologies, by increasing the accuracy by 9,5 or 11.1% compared to not using any information from the ontology. Moreover, when we look into test fidelity, we observe that using the product of information gain and information content from extended ontology we achieve the best fidelity of 93.5, which is 27.6% higher than not using any information from the ontology.

When comparing test accuracy and test fidelity for breast cancer domain using different ontologies and different measures from ontologies, we can observe that using information content from extended ontology instead of information gain increases the accuracy from 78.1818 without knowledge from ontology, to 100. On the other hand, using information content from small ontology instead of information gain achieves the best fidelity of 87.27, which is 17.1% better than using Trepan without any knowledge from ontology.

Syntactic Complexity results

We calculated the syntactic complexity of the resultant trees built using the Trepan algorithm with and without domain knowledge for each domain. In Table 6.8 we present

Method	Heart	Breast	Diabetes	Hepatitis	Stroke	Kidney	Lung
Trepan	78.3333	78.1818	74.5098	66.6667	95.0951	97.4684	83.6066
IG' = IG*RS	**83.8710**	76.3636	74.5098	56.6667	**95.1952**	**100**	83.6066
IG' = RS	67.7419	**96.3636**	66.6667	66.6667	**95.1952**	**100**	83.6066

Table 6.5: Test accuracy in (%) of decision trees extracted with Trepan with and without relevance-score from SNOMED-CT ontologies.

Method	Heart	Breast	Diabetes	Hepatitis	Stroke	Kidney	Lung
Trepan	73.3333	74.5455	91.5033	76.6667	99.7998	97.4684	91.8033
IG' = IG*RS	**87.0968**	67.2727	91.5033	66.6667	**99.8999**	**100**	91.8033
IG' = RS	70.9677	**76.3636**	75.8170	50.0	99.1992	**100**	91.8033

Table 6.6: Test fidelity in (%) of decision trees extracted with Trepan with and without relevance-score from SNOMED-CT ontologies.

two methods for calculating the syntactic complexity of the trees, syntactic complexity TR (calculated from 6.5) and syntactic complexity EDAG (calculated from 6.3). When comparing the syntactic complexity TR of trees built using Trepan with those generated with domain knowledge, we cannot infer whether trees are less complicated than others because the syntactic complexity is nearly always the same or differs by no more than 1. On the other hand, when analyzing syntactic complexity EDAG, we can observe that for the breast cancer dataset, the tree constructed with relevance-score has much lower syntactic complexity than the tree constructed without the relevance-score, with a drop of 49.2%. The syntactic complexity for the heart, stroke, and kidney disease domains is nearly or exactly the same, however the syntactic complexity for the diabetes and hepatitis domains is 2 and 3.2 lower, respectively, compared to the tree when the relevance-score is used.

User-based questionnaires results

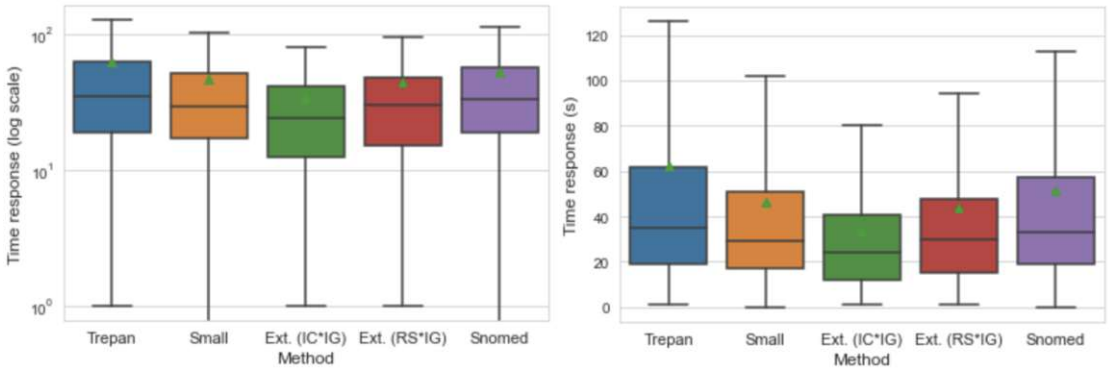
From here on, "Small" refers to decision trees extracted from Trepan with the integration of an ontology, where the ontology is handcrafted in the same way as the authors of Trepan Reloaded did, and the information gain is multiplied by information content $IG' = IG * IC$ during the tree construction. Furthermore, by "Ext. (RS*IG)" and "Ext. (IC*IG)", we indicate taking an existing ontology and expanding it with concepts and roles from the datasets, as well as adjusting the information gain with relevance-score and information content, respectively. And when we say "Snomed", we mean using knowledge from SNOMED-CT extracted ontologies and during the tree construction multiplying relevance-score from the ontology with information gain. Furthermore, we use "Syntactic Complexity TR" and "Syntactic Complexity EDAG" to denote the syntactic complexities

	Domains	Heart		Breast	
Ontology	**Methods**	**Test Accuracy**	**Test Fidelity**	**Test Accuracy**	**Test Fidelity**
None	Trepan	78.3333	73.3333	78.1818	74.5455
Small	IG' = IG*IC	80.6452	90.3226	76.3636	67.2727
	IG' = IG*RS	87.0968	90.3226	76.3636	78.1818
	IG' = IC	80.6452	83.8710	89.0909	87.2727
	IG' = RS	74.1935	77.4194	87.2727	70.9091
	IG' = RS*IC	80.6452	83.8710	87.2727	70.9091
	IG' = IG*IC*RS	87.0968	90.3226	76.3636	78.1818
Extended	IG' = IG*IC	83.8710	93.5484	78.1818	72.7273
	IG' = IG*RS	87.0968	90.3226	76.3636	78.1818
	IG' = IC	54.8387	51.6129	100	80.0
	IG' = RS	77.4194	80.6452	87.2727	70.9091
	IG' = RS*IC	54.8387	51.6129	78.1818	69.0909
	IG' = IG*IC*RS	80.6452	83.8710	78.1818	76.3636
Snomed	IG' = IG*RS	83.8710	87.0968	76.3636	67.2727
	IG' = RS	67.7419	70.9677	96.3636	76.3636

Table 6.7: Test accuracy and test fidelity in (%) of decision trees extracted with Trepan using different heuristics and three different ontologies (Small, Extended and SNOMED-CT)

	Syntactic Complexity TR		Syntactic Complexity EDAG	
Domains	**Trepan**	**IG'=IG*RS**	**Trepan**	**IG'=IG*RS**
Heart	2.42	2.46	24.2	24.4
Breast	7.22	6.54	62.6	31.8
Diabetes	0.94	0.94	16.6	18.6
Hepatitis	0.8	1.08	10.6	13.8
Stroke	1.68	1.68	9.4	9.4
Kidney	1.22	1.36	19.6	19.6

Table 6.8: Syntactic complexities calculated by 6.5 and 6.3 for the trees extracted with Trepan and with domain knowledge from ontologies extracted from SNOMED-CT.



(a) (b)

Figure 6.5: Time response (a) in log scale and (b) in seconds when the ontology is not present, and when it is present on different forms and different ontologies.

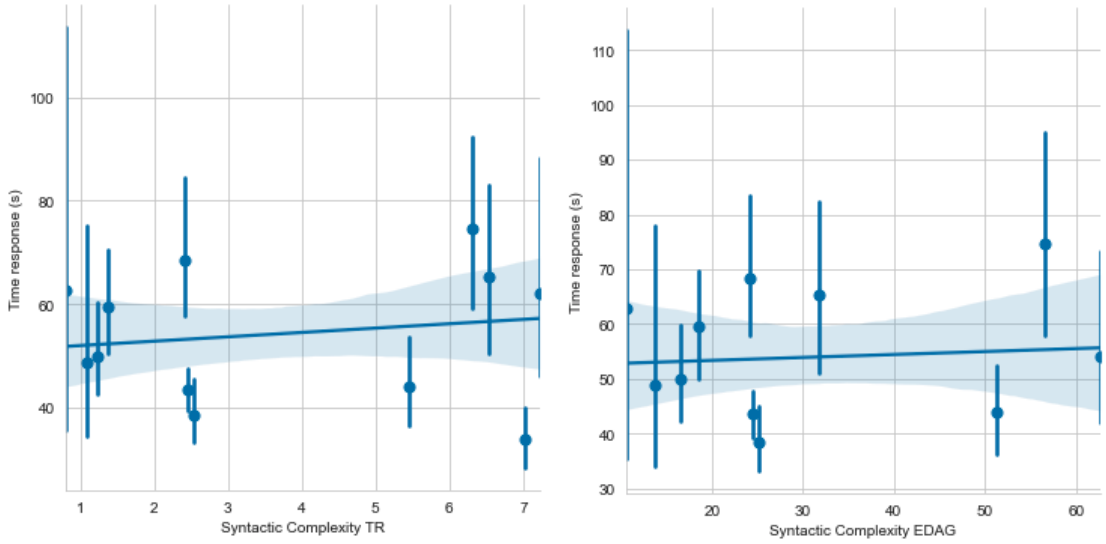
of the trees calculated with 6.5 and 6.3 respectively.

Figure 6.5 depicts time responses to questions asked to participants for the construction of trees in various forms. We can observe that participants spent more time answering questions about trees built using Trepan without the existence of an ontology than about others. Furthermore, participants appear to respond faster to questions about trees built using "Ext. (IC*RS)" than to questions about other trees.

Figure 6.6 illustrates the estimated main effects of time responses and syntactic complexity, for TR and EDAG syntactic complexities. We can observe that as syntactic complexity increases for both TR and EDAG complexities, the time response increases, implying that more complex trees result in longer response times. The increase in time response is more visible when the syntactic complexity TR is increased, whereas for the syntactic complexity EDAG, the complexity of the trees results in a very little increase in time response. In Figure 6.7 and Figure 6.8 we show the estimated main effects of two-way interaction between time responses and syntactic complexity for each task separately for TR and EDAG respectively.

In Figure 6.9 we illustrate the proportions of correct answers for trees constructed with various methods. On average, for classification and inspection tasks, participants responded more accurately for trees created without ontology than trees generated with ontology. Furthermore, trees constructed with the snomed ontology gave more accurate results than other trees with ontologies. Moreover, trees built with the extended ontology with IG'=IG*RS produced less accurate results than the others.

Looking at the estimated effect of syntactic complexity on accuracy in Figure 6.10 we can observe that for both syntactic complexities TR and EDAG, more complex trees produced less accurate responses. Moreover, in Figure 6.11 we show the two-way interactions between task and syntactic complexity TR on correct responses. We can observe that for classification task the decrease of correct answers for more complex trees is less



(a) (b)

Figure 6.6: Estimated main effects of syntactic complexity TR (a) and EDAG (b) on time responses (in seconds).

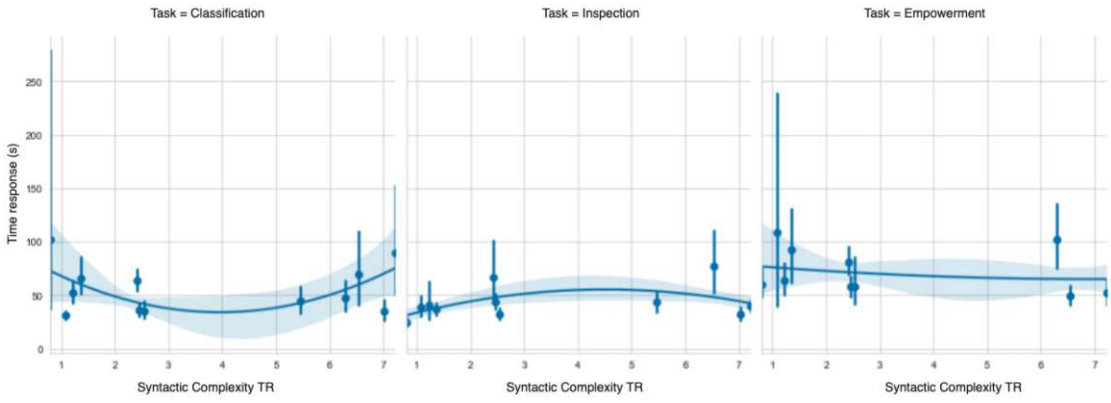


Figure 6.7: Estimated main effects of two-way interactions between task and syntactic complexity TR on time responses

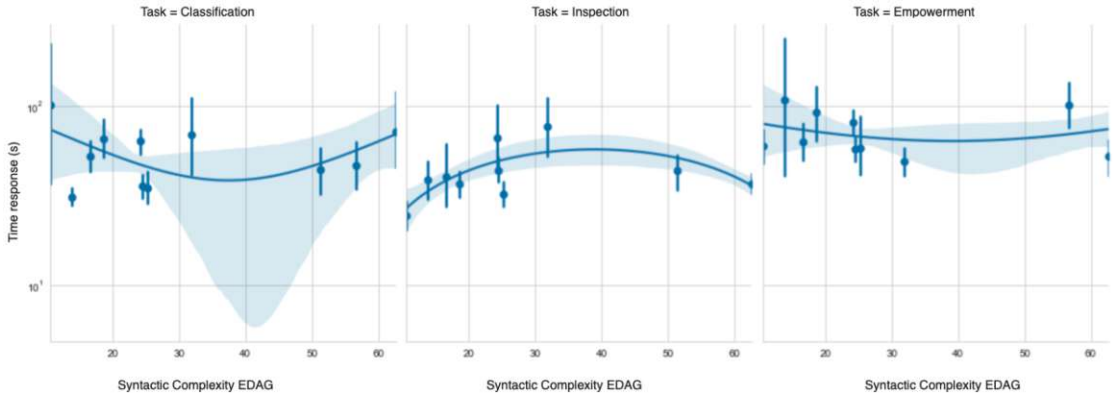


Figure 6.8: Estimated main effects of two-way interactions between task and syntactic complexity EDAG on time responses

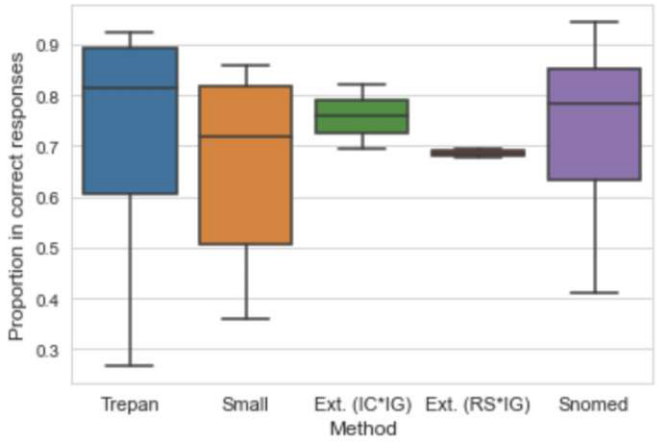
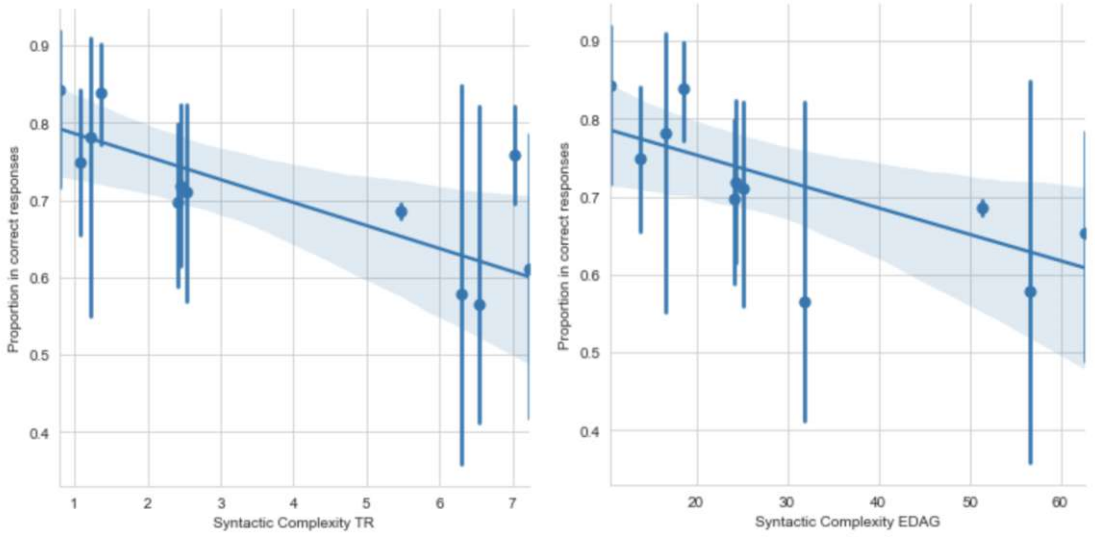


Figure 6.9: Proportion in correct responses for Trepan and different ontology measures.

significant compared to inspection and empowerment tasks, where the proportion of correct answers reduces by roughly 30% for more complex trees. On the other hand, for syntactic complexity EDAG, we can see from Figure 6.12 that the drop of proportions of correct responses when the complexity of the trees is increased is marginal for all three tasks.

Another finding from the questionnaires is how understandable the trees generated with different measurements from different ontologies were for the participants. In Figure 6.13 we show the understandability of the trees for different methods, based on the participants' answers. We assigned a rating of 1 to 5, with 1 being very difficult to understand and 5 being very easy to understand. From the Figure we can observe that trees build without and with snomed ontology seem to be more understandable compare to the others, whereas trees generated with extended ontology using IG'=RS*IG produce less understandable trees compare to others.



(a) (b)

Figure 6.10: Estimated main effects of syntactic complexity (a) TR and (b) EDAG on proportion of correct responses.

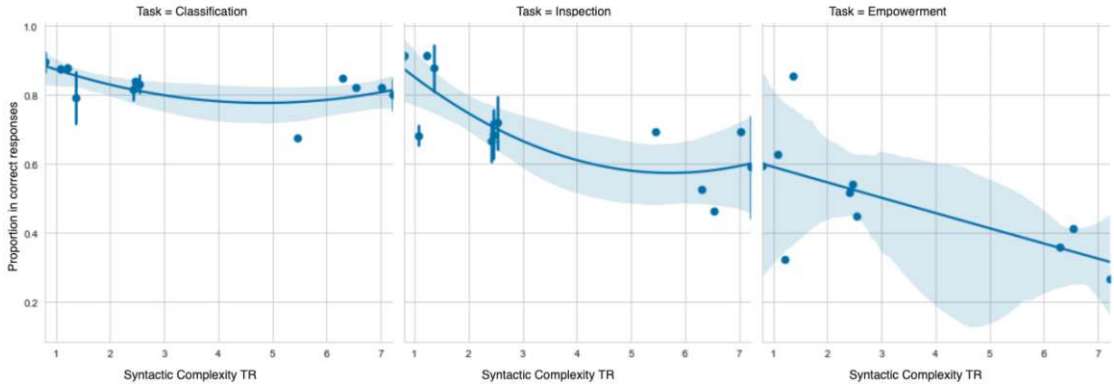


Figure 6.11: Effects of two-way interaction between task and syntactic complexity TR on proportion of correct responses.

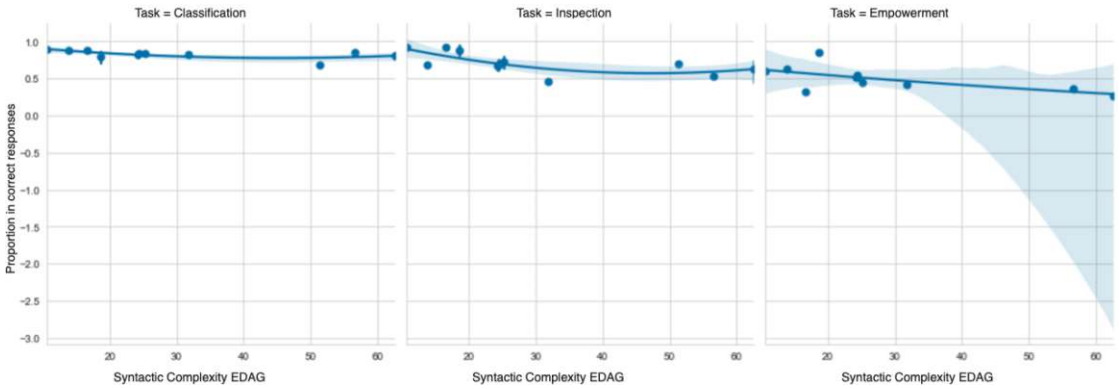


Figure 6.12: Effects of two-way interaction between task and syntactic complexity EDAG on proportion of correct responses.

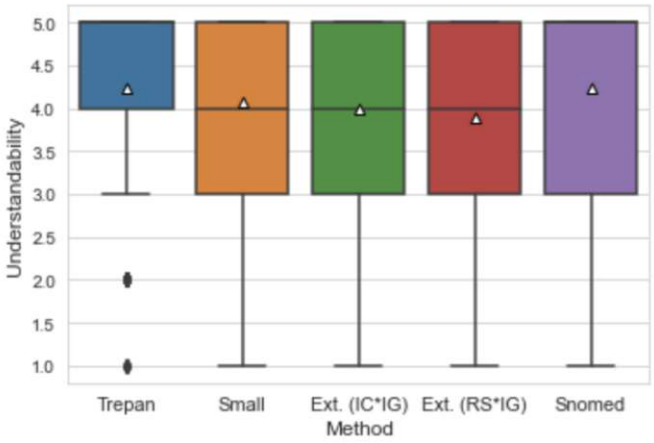
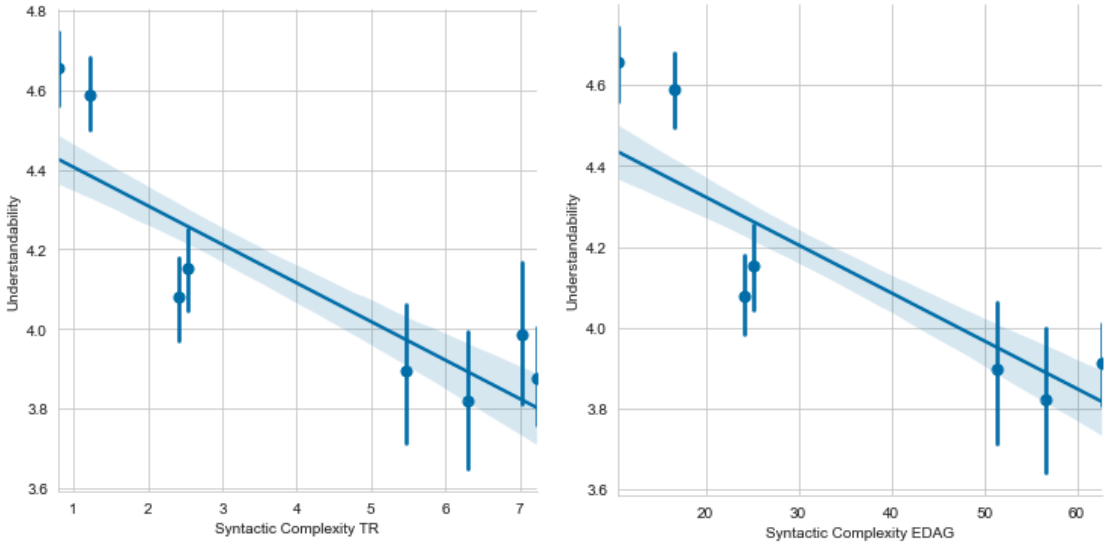


Figure 6.13: Understandability for Trepan and different ontology measures.

When analyzing estimated main effects of syntactic complexity TR and EDAG on understandability reported by users in Figure 6.14, we can conclude that trees are less understandable as they get more syntactically complex. Moreover, when we compare the understandability of the users between classification and inspection tasks shown in Figure 6.15 and Figure 6.16 for syntactic complexity TR and EDAG respectively, we may infer that the users found the inspection task harder to understand compared to classification task.

Table 6.9 shows the mean and standard deviation of correct responses, response time, confidence, and understandability for trees extracted using the Trepan method, for small ontologies with IG'=IG*IC, extended ontologies with IG'=IG*IC and IG'=IG*RS, and snomed ontologies with IG'=IG*RS. We do not have results for extended ontologies in the empowerment task since no tree that was derived from the extended ontologies was shown in the empowerment task in the questionnaires. According to the table, trees



(a) (b)

Figure 6.14: Estimated main effects of syntactic complexity TR (a) and EDAG (b) on understandability reported by users

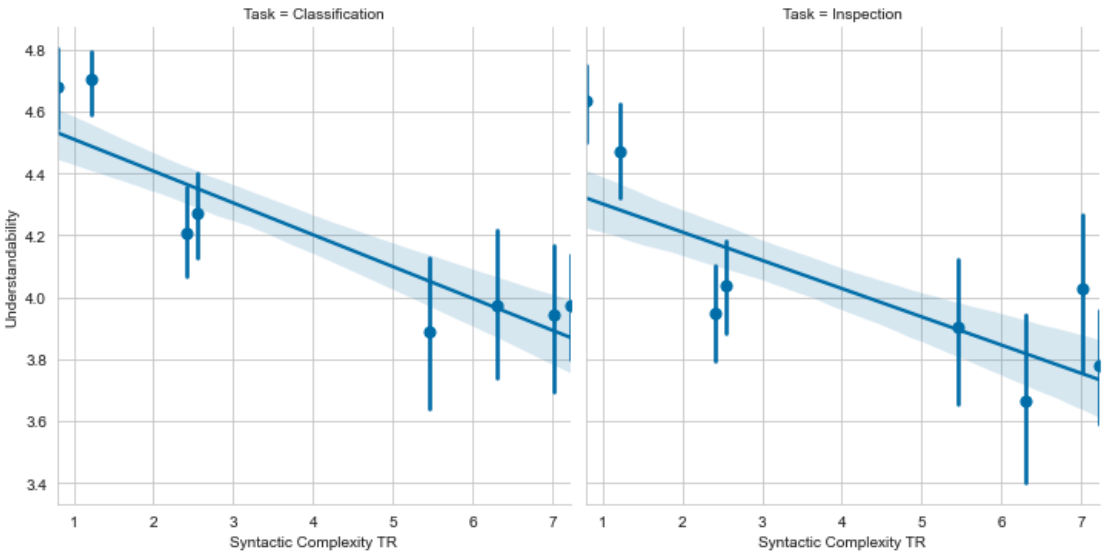


Figure 6.15: Effects of two-way interaction between task and syntactic complexity TR on understandability reported by users.

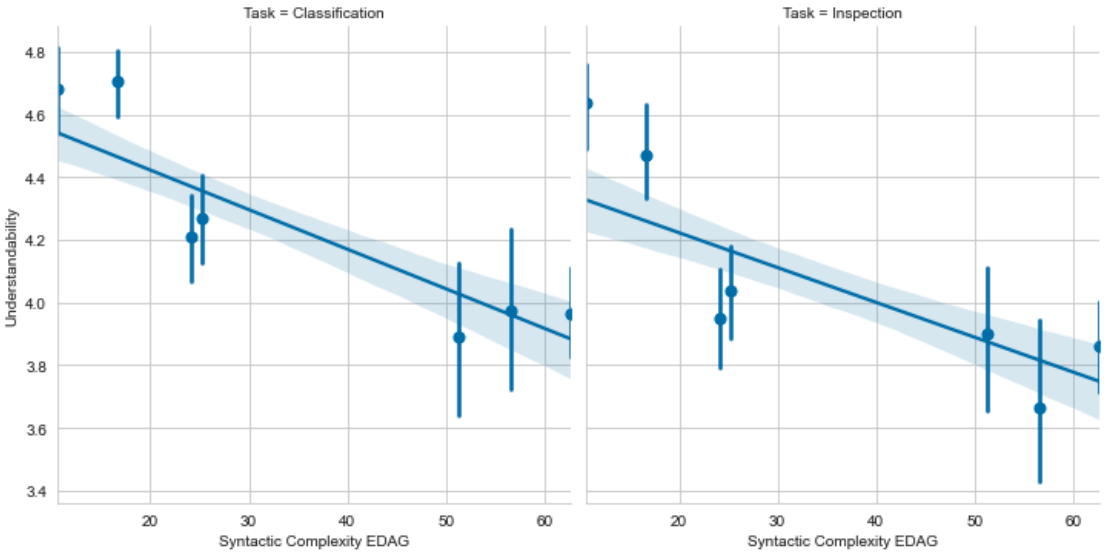


Figure 6.16: Effects of two-way interaction between task and syntactic complexity EDAG on understandability reported by users.

Task	Measure	Trep.	Small	Ext.(IC*IG)	Ext.(RS*IG)	Snomed(IG*RS)
Class.	Correct resp.	0.85 (0.05)	0.84 (0.03)	0.82 (0)	0.68 (0)	0.83 (0.05)
	Time (sec)	75.62 (334.05)	38.13 (58.81)	35.43 (42.92)	44.27 (59.02)	46.45 (82.62)
	Confidence	4.42 (0.9)	4.3 (0.94)	4.08 (1.0)	4.19 (1.03)	4.44 (0.88)
	Understand.	4.33 (0.94)	4.19 (1.03)	3.94 (1.1)	3.88 (1.11)	4.34 (0.94)
Insp.	Correct resp.	0.77 (0.18)	0.65 (0.14)	0.69 (0)	0.69 (0)	0.71 (0.15)
	Time (sec)	46.81 (141.98)	32.2 (37.23)	32.29 (27.64)	43.69 (43.41)	46.21 (69.83)
	Confidence	4.24 (0.93)	4.1 (0.97)	4.22 (0.94)	4.07 (0.83)	4.23 (0.92)
	Understand.	4.14 (1.03)	3.94 (1.09)	4.03 (1.03)	3.9 (1.0)	4.11 (1.0)
Emp.	Correct resp.	0.43 (0.16)	0.4 (0.06)	NA	NA	0.61 (0.19)
	Time (sec)	66.35 (59.65)	76.73 (128.82)	NA	NA	71.28 (196.85)

Table 6.9: Mean values of correct answers, time of response, user confidence, and user understandability for trees extracted with Trepan, using different ontologies and measures.

extracted from snomed with the use of relevance-score in the trees' construction gave a higher proportion of correct responses than other trees, with a significantly higher proportion of correct responses on empowerment tasks and it was among the trees with higher proportions of correct responses on classification and inspection tasks.

In terms of response time, we can see that the empowerment task appears to take longer to complete than classification and inspection. The trees extracted from the extended ontology built using IG'=IG*IC took the shortest time to answer for classification and inspection, whereas for empowerment trees extracted from Trepan when no ontology is used appear to take the shortest time to answer.

In terms of user confidence and understandability, we can see from the Table that for classification task trees extracted from snomed appear to be more understandable and users appear to be more confident on those trees than trees extracted and constructed in other ways. Trees extracted using Trepan, on the other hand, appear to be marginally more understandable for inspection tasks than trees extracted with snomed.

Overall, we can conclude from the table that users find the trees extracted from the snomed ontology constructed using IG'=IG*RS to be more understandable as well as they answered with higher proportion of correct answers than for the trees constructed with other methods.

CHAPTER 7

Conclusion

In this thesis, we investigated ways of incorporating domain knowledge into a machine learning model, more specifically decision trees. We found and proposed heuristics from ontologies which can be used in the decision tree building process.

We extracted seven topic specific ontologies from a high-quality source with corresponding features from domain datasets as concepts and roles. Moreover, we used information content presented in Trepan Reloaded [CWBdPM21] and our modified version of hubscore metric (relevance-score), as well as their combination when building decision trees for the seven domains. Moreover, we tested and compared decision trees built with or without these two heuristics and their combinations when extracting decision trees using Trepan algorithm from neural networks.

In order to evaluate the decision trees built with various heuristics from ontologies we measured accuracy. Moreover, for trees extracted with Trepan algorithm with and without the use of domain knowledge in the tree building process, we measured the fidelity as well as two methods for calculating syntactic complexities of the trees. In addition, in order to measure the understandability of these trees to the users, we performed three user-based questionnaires in four different domains with 234 participants in total. We measured the proportion of correct responses, time response, user confidence and understandability, as well as the dependencies of time response, the proportions of correct answers and understandability from the users with the syntactic complexities.

We found that information content is not applicable to the ontologies we extracted from SNOMED-CT, thus in such cases we used our metric relevance-score.

For constructing direct decision trees, for six out of the seven domains that we experimented with, using knowledge from ontologies improved the accuracy. Moreover, decision trees constructed with our proposed relevance score, performed better for five out of seven domains compared to decision trees constructed without any knowledge from ontologies.

Thus, we conclude that using knowledge from ontologies and the proposed relevance-score can be beneficial to improve the performance of decision trees.

When looking at the accuracy of decision trees extracted from Trepan algorithm with and without ontologies, we can observe that using relevance-score from ontologies extracted from SNOMED-CT ontology in the decision tree building process either increases the accuracy or the accuracy stayed the same.

Another finding from the user-based questionnaires was that users found the trees extracted from the SNOMED-CT ontology constructed using IG'=IG*RS to be more understandable, and gave more correct answers than trees constructed in other methods with or without ontology. Moreover, we observed that users found trees with a higher syntactic complexity less understandable, and the proportions of correct responses from users drops for higher syntactic complex trees.

Even though the improvement on the accuracy were modest when incorporating knowledge from ontologies, the understandability of the decision trees improved based on the the user-based questionnaires results. Moreover, we provided a systematic methodology for using Trepan Reloaded with real-life ontologies, as well as a way to leverage knowledge from the ontologies even in the cases where the information content cannot be used in the decision tree building process.

This research identifies a number of possibilities for further research in this topic. In future work we may focus on discovering new ways to extract knowledge from current ontologies and incorporating it into various machine learning approaches. We also believe that these methods should be tested and compared with other domains as well. Moreover, the domain ontologies could be used for other purposes as well, such as to give explanations of any classifiers' prediction. Furthermore, user-based questionnaires have some limitations due to the fact that there is the need for many users, and the results are very subjective, therefore we need to keep working on better techniques to assess a way to measure decision tree understandability.

List of Figures

List of Tables

Bibliography

[ARS+20] Alejandro Barredo Arrieta, Natalia Díaz Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115, 2020.

[AS17] Ruba Alassaf and Renate A Schmidt. A preliminary comparison of the forgetting solutions computed using scan, lethe and fame. In *SOQE*, pages 21–26, 2017.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BCM05] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. *Ontology learning from text: methods, evaluation and applications*, volume 123. IOS press, 2005.

[BDB08] R Harald Baayen, Douglas J Davidson, and Douglas M Bates. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of memory and language*, 59(4):390–412, 2008.

[BH21] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[Bha15] Subhra Bikash Bhattacharyya. *Introduction to SNOMED CT*. Springer, 2015.

[BHX16] Anila Sahar Butt, Armin Haller, and Lexing Xie. Dwrank: Learning concept ranking for ontology search. *Semantic Web*, 7(4):447–461, 2016.

[BN03] Franz Baader and Werner Nutt. Basic description logics. In *The description logic handbook: theory, implementation, and applications*, pages 43–95. 2003.

[BP21] Vaishak Belle and Ioannis Papantonis. Principles and practice of explainable machine learning. *Frontiers Big Data*, 4:688969, 2021.

[BXS+20] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. Explainable machine learning in deployment. In Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna, editors, *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pages 648–657. ACM, 2020.

[CA21] Bahzad Charbuty and Adnan Abdulazeez. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, 2(01):20–28, 2021.

[CAS+19a] Jieying Chen, Ghadah Alghamdi, Renate A Schmidt, Dirk Walther, and Yongsheng Gao. Modularity meets forgetting: A case study with the snomed ct ontology. In *Description Logics*, 2019.

[CAS+19b] Jieying Chen, Ghadah Alghamdi, Renate A Schmidt, Dirk Walther, and Yongsheng Gao. Ontology extraction for large ontologies via modularity and forgetting. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 45–52, 2019.

[CC05] Matteo Cristani and Roberta Cuel. A survey on ontology creation methodologies. *Int. J. Semantic Web Inf. Syst.*, 1(2):49–69, 2005.

[CFLGP06] Oscar Corcho, Mariano Fernández-López, and Asunción Gómez-Pérez. Ontological engineering: principles, methods, tools and languages. In *Ontologies for software engineering and software technology*, pages 1–48. Springer, 2006.

[CLMW17] Jieying Chen, Michel Ludwig, Yue Ma, and Dirk Walther. Zooming in on ontologies: Minimal modules and best excerpts. In *International Semantic Web Conference*, pages 173–189. Springer, 2017.

[CLW18] Jieying Chen, Michel Ludwig, and Dirk Walther. Computing minimal subsumption modules of ontologies. *GCAI*, 18:41–53, 2018.

[CPC19] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[CS95] Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8:24–30, 1995.

[CWBdPM21] Roberto Confalonieri, Tillman Weyde, Tarek R Besold, and Fermín Moscoso del Prado Martín. Using ontologies to enhance human understandability of global post-hoc explanations of black-box models. *Artificial Intelligence*, 296:103471, 2021.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[Fri91] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

[Gai96] B Gaines. Transforming rules and trees into comprehensible knowledge structures. advances in knowledge discovery and data mining (pp. 205–228), 1996.

[GGB12] AS Galathiya, AP Ganatra, and CK Bhensdadia. Classification with an improved decision tree algorithm. *International Journal of Computer Applications*, 46(23):1–6, 2012.

[GHKS07] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. A logical framework for modularity of ontologies. In *IJCAI*, volume 2007, pages 298–303, 2007.

[GO92] Dov M Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second-order predicate logic. 1992.

[GPSK06] Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In *KR*, pages 198–209, 2006.

[Gru93] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[GSS08] Dov M. Gabbay, Renate A. Schmidt, and Andrzej Szalas. *Second-Order Quantifier Elimination - Foundations, Computational Aspects and Applications*, volume 12 of *Studies in logic : Mathematical logic and foundations*. College Publications, 2008.

[HLD+19] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainability of artificial intelligence in medicine. *WIREs Data Mining Knowl. Discov.*, 9(4), 2019.

[HS06] Hans-Jörg Happel and Stefan Seedorf. Applications of ontologies in software engineering. In *Proc. of Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC*, pages 5–9. Citeseer, 2006.

[KKK13] Ashwini D Khairkar, Deepak D Kshirsagar, and Sandeep Kumar. Ontology for detection of web attacks. In *2013 International Conference on Communication Systems and Network Technologies*, pages 612–615. IEEE, 2013.

[KLWW13] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203:66–103, 2013.

[Koo20] Patrick Koopmann. Lethe: Forgetting and uniform interpolation for expressive description logics. *KI-Künstliche Intelligenz*, 34(3):381–387, 2020.

[KSD+15] Patrick Koopmann, Renate A Schmidt, M Dumontier, B Glimm, R Goncalves, M Horridge, E Jiménez-Ruiz, N Matentzoglu, B Parsia, G Stamou, et al. Lethe: A saturation-based tool for non-classical reasoning. In *Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015)*. RWTH Aachen University, 2015.

[KWW09] Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *IJCAI*, pages 830–835, 2009.

[KWZ10] Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. Logic-based ontology comparison and module extraction, with an application to dl-lite. *Artificial Intelligence*, 174(15):1093–1141, 2010.

[LB02] Ruey-Hsia Li and Geneva G Belford. Instability of decision tree classification algorithms. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 570–575, 2002.

[Lew00] Roger J Lewis. An introduction to classification and regression tree (cart) analysis. In *Annual meeting of the society for academic emergency medicine in San Francisco, California*, volume 14. Citeseer, 2000.

[Lip18] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[LM01] Ora Lassila and Deborah McGuinness. The role of frame-based representation on the semantic web. *Linköping Electronic Articles in Computer and Information Science*, 6(5):2001, 2001.

[LOL+18] Bruno Lepri, Nuria Oliver, Emmanuel Letouzé, Alex Pentland, and Patrick Vinck. Fair, transparent, and accountable algorithmic decision-making processes. *Philosophy & Technology*, 31(4):611–627, 2018.

[LP10] Ion Lungu and Alexandru Pîrjan. Research issues concerning algorithms used for optimizing the data mining process. *J. Inf. Syst. Oper. Manage.*, 4(2):108–125, 2010.

[NB03] Daniele Nardi and Ronald J. Brachman. An introduction to description logics. In Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 1–40. Cambridge University Press, 2003.

[NM+01] N Noy, Deborah L McGuinness, et al. Ontology development 101. *Knowledge Systems Laboratory, Stanford University*, 2001, 2001.

[OdTS13] Fco. Javier Ordóñez, Paula de Toledo, and Araceli Sanchis. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors*, 13(5):5460–5477, 2013.

[PJC09] Jyotishman Pathak, Thomas M Johnson, and Christopher G Chute. Survey of modular ontology techniques and their applications in the biomedical domain. *Integrated computer-aided engineering*, 16(3):225–242, 2009.

[PLGM16] Rok Piltaver, Mitja Lustrek, Matjaz Gams, and Sanda Martincic-Ipsic. What makes classification trees comprehensible? *Expert Syst. Appl.*, 62:333–346, 2016.

[PPP20] Cecilia Panigutti, Alan Perotti, and Dino Pedreschi. Doctor xai: an ontology-based approach to black-box sequential data classification explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 629–639, 2020.

[Qui83] J Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine learning*, pages 463–482. Springer, 1983.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[Qui96] J. Ross Quinlan. Bagging, boosting, and C4.5. In William J. Clancey and Daniel S. Weld, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 1*, pages 725–730. AAAI Press / The MIT Press, 1996.

[Qui14] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[RAA+14] Abdul Razzaq, Zahid Anwar, H Farooq Ahmad, Khalid Latif, and Faisal Munir. Ontology for attack detection: An intelligent approach to web application security. *computers & security*, 45:124–146, 2014.

[RB10] M Ramaswami and R Bhaskaran. A chaid based performance prediction model in educational data mining. *arXiv preprint arXiv:1002.1144*, 2010.

[RLM+06] Daniel L Rubin, Suzanna E Lewis, Chris J Mungall, Sima Misra, Monte Westerfield, Michael Ashburner, Ida Sim, Christopher G Chute, Margaret-Anne Storey, Barry Smith, et al. National center for biomedical ontology: advancing biomedicine through structured organization of scientific knowledge. *Omics: a journal of integrative biology*, 10(2):185–198, 2006.

[RM15] Lior Rokach and Oded Maimon. *Data mining with decision trees: theory and applications*. World Scientific, Hackensack, New Jersey, second edition edition, 2015.

[SAMK05] Irena Spasic, Sophia Ananiadou, John McNaught, and Anand Kumar. Text mining and ontologies in biomedicine: making sense of raw text. *Briefings in bioinformatics*, 6(3):239–251, 2005.

[SG14] Sonia Singh and Priyanka Gupta. Comparative study id3, cart and c4. 5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology (IJAIST)*, 27(27):97–103, 2014.

[SMDE19] Alberto G Salguero, Javier Medina, Pablo Delatorre, and Macarena Espinilla. Methodology for improving classification accuracy using ontologies: application in the recognition of activities of daily living. *Journal of Ambient Intelligence and Humanized Computing*, 10(6):2125–2142, 2019.

[SP18] Andrew Selbst and Julia Powles. "meaningful information" and the right to explanation. In *Conference on Fairness, Accountability and Transparency*, pages 48–48. PMLR, 2018.

[SZP+20] Yulia Svetashova, Baifan Zhou, Tim Pychynski, Stefan Schmidt, York Sure-Vetter, Ralf Mikut, and Evgeny Kharlamov. Ontology-enhanced machine learning: a bosch use case of welding quality monitoring. In *International Semantic Web Conference*, pages 531–550. Springer, 2020.

[TJMG19] Sana Tonekaboni, Shalmali Joshi, Melissa D. McCradden, and Anna Goldenberg. What clinicians want: Contextualizing explainable machine learning for clinical end use. In Finale Doshi-Velez, Jim Fackler, Ken Jung, David C. Kale, Rajesh Ranganath, Byron C. Wallace, and Jenna Wiens, editors, *Proceedings of the Machine Learning for Healthcare Conference, MLHC 2019, 9-10 August 2019, Ann Arbor, Michigan, USA*, volume 106 of *Proceedings of Machine Learning Research*, pages 359–380. PMLR, 2019.

[Van17] Gilles Vandewiele. Enhancing white-box machine learning processes by incorporating semantic background knowledge. In *European Semantic Web Conference*, pages 267–278. Springer, 2017.

[WDL+20] Xuan Wu, Wenxing Deng, Chang Lu, Hao Feng, and Yizheng Zhao. Ui-fame. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3473–3476, 2020.

[WWT+14] Kewen Wang, Zhe Wang, Rodney Topor, Jeff Z Pan, and Grigoris Antoniou. Eliminating concepts and roles from ontologies in expressive descriptive logics. *Computational Intelligence*, 30(2):205–232, 2014.

[XUD+19] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. Explainable AI: A brief survey on history, research areas, approaches and challenges. In Jie Tang, Min-Yen Kan, Dongyan Zhao, Sujian Li, and Hongying Zan, editors, *Natural Language Processing and Chinese Computing - 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9-14, 2019, Proceedings, Part II*, volume 11839 of *Lecture Notes in Computer Science*, pages 563–574. Springer, 2019.

[ZLFC18] Chiara Zucco, Huizhi Liang, Giuseppe Di Fatta, and Mario Cannataro. Explainable sentiment analysis with applications in medicine. In Huiru Jane Zheng, Zoraida Callejas, David Griol, Haiying Wang, Xiaohua Hu, Harald H. H. W. Schmidt, Jan Baumbach, Julie Dickerson, and Le Zhang, editors, *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018, Madrid, Spain, December 3-6, 2018*, pages 1740–1747. IEEE Computer Society, 2018.

[ZS18] Yizheng Zhao and Renate A Schmidt. Fame: an automated tool for semantic forgetting in expressive description logics. In *International Joint Conference on Automated Reasoning*, pages 19–27. Springer, 2018.

[ZSH02] Jun Zhang, Adrian Silvescu, and Vasant Honavar. Ontology-driven induction of decision trees at multiple levels of abstraction. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 316–323. Springer, 2002.