OLAF: An Ontology Learning Applied Framework

Marion Schaeffer, Matthias Sesboüé, Jean-Philippe Kotowicz, Nicolas Delestre, Cecilia Zanni-Merk

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 225 (2023) 2106–2115

Procedia Computer Science

www.elsevier.com/locate/procedia

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023)

OLAF: An Ontology Learning Applied Framework

Marion Schaeffer[a,b], Matthias Sesboüé[a,c], Jean-Philippe Kotowicz[a], Nicolas Delestre[a], Cecilia Zanni-Merk[a]

[a] *INSA Rouen Normandie, LITIS (UR 4108), Normandie Univ., 76000 Rouen, France*
[b] *Wikit, 69005 Lyon, France*
[c] *Traceparts, 76430 Saint Romain, France*

Abstract

Since the beginning of the century, research on ontology learning has gained popularity. Automatically extracting and structuring knowledge relevant to a domain of interest from unstructured textual data is a major scientific challenge. After studying the main existing methods, such as Text2Onto, we propose a new approach with a modular ontology learning framework focusing on automatically extracting knowledge from raw text sources. We consider tasks from data pre-processing to axiom extraction. Whereas previous contributions considered ontology learning systems as tools to help the domain expert craft a reusable ontology, we developed the proposed framework with full automation in mind to build a minimum viable ontology targeted at an application. Ontology Learning Applied Framework (OLAF) has been generically designed to build specific ontologies whatever the application domain, use case and text data. We implement an initial version and test the framework on an ontology-based system, a search engine for technical products.

Keywords: ontology learning; ontology; knowledge acquisition; ontology-based system; framework; automation; NLP

1. Introduction

In computer science, ontologies are widely used to describe a domain of interest and produce a formal knowledge representation. Many well-known ontology definitions have been proposed in [14, 15, 18, 28]. These definitions are arguably vague and originate from logics and the Semantic Web in academic research communities. We consider Navigli et al.'s definition of an ontology [25]: "An ontology specifies a shared understanding of a domain. It contains a set of generic concepts and their definitions and interrelationships".

* Corresponding authors
E-mail address: marion.schaeffer@insa-rouen.fr - matthias.sesboue@insa-rouen.fr

10.1016/j.procs.2023.10.201

Since ontologies are reference domain knowledge, many applications are based on them. However, an ontology is not always available depending on the domain of interest. In that case, a new ontology must be built, or an existing one must be extended. This is a complex and time-consuming task that requires a human domain expert. Ontology Learning (OL) techniques focus on automating as much as possible knowledge acquisition processes [1, 21]. Both knowledge acquisition and OL refer to methods aiming at discovering concepts and relations contained in a source of knowledge. The goal is to structure the knowledge and build a computer-interpretable model. Our work focuses on OL from text data, which implies standard Natural Language Processing (NLP) techniques. This study meets a demand to structure unstructured data with minimal human intervention.

The use case considered in this work is a semantic search engine for a corpus of technical product descriptions. Using an ontology representing the available products has been identified as a promising avenue. To build this ontology, no domain expert or linguist is available and relevant resources only include large text datasets, as it is often the case in many ontology engineering projects. During our literature study, we did not find any maintained tools that would allow us to create a Minimum Viable Ontology (MVO) in a modular and automatic way, using efficient algorithms on technical documentation. We define MVO as the well-known Minimum Viable Product (MVP) adaptation to ontology engineering. Thus, we propose Ontology Learning Applied Framework (OLAF), an OL framework encompassing tasks from data pre-processing to axiom extraction.

Besides our particular use case, automatically created ontologies could serve multiple practical other use cases. In addition to quickly generating potentially production-ready domain knowledge, it could enable domain or knowledge modelling experts to focus on high-value tasks such as axiom definition. An initial version of the introduced framework has been implemented as an open access and collaborative tool[1]. To illustrate its potential usage, we apply our implementation on technical descriptions scrapped from the Schneider Electrics website and the learned ontology is explored and evaluated using Competency Questions (CQs).

The remainder of this paper is organised as follows. Related works are discussed in section 2. Section 3 describes the framework proposed for a modular ontology learning process. We use the framework to build an ontology for a specific use case and evaluate it using CQs in section 4. We conclude with a discussion and some future works in section 5.

2. Related works

This section describes previous work dealing with OL as an end-to-end process. We also consider particular components of the OL process and the evaluation techniques. In the remainder of this paper, we define *pipeline* as a sequence of tasks performed during the OL process. Those tasks are performed and executed by pipeline components composing the pipeline. *Linguistic realisations* are the textual representations of some abstract object in the actual language.

2.1. Ontology learning

OL is the automatic approach to knowledge acquisition methods explored since the acknowledgement of the so-called knowledge acquisition bottleneck in the early 90s [16]. It refers to any automatic or semi-automatic process to construct a knowledge representation from a data source. OL processes can be classified based on the latter sources, considered structured, semi-structured, or unstructured. The boundaries between each category are subjective. However, some globally agreed examples of each category are raw texts such as news articles for unstructured sources, web pages for semi-structured ones, and relational database content for structured ones [17].

In the literature, we encounter different terms related to OL, which are essential to distinguish: Knowledge Graph (KG) construction, knowledge acquisition or ontology population. To the best of our knowledge, though OL and KG construction techniques have much in common, they differ in their goal. KG construction focuses on extracting known entities of interest and building relations and rules based on one or more sources of knowledge. In contrast, OL does not need any knowledge or entities known beforehand. Knowledge acquisition can be seen as a model construction

[1] https://gitlab.insa-rouen.fr/msesboue/ontology-learning/ (Accessed on June 1, 2023)

process where information is extracted and formalised iteratively [2]. As such, OL is one of the possible knowledge acquisition approaches. Finally, ontology population processes assume the existence of an ontology schema or at least define some classes and focus on extracting instances from a knowledge source [6].

Most OL methods are developed to assist the knowledge engineer in ontology construction. In this work, we focus on learning an application ontology. Hence, though not restricted to it, OLAF aims to learn an MVO to be integrated into the first ontology-based system iteration. Our framework tackles the knowledge acquisition component of the operational architecture for Knowledge Graph-based systems introduced in [26]. We seek to minimise human intervention as much as possible and consider the final application during the construction phase. Therefore, we take this into account in the OL framework.

2.2. *Ontology learning from text*

In the work presented here, we focus on techniques based on raw text sources, i.e., an unstructured source of knowledge. These methods are called ontology learning from text and can be seen as a reverse engineering task [6]. OL from text encompasses multiple sub-tasks. We are interested in automated OL techniques that provide a complete pipeline from raw text to structured knowledge representation rather than particular methods focusing on a specific part of the process. As most knowledge is embedded in the vast amount of available text content, many automatic knowledge extraction pipelines include and rely on NLP techniques. Other components are external knowledge sources, e.g., Wikidata [31], ConceptNet [27], or existing standard and company internal classifications.

Text2Onto [7] is a well-known framework for OL. Rather than committing to a particular technology, the ontology structure is represented at a metalevel by defining modelling primitives. The Text2Onto system also introduces the idea of Probabilistic Ontology Models to consider uncertainty while learning ontology. At the same time, it enables the integration of various modules. The result can be translated into multiple ontology representation languages. A significant advantage of this framework is its graphical interface for users to visualise the knowledge model. It is implemented as a plugin for the NeOn project[2] based on GATE [8]. Unfortunately, the Text2Onto implementation is not maintained anymore.

To the best of our knowledge, Text2Onto is the most flexible system, but other methods have been developed using the principle of division by task with different algorithms. They are not frameworks but specific pipelines. For example, OntoLearn [25] and OntoGain [10] are two statistical-based methods. The first one is largely based on the WordNet[3] knowledge source, which is used for terminology extraction and semantic interpretation. The result is a specialised view of WordNet. The second deals with multi-word terms and compares taxonomy construction algorithms and non-taxonomic relation acquisition strategies on two corpora. In [29, 19, 22], different techniques are used respectively based on an iterative focused data crawler, a concept-relation-concept tuple extraction and topic modelling. LExO [30] focuses on expressive ontologies suitable for reasoning. Syntactic and statistical classification methods are used to identify axioms and evaluate their consistency.

Some approaches restrict themselves to certain relations, algorithms, or tools. They are not comparable to our framework but can be included for particular sub-tasks. For example, ASIUM [11] implements a bottom-up clustering approach. Likewise for Mo'K [3], which additionally provides multiple parameters to tune. OntoLT [5] and SPRAT [24] are tool-specific implementations of methods leveraging patterns and mapping rules. FRED[4] [9] considers explicitly the use case of converting text into an RDF/OWL ontology within the context of Linked Data[5]. OntoCmaps [33] differs from other methods as it uses graph theory and provides a visualisation tool.

The majority of OL methods have much in common. The overall process is broken down into several small steps that are often identical from one method to another. For each step, algorithms are identified as more or less relevant and thus popular. The choice of the algorithms largely depends on the corpus and the use case. In the following, we will not specify OL from text for convenience and use OL as we only focus on text knowledge sources.

[2] http://neon-toolkit.org/wiki/Main_Page.html (Accessed on June 1, 2023)
[3] https://wordnet.princeton.edu/ (Accessed on June 1, 2023)
[4] http://wit.istc.cnr.it/stlab-tools/fred/ (Accessed on June 1, 2023)
[5] https://www.w3.org/standards/semanticweb/data (Accessed on June 1, 2023)

2.3. The ontology learning layer cake

Cimiano introduces the ontology learning layer cake [6] to organise the various tasks involved in the process of OL. Even though the 8 layers may convey a sense of order, it might only sometimes be the case. The different layers of the architecture are presented below in the order they are introduced in [6, 4].

1. Term extraction methods focus on extracting linguistic realisations of domain-specific concepts.
2. The synonym extraction layer aims at acquiring semantic term variants with sense disambiguation and domain-specific synonym identification.
3. Concept extraction processes include extracting informal definitions, that is, the text description of the concept from terms and synonyms.
4. The concept hierarchisation step tries to find taxonomic relations between concepts.
5. The relation extraction processes focus on discovering non-taxonomic relations.
6. Relation hierarchisation operations intend to order relations potentially linked to each other.
7. Axiom schemata techniques identify generic rules among concepts and relations.
8. General axioms learning processes try to identify more complex relationships and connections to obtain logical implications.

This model is applied in most papers dealing with OL. Only some of the tasks are necessarily performed, but a clear division into stages can be identified in the layer cake. This architecture is also used in benchmarks [1, 4] where OL techniques are compared for the different sections of the layer cake and their nature, i.e. linguistic, statistical or logical. Eventually, we have to point out that the result obtained on the whole pipeline depends on the ones obtained at each intermediate step of the process and the combination of steps chosen.

2.4. Evaluation

Ontology evaluation uses metrics to assess the ontology produced against the intended knowledge model. The result of the evaluation implicitly describes the produced ontology [20]. As there are many possible OL techniques, it is essential to have comparison criteria for the obtained knowledge representations. This enables us to choose the optimal technique and verify the content of the learned ontology.

Four evaluation techniques are commonly used, as described in [17, 1, 32]. The golden standard-based evaluation assesses and compares the learned ontology through a reference one. The application-based evaluation is task-oriented, exploiting an ontology explicitly built for an application to perform some task. This evaluation technique assesses the ontology quality based on the application performance, independently of ontology structural properties. The data-driven or corpus-based evaluation uses domain-specific knowledge sources to measure the built ontology coverage of a particular domain. Finally, the human or criteria-based one defines some qualitative or quantitative indicators and assesses the ontology against each to compute numerical scores. Regardless of the evaluation type, a set of usual metrics is used in the literature. In [21], Khadir et al. summarise them as consistency, completeness, conciseness, expandability, and sensitiveness. Each of the four described evaluation methods is more or less appropriate to assess each criterion. A domain expert makes the evaluation in [25] and most papers studied in [32]. In [10], OntoGain is evaluated using different evaluation techniques, whereas OntoCmaps [33] evaluation relies on golden standard assessment. The authors implement a data-driven evaluation and a comparison with hand-crafted ontologies. In [29], authors rely on criteria-based evaluation.

Another critical aspect of OL is process automation. Automation of OL tasks can be understood as the degree of human-in-the-loop, i.e., how much human intervention is needed. Several metrics could be used to quantify the latter. We could count the number of steps requiring significant human intervention. The type of human intervention could be qualified to differentiate between long and tedious steps, e.g., labelling, and more straightforward steps, such as verification. Among methods presented in section 2.2, the ones presented in [7, 10, 29, 19, 22] do not include any task requiring a human implication, although [19, 22] suggest a manual evaluation of the results. It is also the case in [25], where one step involving a human is needed to choose the relations considered and tag them in the dataset.

3. Ontology learning framework

The literature review presents multiple OL techniques. Each paper establishes a different combination of tasks leading to the creation of an ontology. However, OL is a task that still presents substantial challenges [21].

Automation is a limitation as it often affects the performance of the ontology produced. The lack of acceptance for uncertainty can be the reason for remaining human involvement in OL tasks. Hence, we focus on OL approaches with a minimum degree of human-in-the-loop to build a MVO that could be enhanced in future project iterations. The considered techniques span from data pre-processing to axiom definition and allow for more noise to justify a higher level of automation.

Another critical limitation is that existing OL pipelines focus more on the ontology than its application use. The latter usage intent influences the ontology structure and must therefore be taken into account during the OL process. In our work, we consistently keep in mind the knowledge representation target application to exploit the framework fully, as we will see in figure 1.

As mentioned in [33], it is essential to check that the proposed framework is domain independent. The fact that an ontology depends on the system in which it is integrated should not prevent the development of a generic creation method. Most existing methods impose non-iterative rigid step order and restrictive algorithms combination. Therefore, we built OLAF for different types of applications combining sub-tasks modularly.

We present OLAF, our ontology learning framework designed to solve the major issues by focusing on learning a MVO. We describe the conceptual terminology definition before detailing our automatic text-based OL framework.

3.1. Terminology

We adopt the vocabulary below in our implementation and the rest of this paper. To illustrate the terminology introduced, we will use the sentence: *"In the future, research should explore merging symbolic and sub-symbolic approaches to semantics."*.

We refer to tokens as the atomic pieces of text resulting from the tokenisation step. In the above sentence, tokens could be <In>, <the>, <future>, <,>, <research> etc. Then, a term denotes a sequence of one or more tokens of interest in the corpus. It can be defined as the linguistic realisation of a domain-specific concept or a relation label. They are extracted during the term extraction phase. In our example, "research", "symbolic" and "sub-symbolic approaches" are terms. Nevertheless, the token <,> would not be one. The result of term extraction is a set of candidate terms, i.e. potential concept or relation linguistic realisations.

The literature on OL defines a synonym as a word or term that denotes the same concept. The associated terms are often found in external data sources, e.g., WordNet or ConceptNet. We found the name "synonym" misleading as it tends to convey the idea that those words should have exactly the same meaning. However, depending on the application, we could link a concept to multiple linguistic realisations close in meaning. Hence, we use the name *term enrichment*. Continuing with our example, "study", "analysis", or "investigation" could be added as a synonym for "research". We might want to add terms such as "scientific paper" depending on the application.

The definition of a concept is controversial. Thus, the extraction process is quite variable from one method to another, depending on the definition used and the use case envisioned for the ontology. We consider concepts a notion of a chosen granularity with related terms as linguistic realisations. Considering our sentence, we could define a concept for *research* with three linguistic realisations: "study", "analysis", and "investigation". Relations enable us to define and make explicit links between known concepts. A relation is defined as a triple, i.e., source concept, relation, and destination concept. It is oriented and labelled. The labels are either induced from text or predefined. Whereas the literature often distinguishes taxonomic and non-taxonomic relations, we introduce relations and metarelations. In our framework, relations are transversal connections whose labels are candidate terms extracted from the source corpus, e.g., "explore" in our illustrative sentence. Metarelations are deduced from linguistic or statistical information whose labels are manually predefined. Some examples of such manually labelled relations are generalisation relations, "related to" relations as there could be between "symbolic" and "sub-symbolic", or mereological relations. The automatic extraction of transversal relations is largely underdeveloped in the literature, mainly focusing on just-defined metarelations.

We distinguish between the ontology resulting from the OL processes and the language (e.g., OWL) or formalism used to represent it. Thus, we will denote the OL process result as the knowledge representation. It is nothing less than a set of concepts, relations (triples), and metarelations (triples).

3.2. OLAF architecture

As previously mentioned, ontology learning should be considered in a practical context. Thus, we build OLAF around this idea. The framework includes different steps in figure 1: pipeline building, optimisation and execution. Pipeline building involves choosing the most relevant algorithms according to the text corpus and the ontology use case(s). Seed ontologies can also link the ontology learned to an upper one. When the algorithms are chosen, they are combined in a generic format allowing the automation of the rest of the process to optimise and execute the constructed pipeline. The framework's output is the learned ontology which can be integrated into the final application. The process can be iterative, and the ontology can be updated through the framework depending on the ontology-based system.

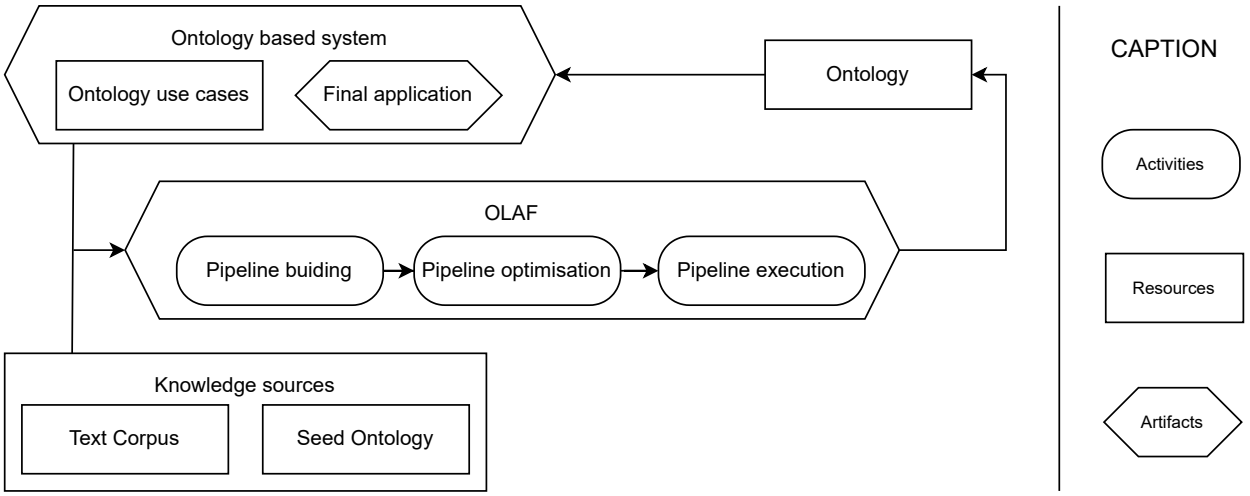

Fig. 1. Ontology Learning Applied Framework in a practical context

Considering OLAF modules, the OL layer cake idea is kept to ease the user learning curve. However, we break down the procedure into different steps, enable an iterative process and adapt the naming. Unlike the meaning conveyed by the ontology learning layer cake, the processes are neither all mandatory nor should they be executed in a determined order. However, the framework implicitly defines some constraints.

Reading figure 2 from left to right, OL begins with a corpus of text documents first handled employing various NLP methods. The *Pre-processing* module is necessarily the first one in an OL pipeline. Its methods start from a raw set of texts and turn them into structured content. The *Corpus* is the data container holding information related to the corpus data. The box *Ontology learning modules* is the framework heart. The modules use the *Corpus* to learn the final *Knowledge Representation* depicted on the right side of the figure.

The *Term Extraction* module extracts *Candidate Terms* before concepts and relations since the *Concept and Relation Extraction* module is based on them. Possible approaches for this task include pattern matching from Part Of Speech (POS) tagging techniques and scored-based methods (occurrences, C-value algorithm, TF-IDF). *Candidate Term Enrichment* is an optional step that adds information fetched from *External Knowledge Resources*, i.e., synonyms and other linguistically linked texts. The external resources are depicted on the top left side of the figure. Any OL module methods might make use of them. Examples of *External Knowledge Resources* are WordNet, ConceptNet and Large Language Models (LLM) from their contextualised word embeddings. The *Concept and Relation Extraction* module feeds the *Knowledge Representation* with concepts or relations built based on *Candidate Terms*. This component is a critical step of the architecture because it is necessary for the related ones. This extraction step can be done with a ConceptNet-based extraction, terms grouping based on synonyms, term cooccurrences-based extraction, similarity-based extraction based on LLM, Hierarchical Clustering or Formal Concept Analysis. The *Hierarchisation*

and Axiom Extraction modules can be used to enrich the *Knowledge Representation*. Hierarchisation enables ordering existing concepts and/or relations to learn taxonomies. Term Subsumption, Hierarchical Clustering and Formal Concept Analysis (FCA) are algorithms of interest for this task. The extraction of rules or axioms provides a higher level of abstraction to the *Knowledge Representation*. This task utility depends on the level of axiomatisation the ontology application requires. Rule-based axiom extraction and Inductive Logic Programming for axiom extraction are selected to perform this task. Regarding the described constraints, all processes in the OL modules in figure 2 can be repeated an indefinite number of times. The literature extensively studied the ontology lifecycle. Knowledge constantly evolves, and so should its computed representations.

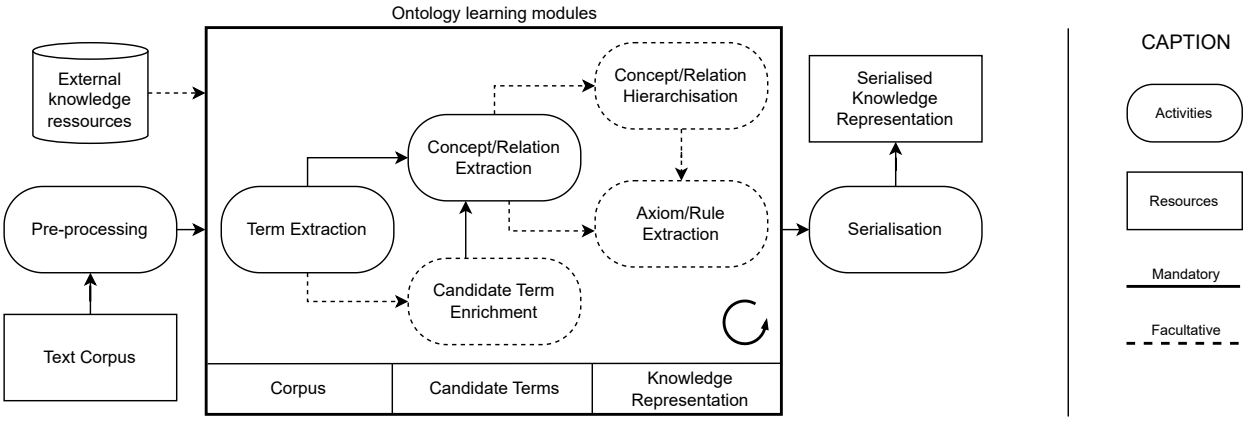

Fig. 2. Ontology Learning Process

We distinguish between two worlds living in parallel: the conceptual one and the linguistic one. In the conceptual one, *Candidate Terms* are the terms of interest extracted from the *Corpus*, which are used to extract concepts and relations. In the linguistic one, we create linguistic realisation from terms during concepts and relations extraction. Thus, terms are linked by their linguistic relations discovered during enrichment. Thereby, OLAF enables the learning of both conceptual ontology and lexicalised ontology. This last one can easily be obtained by creating a concept for each extracted term and a metarelation for each linguistic relation found. With these two worlds, ontology population would only need to use the linguistic realisation to fetch explicit instances.

The *Knowledge Representation* is the data structure containing concepts, relations, metarelations, and any other information to specify the learned ontology. It is typically empty at the beginning of the OL pipeline and is filled by the various processes. However, we can start with a seed ontology, as mentioned previously. Then, the initial *Knowledge Representation* container would hold it. To be used in an application, the knowledge data structure must be converted into a form interpretable by dedicated systems. This is the role of the *Serialisation* module. It can be defined for any representation model, such as the most used RDFS or OWL serialised in an RDF triples format and some proprietary ones, e.g., for Labelled Property Graph.

4. Experiments and results

This section introduces an experiment using OLAF for OL. Though our work focuses on the framework, we implement a pipeline and evaluate our approach on a corpus of technical product descriptions to ground it in practical reality. First, we introduce the corpus domain and the ontology application. Then, the ontology evaluation methodology is given before commenting.

4.1. Application domains

The text corpus used is a collection of 10000 Schneider Electric product descriptions from the Schneider Electric website[6]. We consider enhancing with semantic capabilities an existing search engine solely relying on keyword matching and statistical corpus analysis to retrieve the Schneider products.

To support catalogue search, we aim to build an application ontology structured as a set of domain taxonomies linked together by non-hierarchical relations, as presented in [23]. The corpus is indexed based on the concepts contained in the ontology. The document retrieval process leverages the same ontology by matching concepts in the user text query. The set of concepts is then expanded based on the relations. Finally, the enriched set of concepts is used for search in the corpus.

4.2. Evaluation methodology

The evaluation lets us measure the relevance of using automatic methods to structure knowledge. The learned ontology should represent the knowledge as well as possible to support the application better. We primarily rely on ourselves as domain experts for the ontology evaluation and apply the well-known CQ approach. As described in [20], CQs enable one to check whether the ontology defines enough knowledge to answer questions on the relevant application domain. We restrict our evaluation to a sample of the CQs we present below for convenience.

We aim to enhance a search engine employing an ontology. By default, we expect to find the main concepts mentioned in the one-sentence descriptions of each product range, e.g., *pushbutton*, *relay*, and *tower light* for the Harmony Schneider product range. Besides the latter obvious concepts, we define below a sample of the CQs:

Q1 Which are the Tesys products?
Q2 Which are the different kinds of control units?
Q3 What concepts are related to a specific product range?
Q4 Are protection devices part of the Easy9 product range?
Q5 Is a handle part of a switch?

4.3. Framework implementation details

To implement our framework, we use Python 3 and base the corpus pre-processing on spaCy[7]. We choose Python as it eases access to the vast Python community and its library ecosystem, particularly NLP tools and numerous machine learning libraries. We build the OL pipeline corresponding to our use case. Term extraction for concepts is based on the c-value algorithm [13], which is more relevant than pattern matching through POS tagging for technical data. The extracted candidate terms are enriched using WordNet with a filter on domains under interest. For concept extraction, candidate terms are grouped by synonyms and a hierarchy of concepts is built using a term subsumption strategy [12]. In the following section, we study the learned ontology according to the above-presented sample of CQs.

4.4. Results

The learned ontology contains 68 concepts and 495 metarelations. Since POS tagging is irrelevant to Schneider's technical text and relation extraction is currently based on it, the ontology does not contain any relation.

For *Q4* and *Q5*, the CQs can not be answered because the learned ontology does not contain suitable knowledge. This case often originates in term or/and concept extraction, e.g., *Q4* where the learned ontology does not define any *protection device* concept. One major challenge is also to infer knowledge only implicitly expressed in text. Such a challenge is one potential reason for missing knowledge to answer *Q5*.

Questions *Q1* and *Q3* can partially be answered. The learned ontology contains relevant knowledge but drown in noise. Different cases cause this situation. There can be too many relations or metarelations creating noise, especially

[6] https://www.se.com/ww/en/work/products/master-ranges/ (Accessed on June 1, 2023)
[7] https://spacy.io/ (Accessed on June 1, 2023)

hierarchical and *related to* ones, which are based on the cooccurrence of concepts and are threshold-dependent. This is the case for *Q3* where multiple *related to* relations are deduced. Though quite noisy on this part, the learned model shows concepts such as *residual current*, or *miniature breaker*, related to *Easy9*, a product range. Regarding the specific case of Tesys product range in *Q1*, a concept is extracted and some concepts denoting devices have a linguistic realisation containing the term "Tesys". Sorting can hardly be done automatically in post-processing and would require human intervention. This corresponds to high recall and low precision.

Considering *Q2*, the CQ can be answered. However, completeness can not be ensured for this kind of open question. It is the opposite of the previous remark, i.e., low recall and high precision.

Regarding noise and inaccurate knowledge found in the knowledge representation, we suspect a significant portion originates from the pre-processing stage. Technical texts are challenging for this part of the process. Unfortunately, the mistakes made reverberate throughout the process. Another explanation could be that metarelations extraction is mostly based on patterns in our pipeline. This approach is quite limiting and extracts many noisy links. Other techniques must be explored to use the learned ontology, and this particular use case might benefit from a seed ontology.

5. Conclusion

This paper introduces OLAF, a new framework for ontology learning from text focusing on creating a minimum viable ontology. The literature review displays the limitations of previous attempts. In addition to getting older, many approaches were developed as tools to help knowledge engineers in their work. As such, the human expert is a central part of the system. The proposed framework focuses on automation and is designed with very little, if any, human involvement in mind. Unlike most existing approaches, particular attention is brought to the learned ontology's final use case and framework modularity. A preliminary version of the framework has been implemented. Hence, we apply it to a real-world ontology use case, an ontology-based search engine. The learned ontology is evaluated using CQs.

The experiment results show that the framework provides very little automation on the evaluation task. Though CQs are a valuable tool to evaluate an ontology, they limit the assessment to the knowledge scope. The framework should be extended with further testing at the learned ontology level and each pipeline step to spot the potential sources of low-quality extracted knowledge. One interesting research area might be the adaptation of the software industry's "DevOps" concepts to knowledge management. The latter field is known as "SemOps"[8].

The experiments also demonstrate limited results on relation and axiom extraction tasks. Indeed, those sub-tasks are poorly addressed in the literature. For example, for relation extraction, methods mainly focus on known relations as "is-a" relations or domain-specific identified relations but do not consider the task of extracting unknown relations in data. An exciting perspective could be adding a specific component to our framework to perform this missing task. Regarding general axiom extraction, most frameworks and methods cited in section 2.2 recognise this task as a challenging and briefly addressed issue. LexO [30] is one of the few methods focusing on this part of the layer cake. In an alternative way, Inductive Logic Programming [1] and deep learning techniques to transform natural language sentences into description logic formulas [21] are highlighted as promising methods for automatically inferring axioms and could be integrated into the next version of the framework.

Finally, though already open source, our implementation is a first attempt. We aim to gather feedback and grow a community to develop and test multiple algorithms. Various satellite tools could be developed to enhance the framework implementation.

References

[1] Asim, M.N., Wasim, M., Khan, M.U.G., Mahmood, W., Abbasi, H.M., 2018. A survey of ontology learning techniques and applications. Database 2018. doi:10.1093/database/bay101. bay101.

[2] Bakker, R.R., 1987. Knowledge Graphs: representation and structuring of scientific knowledge.

[3] Bisson, G., Nédellec, C., Cañamero, D., 2000. Designing clustering methods for ontology building: The mo'k workbench, in: Proceedings of the First International Conference on Ontology Learning - Volume 31, CEUR-WS.org, Aachen, DEU. p. 13–28.

[8] https://www.semanticarts.com/the-data-centric-revolution-the-role-of-semops-part-1/ (Accessed on June 1, 2023)

[4] Buitelaar, P., Cimiano, P., Magnini, B., 2005. Ontology learning from text: An overview, in: Methods, Evaluation and Applications, Amsterdam: IOS Press. p. 3–12.

[5] Buitelaar, P., Olejnik, D., Sintek, M., 2004. A protégé plug-in for ontology extraction from text based on linguistic analysis, in: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (Eds.), The Semantic Web: Research and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 31–44.

[6] Cimiano, P., 2006. Ontology Learning from Text. Springer US, Boston, MA. chapter 3. pp. 19–34. doi:10.1007/978-0-387-39252-3_3.

[7] Cimiano, P., Völker, J., 2005. Text2onto, in: Montoyo, A., Muńoz, R., Métais, E. (Eds.), Natural Language Processing and Information Systems, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 227–238.

[8] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., 2002. A framework and graphical development environment for robust nlp tools and applications, in: Annual Meeting of the Association for Computational Linguistics.

[9] Draicchio, F., Gangemi, A., Presutti, V., Nuzzolese, A.G., 2013. Fred: From natural language text to rdf and owl in one click, in: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (Eds.), The Semantic Web: ESWC 2013 Satellite Events, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 263–267.

[10] Drymonas, E., Zervanou, K., Petrakis, E.G.M., 2010. Unsupervised ontology acquisition from plain texts: The ontogain system, in: Hopfe, C.J., Rezgui, Y., Métais, E., Preece, A., Li, H. (Eds.), Natural Language Processing and Information Systems, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 277–287.

[11] Faure, D., Nédellec, C., 1998. A corpus-based conceptual clustering method for verb frames and ontology acquisition, in: Velardi, P. (Ed.), LREC workshop on Adapting lexical and corpus ressources to sublanguages and applications, Granada, Spain. pp. 5–12.

[12] Fotzo, H.N., Gallinari, P., 2004. Learning "generalization/specialization" relations between concepts: Application for automatically building thematic document hierarchies, in: Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval, Paris, FRA. p. 143–155.

[13] Frantzi, K., Ananiadou, S., Mima, H., 2000. Automatic recognition of multi-word terms: the c-value/nc-value method. International Journal on Digital Libraries 3, 115–130. doi:10.1007/s007999900023.

[14] Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5, 199–220. doi:10.1006/knac.1993.1008.

[15] Guarino, N., 1998. Formal ontology and information systems, in: Frontiers in Artificial intelligence and Applications, Amsterdam: IOS Press. p. 3–15.

[16] Gutierrez, C., Sequeda, J.F., 2021. Knowledge graphs. Communications of the ACM 64, 96–104. doi:10.1145/3418294.

[17] Hazman, M., El-Beltagy, S., Rafea, A., 2013. A survey of ontology learning approaches. International Journal of Computer Applications 22, 36–43. doi:10.5120/2610-3642.

[18] Horrocks, I., Patel-Schneider, P.F., van Harmelen, F., 2003. From shiq and rdf to owl: the making of a web ontology language. Journal of Web Semantics 1, 7–26. doi:10.1016/j.websem.2003.07.001.

[19] Jiang, X., Tan, A.H., 2010. Crctol: A semantic-based domain ontology learning system. J. Assoc. Inf. Sci. Technol. 61, 150–168.

[20] Kendall, E.F., McGuinness, D.L., 2019. Ontology Engineering. volume 9. Morgan & Claypool Publishers LLC. doi:10.2200/s00834ed1v01y201802wbe018.

[21] Khadir, A.C., Aliane, H., Guessoum, A., 2021. Ontology learning: Grand tour and challenges. Computer Science Review 39, 100339. doi:10.1016/j.cosrev.2020.100339.

[22] Khemiri, A., Drissi, A., Tissaoui, A., Sassi, S.B., Chbeir, R., 2021. Learn2construct: An automatic ontology construction based on lda from texual data. Proceedings of the 13th International Conference on Management of Digital EcoSystems .

[23] li, Z., Raskin, V., Ramani, K., Mem, P., Asme, 2008. Developing engineering ontology for information retrieval. Journal of Computing and Information Science in Engineering - JCISE 8. doi:10.1115/1.2830851.

[24] Maynard, D., Funk, A., Peters, W., 2009. Sprat: a tool for automatic semantic pattern-based ontology population.

[25] Navigli, R., Velardi, P., Gangemi, A., 2003. Ontology learning and its application to automated terminology translation. IEEE Intell. Syst. 18, 22–31.

[26] Sesboüé, M., Delestre, N., Kotowicz, J.P., Khudiyev, A., Zanni-Merk, C., 2022. An operational architecture for knowledge graph-based systems. Procedia Computer Science 207, 1667–1676. doi:10.1016/j.procs.2022.09.224. knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES2022.

[27] Speer, R., Chin, J., Havasi, C., 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. arXiv.org doi:10.48550/ARXIV.1612.03975.

[28] Studer, R., Benjamins, V., Fensel, D., 1998. Knowledge engineering: Principles and methods. Data & Knowledge Engineering 25, 161–197. doi:10.1016/S0169-023X(97)00056-6.

[29] Venu, S.H., Mohan, V., Urkalan, K., Tv, G., 2017. Unsupervised domain ontology learning from text, in: Mining Intelligence and Knowledge Exploration, pp. 132–143. doi:10.1007/978-3-319-58130-9_13.

[30] Völker, J., Haase, P., Hitzler, P., 2008. Learning expressive ontologies, in: Ontology Learning and Population.

[31] Vrandečić, D., Krötzsch, M., 2014. Wikidata: A free collaborative knowledgebase. Commun. ACM 57, 78–85. doi:10.1145/2629489.

[32] Watróbski, J., 2020. Ontology learning methods from text - an extensive knowledge-based approach. Procedia Computer Science 176, 3356–3368. doi:10.1016/j.procs.2020.09.061. knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020.

[33] Zouaq, A., Gasevic, D., Hatala, M., 2011. Towards open ontology learning and filtering. Inf. Syst. 36, 1064–1081. doi:10.1016/j.is.2011.03.005.