NEOntometrics – A Public Endpoint For Calculating Ontology Metrics

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Achim Reiz[a*] and Kurt Sandkuhl[a]
[a] *Chair of Information Systems, Rostock University, 18051 Rostock, Germany*

Abstract. Ontologies, the cornerstone of the semantic web, are available from various sources, come in many shapes and sizes, and differ widely in their attributes like expressivity, degree of interconnection, or the number of individuals. As sharing knowledge and meaning across human and computational actors emphasizes the reuse of existing ontologies, how can we select the ontology that best fits the individual use case? How to compare two ontologies or assess their different versions? Automatically calculated ontology metrics offer a starting point for a quality assessment. In the past years, a multitude of metrics have been proposed. However, metric implementations and validations for real-world data are scarce. For most of these proposed metrics, no software for their calculation is available (anymore). This work aims at solving this implementation gap. We present NEOntometrics, an open-source, flexible metric endpoint that offers (1.) an explorative help page that assists in understanding and selecting ontology metrics, (2.) a public metric calculation service that allows assessing ontologies from online resources, including git-based repositories for calculating evolutional data, with an (3.) adaptable architecture to adopt new metrics quickly. We further take a quick look at an existing ontology repository that outlines the potential of the software.

Keywords: Ontology Metrics, Ontology Quality, Semantic Web, owl, rdf

1. Introduction

Ontologies facilitate the communication of meaning between human and computational actors. They allow to structure and contextualize data to detect implicit knowledge, access this knowledge using complex queries, and integrate and interlink data from various sources while facilitating a common understanding.

Over time, the semantic web community developed countless ontologies. To give a perspective, the vocabulary repository "Linked Open Vocabulary (LOV)" [1] contains 773 ontologies. The portal "ontologydesignpatterns.org" collects small, reusable ontology patterns and provides 236 artifacts. Bioportal [2], a large repository for biomedical ontologies, contains 996 ontologies. Moreover, many more ontologies are available on different sources like GitHub or private company repositories.

Nevertheless, while the number of developed ontologies is extensive, there are just a few means available to assess and control the quality of these artifacts. For the development team that likes to integrate an ontology into their system, comparing two or more different available ontologies that serve the same purpose is complicated. For the knowledge engineer, the missing assessment capabilities hinder the tracking of change impacts throughout the ontology lifetime on the ontology as a whole.

As we will show in the next section, the lack of means for quality assessments does not originate from a lack of proposed metrics. Over time, a variety of ontology quality frameworks have been developed. However, what is missing are practical implementations of these metrics.

This work aims at closing this gap by presenting a flexible, extensible metric calculation endpoint for rdf and owl-based ontologies. The software enables

*Corresponding author. E-mail: achim.reiz@uni-rostock.de

metrics to be calculated and retrieved using a graphical user interface (GUI) or an application programming interface (API). It further aids users in learning about different metrics, their calculations, and possible interpretations through an interactive explorer for ontology metrics. If several versions of an ontology are available, the development of the metric values over time can be tracked. From a user's perspective, we believe this software can contribute to a more intense use of ontology metrics to control the development process better. From a researcher's perspective, it will allow us to study how ontologies evolve and how the metrics perform in an actual application.

The paper is structured as follows: Section two summarizes the current state-of-the-art regarding ontology metrics and calculation software. Section three presents NEOntometrics with its architecture and usage of the API and GUI. In section four, we illustrate the use of NEOntometrics by presenting a case study showing what evaluations are possible using the service through a preliminary analysis of a calculated dataset. The final section concludes the paper.

2. Related Work

This section covers the previously published work relevant to this paper. Significant for our research are metric calculation proposals and ontology metric frameworks, covered in the first part of this section, and calculation implementations, covered in the second part.

There are many different evaluation methods available. Please note that we only consider criteria-based frameworks that allow for an automatic evaluation based on the structural attributes of the ontology. Metrics that need human intervention or additional input parameters are not considered relevant. That excludes evaluation methods based on a gold-Standard *(additional input parameter is a "perfect" ontology),* task-based *(additional input parameter is the task fulfillment level that an ontology can provide in a given context)*, or corpus-based *(additional input parameters are domain-related documents like a text corpus)*. An overview of these evaluation categories can be found in [3].

2.1. Related Quality Frameworks

Lozano-Tello and Gómez-Pérez published the Ontometric framework in 2004 [4]. It proposes evaluation attributes in five criteria: *(Development) tool, (ontology) language*, *context*, *methodology,* and *cost*. Arguably, some metrics have become obsolete due to standardization in the past years. In 2004, the web ontology language (owl) was just released, and other representations like OIL, DAML+OIL, and SHOE were still actively used. Here, Ontometric is targeted to make the influences of the languages explicit and comparable. Today though, regarding the category *languages*, rdfs-based ontologies can be considered state-of-the-art and are mostly compatible with each other. Further, the standardization decoupled the *tools* from the ontology. Thus, the tool capabilities do not influence the semantic artifact.

Other proposed elements, however, are still significant today and can be supported by an automatic calculation, like the metric *maximum depth* in the category *content.* While the framework's relevance today might be somewhat limited, Ontometric considerably influenced the newer frameworks.

Gangemi et al. proposed an ontology evaluation framework based on a semiotic meta-ontology O^2 that provides a formal definition for ontologies and their usage. Further, the authors define an ontology evaluation design pattern (oQual). Based on their O^2 definition, measurements assessing *structural*, *task*, *corpus*, and *usability*-based attributes are proposed [5]. A technical report by the same authors [6] further suggested 32 metrics in seven categories assessing mostly graph-related structures like *depth*, *width*, *modularity*, the *distribution of siblings*, or *tangledness*.

In 2005, Burton-Jones et al. presented the semiotic metric suite [7]. It comprises four main categories (*syntactic*, *semantic*, *pragmatic*, and *social* quality) and ten quality metrics. While some of these metrics are based solely on the structure of the ontology itself, others need further additional external information. Nine of these measurements, in theory, can be calculated automatically. Practically, some of the required data for some measures will probably not be available. Examples are the access count of the ontology or the links from other ontologies to the currently assessed one.

OQuaRE was first proposed in 2011 by Duque-Ramos et. al [8]. It was since used in several publications, always involving the core team of the proposing authors. OQuaRE offers 19 calculatable metrics and associates these metrics with quality dimensions like readability or accountability. Further, the framework ties metric results to quality ratings, thus providing an interpretation for the measurements. This holistic approach to quality is a unique characteristic.

However, during implementation, we experienced several heterogeneities in the metric definitions of the

framework, with metrics with the same name by the same authors being defined differently in the publications. While we made efforts to align the metrics [9], it remains yet to be seen if the quality implications stated by the framework can be observed in an empirical analysis. So far, no applications of OQuaRE have been made by authors that are not part of the team that proposed the framework.

Tartir et al. published 19 metrics in the OntoQA framework in 2005 and 2007 [10,11]. Further, the authors propose measurements applicable not to the ontology as a whole but to the elements in an ontology. Here, OntoQA defines class- and relation-specific measurements. The *relationship importance*, for example, is calculated for each relationship. While the framework does not provide a grading system for the metrics like OQuaRE, it aids the interpretation by describing how modeling decisions influence the metric results.

In his Ph.D. thesis, Vrandečić[12] gathered the state-of-the-art in ontology evaluation and introduced means to connect the various evaluation and improvement efforts. He condensed the quality proposals into eight quality characteristics *(accuracy, adaptability, clarity, completeness, computational efficiency, conciseness, consistency, and organizational fitness)* and proposed validation methods for these characteristics. These methods are primarily based on manual evaluation approaches. While he selected a few autonomous numeric calculation values, the thesis focuses primarily on providing methodologies for the knowledge engineer to improve ontologies in general without a particular focus on metrics.

2.2. Related Metric Calculation Software

As presented in the previous section, various metric frameworks have been developed over the past years. Some of these frameworks are merely theoretical in their proposals; others come with prototypical implementations. Further, tools that do not correspond to one of the proposed frameworks have been developed. The following section shows historical and current software for ontology metric calculation.

OntoKBEval by Lu and Haarslev [13] analyzes the structure of ontologies by providing graph-related measures like the number of levels or the number of concepts per level. The tool offers means to grasp clusters in the ontology and developed its own visualization “Xmas”-tree.

Tartir et al. developed a standalone java application for the OntoQA framework [11]. The application implements measures of the OntoQA framework, including metrics for the individual classes.

OntoCat, proposed by Cross and Pal [14], is a plugin for the Protégé editor and provides size- and structure-related metrics. They allow the assessment of the ontology as a whole but also provide metrics concerned with specific subsets of the given ontology.

Table 1

The availability of the developed metric software *(type: S: Standalone, P+: Protégé plugin, API: REST-API, WT: Web Tool)*

Tool	Date	Type	Available	Open Source	Ref.
OntoKBEval	2006	S	No	No	[13]
OntoQA	2005	S	Yes[1]	(No)[2]	[11]
OntoCat	2006	P+	No	No	[14]
S-OntoEval	2008	S		No	[15]
Protégé	2015	P	Yes		[16]
OQuaRE	2018	WT, API	Yes	No	[17]
OntoKeeper	2017	WT	No	No	[18]
OOPS	2012	WT, API	Yes[3]	No	[19]
OntoMetrics	2015	WT, API	Yes[4]	No	[20,21]

S-OntoEval by Dividino et al. [15] serves as a calculation tool for, among others, the framework of Gangemi et al. Its primary focus is on structural evaluation. However, the tool also calculates usability based on annotations and task performance based on ontology querying.

The Protégé editor [16] offers basic metrics on its landing page that counts the usage of owl-specific language constructs like the number of object property domain declarations or the number of classes.

The developers of the OQuaRE framework introduced a web tool to calculate their proposed metrics. It integrates a statistical correlation analysis of the metrics and a web service. Unfortunately, the tool suffers from the same issues as the framework [9].

[1] https://github.com/Samir-Tartir/OntoQA

[2] The binary .jar files are available under CC license. The source code itself is not made public.

[3] http://oops.linkeddata.es/

[4] https://ontometrics.informatik.uni-rostock.de/, opi.informatik.uni-rostock.de/

Amith et al. developed the Semiotic-based Evaluation Management System (SEMS), later renamed OntoKeeper [18], which implements the semiotic suite by Burton-Jones et al.

The "OntOlogy Pitfall Scanner" (OOPS) by Poveda-Villalón et al. [19] approaches automatic ontology evaluation differently: They do not calculate ontology metrics but detect common modeling pitfalls like the use of *is* relationship instead of *rdfs:subClassOf* or wrongful equivalent relationships.

OntoMetrics, first developed by Lantow [21], is a web service for calculating several ontology metrics. It covers most of the OntoQA and oQual ontology metrics and integrates the owl-based axiom counts that are also part of Protégé. It was later extended with a web service by Reiz et al. [20].

2.3. The Need for Another Calculation Tool

As the previous section has shown, many frameworks and tools have been developed over the past years. That raises the question of whether a new calculation tool is necessary. We argue that our application fills essential gaps:

Missing Practicality. Most of the developed tools are no longer available. Even if they are available, their usability is often low. Many of the tools were used for the authors' evaluation efforts and do not come with a state-of-the-art user interface or support a variety of ontology sources. Further, most of the software is not maintained. This problem is amplified by the fact that most of the software is:

Closed Source. None of the evaluated tools is fully open source. Not only hinders this reproducibility. It also prevents the community from maintaining the software and building on this previous research. If there is a need for another kind of evaluation, one has to start from scratch. We, thus, argue that the closed source leads to:

Isolation. The implementation efforts have stayed mainly isolated from one another. Hardly any tool has reached a broad acceptance within the community, and the ontology evaluation efforts of researchers using different tools are often not comparable. While there is a consensus that ontology evaluation is meaningful, there is no common understanding of how to do it.

3. NEOntometrics

Our proposed approach targets to solve many of the previously named challenges. The software provides a frontend with a state-of-the-art user experience, comes with a GraphQL endpoint, calculates a variety of metrics, is quickly extensible through the use of an ontology for creating and describing calculated metrics, and is fully open source[5]. We, thus, believe it has the potential to become a community-accepted tool for calculating ontology metrics.

The following section details the software itself: it presents the different components of the service, how they interact, and the underlying development decisions. We also present how our ontology-based metric calculations are extensible for future usage. Afterward, we give an overview of how to put the software to use.

3.1. The Architecture of the Metric Calculation

One design goal was to create a flexible application for integrating new metrics. A researcher shall be able to adapt the application to their individual needs and quickly implement the required metrics.

To achieve this adaptability, we do not directly implement the metrics of the various frameworks but decompose them into their building blocks. For example, the metric *axiom/class ratio* is not calculated at the time of the ontology analysis. Instead, their building blocks *axioms* and *classes* are saved in the database. The compositional values are then calculated at the time of querying.

Figure 1 presents an example of this kind of decomposition and its representation in the ontology[6]. *Elemental Metrics* contain the atomic elements that build the further compositions. For *OntoQA Class Inheritance Richness*, the *Elemental Metrics* are *Classes* (the number of classes) and *Sub Class Declarations*. The ontology further specifies mathematical relationships between the metrics. In the given example, *OntoQA Class Inheritance Richness* is the *subClassOf (divisor only Classes) and (numerator only Sub Class Declarations)*. The *Elemental Metrics* are connected to metric instances named identically to the implementation names in the calculation service and the elements in the database. In the example of the *Sub Class Declarations*, this element has a relationship *implementedBy value subClassOfAxioms*.

All elements have rich annotations, providing human-centered meaning to the metrics. Additionally,

[5] https://github.com/achiminator/NEOntometrics – Upon acceptance, we will create a permanent ZENODO-DOI as well.

[6] http://ontology.neontometrics.com – *link to ontology in Github-repository*

links to further online resources or scientific publications are provided. These annotations are the foundation for the *Metric Explorer*, where users find guidance on the available metrics.

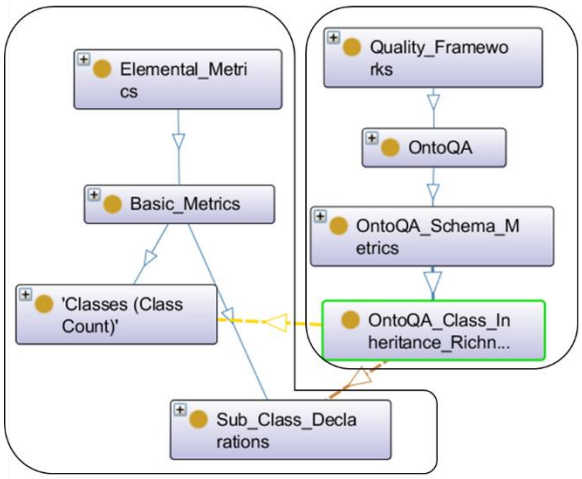

Figure 1. The metric ontology: The left side shows the atomic *Elemental Metrics*, the right side represents the custom defined metric frameworks.

New metrics that build upon the available *Elemental Metrics* can be modeled in the ontology. Upon start, the application will make these custom metrics automatically available in the front- and backend.

3.2. The Architecture Of Application

The application is based on a dockerized microservice architecture and consists of five components: the calculation-unit OPI (Ontology Programming Interface), the API, the worker application, a database for storing the calculated metrics, and a Redis interface for queueing. The API and worker share a common codebase. Figure 2 depicts the interaction of the involved services.

The **frontend** provides the GUI. It is written using the multi-platform UI language *flutter* with its underlying client language *dart*[7]. Upon loading, the frontend first queries the API for available ontology metrics based on the metric ontology. This data fills the help section *Metric Explorer*, which allows users to inform themselves about the various available metrics and the options for the calculation page. Afterward, the user can retrieve the requested ontology metrics or put them into the queue if they do not yet exist.

The **API** is the *django*-based[8] endpoint for accessing already calculated metric data or requesting the analysis of new repositories. During the startup of the software, the application queries the metric ontology. It builds the frontend data and dynamically creates calculation code to provide the measurements of the frameworks that build upon the *Elemental Metrics*. After startup, a client can exploit GraphQL to check whether the data he requests exists already in the database. If so, he is able to retrieve all the selected metrics for a given repository. If not, it is possible to put the calculation of a given repository in the queue and track its progress.

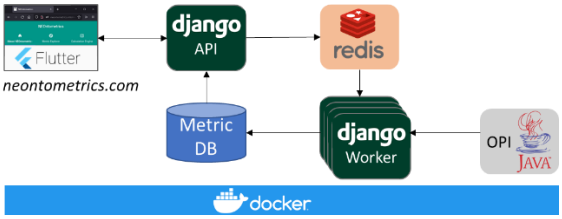

Figure 2. The NEOntometrics microservice architecture

The **worker** is responsible for the calculation of the metrics itself. It checks whether jobs are available in the **scheduler** Redis database. If that is the case, it starts the analysis by first downloading the repository, then iterating through every file and commit, analyzing the owl ontologies using the **OPI** metrics endpoint. Afterward, the calculation results are stored in the **database**. The scheduling mechanism is based on *django-rq*[9]. Even though it shares a code base with the api, it runs as a separate application. The number of parallel calculations can be scaled by increasing the number of workers.

The calculation service **OPI** is responsible for calculating metrics out of ontology documents. While it is based on the calculation service published in [20], most underlying code has been replaced. The old application struggled with ontology files larger than 10 MB due to inefficient memory allocation, had no separation of the calculation of the *Elemental Metrics*, and the *composed metrics* of the metric frameworks, and lacked support for reasoning. The old application was designed as standalone software, while the new calculation engine is hidden from the user and only accessed by the **API**.

[7] https://flutter.dev/, https://dart.dev/
[8] https://www.djangoproject.com/, https://www.django-rest-framework.org/
[9] https://python-rq.org/patterns/django/

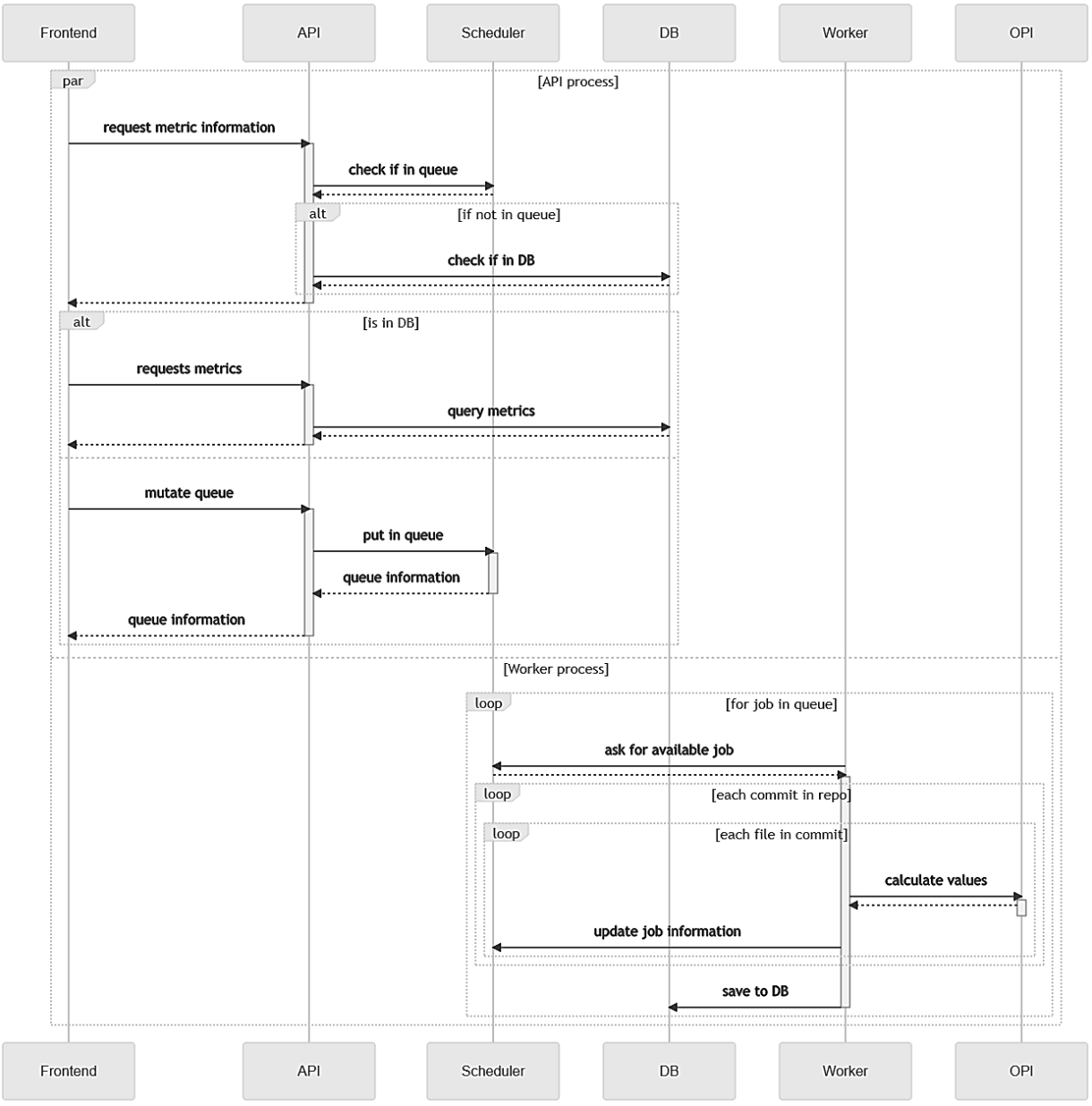

Figure 3. The process of analyzing and retrieving ontologies with NEOntometrics (without application startup)

The backend application utilizes two languages: The **API** is written in python, and the calculation service **OPI** builds on Java. While this adds complexity to the application design, it allows the integration and use of the OWL API[10]. This library provides many convenient functions for handling OWL and RDF-Files like an entity searcher or the automatic calculation of some axiom-based metrics.

The calculation and retrieval process as a whole is depicted in Figure 3. At first, the frontend requests whether an ontology is already known in the system. Afterward, it either returns the queue information to the end-user or starts another request for the ontology metrics. At the same time, the worker applications and OPI work on the queued tasks.

[10] http://owlcs.github.io/owlapi/

3.3. The Metric Explorer

The page *Metric Explorer* (Figure 4) is a repository of available metrics in NEOntometrics and beyond. The two main categories are *Elemental Metrics* and *Quality Frameworks*. The former contains the underlying atomic measurements of the ontologies. The authors of the software create all information shown in this category. *Quality Frameworks*, on the opposite, present the ontology quality metrics developed by other researchers, like the OntoQA Metrics [11] by Tartir et al., shown in section 2.2. Here, all information originates from the authors of the given frameworks.

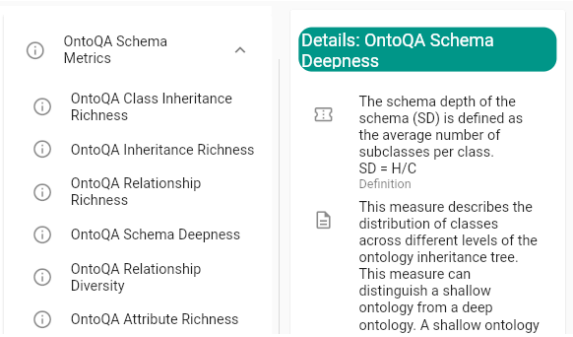

Figure 4. The metric explorer page in NEOntometrics

The page provides information on five categories (though not all of them are filled for all the metrics). *Metric Definition* contains the formal definition of the metrics and how they are calculated, while *Metric Description* supplements a more human-readable explanation and, at times, an example. *Metric Interpretation* guides practical usage. *Calculation* explains their decompositions into the *Elemental Metrics* using the metric names that are returned by the API, and *seeAlso* links to further resources like the corresponding papers or additional reads.

3.4. Using the Calculation-Frontend

The tab *Calculation Engine* as shown in Figure 5 is the main entry point for the metrics calculation. The end-user first selects the required metrics. Hovering over the elements shows additional information to the metrics. Checking the box on the right enables the calculation of all metrics of a given category. The "Already Calculated" button shows the repositories that are already stored in the database. While these repositories can be a starting point for further exploration, the user can also place a URL in the textbox that points to a new git repository or an online location of an ontology file.

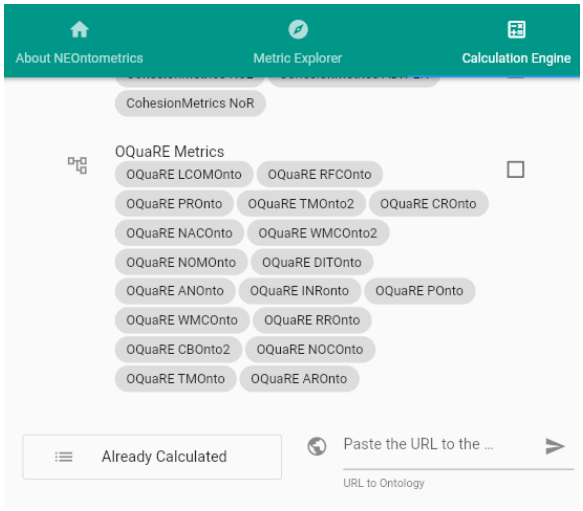

Figure 5. Selecting ontology metrics in NEOntometrics

A click on the arrow starts the metric request. If the metric is unknown in the system, the application asks to queue the calculation task. If it is already in the queue, a notification informs of the progress. As soon as the data is analyzed, a click on the arrow leads to the metric results presented as a paginated table, representing the metric values for the different ontology versions. A drop-down menu in the header allows for selecting the various ontology files, the download button exports the metrics into a .csv.

3.5. Using the API

The GraphQL is accessible through a browser on a GraphiQL interface or by using any other GraphQL client. Typically, one would first query the node *queueInformation* to check whether the data is already in the database or the calculation queue, then run a mutation to put the data into the queue or fetch the data from the node *getRepository*. The GraphQL endpoints further provide documentation on the various available requests and possible return values, thus enabling the guided development of new queries.

4. A Case Study for Analyzing Ontologies with NEOntometrics

The following section presents examples of analysis that are possible with NEOntometrics. We analyzed

the *Evidence and Conclusion Ontology (ECO)*. *ECO* captures the biological coherences like "gene product X has function Y as supported by evidence Z" [22]. Please note that the NEOntometrics authors have no affiliation with the authors of the ontology nor with the biomedical field of research. Further, the goal of the section is not to evaluate quality but to observe the development of the ontology over time, to give an impression of possible assessments.

The corresponding repository has 856 commits in 17 ontology files. For this analysis, we were interested in the main ontologies in this repository. Thus, we only assessed the ontologies in the root structure, resulting in three ontology files. *eco.owl* with 89 versions, *eco-basic.owl* with 44, and *eco-base.owl* with 45 versions.

We first examined the axiom count of the ontologies and then used Tartir et al.'s OntoQA framework [10,11] for further analysis. The corresponding source files for the analysis in Jupyter Notebooks are available online[11].

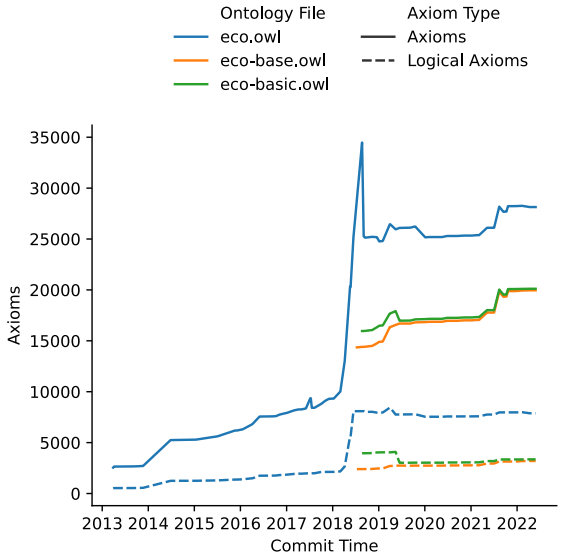

Figure 6. The change of axioms over time in the ECO-ontologies

The first analysis is concerned with the development of the ontology size. Figure 6 presents the three ontology files in their different versions and plots the development of axioms with the time. While the solid line represents all axioms overall, the dashed line only accounts for such that incorporate a logical meaning in owl syntax. The difference between the dashed and the solid lines are, thus, annotations or custom-defined properties.

A first insight of the chart in Figure 6 is the variances of the logical axioms and the axioms in general. While the size of the ontology overall fluctuates intensely, the number of parts of the ontology that incorporates logical meaning stay relatively stable. One significant spike occurred between 2018 and 2019, which we will scrutinize further. Analysis reveals that a more extensive restructuring of the ontology drives this increase in logical axioms.

At first, the classes in *eco* doubled from around 900 to first a little over 2000, then further to over 3000. The number of defined object properties jumps from three to above 50, and the relation on classes through object properties increases from 350 to almost 2500, then drop to around 1600. This change event also marked the introduction of *eco-base* and *eco-basic*.

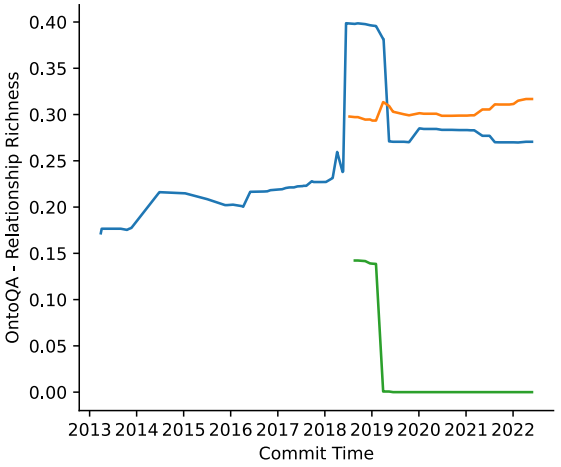

Figure 7. The development of OntoQA's Relationship Richness

Figure 7 and Figure 8 show the relationship richness and schema deepness defined in the OntoQA framework. The former is the number of non-inheritance relationships divided by the sum of non-inheritance relationships and inheritance relationships; the latter is the number of subclasses per class [11]. They directly reflect the changes named above:

After an initial increase due to the rise in object properties, the relationship richness of *eco* drops with the increase in classes and subClassOf statements. Also, object properties were introduced. Later on, a decline in object properties, combined with the further

[11] Data: https://github.com/achiminator/NEOnto-Evaluation. Upon acceptance, we will upload the data permanently to Zenodo.

increase in classes and subClassOf statements, reverses the growth partially.

Figure 8 provides more insights into the role of subclass relationships, as it is calculated by dividing the number of inheritance relations by the number of classes. At first, a lot more subClassOf relationships than classes were introduced. However, a little later, the number of subClassOf relations stagnated, even getting smaller. The number of classes, however, increased steadily. This suggests that the rebound in the relationship richness is driven more by the decline in object properties than the increase in subClassOf relationships.

Figure 7 and Figure 8 reveal that, while the number of logical axioms is more or less stable, the underlying logical attributes of the ontology that constitute how the ontology is structured are still subjected to changes.

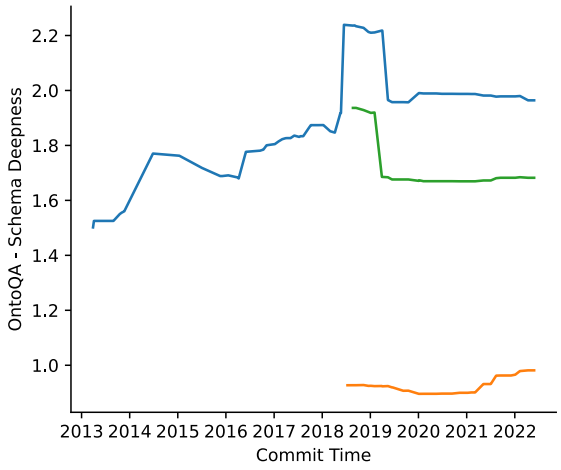

Figure 8. The development of OntoQA's Schema Deepness

There are many more aspects that one could analyze for the given repository. For instance, we have only analyzed one file thoroughly and examined the impact of one main change event. As the last diagram already indicates, many more fluctuations are worth looking at. The variations affect the relationships between non-hierarchical and hierarchical relationships and classes and graph-related structures like the width or depth, individuals, or data properties. NEOntometrics provides the necessary tool to examine these questions further.

5. Conclusion

Ontologies are in use in various applications, facilitating meaning between human and computational actors and enabling these actors to harness the full potential of structured knowledge. The rising number of developed ontologies emphasizes the need for quality control.

The research community has identified this need for quite some time. Many frameworks for controlling the quality of ontologies have been proposed, assessing several different attributes of an ontology. However, implementations of these frameworks have been scarce. The missing software hinders the research progress: While the definition of measurements is an important step, it is crucial to put these metrics into use.

The application proposed in this paper aims at closing this gap. We presented NEOntometrics, an open-source software to calculate ontology metrics. The application integrates several metric frameworks and is easily extensible. It is possible to analyze the development of metrics over time by analyzing git-based ontology repositories. Further, the user can inform themselves of available calculations and possible implications using an interactive metric explorer. The ontology metrics can be calculated and retrieved either using a graphical user interface or a GraphQL-API. While the former is targeted at the knowledge engineers, the latter shall allow developers of semantic-based applications to integrate metrics into their software.

The preliminary case study shown in the previous section just briefly presents the possible analysis applications. There are many more aspects worth looking at, like the comparison of typical development processes in different fields (e.g., industrial vs. biomedical ontologies), the usefulness of the proposed frameworks, and the modeling preferences of different persons, to name a few.

Further research will be concerned with analyzing the metric data itself. However, we will continue to work on the application, to add more metric sources, metrics on the specific elements within an ontology like class-specific and relation-specific measurements, and integrate visualization capabilities. In this regard, we are interested in the aspects that the community would like to see implemented.

In the long term, we hope that NEOntometrics impacts the use and research of ontology metrics and believe it can help us empirically understand ontology modeling better.

References

[1] P.-Y. Vandenbussche, G.A. Atemezing, M. Poveda-Villalón, and B. Vatant, Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web. Semant Web **8** (2016), 437–452.

[2] P.L. Whetzel, N.F. Noy, N.H. Shah, P.R. Alexander, C. Nyulas, T. Tudorache, and M.A. Musen, BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. Nucleic Acids Res **39** (2011), W541-5.

[3] J. Raad and C. Cruz, A Survey on Ontology Evaluation Methods. In *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management: Lisbon, Portugal, November 12 - 14, 2015*, A. Fred, ed. SciTePress, Setúbal, 2015, pp. 179–186.

[4] A. Lozano-Tello and A. Gómez-Pérez, ONTOMETRIC: A Method to Choose the Appropriate Ontology. Journal of Database Management **15** (2004), 1–18.

[5] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann, Modelling Ontology Evaluation and Validation. In *The semantic web: research and applications: 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11 - 14, 2006 ; proceedings*, Y. Sure, ed. Springer, Berlin, 2006, pp. 140–154.

[6] A. Gangemi, C. Catenacci, M. Ciaramita, J. Lehmann, R. Gil, F. Bolici, and Strignano Onofrio, *Ontology evaluation and validation: An integrated formal model for the quality diagnostic task*, Trentino, Italy, 2005.

[7] A. Burton-Jones, V.C. Storey, V. Sugumaran, and P. Ahluwalia, A semiotic metrics suite for assessing the quality of ontologies. Data & Knowledge Engineering **55** (2005), 84–102.

[8] A. Duque-Ramos, J.T. Fernández-Breis, R. Stevens, and N. Aussenac-Gilles, OQuaRE: A square-based approach for evaluating the quality of ontologies. Journal of Research and Practice in Information Technology **43** (2011), 159–176.

[9] A. Reiz and K. Sandkuhl, Harmonizing the OQuaRE Quality Framework. In *Proceedings of the 24th International Conference on Enterprise Information Systems*, 2022, pp. 148–158.

[10] S. Tartir and I.B. Arpinar, Ontology Evaluation and Ranking using OntoQA. In *International Conference on Semantic Computing, 2007: ICSC 2007 ; 17 - 19 Sept. 2007, Irvine, California ; proceedings ; [held in conjunction with] the First International Workshop on Semantic Computing and Multimedia Systems (IEEE-SCMS 2007)* IEEE Computer Society, Los Alamitos, Calif., 2007, pp. 185–192.

[11] S. Tartir, I.B. Arpinar, M. Moore, A.P. Sheth, and B. Aleman-Meza, OntoQA: Metric-Based Ontology Quality Analysis. In *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, D. Caragea, V. Honavar, I. Muslea, and R. Ramakrishnan, eds., 2005.

[12] D. Vrandecic, *Ontology Evaluation*, Karlsuhe, 2010.

[13] Qing Lu and Volker Haarslev, OntoKBEval: A Support Tool for DL-based Evaluation of OWL Ontologies. In *OWL: Experiences and Directions*, B. C. Grau, P. Hitzler, C. Shankey, and E. Wallace, eds., 2006.

[14] V. Cross and A. Pal, OntoCAT: An Ontology Consumer Analysis Tool and Its Use on Product Services Categorization Standards. In *Proceedings of the First International Workshop on Applications and Business Aspects of the Semantic Web*, E. Simperl, M. Hepp, and C. Tempich, eds. CEUR-WS.org, Aachen, DEU, 2006, pp. 1–15.

[15] R. Dividino, M. Romanelli, and D. Sonntag, Semiotic-based ontology evaluation tool S-OntoEval. In *Proceedings of the International Conference on Language Resources and Evaluation*, N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, and D. Tapias, eds., 2008.

[16] M.A. Musen, The Protégé Project: A Look Back and a Look Forward. AI Matters **1** (2015), 4–12.

[17] M. Quesada-Martínez, A. Duque-Ramos, M. Iniesta-Moreno, and J.T. Fernández-Breis, Preliminary Analysis of the OBO Foundry Ontologies and Their Evolution Using OQuaRE. In *Informatics for health: Connected citizen-led wellness and population health*, R. Randell, R. Cornet, C. McCowan, N. Peek, and P. J. Scott, eds. IOS Press, Amsterdam, Washington DC, 2017, pp. 426–430.

[18] M. Amith, F. Manion, C. Liang, M. Harris, D. Wang, Y. He, and C. Tao, OntoKeeper: Semiotic-driven Ontology Evaluation Tool For Biomedical Ontologists. J Biomed Semantics **8** (2017), 1614–1617.

[19] M. Poveda-Villalón, A. Gómez-Pérez, and M.C. Suárez-Figueroa, OOPS! (OntOlogy Pitfall Scanner!). Semantic Web and Information Systems **10** (2014), 7–34.

[20] A. Reiz, H. Dibowski, K. Sandkuhl, and B. Lantow, Ontology Metrics as a Service (OMaaS). In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, J. Filipe, D. Aveiro, and J. L. Dietz, eds., 02.11.2020 - 04.11.2020, pp. 250–257.

[21] B. Lantow, OntoMetrics: Putting Metrics into Use for Ontology Evaluation. In *Proceedings of the 8th IC3K 2016 International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, J. Filipe, D. Aveiro, and J. L. Dietz, eds., 2016, pp. 186–191.

[22] M. Giglio, R. Tauber, S. Nadendla, J. Munro, D. Olley, S. Ball, E. Mitraka, L.M. Schriml, P. Gaudet, E.T. Hobbs, I. Erill, D.A. Siegele, J.C. Hu, C. Mungall, and M.C. Chibucos, ECO, the Evidence & Conclusion Ontology: community standard for evidence information. Nucleic Acids Res **47** (2019), D1186-D1194.