NEOntometrics: A Flexible and Scalable Software for Calculating Ontology Metrics

Achim Reiz[1] and Kurt Sandkuhl[1]

[1] *Rostock University, 18051 Rostock, Germany*

Abstract
Metrics allow to empirically assess ontologies. They enable the knowledge engineer to quickly grasp changes and differences between two different ontologies or two different versions of one ontology and can guide developing or reusing decisions.
Calculating ontology metrics requires specialized software. Today, there is a lack of application support, as none of the previously developed software is open source, and most applications are not available anymore. This paper presents a flexible, scalable architecture for future-proof metric calculation software. We first depict the missing availability and lack of functionality in the current approaches. Afterward, we abbreviate a new scalable and flexible architecture, which is the underpinning of the new NEOntometrics software. The software is open source and comes with a public metric calculation endpoint.

Keywords
Ontology Metrics, Ontometrics, NEOntometrics, Ontology Quality

1. Introduction

Ontologies come in various shapes, sizes, degrees of interconnection, or logical complexities. Comparing ontologies of different domains or various versions can be a tedious task. Here, metrics offer a great way to quickly grasp a set of attributes. They provide a replicable, objective way to assess ontologies and offer condensed information, such as the ratio of classes to relations, the number of annotations, the depth and breadth of the graph, and much more.

Calculating these values is not trivial. While some basic measurements are available in popular ontology editors like protégé, specialized software is needed for most metrics. Furthermore, even though some software has been developed in the past years, only a few are still available and usable today. One of them is Ontometrics, which is available as a graphical user interface (GUI) application [1] and an API[2] [2]. It originated from a student project and is since maintained by the authors of this paper.

However, further research in ontology metrics brought new requirements that the existing applications struggled to fulfill. Ontometrics has problems with the efficient analysis of large ontologies, does not allow for the calculation of evolutional metrics, and has no interface to adopt new measurements. These shortcomings initiated the development of NEOntometrics[3].

The paper is structured as follows: First, we present previously developed metric calculation software and derive our need for a new development based on the research requirements and the shortcomings of the current Ontometrics application. These shortcomings then motivate the newly proposed application architecture. The research concludes with an overview of upcoming research activities.

SEMANTICS 2022 EU: 18th International Conference on Semantic Systems, September 13-15, 2022, Vienna, Austria
EMAIL: achim.reiz@uni-rostock.de (A. 1); kurt.sandkuhl@uni-rostock.de (A. 2)
ORCID: 0000-0003-1446-9670 (A. 1); 0000-0002-7431-8412 (A. 2)

[2] ontometrics.informatik.uni-rostock.de/, opi.informatik.uni-rostock.de/
[3] neontometrics.com

2. Related Work And New Requirements For Metric Based Ontology Research

The field of automatic metric calculation has seen some activity in the past years. However, many approaches are unavailable and lack the features we deem necessary for our research. Table 1 below gives an overview of previously published ontology evaluation software.

Table 1
Ontology Metric Calculation Software and Their Availability.

Software	Type	Open Source	Still Available	Citation
OntoKBEval	Standalone	No	No	[3]
OntoQA	Standalone	No[4]	No	[4]
S-OntoEval	Standalone	No	No	[5]
OntoMetrics	Web Application	No	Yes	[1,2]
DoORS	Web Application	No	Yes	[6]
OQuaRE	Web Application	No	Yes	[7,8]

None of the software named above is open source, and most are no longer available. The lack of sources and applications hinders today's research from reusing significant parts of the already developed body of knowledge. Thus, the research approaches are often isolated from one another. As a result, an assessment that applies the OntoQA framework is hardly comparable with an assessment that utilizes metrics from the OQuaRE framework.

As part of a broader research perspective [9], we would like to: (**A.**) Analyze historical metric data for a variety of different ontologies. As git has become the de facto standard for sharing codebases and distributed development, we argue that the software should allow the git protocol. (**B.**) Support a variety of existing and potential future calculation methodologies and frameworks. (**C.**) Provide helpful resources on the various available metrics. (**D.**) Have convenient interfaces for both humans and machines.

As the Ontometrics source code is available at our institution, we selected it as the basis for our future software. Furthermore, at first glance, Ontometrics already fulfills many of the given requirements: It comes with a web GUI and an API. The latter could be extended with a script to calculate historical data. The corresponding wiki provides helpful resources on the metrics.

However, the underlying metric calculation does not scale well, and the implemented calculation algorithms are neither easily extensible nor flexible. The missing scalability originates from the implementation of the metric calculation itself: Each calculated metric is represented as an object with an internal representation of the given ontology. This architecture creates a massive overhead of memory usage for larger ontologies, which were at risk of overloading the memory stack. This inefficient memory handling further hindered the parallelization of calculations.

The metric calculations themselves were hardcoded into the software. While this, arguably, does not hinder the extension of the software, it adds a layer of complexity for future efforts. Regarding the API, which is REST-based, adding new metric calculations would require adding versioning to keep the endpoint consistent for the consumers.

There are a few more issues, like the old-fashioned web interface and the isolated help pages on the wiki. The application was not future-proofed in the old state, and we argue that a complete rework is more promising than a somewhat limited evolution for all of the given requirements.

3. A New Calculation Architecture

As shown before, the old Ontometrics application is not sufficient to fulfill the upcoming requirements. The following section outlines the architecture for Ontometrics' successor NEOntometrics. The software is open-source and available on GitHub[5].

[4] Java binaries available on GitHub, but no source code or license attached.
[5] https://github.com/achiminator/NEOntometrics, https://doi.org/10.5281/zenodo.6984839

3.1. A Metric Ontology For Ontology Metrics

Fundamental to the new metric application is a central place for storing knowledge on ontology metrics. NEOntometrics stores the information in the form of an ontology. The resource contains human-readable data, like metric descriptions and definitions, but also the underlying calculation methodologies that the computer will use to set up the calculation engine at the start of the application.

The ontology contains two main sets of different metrics: The *Elemental Metrics* represent the atomic attributes of the ontology, like the *Number of Classes* or *Object Property Domain Axioms*. These measurements are connected to individuals representing the metric name in the database and the calculation software.

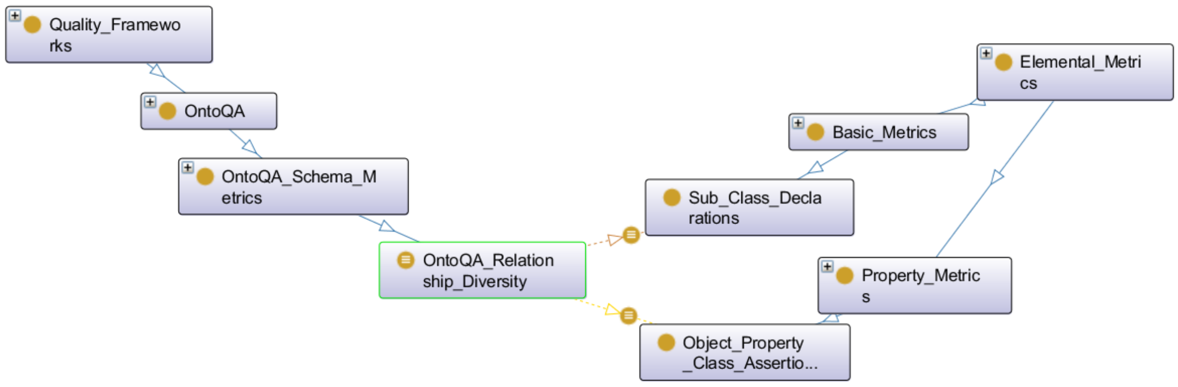

Figure 1: The Modeling of OntoQA's Relationship Diversity Metric

The section *Quality Frameworks* upholds metrics proposed in the literature, namely the various frameworks like OntoQA by Tartir et al. [4] or the measurements by Gangemi et al. [10]. These elements contain annotations representing the textual description found in their papers and the calculation of the metrics in the form of relations with the *Elemental Metrics*. Figure 1 shows Tartir et al.'s *Relationship Diversity*. The calculation is formalized using the object properties *divisor* and *numerator*, connected to the elemental metrics *Sub Class Declarations* and *Object Property Class Assertions*.

As we will show in the next section, the ontology aligns the various (micro-)services and provides the domain knowledge responsible for the functioning of the services. It ensures the flexibility to describe and implement new metrics quickly. As long as there is no need to add other *Elemental Metrics*, newly implemented *Quality Frameworks* can be instantly applied to already calculated datasets.

3.2. A Scalable Calculation Unit

The new service is built in a microservice architecture, encapsulating the various functionalities in separate containers. Figure 2 gives an overview of the application stack and the underlying technologies.

The **web frontend** is built using the multi-platform language *flutter* and the underlying client language *dart*[6]. The Web-Framework complies with the *Material* design, which is known chiefly as the primary design language of Android Apps. It thus provides a familiar navigation experience. *flutter* allows us to utilize a state-of-the-art web design while building the UI with a typed, object-oriented programming language. The metric-related parts, like the help and calculation page, are created dynamically when visiting the frontend webpage.

The **API** part is built using *django* and *django rest framework*[7] and handles all incoming requests. At startup, the API first extracts the relevant information from the **Metric Ontology**. Afterward, it prepares the data for the frontend help page and metric calculation options and augments the *Elemental*

[6] https://flutter.dev, https://dart.dev
[7] https://www.djangoproject.com, https://www.django-rest-framework.org

Metrics available in the database with the *Quality Framework* metrics from the ontology using automated code generation.

For accessing ontology data, the client first examines whether a requested ontology analysis is already known in the system. Three states can occur: If it is in the (redis-) queue, the API returns either the current position of the job in the queue or, if the calculation has already started, its progress. If the metrics are already completely calculated and stored in the database, they can be retrieved with a second request. The user can put the calculation into the **queue** if the requested ontology (repository) is unknown to the system**.**

Already, NEOntometrics is currently able to calculate over 160 metrics – returning all of them would produce a significant over-fetching for most situations. Thus, the API provides its services to the frontend and other consumers using a GraphQL interface. It empowers the service consumer to decide how the response is structured and which information and metrics shall be included.

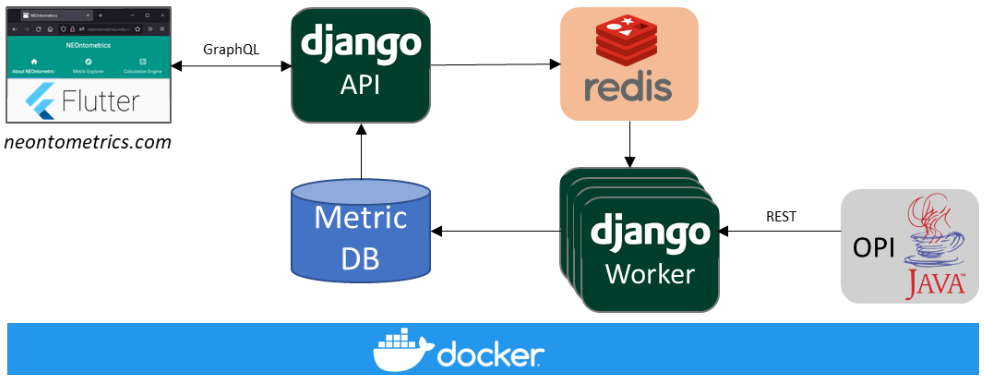

Figure 2: The Microservice Architecture Ensures Scalability

The **queue** is managed by a *Redis*[8] instance, an in-memory key-value database. It stores information on the required parameters and meta information on the upcoming jobs, the workers that handle the actual calculation, and their progress. The latter includes information like the number of already analyzed and analyzable files.

The **workers** handle the actual calculation of the jobs. They register themselves in the queue and retrieve open jobs. To analyze a git repository, they clone it, then collect the files that end with *.owl*, *.rdf*, or *.ttl*. Afterward, the service calls the OPI calculation instance for every commit of every collected file to calculate the underlying elemental ontology metrics. After a successful calculation, the metrics are stored in the database. The **worker** and **API** share a common codebase, which can either be started as a *django* instance (API) or as a worker. The asynchronous calculation is managed using the *django-rq*[9] package.

The **OPI** service carries out the actual analysis of the ontologies. The worker sends an ontology to OPI using an HTTP-POST request and receives the corresponding metric data. OPI is the successor of the OntoMetrics API [2]. However, the service does not have much in common with its predecessor. First and foremost, the metric calculation has been reworked: Each metric now works on the same internal ontology object, thus, avoiding inefficient memory allocation. Further, the service supports reasoning capabilities. At last, we removed all non-atomic metrics and homogenized and aligned the names of the calculated metrics with the other services.

The relational **database** stores the django-related information and the calculated metrics. It currently builds upon *MariaDB*[10]. However, django is highly flexible regarding the endpoint and allows the user to adapt it to other technologies without changing the code.

Docker[11] handles the provisioning of the services. The use of containers ensures portability, isolation of services, and scalability. For instance, the number of parallel analyses can be scaled up by increasing the number of workers (currently: 3). As OPI creates a new thread for every request, we can scale the number of parallel computations on the machine by scaling up the workers. Furthermore, the

[8] https://redis.io
[9] https://github.com/rq/django-rq
[10] https://mariadb.org
[11] https://www.docker.com

system is future-proofed for more upcoming demand as it could be scaled out horizontally, e.g., by adding it to a docker swarm or Kubernetes cluster with load balancing. More API nodes handle more frontend requests, and the calculation capacity increases with adding more OPI nodes.

4. Conclusion

Calculating ontology metrics requires specialized software. Depending on the ontology size and complexity, the analysis of ontologies can require considerable computational resources. It is especially the case if we target not only one ontology at a time but an ontology repository with all available files and versions.

This paper presented the architecture of NEOntometrics, a responsive, flexible, and scalable application for calculating ontology metrics. We believe that it has the potential to strengthen the use of ontology metrics and enable us to answer many research questions of today and tomorrow.

Future research will analyze the results of the metric calculations. However, we still plan to extend the application with functionalities like the analysis of SPARQL endpoints, dashboarding capabilities, or private repositories.

5. References

[1] 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, Porto, Portugal, 2016, pp. 186–191.

[2] A. Reiz, H. Dibowski, K. Sandkuhl, B. Lantow, Ontology Metrics as a Service (OMaaS), in: Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Budapest, Hungary, 02.11.2020 - 04.11.2020, pp. 250–257.

[3] Qing Lu, Volker Haarslev, OntoKBEval: A Support Tool for DL-based Evaluation of OWL Ontologies, in: OWL: Experiences and Directions, Athens, Georgia (USA), 2006.

[4] S. Tartir, I.B. Arpinar, M. Moore, A.P. Sheth, B. Aleman-Meza, OntoQA: Metric-Based Ontology Quality Analysis, in: IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources, Houston, 2005.

[5] R. Dividino, M. Romanelli, D. Sonntag, Semiotic-based ontology evaluation tool S-OntoEval, in: Proceedings of the International Conference on Language Resources and Evaluation, Marrakech, Morocco, 2008.

[6] M. McDaniel, V.C. Storey, V. Sugumaran, Assessing the quality of domain ontologies: Metrics and an automated ranking system, Data & Knowledge Engineering 115 (2018) 32–47. https://doi.org/10.1016/j.datak.2018.02.001.

[7] A. Reiz, K. Sandkuhl, Harmonizing the OQuaRE Quality Framework, in: Proceedings of the 24th International Conference on Enterprise Information Systems, online, 2022, pp. 148–158.

[8] A. Duque-Ramos, J.T. Fernández-Breis, R. Stevens, 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, An Evolutional Based Data-Driven Quality Model for Ontologies, in: Proceedings of the ISWC 2020 Doctoral Consortium, Athens, Greece/online, 2020.

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