Structural Quality Metrics to Evaluate Knowledge Graph Quality

Sumin Seo\textsuperscript{a,\ast}, Heeseon Cheon\textsuperscript{a}, Hyunho Kim\textsuperscript{a} and Dongseok Hyun\textsuperscript{a}

\textsuperscript{a}NAVER Corporation, Republic of Korea

E-mails: sumin.seo@navercorp.com, heeseon.cheon@navercorp.com, kim.hh@navercorp.com, dustin.hyun@navercorp.com

Abstract. This work presents six structural quality metrics that measures the quality of knowledge graphs and apply the metrics to six knowledge graphs: four cross-domain knowledge graphs on the web (Wikidata, DBpedia, YAGO, Freebase), Google Knowledge Graph, and Naver’s integrated knowledge graph (Raftel). The ‘Good Knowledge Graph’ should define specific classes and properties in its ontology so that it can abundantly express knowledge in the real world. Also, Knowledge Graph should use the classes and properties actively. We tried to examine the internal quality of knowledge graphs by focusing on the structure of the ontology, which is the schema of knowledge graphs, and the degree of use thereof. As a result, We have found the characteristics of a good knowledge graph that could not be known only by scale-related indicators such as the number of classes and properties.

Keywords: Ontology, Knowledge Graph Evaluation, Wikidata, Semantic Web

1. Introduction

A knowledge graph is a data system that consists of RDF triples which are in the form of ‘subject-predicate-object’ between entities in the real world and their relationships. For example, the fact that ‘the capital of Korea is Seoul’ can be expressed as ‘Korea - capital - Seoul’.

Ontology is a schema that defines the structure of a knowledge graph. An ontology defines the hierarchical relationship between the classes and the properties that classes can have. Class is an abstract concept that encompasses entities with similar characteristics. For example, Seoul is the instance(=entity) of the ‘City’ class, and Korea is the instance of the ‘Country’ class. The hierarchical relationship between classes is expressed with ‘subclass of’ predicate between superclass, which means a higher concept, and subclass, which means a lower concept. In addition, ‘is-a’ relationship must be established between the two. For example, the ‘Book’ class and the ‘Movie’ class are subclasses of the ‘Creative Work’ class and are defined in the ontology as ‘Book - subclass of - Creative Work’ and ‘Movie - subclass of - Creative Work’. Property is an attribute that each class can have. For instance, the ‘Country’ class can have ‘capital’, ‘population’, and ‘president’ as properties. An ontology defines the properties that each class can have.

Knowledge graphs have an important role in search systems such as Google’s knowledge panel ([1]), and recently attention is being drawn from NLP tasks such as Question Answering ([2]), recommendation systems ([3]), and explainable AI ([4]). In this regard, Wikidata ([5]), Freebase([6]), DBpedia ([7]), and YAGO([8]) are examined for various tasks in many studies.

\textsuperscript{\ast}Corresponding author. E-mail: sumin.seo@navercorp.com.
This study outlines a "good knowledge graph" and offers metrics to quantify it. In contrast to the current knowledge graph evaluation studies, which mainly focused on the size and distribution of the data, this study devised a measure to compare the quality between knowledge graphs with the point of view that structure (=ontology) is a key factor that determines the quality of knowledge graphs. Based on these indicators, we compared knowledge graphs on the web (Wikidata, Freebase, DBpedia, YAGO, Google Knowledge Graph) and Naver's knowledge graph (Raftel).

2. Related Works

Ontology and knowledge graph evaluation methods can be classified as follows. ([9], [10], [11])

- gold standard evaluation: a method of comparing knowledge graphs to high-quality knowledge graphs with the same topic.
- data driven evaluation: a method that selects important words by extracting keywords from documents dealing with the same domain of knowledge graphs and measures how much information knowledge graphs contain.
- application/task based evaluation: a method that evaluates the downstream task performance of the knowledge graph.
- user based evaluation: a method that evaluates quality from the perspective of knowledge graph users.
- structure based evaluation: a method that evaluates knowledge graphs through metrics that represents the structure or statistical properties of the ontology and knowledge graph. ([12], [13])
- data quality evaluation: a method that defines data quality with various point of views including accuracy and consistency of data with indicators to measure data quality. ([14], [11])

Comparative studies on cross-domain knowledge graphs on the web mainly focuses on structure based evaluation and data quality evaluation. ([15], [16]) First of all, the structure based evaluation uses schema metric, instance/knowledgebase metric, class metric, graph metric and complexity metric to compare knowledge graphs. Schema metric calculates the number of classes, the number of properties, and the number of properties per class, focusing on an ontology. Instance/knowledgebase metric calculates statistics like the average number of instances per class considering instances with an ontology. In the case of class metric, the number of instances of the 'Person’ class or the 'Person’ class's degree of connection with other classes class is calculated to represent the characteristics of each class. Graph metric is the application of basic statistics of graph theory such as cohesion and cardinality.

Regarding data quality evaluation methods, [17] analyzed the knowledge graph’s data quality dimensions according to the framework presented by [18]. There are many works that evaluate knowledge graphs on the data quality perspective: An accuracy perspective which means how accurately the knowledge graph reflects real-world information ([19], [20]), a consistency perspective that focuses on how consistent the data in the knowledge graph is ([21], [22]), an ease of understanding perspective including how many languages a knowledge graph provides ([23]), an interlinking perspective which means how much a knowledge graph can be connected to other knowledge graphs ([8], [15]).

As a complement to the shortcomings of current structure-based metrics and data-quality-based assessment techniques, this work introduces the structural quality metrics to evaluate knowledge graph quality. Most of the researches comparing knowledge graphs on the web mainly focus on the size of knowledge graphs (number of RDF triple, class, and instance). In addition, numerical indicators related to structure simply describe the distribution of data such as the number of instances per class. It is easy to grasp the size and approximate structure of the knowledge graph with this approach, but difficult to decide either side is better in terms of quality. Graph metrics such as cohesion and cardinality can be used to evaluate the quality of knowledge graphs, such as how concisely organized knowledge graphs are and how rich relations a knowledge graph has. However, these metrics are not specialized for the quality of knowledge graphs. Studies based on data quality dimension analyzed knowledge graphs on the web in various aspects, but there are no quality indicators based on the knowledge graph structure, ontology. Therefore, we propose structural quality metric that can numerically represent the internal quality of knowledge graphs, focusing on the ontological structure and its degree of utilization.
Table 1
Example of Freebase Ontology data extraction

| subject | predicate | object |
|---------|-----------|--------|
| class   | <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | rdf-schema#Class > |
|         | <http://rdf.freebase.com/ns/film.film> | /ns/film.film > |
| predicate | film.directed_by  | <http://www.w3.org/2000/01/rdf-schema#domain> |

3. Data Introduction and Basic Statistics

3.1. Data Introduction and Data Preparation

Wikidata, DBpedia, Freebase, YAGO, and Google Knowledge Graph are analyzed in the work and have the following characteristics. ([24])

Wikidata
Wikipedia was launched in 2012 by Wikimedia Deutschland, which provides information associated with Wikipedia and allows users to participate directly in creating and editing data. Although ontology information is not provided as a separate file, the hierarchical structure between entities can be known through subclass of property for each entity.

DBpedia
DBpedia is a knowledge graph launched in 2007 by Free University of Berlin and the University of Leipzig. It is created by automatically extracting structured information contained in Wikipedia, and it builds and manages its own ontology structure.

Freebase
Freebase was launched by MetaWeb Technologies, Inc. in 2007 and merged into Wikidata by the Wikimedia Foundation and Google in 2015. ([25]) Ontology is constructed in a human readable manner with the structure of 'domain/class/predicate'.

YAGO
Developed by Max Planck in 2007, data is generated by extracting information from Wikipedia infobox and WordNet in various languages. The ontology structure is built on WordNet.

Google Knowledge Graph
Google Knowledge Graph (Google KG) was developed by Google in 2012 to understand the meaning of search terms in the search engine. Through the API, information such as name, description, image, and type which informs the class can be obtained for a specific keyword. There is no self-defined ontology, and the type is configured based on schema.org. Even though Google introduced Knowledge Vault ([26] in 2014, which integrates Wikipedia, YAGO, Microsoft’s Satori, and Google Knowledge Base, we used Google Knowledge Graph data which is accessible by API (Google Knowledge Graph Search API) since Knowledge Vault is not published to the public.

Raftel, introduced in this work, is a knowledge graph that integrates Wikidata and Naver’s databases on various domains. Based on the class structure and properties of Wikidata, an active and fast-generating knowledge graph constructed from the community, the basis ontology of Raftel was designed. For the following steps, classes were added and removed, hierarchical relationships were adjusted, and attributes were added and removed to alleviate the complexity of the ontology and increase consistency.

In addition, since Raftel is a knowledge graph generated based on Korean data, other knowledge graphs were filtered by the entity with the Korean label. Since Google Knowledge Graph API can be queried with keywords, data with Korean labels that exist in Wikidata were imported.

There are cases that a knowledge graph provides individual ontology and ontology is included in the knowledge graph. DBpedia and YAGO have the ontology file provided by themselves. In the case of YAGO, classes are defined in ontology file as well as in the knowledge graph’s RDF triples with the format of ‘A - instance of - B’. However, in this study, only the classes in the ontology file are considered to calculate metrics. The ontology of Google Knowledge Graph is based on schema.org, so schema.org’s ontology data was used for Google KG. [1] For Wikidata and Freebase, where ontology data is not provided with a separate file, the ontology was constructed from knowledge.

[1] https://developers.google.com/knowledge-graph
graph’s RDF triples. Wikidata’s ontology was extracted using the ‘subclass of’ property. For the properties for each class, the properties used in the class were collected by mapping the class to the instances of the RDF triples. Freebase’s classes are represented in RDF triple’s subject with a property of ‘<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>’ and an object of ‘<http://www.w3.org/2000/01/rdf-schema#Class>’. In addition, Freebase does not specify a hierarchy between classes, so there is no root class for Freebase’s ontology([27]). To calculate metrics, we created a root class and connected all classes as subclasses of the root class. Freebase’s properties are derived from RDF triples with a property of ‘<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>’ whose object ends with ‘Property>’, and only those with domain property, which is ‘<http://www.w3.org/2000/01/rdf-schema#domain>’ (Table 1).

### Table 2

|                | Rafel | Wikidata | DBpedia | YAGO | Google KG | Freebase |
|----------------|-------|----------|---------|------|-----------|----------|
| number of classes | 273   | 59,662   | 804     | 266  | 910       | 53,091   |
| number of properties | 607   | 7,476    | 21,607  | 141  | 1,447     | 23,446   |
| number of RDF triples | 253,566,996 | 27,258,977 | 11,137,852 | 348,094,663 | 48,348,838 | 48,292,483 |
| number of instances | 17,653,785 | 1,323,452 | 287,752 | 19,707,176 | 1,390,438 | 33,535,913 |

3.2. Basic Statistics

Table 2 shows the basic statistics for knowledge graphs. The number of classes and properties were calculated from ontology, and the number of RDF triple and instances were calculated from knowledge graph RDF triple data. It includes an RDF triples having instances of classes and properties that were not defined in the ontology. All classes and properties contained in the ontologies were included for the calculation even if they don’t have Korean label. Other analyses include RDF triples of the instance with a Korean label among all RDF triples of the knowledge graph. Numbers in parentheses are values for the entire language. For Google KG, since only entities with Korean names are included, analysis of the entire language was not conducted.

In terms of classes and properties that construct the ontology, Wikidata and Freebase have more than 50,000 classes, which is about 200 times more than Rafel and YAGO. In the case of properties, DBpedia and Freebase have more than 20,000 properties, 100 times more than YAGO, which has the least properties, and 30 times more than Rafel, which has the second least properties.

On the other hand, Rafel and YAGO have relatively large amount of RDF triples and instances. Compared to DBpedia, which has the lowest RDF triple count, the number of RDF triples of YAGO is more than 30 times higher, and in the case of Rafel, it is more than 20 times higher. In the case of the number of instances, Freebase is the largest, but for Freebase, the number of RDF triples is small because most RDF triples are composed of ‘instance of’ relationship. It can be seen from the fact that the number of an RDF triple is about 1.4 times the number of instances. Next, the number of YAGO and Rafel’s instances is 19 million and 17 million, respectively, more than 60 times that of DBpedia, which has the lowest number of instances. For YAGO, there are about 18.46 million instances belonging to the ‘scholarArticle’ class, accounting for 97% of the total, and RDF triples with the ‘scholarArticle’ instance as the subject account for more than 90% (312,203,867).

4. Structural Quality Metrics

4.1. What is a good Knowledge Graph?

A good knowledge graph should have a fine-grained ontology structure that can precisely express information in the real world, and instances and triples should make full use of the ontology’s classes and properties. By categoriz-
ing this perspective into the four categories listed below, a structural quality metrics that can quantify each content was developed.

First, class hierarchy must be abundantly subdivided in the ontology. Taking the ‘Person’ class as an example, the more the Person is divided into horizontals like ‘Artist, Athlete, Politician, Doctor’, and the more split into verticals like ‘Person → Artist → Musician’, the more information the ontology can include. Compared to the case where only ‘Person’ class exists for the class related to people, if it is divided into more classes according to occupation, ontology would be more powerful in various tasks by narrowing the scope of the classification of entities. For example, when in comes to entity disambiguation task, it is easy to disambiguate entities with the same name because the class range that the person with the same name belongs is specified. If the ‘Musician’ and ‘Author’ classes are added as a subclass to ‘Artist’, which is a subclass of ‘Person’, the ontology will provide more specific information to application tasks.

Second, it is better when there are more properties in subclass that are not in superclass. In the case of vertically splitting the class, the number of properties that the subclass has should increase. For example, when Athlete class is a subclass of Person class, the Person class has properties such as "parent", "birth date", and "birth place". When the Athlete class has more specific properties (e.g. "back number", "team", "world ranking", "league", etc.), the ontology’s quality is enhanced since the information that can be obtained by adding a class increases.

Third, classes and properties defined in the ontology must be used sufficiently in the knowledge graph. Even if classes and properties are defined in detail, ontologies are only useful if they are applied to the knowledge graph. Even though there can be subclasses of the Person class like "Chef of Chinese Restaurant in Seoul" or "4th grade music teacher of the elementary school", if the classes have only tiny amount of instances, it is difficult to say that adding these class is beneficial. Likewise, specific properties such as "number of debut songs sung at concerts" and "administrative district where the most fans live" can be defined for the Musician class, but if data does not exist and is not used as actual RDF triples, it is better not to add them.

Fourth, though the quality increases as the class is subdivided, it is negative if the complexity increases in this process. Multiple inheritance is one of the factor that describes ontology complexity. Multiple inheritance means that one class has several superclasses. For example, the Hospital class is a subclass of ‘Facility’, a space that provides a specific function, and a subclass of ‘Organization’, a group of employees including doctors and nurses. Avoiding multiple inheritance is preferable, unless it is necessary, like in the case of the Hospital where both location information which is a facility characteristic, and member information, which is an organization characteristic, are crucial. This is because when subclass and superclass are connected in a many-to-many relationship, the complexity of understanding and application of ontology increases.

4.2. Structural Quality Metrics

This work shows a structural quality metric that can measure the quality of good knowledge graphs presented in 4.1. 4.2.1, 4.2.2 have been examined in previous studies, and 4.2.3, 4.2.4, 4.2.5, 4.2.6, are newly introduced in this work.

4.2.1. Instantiated Class Ratio

\[ ICR(Ontology) = \frac{N(IC)}{N(C)} \]  

(1)

4.2.2. Instantiated Property Ratio

Instantiated Property Ratio refers to the ratio of properties actually used in RDF triple among the properties defined in the ontology. It is an indicator of how well the properties of the ontology are actually being used. In
obtaining *Instantiated Property Ratio* for ontology Eq. (2), \( N(P) \) denotes the total number of properties of the Ontology, and \( N(IP) \) denotes the number of properties used in RDF triples.

\[
IPR(\text{Ontology}) = \frac{N(IP)}{N(P)} \tag{2}
\]

4.2.3. **Class Instantiation**

**Class Instantiation** is a metric that assesses how much in detail classes are defined in the ontology and how much they are actually instantiated. For each class included in the knowledge graph, the class instantiation is calculated and each metric is summed up to represent the knowledge graph as a whole. In Eq. (3) to obtain **Class Instantiation** for a particular Class, \( n_c \) means the number of subclasses that the Class has, \( ir(c) \) means instantiated ratio, which is 'number of instances of the Class' divided by 'number of all the instances in knowledge graph', \( c_i \) is the \( i \)-th subclass the Class has, \( d \) means the distance between the Class and \( c_i \).

\[
CI(\text{Class}) = \sum_{i=1}^{n_c} \frac{ir(c_i)}{2^d(c_i)} \tag{3}
\]

The process of calculating **Class Instantiation** for ‘Person’ class in a knowledge graph such as Fig. 1 is as follows. The total number of instances of the knowledge graph is 500, of which the number of instances of ‘Person’ and Person’s subclass is 200. For all subclasses under the ‘Person’ class, the proportion of the class’s instances to the all instances is calculated. The proportion is considered as "weight". For example, to calculate the weight for ‘Artist’, use the number of direct instances of the Artist, not the instances of ‘Actor’, ‘Musician’, or ‘Author’. ‘Ariana Grande’ instance is not a direct instance of the ‘Artist’ because the singer is an instance of the Artist’s subclass,
‘Musician’. Since ‘Pablo Picasso’ instance does not belong to the ‘Artist’’s subclasses, it becomes a direct instance of the ‘Artist’. In Fig. 1, the class is represented with rectangle box, and in the box and (number of instances → weights) is denoted.

Person’s Class Instantiation accumulates weights from the subclass farthest from ‘Person’. Weights are not added as they are, but divided by \(2^{\text{DepthFromPersonClass}}\). The class’ weight become smaller when it is farther away from ‘Person’. As a result, Class Instantiation of ‘Person’ is calculated as \(0.1 + (0.02 + 0.01 + 0.06) \times \frac{1}{2} + (0.06 + 0.1 + 0.04) \times \frac{1}{2}\). Class Instantiation is obtained in the same way as above for all the classes including ‘Artist’, ‘Musician’, and ‘Creative Work’ that exist in the ontology.

When classes are divided more specifically and each class has more instances, Class Instantiation becomes higher. In addition, the penalty according to depth was applied to prevent the side effects of increasing the score as the class is subdivided into verticals unconditionally.

4.2.4. Subclass Property Acquisition

Subclass Property Acquisition is a metric that measures how many properties are defined in the subclass that is not in the superclass in the ontology. For example, if the ‘Person’ class is a subclass of the ‘Entity’ class, properties that are not defined in the ‘Entity’ class such as ‘children’, ‘academic degree’, and ‘spouse’, can be added. Furthermore, if the ‘Actor’ class is a subclass of the ‘Person’ class, properties like ‘character role’, ‘cast member of’ can be added. The Subclass Property Acquisition is the average value obtained by number of newly added properties that are not in the superclass for all classes of the ontology, except for the root class (e.g., Entity class).

In the Eq. (4) for obtaining Subclass Property Acquisition for Ontology, \(P\) denotes property set and \(N(P)\) is the number of elements in the property set. For all ‘superclass-subclass’ relationships present in Ontology, \(N(P_{\text{subclass}} - P_{\text{superclass}})\) is calculated, and the number of properties present in the subclass is obtained and summed. For all classes in Ontology, Subclass Property Acquisition is calculated and divided by \(N(C)\), which denotes the number of classes.

\[
\text{SPA(Ontology)} = \frac{\sum(N_i(P_{\text{subclass}} - P_{\text{superclass}}))}{N(C)} \tag{4}
\]

4.2.5. Subclass Property Instantiation

Subclass Property Instantiation quantifies how many properties are used in the RDF triples when the properties of the subclass that are not in the superclass are defined in the ontology. For example, if an ‘Actor’ class adds ‘cast member of’ and ‘character role’ properties that are not in the superclass ‘Person’ class, the more unique properties of the actor are used in RDF triples, such as "Tom Cruise - cast member of - Mission Impossible" and "Tom Cruise - character role - Ethan Hunt", the better structure knowledge graph has. Knowledge graph’s Subclass Property Instantiation is the average of Subclass Property Instantiation of all the classes.

In the Eq. (5) for obtaining Subclass Property Instantiation for a particular Class, \(T\) is a set of RDF triples, and \(N(T)\) is the number of RDF triples. \(N(T_{\text{class}} - T_{\text{class, superclass}})\) is the number of RDF triples of Class excluding RDF triples which uses predicates defined for superclass.

\[
\text{SPI(Class)} = \frac{N(T_{\text{class}} - T_{\text{class, superclass}})}{N(T_{\text{class}})} \tag{5}
\]

To compute a Subclass Property Instantiation for an ‘Actor’ class, first, count the number of all triples with an ‘Actor’ class’s instance as the subject. In addition to RDF triples like "Tom Cruise - cast member of - Mission Impossible" and "Tom Cruise - character role - Ethan Hunt", count all the RDF triples including "Tom Cruise - Birth Place - Syracuse", "Tom Cruise - Nationality - United States", and "Tom Cruise - Name - Tom Cruise". This becomes denominator of the Subclass Property Instantiation. Next, count the number of triples in which the properties added in the ‘Actor’ class are used. Except for "Tom Cruise - Birth Place - Syracuse", "Tom Cruise - Nationality - United States", and "Tom Cruise - Name - Tom Cruise", only RDF triples such as "Tom Cruise - cast member of - Mission Impossible" and "Tom Cruise - character role - Ethan Hunt" are considered. This is the numerator of Subclass Property Instantiation. By dividing the number of RDF triples used by the property added in the ‘Actor’ by the
Table 3

|                        | Raftel | Wikidata | DBpedia | YAGO     | Google KG | Freebase |
|------------------------|--------|----------|---------|----------|-----------|----------|
| Instantiated Class Ratio | 0.941  | 0.470    | 0.820   | 0.099    | 0.046     | (0.314)  |
|                        | (0.334)| (0.540)  | (0.966) |          |           |          |
| Instantiated Property Ratio | 1     | 0.99     | 0.90    | 1        | 0.002     | (0.003)  |
|                        | (1)    | (1)      | (0.96)  |          |           |          |
| Class Instantiation    | 0.941  | 0.716    | 0.900   | 0.886    | 0.660     | 0.874    |
|                        | (0.743)| (0.949)  | (0.616) |          |           | (0.749)  |
| Inverse Multiple Inheritance | 0.975 | 0.962    | 0.971   | 0.942    | 0.952     | -        |
|                        | (0.743)| (0.949)  | (0.616) |          |           |          |
| Subclass Property Acquisition | 6.54  | 40.94    | 63.57   | 2.23     | -         | 1        |
|                        | (0.0133)| (0.0841)| (0.0003)|          |           |          |
| Subclass Property Instantiation | 0.0857| 0.0133   | 0.0841  | 0.0003   | 0.0       | (0.0)    |
|                        | (0.0001)| (0.0668)| (0.0001)|          |           |          |

total number of triples in the ‘Actor’, we can see how the unique property increases in RDF triple as the ‘Actor’ was subdivided from ‘Person’. For example, if the Actor’s Subclass Property Instantiation is 0.05, it means that the RDF triple of ‘Actor’ with new properties has increased by 5% compared to the superclass ‘Person’.

4.2.6. Inverse Multiple Inheritance

Inverse Multiple Inheritance evaluates the simplicity of the knowledge graph. If multiple inheritance occurs frequently in which a single class has numerous superclasses, it might make it challenging to use the knowledge graph because of the complexity of the class relationship. Inverse multiple inheritance was devised to measure how little multiple inheritance appears. The average number of superclasses per class is computed to obtain the average multiple inheritance, and take the reciprocal of it. Therefore, the higher the Inverse Multiple Inheritance, the simpler the knowledge graph is. In Eq. (6), $N_c$ represents the total number of classes in the ontology, $C_i$ represents each class in the Ontology, and $nsup(C)$ represents the number of direct superclasses in the class.

$$IMI(Ontology) = \frac{1}{\sum_{i=1}^{N_c} nsup(C_i) / N_c}$$

The six structural quality metrics determine whether knowledge graph can express knowledge abundantly through a detailed ontology. Among them, Class Instantiation and Subclass Property Instantiation have the characteristics of a comprehensive indicator that can reflect classes or attributes’ degree of subdivision and actual utilization.

5. Structural Quality Metric Evaluation Result

Table 3 is the analysis of structural quality metric with six knowledge graphs. First, YAGO and Raftel have the smallest number of classes in Table 2, but more than 80% of classes were instantiated. On the other hand, Wikidata, Freebase, and Google KG instantiated less than 10% compared to the large number of classes defined in the ontology. According to the analysis related with Class Instantiation, DBpedia and Raftel fully utilize fine-grained classes and properties in the knowledge graph. When comparing DBpedia and YAGO, even though YAGO has higher Instantiated Class Ratio, DBpedia’s Class Instantiation shows that it is divided into vertical and horizontal classes, and the classes are actively used in the knowledge graph. For Google KG, the low number of classes that could be imported through the API was reflected in the low Instantiated Class Ratio and Class Instantiation. Freebase does not define the hierarchy between classes, so it seems to affect the degree of Class Instantiation to be lower. When it comes to Inverse Multiple Inheritance, Freebase is calculated as 1 because classes do not have parent classes, and
has the most concise ontology structure. Schema.org (Google KG’s ontology) and YAGO have a high complexity
due to its relatively frequent multiple inheritance.

Referring the property-related metrics, DBpedia and Wikidata have a large number of properties defined in the
ontology (Table 2) and Subclass Property Acquisition shows the large number of properties are added as classes
are subdivided. On the other hand, for Freebase, the number of properties of Table 2 is large, but the value of
Subclass Property Acquisition is low because the ontology has no class hierarchy. Google KG only provides basic
information about ‘name, image, description, class’ through the API, so Subclass Property Acquisition appears low.
Looking at Subclass Property Instantiation, Wikidata has richly defined properties according to class segmentation
in the ontology, but the degree of use is relatively low. DBpedia has abundant properties and those properties are
well used in RDF triple. YAGO has small number of properties (Table 2). Also, Subclass Property Acquisition and
Subclass Property Instantiation infers that the degree of segmentation is low.

RafTel appears to have high scores in Class Instantiation and Subclass Property Instantiation, which are the
comprehensive score, since it organized classes and properties according to the criteria of 4.1.

Table 4 shows comprehensive analysis of structural quality metric by categorizing metrics to Class Metric (CM)
including Instantiated Class Ratio, Class Instantiation, Inverse Multiple Inheritance and Property Metric (PM) in-
cluding Instantiated Property Ratio, Subclass Property Acquisition, Subclass Property Instantiation and calculating
weighted average of them. CM consists of metrics that measure how systematic and detailed the knowledge graph is
constructed. PM metrics indicate how knowledge graph can express real world’s knowledge with diverse properties.
Knowledge graphs with higher CM would be preferred for downstream tasks such as NER and Entity Linking which
need fine-grained ontology structure. On the other hand, knowledge graph with higher PM can be a good reference
for new knowledge graph structure because they give abundant information according to each class.

Each metric of the structural quality metric was normalized to have a maximum value of 1 and a maximum
value of 10, and then metrics belonging to Class Metric and metrics belonging to Property Metric were averaged
to obtain a representative value. After that, the characteristics of the knowledge graph were examined by varying
the weights of CM and PM. When the proportion of PM is large, DBpedia showed the highest score, and when the
weighted average was the same or the proportion of CM was larger, RafTel showed the highest score. Through this,
if the degree of segmentation of the property is important, the quality of DBpedia can be judged to be high, and
if the degree of segmentation of the class is important, the quality of RafTel can be judged to be high. Fig. 2 is a
visualization of Table 4.

|                  | RafTel | Wikidata | DBpedia | YAGO | Google KG | Freebase |
|------------------|--------|----------|----------|------|-----------|----------|
| 0.0 × CM + 1.0 × PM | 7.31   | 6.40     | **9.91** | 3.81 | 4.00      | 1.04     |
| 0.25 × CM + 0.75 × PM | 7.66   | 5.47     | **9.07** | 4.37 | 3.45      | 2.39     |
| 0.5 × CM + 0.5 × PM | **8.01** | 4.51     | 8.23    | 4.92 | 2.90      | 3.72     |
| 0.75 × CM + 0.25 × PM | **8.36** | 3.57     | 7.40    | 5.47 | 2.34      | 5.06     |
| 1.0 × CM + 0.0 × PM | **8.71** | 2.63     | 6.56    | 6.03 | 1.79      | 6.4      |

6. Conclusion

In this study, six structural quality metrics were proposed as indicators to evaluate the quality of knowledge
graphs. Reflecting the perspective that ‘Knowledge graphs should have the ontology that can express knowledge
in the real world, and the knowledge graph RDF triples should utilize the ontology sufficiently’, we present the
Instantiated Class Ratio, Instantiated Property Ratio, Class Instantiation, Subclass Property Acquisition, and Sub-
class Property Instantiation. Also, Inverse Multiple Inheritance was introduced to measure the complexity of the
ontology.
In addition, the structural quality metric was applied to five cross-domain knowledge graphs on the web and Naver’s integrated knowledge graph, Raftel for the comparative analysis. Compared to the structure evaluation conducted only in terms of the size and distribution of graphs, it was able to gain in-depth insights on the quality of knowledge graphs.

Structural quality metric sees ‘structure’ as an important factor in determining the quality of knowledge graphs. According to the results of the structural quality metric analysis, some knowledge graphs with many classes and properties in their ontology have low degree of segmentation and instantiation. On the contrary, some knowledge graphs that have less classes and properties compared to others described knowledge in detail with specified classes and their distinct characteristics. Of course, since each knowledge graph has a different orientation, knowledge graphs with a low score in the structural quality metric can also show good scores in the quality metric in different dimensions. In future studies, it is expected that the strengths and weaknesses of each knowledge graph to be examined with multi-dimensional point of view by applying various metrics from the data quality perspective presented in previous works.
Appendix A. Graphs for Statistics (target language: Korean)

A.1. Basic Statistics

A.1.1. Number of Classes

| Dataset     | Number of Classes |
|-------------|-------------------|
| Raftel      | 274               |
| Wikidata    | 59662             |
| DBpedia     | 804               |
| YAGO        | 266               |
| GoogleKG    | 910               |
| Freebase    | 53091             |

A.1.2. Number of Properties

| Dataset     | Number of Properties |
|-------------|----------------------|
| Raftel      | 697                  |
| Wikidata    | 7467                 |
| DBpedia     | 21607                |
| YAGO        | 141                  |
| GoogleKG    | 1447                 |
| Freebase    | 23446                |
A.1.3. Number of RDF Triples

A.1.4. Number of Instances
A.2. Structural Quality Metrics

A.2.1. Instantiated Class Ratio

![Instantiated Class Ratio Graph](image)

A.2.2. Instantiated Property Ratio

![Instantiated Property Ratio Graph](image)
A.2.3. Class Instantiation

![Class Instantiation Chart]

A.2.4. Inverse Multiple Inheritance

![Inverse Multiple Inheritance Chart]
A.2.5. Subclass Property Acquisition

![Bar chart showing subclass property acquisition for different datasets.]

A.2.6. Subclass Property Instantiation

![Bar chart showing subclass property instantiation for different datasets.]

Raftel Wikidata DBpedia YAGO GoogleKG Freebase

Raftel Wikidata DBpedia YAGO GoogleKG Freebase

Subclass Property Acquisition

Subclass Property Instantiation
References

[1] X. Zou, A Survey on Application of Knowledge Graph, Journal of Physics: Conference Series 1487(1) (2020), 012016. doi:10.1088/1742-6596/1487/1/012016.

[2] X. Huang, J. Zhang, D. Li and P. Li, Knowledge Graph Embedding Based Question Answering, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM ’19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 105–113. ISBN 9781450359405. doi:10.1145/3289600.3290956.

[3] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong and Q. He, A Survey on Knowledge Graph-Based Recommender Systems, IEEE Transactions on Knowledge and Data Engineering 34(8) (2022), 3549–3568. doi:10.1109/TKDE.2020.3028705.

[4] I. Tiddi and S. Schlobach, Knowledge graphs as tools for explainable machine learning: A survey, Artificial Intelligence 302 (2022), 103627. doi:https://doi.org/10.1016/j.artint.2021.103627. https://www.sciencedirect.com/science/article/pii/S0004370221001788.

[5] D. Vrandečić and M. Krötzsch, Wikidata: A Free Collaborative Knowledge Base, Communications of the ACM 57 (2014), 78–85. http://acm.acm.org/magazines/2014/10/178785-wikidata/fulltext.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD ’08, Association for Computing Machinery, New York, NY, USA, 2008, pp. 1247–1250. ISBN 9781595938849. doi:10.1145/1376616.1376746.

[7] M. Auer, C. Bizer, G. Kobieluk, U. Lehmann, R. Cyganiak and Z. Ives, DBpedia: A Nucleus for a Web of Open Data, 2007, pp. 722–735. ISBN 978-3-540-76297-3. doi:10.1007/978-3-540-76298-0_2.

[8] F.M. Suchanek, G. Kasneci and G. Weikum, Yago: A Core of Semantic Knowledge, in: Proceedings of the 16th International Conference on World Wide Web, WWW ’07, Association for Computing Machinery, New York, NY, USA, 2007, pp. 697–706. ISBN 9781595936547. doi:10.1145/1242572.1242667.

[9] J. Brank, M. Grobelnik and D. Mladenić, A Survey of Ontology Evaluation Techniques, in: Proc. of 8th Int. multi-conf. Information Society, 2005, pp. 166–169.

[10] J. Raad and C. Cruz, A Survey on Ontology Evaluation Methods, in: KEOD, 2015.

[11] M. Färber and A. Rettinger, Which Knowledge Graph Is Best for Me?, in: Proceedings of the 15th International Conference on Inventive Systems and Control (ICISC) (2018), 1415–1421.

[12] G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun and W. Zhang, Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion, in: Proceedings of the 13th International Symposium on Open Collaboration, OpenSym ’19, Association for Computing Machinery, New York, NY, USA, 2019. ISBN 9781450363198. doi:10.1145/3306433.3340822.

[13] R.Y. Wang and D.M. Strong, Beyond Accuracy: What Data Quality Means to Data Consumers, J. Manage. Inf. Syst. 12(4) (1996), 5–33-. doi:10.1007/978-1-4471-4904-1_1.

[14] F Nielsen, D. Mietchen and E. Willighagen, Scholia, Scientometrics and Wikidata, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’14, New York, NY, USA - August 24 - 27, 2014, 2014, pp. 601–610. Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Gernery Heitz http://www.cs.cmu.edu/~nlao/publication/2014.kdd.pdf.
[27] N. Chah, OK Google, What Is Your Ontology? Or: Exploring Freebase Classification to Understand Google’s Knowledge Graph, ArXiv abs/1805.03885 (2018).