Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information)
16/1 (2023), 15-46. DOI: http://dx.doi.org/10.21609/jiki.v16i1.1096

LexID: The Metadata and Semantic Knowledge Graph Construction of Indonesian Legal Document

Nur Siti Muninggar[1], Adila Alfa Krisnadhi[2]

Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia

Email:nur.siti@ui.ac.id[1], adila@ui.ac.id[2]

Abstract

The *Legal Fiction* principle stipulates that the government needs to ensure the public availability of all of their legal documents. Unfortunately, the text-based search services they provide cannot return satisfactory answers in retrieval scenarios requiring proper representation of relationships between various legal documents. A key problem here is the lack of explicit representation of such relationships behind the employed retrieval engines. We aim to address this problem by proposing LexID knowledge graph (KG) that provides an explicit knowledge representation for Indonesian legal domain usable for such retrieval purposes. The KG contains both legal metadata information and semantic content of the legal clauses of the legal document's articles, modeled using formal vocabulary from the LexID ontology also presented in this paper. The KG is constructed from thousands of Indonesian legal documents. Since the procedure of writing a legal document regulated by the government is clear and detailed, we use a rule-based approach to construct our KG. At the end, we describe several use cases of the KG to address different retrieval needs. In Addition, we evaluated the quality of our KG by measuring its ability to answer questions and got that LexID can answer questions with the macro average F1 score is about 0.91.

Keywords: *Indonesian Legal Ontology, Knowledge Graph Construction*

1. Introduction

Indonesian law follows the so-called *Legal Fiction* principle or *Fictie Rechts* in which everyone is assumed to know the law once it is enacted — ignorance does not exempt a person from being subjected to lawsuits. Consequently, the state has an obligation to ensure that all citizens of the state are aware of all of the state's legal products, including the constitution, statutes, regulations, and various legal codes [1, 2]. One way to achieve this is by providing a direct access to such legal documents online [3].

Two online government services, namely JDIHN[1] and peraturan.go.id[2], currently provide directory services of 50 thousand legal documents covering various kinds of regulations enacted by state bodies such as the Indonesian parliament, the president's office, ministries, Indonesian central bank, as well as provincial and district-level government regulations from all over the country. Both systems also provide a simple search service based on the title of the regulation. Some private organizations, such as hukumonline.com[3] and eclis.id,[4] also provide similar, but slightly better, search services whereby users can also perform a search over the content of those legal documents. However, we have to sign up for a paid membership to get the complete services of those sites.

Despite the public availability of the aforementioned services, retrieving legal information remains a challenge, particularly when the retrieval needs to take into account a number of separate, but interrelated legal documents. As an example, consider article 16A of Act number 8 of 1983 concerning

[1]https://jdihn.go.id/
[2]https://peraturan.go.id/peraturan/index.html?
[3]https://www.hukumonline.com/
[4]https://eclis.id/

Goods and Services Tax and Sales Tax on Luxury Goods. Act number 8 of 1983 (or Act 8/1983[5]) has in fact been amended three times by another law, namely by Act 11/1994, then by Act 18/2000, and finally by Act 42/2009. Moreover, article 16A originally was not part of Act 8/1983. Rather, it was added to the law by the first amendment, i.e., as directed by Act 11/1994, and not only that, the content of that article was then changed by the second amendment, i.e., according to Act 18/2000. Now, suppose one wishes to query for information about that article, e.g., by using the following search phrase *'Pasal 16A Undang-Undang nomor 8 tahun 1983'*, to the four online services mentioned earlier.[6] Out of the four services, three return no results. The only one returning result is only hukumonline.com that returns Act 8/1983 ranked at the 1st position in the search results. Unfortunately, even though the amendments are also returned, they are ranked very low: the first amendment is ranked at the 56th position, the second amendment is ranked at the 70th position, and the third amendment does not even appear higher than the 200th position.

Unfortunately, the above situation is contrary to what is reasonably expected by the user. From a user's perspective, to understand the intent of article 16A above, one would naturally need to access the latest version the article as it appears in the latest amendment. In terms of search results, we wish not only to get Act 8/1983 ranked at the 1st position, but also to have all the three amendments of that law appearing close to the top ranked position, e.g., within the top-10 results.

One reason leading to this problem is the fact that purely text-based search does not consider such referencing relationships between two different legal documents when determining the relevance of a candidate result. On the other hand, almost all legal documents would contain passages that indicate referencing relationships either between parts of the same document or with a part of (or the whole of) another legal document. In order to make use such relationships, e.g., for retrieval/search purposes, a more explicit representation is needed.

Intuitively, one could build a graph that captures a variety of referencing relationships within and between legal documents, e.g., when one article refers to another article in the same law, or when a law changes or repeals an earlier law. In fact, not just referencing relationships can be considered, but also semantic relationships between entities occurring in the text in a legal document.

[5]We use this pattern as a shorthand for the name of a law.

[6]performed using free service on March 2, 2022

The study of using graphs for knowledge representation in recent years brings about the idea of *knowledge graph*. Hogan et al. [4] define a knowledge graph (KG) as "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities." Further, they consider such a KG to be "potentially enhanced with representations of schema, identity, context, ontologies and/or rules." Thus, a KG may concern both knowledge representation at the data level such as instances and relations between them, as well as the more abstract schema level such as classes, terminological relations, and logical constraints imposed on them.

As a way of structuring data and knowledge that enables extensive linking and integration, KGs have found a wide-ranging use in applications [4]. For example, Google uses the knowledge graph to improve the performance of its search engine. Some commerce platforms such as Amazon, eBay, Airbnb, and Uber Eats also utilize knowledge graphs on their search and recommendation systems to improve user experience. Semantic search applications such as question answering systems, information retrieval systems, and recommendations systems are the most popular knowledge graph applications [5].

In the legal domain, the use of KG has been preceded by research on the use of ontologies to represent concepts and relationships existing in legal text pioneered by Functional Ontology of Law (FOLaw) [6] and Frame-Based Ontology (FBO) [7]. Further advances were done in developing legal core ontologies such as LRI Core [8] and LKIF Core [9]. Most of these research focused on capturing notions in the legal domain as schema-level knowledge without caring too much about knowledge at the data level.

More recently, however, works on legal knowledge graphs put more emphasis on capturing data-level knowledge, not just on the schema-level knowledge. Such works are often motivated by use cases in government and private organizations. For example, Filtz et al. [10] mention that more than 20 European countries built their legal information system around the ELI [11] and ECLI [12] ontologies. Specifically, they describe the use of ELI and ECLI to build a knowledge graph based on Austrian legal documents. This KG was constructed according to the the vocabulary and knowledge model espoused by these ontologies, and then populated using instance data extracted from the documents. On top of these ontologies, a search service was developed. Another example includes Finlex[7] [13], Finland's legal in-

[7]`https://www.finlex.fi/fi/`

formation system that employs both ELI and ECLI as the underlying metadata model. Finlex features a KG-based search that is capable not only returning a relevant legal document given the search keywords, but also an appropriate context information on the results, such as the relationship between the returned legal text with other legal text or information on whether the legal text has been amended or repealed, together with links to the documents that amend or repeal the legal text.

For Indonesia, on the one hand, no comprehensive legal KG has been developed. On the other hand, there are at least tens of thousands Indonesian legal documents covering regulations enacted by various government bodies, not to mention documents containing court decision at various levels. As illustrated by the example at the beginning of this section, retrieving legal information from such a large collection of documents is difficult due to the lack of linkages between content in different legal documents. A legal KG for Indonesia can significantly help such a retrieval service.

As part of an effort to fulfill the aforementioned needs, we present in this paper the LexID KG, an **Indonesian legal KG** that covers a significant part of Indonesian legal regulations enacted by various government bodies and agencies. Specifically, this KG is constructed from more than 20 thousands documents of legal regulations available from peraturan.go.id, which is an official repository of Indonesian legal documents. The construction method we employ can be viewed as a rule-based approach exploiting the structural as well as lexical patterns contained in the legal documents. We do not choose a statistical, machine learning-based approach due to the lack of training dataset to build a good model.

In addition to the KG, which contains data-level statements, we also include the LexID ontology, a **novel legal ontology for Indonesian legal domain** to enrich the LexID KG with schema-level knowledge. This ontology is designed by combining features of three existing legal ontologies, namely FBO, ELI, and ECLI ontologies. From the first, we adapt its approach for modeling knowledge contained semantically by the legal text, while the latter two inspire the metadata structure, which we adapt to be more suitable with the Indonesian case. Both the KG and the ontology are available online.[8]

Our work is not the only recent effort for constructing Indonesian legal ontology. Another work with a similar objective as ours is Lex2KG [14][9]. However, their work focuses only on metadata representation and legal document structure. Lex2KG are not explicitly represents the relationships between legal documents also does not cover the semantic representation of the legal clause of the legal document articles.

We also include an evaluation of the proposed KG and ontology. The evaluation aims to see whether the knowledge modeled by the KG can satisfy a variety of information needs. Such information needs are expressed by a number of specific questions and we consider the KG to be adequately modeled if an appropriate formal query expressing those questions can be correctly and completely answered by using vocabulary and instance data in the KG. Note that, we do not propose a system for legal information retrieval that employs a KG and ontology in the background. Rather, we present in this paper a necessary building block for realizing such a system that has a potential to have a better performance than existing legal search systems, especially for Indonesian legal documents. Note also that the term *legal document* in this paper refers to those legal documents produced by government agencies. Thus, it does not cover those produced by private parties, e.g., legal contracts or agreements between private parties. In addition, it also does not include court decisions and jurisprudence as the knowledge source of the KG does not contain such documents. We leave the KG construction from such documents for a future work.

This paper is organized as follows. After presenting the motivation in this section, Section 2 introduces a number of basic notions relevant for this paper. Section 3 describes the ontology we use to express schema-level knowledge we need. Section 4 details the rule-based approach we employ to construct the KG. The description of some query examples and the evaluation result of our ontology are in Section 5 and Section 6, respectively. And finally, in Section 7, we present our conclusion.

2. Basic Terminologies and Related Works

In this section, we introduce a number basic notions, including the standard structure of Indonesian legal documents, the notion of knowledge graph, as well as examples of existing legal ontologies.

2.1. Structure of Indonesian Legal Document

The structure of a legal document in the Indonesian legal system is formally regulated by the Indonesian government through Act 12/2011 [1].

[8]`https://github.com/ninggar17/Lexid.git`
[9]`https://github.com/aabccd021/legal-kg`

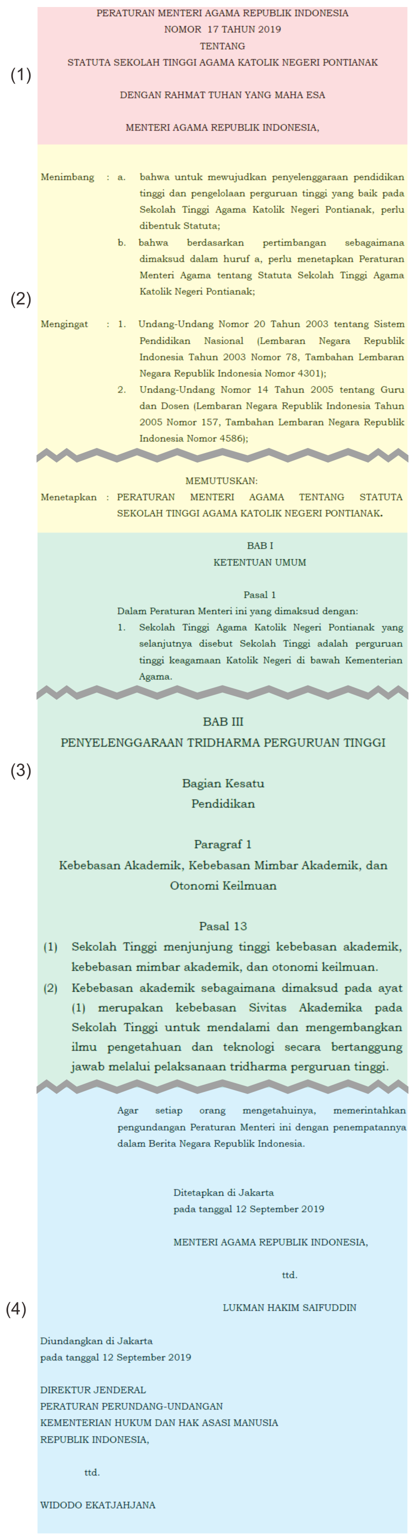
Generally, such a document consists of four mandatory segments, namely the title, the preamble, the body, and the closing. Fig. 1 depicts the structure of a legal document with their segments partially shown.

The title is composed of the type, year, number, and name of the regulation, respectively. In Fig. 1, the type part of the title is *"PERATURAN MENTERI AGAMA REPUBLIK INDONESIA"*. The year is 2019. The number is 17. The name of the regulation is *"STATUTA SEKOLAH TINGGI AGAMA KATOLIK NEGERI PONTIANAK"*, while the creator is *"MENTERI AGAMA REPUBLIK INDONESIA"*.

The preamble contains the consideration matters, legal basis, and dictum. In Fig. 1, the consideration matters always start with the heading *"Menimbang"* followed by an enumerated list of statements. The legal basis part starts with the heading *"Mengingat"* followed by an enumerated list of existing regulations that form the legal basis for this legal document. The preamble is closed by the dictum, which is indicated by the heading *"MEMUTUSKAN"* followed typically by the statement of formal issuance of the regulation. In older legal documents (prior to the enaction of Act 12/2011), the preamble may also contain an enumerated list preceded by the heading *"Memperhatikan"*. This list also contains existing regulations used as a legal basis for the legal document in concern.

The body is the core part of the document that follows after the preamble. It consists of a set of complete sentences called *legal clauses*. An *article* (*'Pasal'*) consists of either a single clause or more than one clauses. In the latter case, each clause in the same article is designated as a *section* (*'Ayat'*) of that article. Thus, an article that consists only of a single clause will not be divided into sections. Several articles may be grouped into *chapters* (*'Bab'*), *parts* (*'Bagian'*), or *paragraphs* (*'Paragraf'*). Every paragraph must belong to a part, and every part must belong to a chapter. These three groupings are optional. However, a paragraph cannot exist without an enclosing part, and a part cannot exist without enclosing chapter.

The numbering of the groups also follow a standard scheme. Chapters employ Roman numbering with chapter title. Parts use the verbalized form of the numbers, e.g., "Kesatu", which means the "first", also with part title. Paragraphs, articles, and sections employ arabic numbering. Of these three, paragraphs always have a title, while section numbers are always written inside parentheses. In addition, an article or a section may be expressed in the form of a compound sentence that contains an enumerated list of subordinate clauses. In such a case, the subordinate



(1)

PERATURAN MENTERI AGAMA REPUBLIK INDONESIA
NOMOR 17 TAHUN 2019
TENTANG
STATUTA SEKOLAH TINGGI AGAMA KATOLIK NEGERI PONTIANAK

DENGAN RAHMAT TUHAN YANG MAHA ESA

MENTERI AGAMA REPUBLIK INDONESIA,

(2)

Menimbang : a. bahwa untuk mewujudkan penyelenggaraan pendidikan tinggi dan pengelolaan perguruan tinggi yang baik pada Sekolah Tinggi Agama Katolik Negeri Pontianak, perlu dibentuk Statuta;
b. bahwa berdasarkan pertimbangan sebagaimana dimaksud dalam huruf a, perlu menetapkan Peraturan Menteri Agama tentang Statuta Sekolah Tinggi Agama Katolik Negeri Pontianak;

Mengingat : 1. Undang-Undang Nomor 20 Tahun 2003 tentang Sistem Pendidikan Nasional (Lembaran Negara Republik Indonesia Tahun 2003 Nomor 78, Tambahan Lembaran Negara Republik Indonesia Nomor 4301);
2. Undang-Undang Nomor 14 Tahun 2005 tentang Guru dan Dosen (Lembaran Negara Republik Indonesia Tahun 2005 Nomor 157, Tambahan Lembaran Negara Republik Indonesia Nomor 4586);

MEMUTUSKAN:
Menetapkan : PERATURAN MENTERI AGAMA TENTANG STATUTA SEKOLAH TINGGI AGAMA KATOLIK NEGERI PONTIANAK.

(3)

BAB I
KETENTUAN UMUM

Pasal 1
Dalam Peraturan Menteri ini yang dimaksud dengan:
1. Sekolah Tinggi Agama Katolik Negeri Pontianak yang selanjutnya disebut Sekolah Tinggi adalah perguruan tinggi keagamaan Katolik Negeri di bawah Kementerian Agama.

BAB III
PENYELENGGARAAN TRIDHARMA PERGURUAN TINGGI

Bagian Kesatu
Pendidikan

Paragraf 1
Kebebasan Akademik, Kebebasan Mimbar Akademik, dan Otonomi Keilmuan

Pasal 13
(1) Sekolah Tinggi menjunjung tinggi kebebasan akademik, kebebasan mimbar akademik, dan otonomi keilmuan.
(2) Kebebasan akademik sebagaimana dimaksud pada ayat (1) merupakan kebebasan Sivitas Akademika pada Sekolah Tinggi untuk mendalami dan mengembangkan ilmu pengetahuan dan teknologi secara bertanggung jawab melalui pelaksanaan tridharma perguruan tinggi.

(4)

Agar setiap orang mengetahuinya, memerintahkan pengundangan Peraturan Menteri ini dengan penempatannya dalam Berita Negara Republik Indonesia.

Ditetapkan di Jakarta
pada tanggal 12 September 2019

MENTERI AGAMA REPUBLIK INDONESIA,

ttd.

LUKMAN HAKIM SAIFUDDIN

Diundangkan di Jakarta
pada tanggal 12 September 2019

DIREKTUR JENDERAL
PERATURAN PERUNDANG-UNDANGAN
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA
REPUBLIK INDONESIA,

ttd.

WIDODO EKATJAHJANA

Figure 1. Structure of Indonesian legal documents: (1) title, (2) preamble, (3) body, and (4) closing.

clauses may be listed using *numbers* (*'Angka'*) or *letters* (*'Huruf'*). All groupings, including the aforementioned numbers and letters, may be referenced by text in any part of the document.

After the body, the document contains the closing. This segments is begun by a statement that formally close the document. This closing statement may vary between different legal documents, but most of the time states that the enacted regulation is promulgated in an official promulgation place and thus, must be known by everyone. The promulgation place includes *'Lembaran Negara'*, *'Berita Negara'*, *'Lembaran Daerah'*, and *'Berita Daerah'* — which one is being used depends on the type of the legal document. A known exception to this is the constitution of which the promulgation place is not mentioned. After the above closing statement, the document mentions the city name where the law is enacted together with the corresponding date and the name of both the office and the official who enacts the law. Finally, the closing segment of the document is ended by a sub-segment providing the city name and date of the promulgation of the legal document, as well as the name of both the office and the official who promulgates the law.

2.2. Knowledge Graph

Knowledge graph (KG) refers to a way to represent large-scale data and knowledge in a graph-based model [4]. In such a model, vertices and edges represent entities and relationships between entities, respectively. A graph data model may be equipped with a schema, but unlike relational data model, we can still work with a graph data model when the schema is incomplete or even absent. Historically, the term knowledge graph has been used since 1972, but its current usage was popularized in 2012 when Google used it to name its graph data model.

KG provides a lot of flexibility when there is a new data source to be integrated into the KG. This is in contrast to relational data model where the schema must be defined first and the data (including new ones) must comply to it. In relational data model, if new data comes in, but does not comply with the schema, then we are forced to either discard it or modify the schema. Moreover, relational data model also discourages circular relationships, while KG can easily and seamlessly model it without issues,

2.2.1. KG data model. One major data model used by KGs is based on *directed edge-labeled graph* (*DELG*). A DELG consists of a set of labeled vertices and labeled directed edges. Labeled vertices represent entities, e.g., West Java, East Java, Bandung, Tasikmalaya, Depok, Surabaya, and Malang. Meanwhile, labeled directed edges represent relationships, which are simply a set of triples (u, p, v) where u and v are labeled vertices and p is the edge label, e.g., (Bandung, cityIn, WestJava), (Tasikmalaya, cityIn, WestJava), (Depok, cityIn, WestJava), (Surabaya, cityIn, EastJava), and (Malang, cityIn, EastJava).

World Wide Web Consortium (W3C) published a standard graph data model based on directed edge-labeled graph, called Resource Description Framework (RDF) [15]. In RDF, a graph (i.e., *RDF graph*) is a collection of triples (s, p, o) where s (called *subject* of the triple) and o (called *object* of the triple) correspond to vertices of the graph, and p (called *predicate* of the triple) corresponds to the directed edge from s to o.

RDF defines three types of vertices. The first type is Internationalized Resource Identifiers (IRIs), used for the global identification of entities on the Web, e.g., `http://www.w3.org/2002/07/owl#Thing`. Such an IRI may be abbreviated with a namespace prefix that can be defined at the beginning of the an RDF document. For example, one can define `owl:` as a namespace prefix that abbreviates `http://www.w3.org/2002/07/owl#`. Thus, `http://www.w3.org/2002/07/owl#Thing` can be written as `owl:Thing`.

The second type is literal, used to represent strings (with or without language tags) and values of other data types (integers, dates, etc.), e.g., `"Jakarta"`, `"200"8sd:int`, or `"Jawa Barat"@id`. The last type is a blank node, used to represent anonymous vertices and does not need to have an identifier. IRIs and blank nodes may appear as a subject or an object of a triple, while literals may only appear as an object of a triple.

Meanwhile, the label of all edges in the graph, i.e., every triple's predicate, is always represented by an IRI. Such an IRI is called a *property*. Given an RDF triple (s, p, o), we sometimes call o the *value* of the property p. We can see from this definition that RDF allows an IRI to appear in an RDF graph as both a vertex and an edge.

Publishing RDF graphs can be done by serializing them as RDF files, which are then made available on the web, or by exposing them via an endpoint of some RDF store that can be queried using SPARQL (discussed briefly next). There are several alternative syntaxes for RDF serialization, including RDF/XML, N-Triples, Turtle, TriG, N-Quads, JSON-LD, and RDFa [16].

In addition, the four linked data principles [17] stipulate that each IRI needs to be a HTTP IRI that is web-resolvable. That is, if one access the IRI of an entity or property either via standard web browser or programmatically, an appropriate human-readable or machine-readable response needs to be returned. This implies that an appropriate web infrastructure must be set up to really make RDF graphs publicly available [18].

2.2.2. KG Query. Data retrieval from a KG, like in other data models, is done via a query language. Such a query language is closely tied to the underlying graph data model being employed to represent the data. In the RDF case, the standard query language is called SPARQL [19]. SPARQL not only defines standard relational query operations such as join, union, projection, etc., but also traversal operations that enable users to find entities connected recursively through long paths.

Formally, a SPARQL query is simply a set of *basic graph patterns* (BGPs). A BGP is like an RDF triple, but may contain variables (indicated by a question mark). For example, the BGP (?x, `http://ex.org/cityIn`, `http://ex.org/WestJava`) expresses the query of finding entities that is a city in West Java. When this query is executed, the engine will try to match ?x with the subject of any triple whose predicate is `http://ex.org/cityIn` and whose object is `http://ex.org/WestJava`. SPARQL provides a set of operators that can be employed to express various complicated queries. The actual syntax is slightly more elaborated. Examples are given in Section 5.

2.2.3. KG Schema (Ontology). Despite modeling a graph do not need a schema, we can create a graph schema to set the structure or semantics as a rule while developing graph data. One of the graph schemas is a semantic schema. It defines the meaning of higher-order terms of the graph and facilitates reasoning using those terms. Suppose we found several groups of nodes based on the type of entity. Thus, we can decide to define a class schema. We can also define several other schemas. RDF Schema (RDFS) is a standard schema to define the semantic schema of RDF. Some of the schemes we can use are sub-class, sub-property, domain, and range. We can see RDFS itself as RDF. Other than that, we can also define more deeply schemas using standard schemas such as the Web Ontology Language (OWL). A semantic schema is generally an incomplete schema. Therefore the semantic scheme adopts Open World Assumption (OWA) compared to Closed World Assumption (CWA). From the OWA point of view, we can not say it is prohibited of something not defined in the schema.

Graph-based knowledge representation is can also be called Knowledge Graph (KG). It contains several domain topics. KG models all possible entities and relations in an ontology. There is no restriction on determining the relationships. We can construct the ontology in various ways, such as curation, crowdsourcing, and data extraction [20]. CyC is an example ontology constructed using the curation method. Another examples is Wikidata, which is from crowdsourcing. DBpedia and YAGO are in which the construction is by extracting semi-structured web data like Wikipedia. Various ontology-based intelligent systems have been developed, such as information acquisition, semantic search, chatbot, question and answer, and recommendation systems [5].

2.3. Existing Legal Ontologies

The conceptualization and representation of legal documents have received attention for a long time, especially in the EU, which aims to integrate the laws of each country member. There is some initiation of several projects related to legal conceptualization and representation to reach the goal. One of the projects is Estrella[10]. It's a project to standardize legal data of the European to a standard XML-based semantic web, like RDF and OWL, and develop a knowledge base application to utilize the standardized data. Other project is Lynx[11] which aims to build a legal intelligence system for the EU. As part of the project, they construct a knowledge graph from legal data as a basis of the project.

In recent years, several works are proposing legal ontologies. Based on the kind of data has used, there are two types of ontology, legal semantic and metadata ontologies. Breuker et al. found at least 23 existing legal semantic ontologies and grouped them based on the purpose, characteristic, and legal domain [21]. Some of them are as follows FO-Law [6], FBO [7], LRI Core [8], CLO [22], LKIF Core [9]. The FBO (Frame-Based Ontology) is one of the pioneers of legal ontology to represent legal texts semantically. As legal semantic ontology, FBO is more simple compared to others. It only defines three frames, which are Norm, Act, and Concept. Each entity of Norm represents a legal norm clause, while each entity of Act represents an action performed by someone or institution in the norm clause. And last, each entity of Concept describes a legal term contained in the norm clause.

Other legal ontologies are the European Legal-Document Identifier (ELI) [11] and The European Case Law Identifer (ECLI) [12]. Both are some examples of existing legal metadata ontologies. Both

[10]`http://www.estrellaproject.org/`

[11]`https://lynx-project.eu/project/summary`

describe legal documents using their metadata information such as the title, type, person or institution issuing, issued date, and place of publication.

The Indonesian legal ontology has initiated by Lex2kg [14]. The ontology is a legal metadata ontology and adopts the existing ontology, ELI. However, Lex2kg makes adjustments since classes and properties defined by ELI are not complete enough and designed only for legal documents around the EU. Lex2kg defines the class structure of the legal document, which ELI doesn't have. The class structure defined is based on the type of documents. Lex2kg defines some classes representing the group of norm clauses corresponding to the document body structure. It also adjusts the property of classes with metadata provided by the document.

We use the previous work as a basis of our ontology. Our ontology consists of both legal metadata or semantic ontology. It also can capture the relationship between two legal documents or clause norms since Lex2kg doesn't explicitly show it. In addition, we construct our ontology automatically.

3. LexID Ontology

In this section, we present the structure of LexID ontology, which is designed as a schema for LexID KG. We elaborate the classes and properties in LexID ontology next. Here and throughout the paper, we use the following namespace prefixes

- `lexid-s:` for `https://w3id.org/lexid/schema/`,
- `lexid:` for `https://w3id.org/lexid/data/`,
- `rdf:` for `http://www.w3.org/1999/02/22-rdf-syntax-ns#`,
- `rdfs:` for `http://www.w3.org/2000/01/rdf-schema#`,
- `owl:` for `http://www.w3.org/2002/07/owl#`,
- `dct:` for `http://purl.org/dc/terms/`,
- `xsd:` for `http://www.w3.org/2001/XMLSchema#`, and
- `wd:` for `http://www.wikidata.org/entity/`.

Unless stated otherwise, `lexid-s:` is intended for class and property IRIs, while `lexid:` is for IRIs of instance data. Also, `rdf:`, `rdfs:`, `owl:`, and `dct:` are namespace prefixes for vocabulary terms taken from the RDF, RDFS, OWL, and Dublin Core vocabulary, respectively.

We define eight top-level classes as shown in Fig. 2, some of which have several subclasses. We express the ontology using Web Ontology Language (OWL), which defines `owl:Thing` as the superclass of every classes. Details of each of these classes are explained separately in their own section later.

In addition, LexID ontology provides a collection of properties (i.e., binary relations) to capture

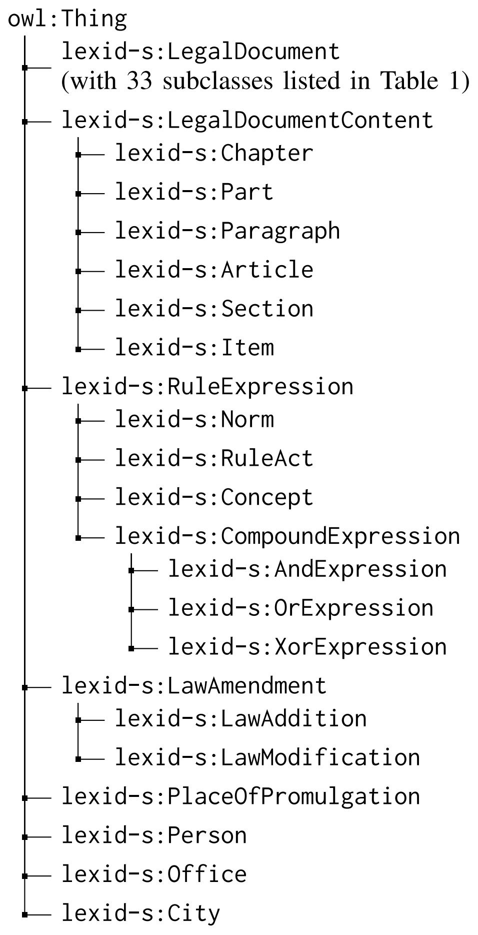


Figure 2. Top-level classes of LexID ontology.

different types of relationships between objects it models. The complete list of these properties is given by Table A1-Table A5. We explain some of these properties in the following section.

Because of the OWA adoption, we can still dynamically modify our ontology scheme even though we have finished the construction. It is very beneficial since the legal systems in Indonesia can be changed in the future.

3.1. General metadata properties

LexID ontology defines a number of properties, some of which are taken from RDFS, OWL, and Dublin Core vocabulary, to describe general metadata information of a legal document, see Table A1 for more detail.

As mentioned in the Section 2.1, the government determines several places to promulgate the legal document according to its type. The class `PlaceOfPromulgation` represents places of legal document promulgation. Instances of this class include:

- lexid:Lembaran_Negara and lexid:Berita_Negara representing the Indonesian state gazette and state report collection;
- 34 pairs of instances representing provincial gazettes and provincial report collections (since Indonesia has 34 provinces), e.g., lexid:Lembaran_Daerah_West_Java and lexid:Berita_Daerah_West_Java;
- 416 pairs of instances representing regency gazettes, such as lexid:Lembaran_Daerah_Kabupaten_Bogor, and regency report collections, such as lexid:Berita_Daerah_Kabupaten_Bogor; and
- 98 pairs of instances of representing city gazettes and city report collections, e.g., lexid:Lembaran_Daerah_Kota_Semarang and lexid:Berita_Daerah_Kota_Semarang.

3.2. Modeling Legal Document and Their Relationships

The class LegalDocument represents the abstraction of Indonesian legal documents (i.e., not the physical documents). Subclasses of LegalDocument, as seen in Table. 1, are introduced mainly based on Act 12/2011, which defines the types of legal document in the Indonesian law system as well as the hierarchical precendence of between them. Names of these classes are directly taken from the legal document type names we explain next.

First, article 7 of Act 12/2011 stipulates the following nine types of legal document ordered according to their legal precedence (from high to low).

- The national constitution.
- People's Consultative Assembly resolutions (*'Ketetapan Majelis Permusyaratan Rakyat'* or *'Tap MPR'*).
- Acts (*'Undang-undang'* or *'UU'*) and government regulations in-lieu-of-acts (*'Peraturan Pemerintah Pengganti Undang-undang'* or *'Perppu'*).
- Government regulations (*'Peraturan Pemerintah'* or *'PP'*)
- Presidential regulations (*'Peraturan Presiden'* or *'Perpres'*)
- Provincial regulations (*'Peraturan Daerah Provinsi'* or *'Perprov'*)
- Municipality regulations (*'Peraturan Daerah Kota'* or *'Perkot'*) and regency regulations (*'Peraturan Kabupaten'*

Note that two pairs of legal document types are listed together, indicating that they have an equal legal precedence. Each of these legal document types has an associated LexID class.

In addition, article 8 of Act 12/2011 recognizes other regulations that are law-binding so long as those regulations are put into law by the order of a regulation with a higher precedence or due to the official authority of the government institution/agency that issues them. They are regulations (*'Peraturan'*) issued by the People's Consultative Assembly, the House of Representative (*'Dewan Perwakilan Rayat (DPR)'*), the Regional Representative Council (*'Dewan Perwakilan Daerah (DPD)'*), the Supreme Court (*'Mahkamah Agung (MA)'*), the Constitutional Court (*' Mahkamah Konstitusi (MK)'*), the Audit Board (*'Badan Pemeriksa Keuangan (BPK)'*), the Judicial Commission (*'Komisi Yudisal (KY)'*), Bank Indonesia (BI), government ministries, government agencies and commissions (*'Badan'*, *'Lembaga'*, and *'Komisi'*), the provincial people's representative councils (*'Dewan Perwakilan Rakyat Daerah (DPRD) Provinsi'*), (provincial) governors (*'Gubernur'*), the regional legislative council of regencies and municipalities (*'DPRD Kabupaten/Kota'*), regents/mayors (*'Bupati/Walikota'*), and other govern-

Table 1. Subclasses of LegalDocument (33 of them), all are defined in the lexid-s namespace.

Constitution
AmendmentToTheConstitution
PeoplesConsultativeAssemblyResolution
Act
GovernmentRegulationInLieuOfAct
GovernmentRegulation
PresidentialRegulation
PresidentialDecree
PresidentialInstruction
PeoplesConsultativeAssemblyRegulation
HouseOfRepresentativeRegulation
RegionalRepresentativeCouncilRegulation
SupremeCourtRegulation
ConstitutionalCourtlRegulation
AuditBoardRegulation
JudicialCommissionRegulation
BankIndonesiaRegulation
AgencyRegulation
MinisterialRegulation
MinisterialDecree
JointRegulation
ProvincialLegislativeCouncilRegulation
ProvincialRegulation
GovernorRegulation
GovernorDecree
MunicipalityLegislativeCouncilRegulation
RegencyLegislativeCouncilRegulation
MunicipalityRegulation
RegencyRegulation
MayorRegulation
RegentRegulation
MayorDecree
RegentDecree

ment officials/institutions of similar authorities. The legal precedence of these regulations are not explicitly stated in Act 12/2011, but based on Indonesian legal convetion, are placed in accordance to the level of authority and legal standing of the issuing government officials/agencies.

Other than the above regulations, the Indonesia law system also recognizes regulations called decrees (*'Keputusan'*) or instructions (*'Instruksi'*), such as presidential decrees, presidential instructions, ministerial decrees, governor decrees, regent/mayor decrees, and other decrees by government officials. By article 100 of Act 12/2011, these regulations are considered to have the same legal precedence as regulations by the corresponding government official. So, for example, presidential decrees have the same legal precedence as presidential regulations.

Finally, it is possible for more than one government officials or agencies to issue a joint regulation (*'Peraturan Bersama'*). Such a regulation is viewed as if there are multiple regulations issued by different officials but the content of the regulations is exactly the same. For example, if the minister of commerce and the minister of maritime affairs and fishery jointly issue a joint regulation, then that regulation would have two designated regulation numbers associated to each ministry. The legal precedence is thus equal to other ministry regulations.

Next, we also model relationships between different legal documents, see Table A2 for more detail. This is achieved through the use of properties P (all in `lexid-s` namespace) listed below in a triple of the form (D, P, D') in the LexID KG where D and D' are instances of legal documents. Also, if such a triple (D, P, D') is asserted in the KG, then the triple (D', P^{-}, D) holds implicitly in the KG where P^{-} is the inverse of P.

- P = `hasLegalBasis` (with `legalBasisOf` as inverse) indicates that D has D' its legal basis. The latter is always of equal or higher precedence than the former.
- P = `implements` (with `implementedBy` as inverse) indicates that D is an implementing regulation of D' whose precedence is higher.
- P = `amends` (with `amendedBy` as inverse) indicates that D is an amendment of D' whose precedence is equal or lower.
- P = `repeals` (with `repealedBy` as inverse) indicates that D repeals D' whose precedence is equal or lower.

Note that the amendment and repeal relationships above do not capture the details of the amendment and repeal themselves as they are given by some of the articles in D. Such details imply richer relationships between elements of the content of different legal documents. Modeling of such relationships are discussed next.

3.3. Modeling Legal Document Content Structure

The `LegalDocumentContent` class represents types of grouping of legal clauses that exist in the body segment of a legal document, as mentioned in Section 2.1. It has six subclasses, namely `Chapter`, `Part`, `Paragraph`, `Article`, `Section`, and `Item`, which represent chapters (*'Bab'*), parts (*'Bagian'*), paragraphs (*'Paragraph'*), articles (*'Pasal'*), section (*'Ayat'*), and items in the body segment of the document. Here, items correspond to subclauses appearing as elements of an enumerated list inside an article or section. Each item is indicated by a number (*'Angka'*) or a letter (*'Huruf'*). Note that instances of all the aforementioned classes are not the text of the legal clauses themselves, but rather, objects named with some IRI to which the textual content of the legal clauses are attached via the property `description`. In addition, we also modeled properties from and to `LegalDocumentContent` in Table A3.

3.4. Representing Legal Document Changes at Content Level

In Section 3.2, we have explained the properties for modeling the relationship (implementation, amendment, and repealing) between legal documents at the document level. However, note that similar, but more fine-grained, relationships may occur at the content level, For example, the consideration matters of a legal document may explicitly say that the legal document implements a particular article of another legal document, or an article of a legal document explicitly asserts that another article in a different legal document must be changed or even deleted. We describe how LexID ontology models such relationships in this section — all properties and classes are in the `lexid-s` namespace, see Table A4 for the detail.

First, the property `implements` may also be used in a triple of the form $(D, \texttt{implements}, y)$ to indicate that a legal document D implements y where y is an `Article` or `Section`. Similarly, the property `repeals` may also be used in a triple of the form $(x, \texttt{repeals}, D')$ to indicate that x repeals the legal document D' where x may be an `Article` or `Section`. Note that the semantic of `repeals` and also of the property `implements` above are defined in addition to their semantic given in Section 3.2.

Next, fine-grained changes to legal document content consist of deletions, additions, and modifications. We define the property deletes, as shown in Fig. A.1, which can be used in a triple of the form $(x, \texttt{deletes}, y)$ with the following semantics: an article or section x (in one legal document) states that y (from another legal document) is deleted from the legal document where it belongs. Here, y can be a chapter, a part, a paragraph, an article, a section, or an item.

Addition of legal document content cannot be modeled via a simple property because it involves more than two components, namely the article/section that expresses the addition, the textual description of the legal clause(s) to be added, and the location (in another legal document) where the new content needs to be added. For this reason, we create a class Addition to represent the addition of a new legal content. Together with this class, we also define three properties: adds, hasAdditionTarget and hasAddedContent.

Similar to addition, modification also involves more than two components, namely the article/-section that expresses the modification, the textual description of the legal clause(s) to be modified, and the old version of content needs be modified. therefore, we also create a class Modification to represent the modification of legal content. Together with this class, we also define three properties: modifies, hasModificationTarget and hasModificationContent.

3.5. Modeling Legal Clauses

The class RuleExpression captures the semantic content of the legal clauses in the legal document. We adopt the approach used in the FBO ontology, which employs fewer frames and a simpler structure compared to other existing legal ontologies.

Following FBO, the class RuleExpression has three sub-classes, namely Norm, RuleAct, and Concept. In addition, as shown in Fig. 2, we also add a new class called CompoundExpression with three subclasses AndExpression, OrExpression, and XorExpression to model conjunctive or disjunctive expressions appearing in a legal clause. The relationships between instances of RuleExpression are modeled in Table A5.

4. LexID KG Construction

In this section, we describe the steps we conduct to construct LexID KG from legal documents. A high-level workflow of this process is illustrated by Fig. 3. We start with 32,218 legal documents from downloaded from https://peraturan.go.id containing most types of regulations described in Table 1.

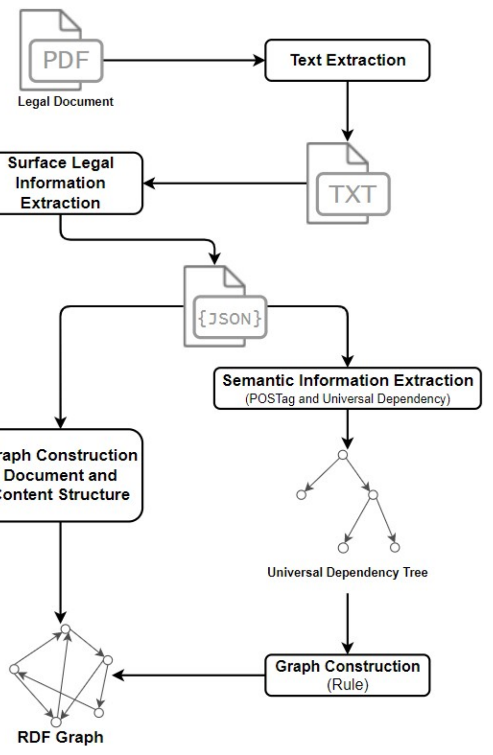

PDF
Legal Document
Text Extraction
TXT
Surface Legal Information Extraction
{JSON}
Semantic Information Extraction
(POSTag and Universal Dependency)
Graph Construction Document and Content Structure
Universal Dependency Tree
Graph Construction
(Rule)
RDF Graph

Figure 3. KG construction flowchart

4.1. Initial Processing

In the initial processing, we extract the textual content of the aforementioned 32,218 legal documents using the PDFBox library.[12] As a result of this initial processing step, we obtain 30,478 documents whose textual content is successfully extracted. We then construct LexID KG by processing the resulting text documents through the subsequent steps in the workflow. Each document yields a subset of the KG. At the end of the workflow, LexID KG is obtained as a union of those subsets. IN this initial processing steps, we failed to extract 2,427 documents due permission restrictions imposed on those documents.

4.2. Surface Legal Information Extraction

Structure and format of Indonesian legal documents is formally defined by Act 12/2011. We parse 30,478 legal documents (already text-formatted) and successfully extract legal information from 27,596

[12]https://pdfbox.apache.org/

documents. The remaining documents could not be parsed because either they are empty, their format is incorrect, or they are not the typical document we want to extract such as court decisions, documents containing explanation of other legal documents, or separate attachments to some legal documents.

The parsing process itself proceeds sequentially from the beginning to the end of the document. That is, we start by parsing the title part of the document, followed by parsing the preamble, then followed by parsing the body segment, and end with parsing the closing segment of the document. In all parts, we extract a variety of information according to the parsing rules listed in Table A7 and Table A8. The parsing result is stored as a JSON-formatted output with fields detailed in Table A9.

4.3. Instance IRI Naming Scheme

Before construction, we design an IRI naming scheme of the instance in the KG, described in Table A6 together with examples. As noted at the beginning of Section 3, instance IRIs reside in the namespace given by `https://w3id.org/lexid/data/`, abbreviated using the `lexid`: prefix. Instances of the class `Item` and `Concept`, in particular, have three and four IRI patterns, respectively. The IRI pattern of an instance of `Item` depends on two things: its numbering type (numbers or letters) and the part of the document structure that directly encloses it.

4.4. Graph Construction for Legal Document and Content Structure

At this stage, we construct RDF triples for LexID KG by using properties and classes specified in Section 3.1, 3.2, 3.3, and 3.4. The instance information are taken from the JSON-encoded result of surface information extraction explained in Section 4.2 (JSON fields are given in Table A9). Instance IRIs are constructed based on the naming scheme in in the Table A6. Figure A.5 shows examples of triples constructed in this process. In addition, from the JSON field 'document_structure', we construct a set of triples representing the document structure. For example, Figure A.6 shows some of the triples that represent the legal content structure of the Minister of Religion 17/2019. Note that this phase only deals with surface information as well as content structure, not the semantic content of the legal clauses in the document.

4.5. Semantic Information Extraction

The aim of this phase is to parse the semantic content from the text of legal clauses in the form POS tags and the relationships between the phrases as determined via universal dependency (UD) parsing. The POS tags and the UD parse tree are employed in the subsequent phase to construct the LexID KG triples that model the legal clauses. We use Stanza NLP Package[13] to obtain the UD parse tree and the POS tags of each token.

4.6. UD Tree Transformation for Legal Clauses

At this stage, each legal clause has been represented by a UD tree where each token has an associated POS tag. We then proceed by applying 10 transformation rules illustrated to the tree. Each rule is a pattern that transforms any part of the tree to which the pattern is applicable. The application is done sequentially and exhaustively. That is, we first apply rule (1) exhaustively, before applying rule (2) exhaustively, followed by applying rule (3) exhaustively, and so on. The rules are detailed below. The result of the transformation is a representation of the legal clause in the form of a graph structure from which RDF triples can be later created. Note that since we work with a UD tree, we note that originally each node in the tree is associated with a type (e.g., Noun, Verb, etc.) and a piece of text. During transformation, we may introduce new nodes, which also has a type and a text content. The type of these new nodes may come from the UD tree node types or from the following new types: Concept, RuleAct, and Norm. Figure 4 and 5 visualize the rule specification

1) ***Phrase contraction***: We perform phrase contraction by finding a multiword expression in the tree. The multiword expression pattern is recognized by the existence of an edge with label `COMPOUND`, `FIXED`, or `FLAT`. The rule then merge the two nodes connected by that edge into a new node representing a multiword expression containing the words in both original nodes. This resulting node has type Verb or Case if the one of the original node has type Verb or Case, respectively (with Verb takes typing precedence over Case). Otherwise, we simply fixed the new node type to Noun. The text content of the new node is the phrase obtained by concatenating the content of the original two nodes.

[13]`https://stanfordnlp.github.io/stanza/index.html`

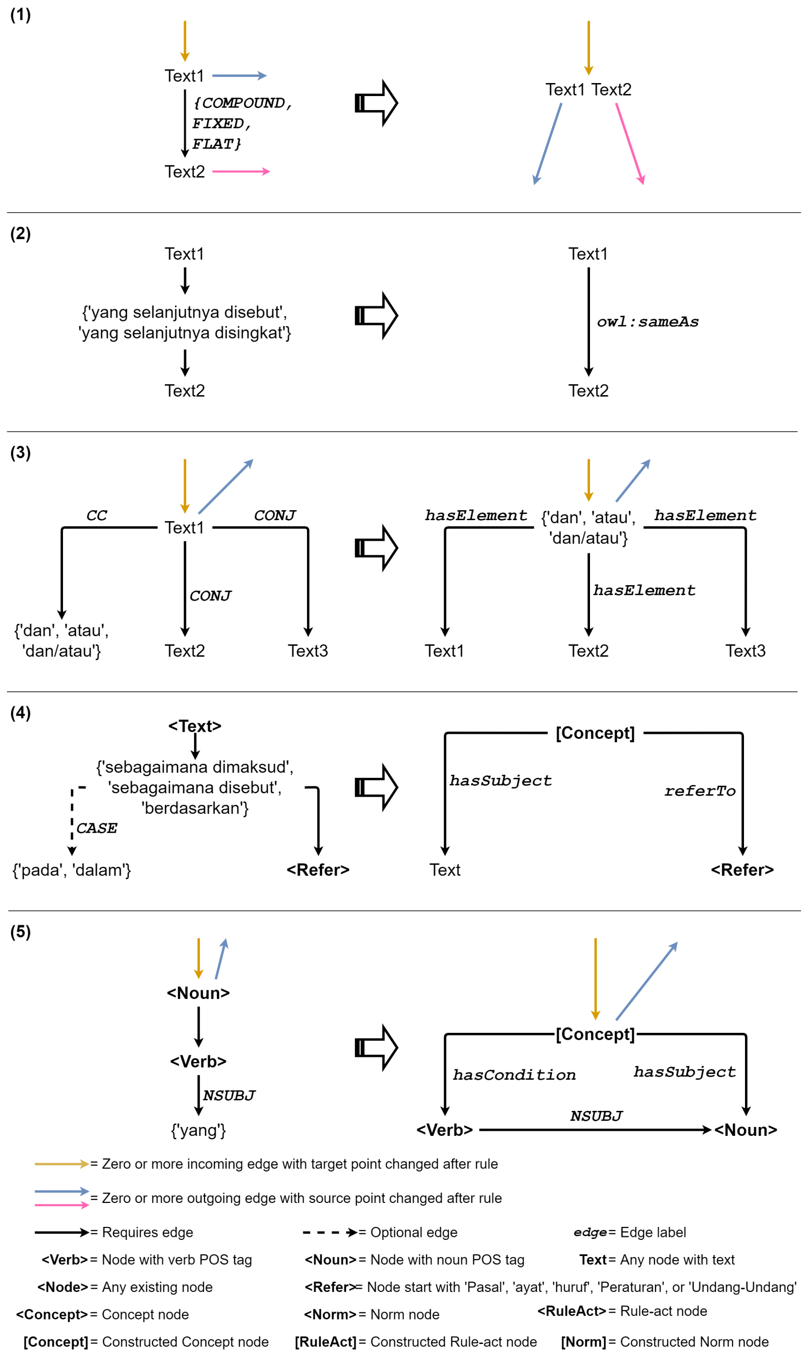

(1)
Text1
{COMPOUND, FIXED, FLAT}
Text2
Text1 Text2
(2)
Text1
{'yang selanjutnya disebut', 'yang selanjutnya disingkat'}
Text2
owl:sameAs
(3)
CC
CONJ
Text1
{'dan', 'atau', 'dan/atau'}
Text2
Text3
hasElement
(4)
<Text>
{'sebagaimana dimaksud', 'sebagaimana disebut', 'berdasarkan'}
CASE
{'pada', 'dalam'}
<Refer>
[Concept]
hasSubject
referTo
Text
(5)
<Noun>
<Verb>
NSUBJ
{'yang'}
[Concept]
hasCondition
hasSubject
= Zero or more incoming edge with target point changed after rule
= Zero or more outgoing edge with source point changed after rule
= Requires edge
= Optional edge
edge = Edge label
<Verb> = Node with verb POS tag
<Noun> = Node with noun POS tag
Text = Any node with text
<Node> = Any existing node
<Refer>= Node start with 'Pasal', 'ayat', 'huruf', 'Peraturan', or 'Undang-Undang'
<Concept> = Concept node
<Norm> = Norm node
<RuleAct> = Rule-act node
[Concept] = Constructed Concept node
[RuleAct] = Constructed Rule-act node
[Norm] = Constructed Norm node

Figure 4. Ten transformation rules of a UD tree from a legal clause to yield a graph from which RDF triples can be obtained: (1) phrase contraction, (2) alias detection, (3) reformation of coordinating conjunction, (4) reference detection, (5) conditional concept detection. Rule (6)-(10) are given in Figure 5. The left trees describe patterns to be matched, while the right trees describe the resulting shape after rule application. Patterns are applied to any applicable part of the tree without changing the source and target of any edges, except for the ones indicated by the yellow/blue/magenta arrows.

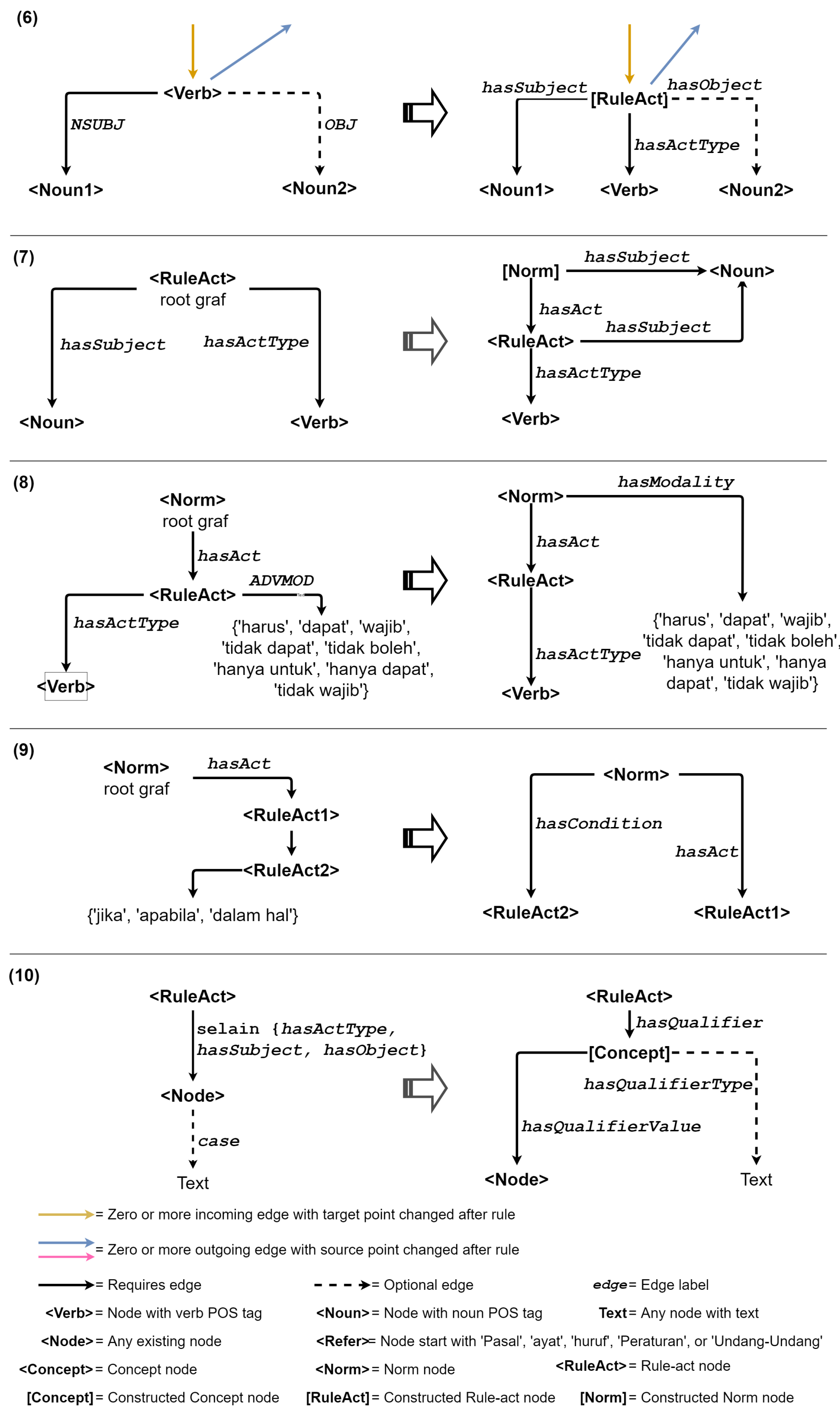

(6)
<Verb>
NSUBJ
OBJ
<Noun1>
<Noun2>
hasSubject
[RuleAct]
hasObject
hasActType
<Noun1>
<Verb>
<Noun2>
(7)
<RuleAct>
root graf
hasSubject
hasActType
<Noun>
<Verb>
[Norm]
hasSubject
<Noun>
hasAct
<RuleAct>
hasSubject
hasActType
<Verb>
(8)
<Norm>
root graf
hasAct
ADVMOD
<RuleAct>
hasActType
{'harus', 'dapat', 'wajib', 'tidak dapat', 'tidak boleh', 'hanya untuk', 'hanya dapat', 'tidak wajib'}
<Verb>
<Norm>
hasModality
hasAct
<RuleAct>
hasActType
<Verb>
{'harus', 'dapat', 'wajib', 'tidak dapat', 'tidak boleh', 'hanya untuk', 'hanya dapat', 'tidak wajib'}
(9)
<Norm>
root graf
hasAct
<RuleAct1>
<RuleAct2>
{'jika', 'apabila', 'dalam hal'}
<Norm>
hasCondition
hasAct
<RuleAct2>
<RuleAct1>
(10)
<RuleAct>
selain {hasActType, hasSubject, hasObject}
<Node>
case
Text
<RuleAct>
hasQualifier
[Concept]
hasQualifierType
hasQualifierValue
<Node>
Text
= Zero or more incoming edge with target point changed after rule
= Zero or more outgoing edge with source point changed after rule
= Requires edge
= Optional edge
edge = Edge label
<Verb> = Node with verb POS tag
<Noun> = Node with noun POS tag
Text = Any node with text
<Node> = Any existing node
<Refer> = Node start with 'Pasal', 'ayat', 'huruf', 'Peraturan', or 'Undang-Undang'
<Concept> = Concept node
<Norm> = Norm node
<RuleAct> = Rule-act node
[Concept] = Constructed Concept node
[RuleAct] = Constructed Rule-act node
[Norm] = Constructed Norm node

Figure 5. Continuation of Fig. 4. This depicts the following rules: (6) rule-act detection, (7) norm detection, (8) modality detection, (9) conditional norm detection, and (10) qualifier detection.

2) ***Alias detection***: We detect an alias of a word or phrase by looking a chain of three nodes where the middle one contains one of the following phrases: '*yang selanjutnya disebut*' or '*yang selanjutnya disingkat*'. This indicates that the expression represented by the first and the last nodes are aliases of the other. So, the rule removes the middle node and then makes a new direct edge between the first and last nodes labeled with `owl:sameAs`.
3) ***Reformation of coordinating conjunction***: This rule is applied if we have a coordinating conjunction indicated by an edge labeled with `CC` that points to a node C containing any one of the following words/phrases: {*'dan'*, *'atau'*, and *'dan/atau'*}. Let *text1* be the source node A_1 of that `CC` edge. Then, this node will be connected to one or more nodes $A_2, \ldots, A_N$ respectively containing text *text2*, ..., *textN* via edges with label `CONJ`. The rule thus reshaped the tree as follows: (i) all of `CC` edge and `CONJ` edges are removed; (ii) all other incoming and outgoing edges for A_1 become incoming and outgoing edges for C; and (iii) node C is connected to all $A_1, \ldots, A_N$ via edges labeled with `hasELement`.
4) ***Reference detection***: A part of a legal clause can refer to other clauses. We detect it through the existence of a node C containing one of the following phrases: *'sebagaimana dimaksud'*, *'sebagaimana disebut'*, and *'berdasarkan'*. Such a node always has one in-neighbor node A containing some text, and one out-neighbor node B containing text started by the word/phrase *'Pasal'*, *'ayat'*, *'huruf'*, *'Peraturan'*, or *'Undang-Undang'*, which we call the reference node. In addition, it may also have an out-neighbor node containing the word *'pada'* or *'dalam'* connected via an edge labeled by `CASE`. The rule then generates a new concept node (i.e., of type concept) and connects it to A via a `hasSubject` edge and to B via a `referTo` edge.
5) ***Conditional concept detection***: This rule is applicable when a chain of noun node, verb node, and another node with text `'yang'`. This indicates a concept defined in terms of the noun phrase conditioned by the verb expression. The rule thus generates a new concept node, and then connects it to the noun phrase via `hasSubject` edge and to the verb node via `hasCondition` edge. The original edges connecting those three nodes as well as the node with text `'yang'` are removed. In addition, other incoming and outgoing edges originally of the noun node are now attached to the new concept node.
6) ***Rule-act detection***: The pattern enabling application of this rule consists of a verb node connected to a noun node through a `NSUBJ` edge. Optionally, the verb node may also be connected to the second noun node through an `OBJ` edge. This rule then creates a structure representing rule-acts by constructing a new rule-act node, and then connects it to the first noun node via and `hasSubject` edge, to the verb node via a `hasActType` edge, and if the second noun node exists, to that second noun node via a `hasObject` edge. The other incoming and outgoing edges of the verb node are now attached to the new rule-act node. The `NSUBJ` and `OBJ` edges above are removed.
7) ***Norm detection***: We identify that a legal clause is a norm if the graph root is rule-act node. Therefore, for each rule-act node as a graph root found, we create a norm node and place it in the root replacing the rule-act node. The norm node then connects to the rule-act node with the `hasAct` edge. The subject of the rule-act node will also be the subject of the norm node.
8) ***Modality detection***: The modality node usually represents the following phrase: *'harus'*, *'dapat'*, *'wajib'*, *'tidak dapat'*, *'tidak boleh'*, *'hanya untuk'*, *'hanya dapat'*, and *'tidak wajib'*. If we find rule-act of the norm connected to the modality node with the `ADVMOD` edge, the norm node then connects to the modality node using the `hasModality` edge. Next, we remove The `ADVMOD` edge that connects the modality node to the rule-act node.
9) ***Conditional norm detection***:Let a chain of four nodes. The first node is norm node. The second and third nodes are rule-act node. And, the fourth node is a node that represents the following phrases: *'jika'*, *'apabila'*, and *'dalam hal'*. The rule-act node that lies in the third node of the chain is the condition that must meet to apply the norm. The norm node is then connected to the rule-act node by `hasCondition` edge and unbinds the connection to the other rule-act node that lies in the second position of the chain. Next, we remove the fourth node of the chain from the graph.
10) ***Qualifier detection***: We detect the qualifier of the act if there is an act-rule node that has a connection to another node, let v, by any edge label, exclude `hasActType`, `hasSubject`, `hasObject`. We then create a new concept node, connect the act-rule node to the concept node

by the `hasQualifier` edge, and the concept node to v by the `hasQualifierValue` edge. If v has a connection to the text node by `CASE` edge, we connect the concept node to the text node using `hasQualifierType` edge and remove the `CASE` node.

4.7. Graph Construction for Legal Clauses

After transformation, we obtain a graph that in principle can be directly written as as set of RDF triples, provided that we mint all required URIs for the nodes and edges in the graph. For each edge in the graph, except for the ones labeled `owl:sameAs`, we create a URI of the form `lexid-s:{edge label}` pattern. Note that at the end of the transformation, edge labels already conform to the property names in the LexID ontology.

For each node, we create a URI following appropriate patterns according to the IRI scheme in Fig A6. For each norm node and rule-act, we formulate its URI using the following URI patterns of class `Norm`, `RuleAct`. For each refer node, a URI is created following pattern of class `Article`, `Section` or `Item` depending on its text. Each node that representing *'dan'*, *'atau'*, and *'dan/atau'* will use URI patterns of class `AndExpression`, `XorExpression`, and `OrExpression`, respectively. Then, we form a typing triple for each norm node, rule-act node, and reference node, asserting them as an instance of `Norm`, `RuleAct`, and `LegalDocumentContent`, respectively. We also assert each node representing *'dan'*, *'atau'*, and *'dan/atau'* as an instance of `AndExpression`, `XorExpression`, and `OrExpression`.

Next, we assert each concept node as an instance of `Concept` with URI pattern `Concept_{CONCEPT_ID}_{Article or Section or Item}`. The remaining nodes, i.e., any node with text, are also asserted as an instance of `Concept`. The URI pattern of a text node depends on the position of the nodes in the graph. If the text node is not the graph root, then the URI pattern of the node uses the `Concept_{TEXT CONCEPT}_{LegalDocument}` pattern. Otherwise, the URI of the text node has the `Concept_{TEXT CONCEPT}` pattern. It indicates that the legal clause is a definition clause in which the text node is the concept that has the definition (The definition clause is applied only in the legal document containing the clause).

By asserting the corresponding triples, each root node will be connected to the appropriate instance of `LegalDocumentContent` via the property `ruleOf` (with the property `hasRule` connects in the inverse direction). That instance of `LegalDocumentContent` can actually be an instance of `Article`, `Section`, or `Item`, which represents the article, section, or item that contains the legal clause.

Table 2. SPARQL query and answers of question *"Who enacts The Regional Regulation of Jambi Province No. 11 of 2008?"*

```
SELECT distinct (coalesce(?label, ?ans)
                 as ?answer)
WHERE {
  ?LegalDocument a lexid-s:
      LegalDocument ;
     lexid-s:hasEnactionOfficial ?ans ;
     rdfs:label "Peraturan Daerah
         Provinsi Jambi Nomor 11 Tahun
         2008"^^xsd:string.
  OPTIONAL{
    ?ans rdfs:label ?label .
  }
}
```

answer
"Zulkifli Nurdin"

5. Use Cases

To show how to work with our ontology, we provide some question examples for which the KG (and the ontology) can answer via SPARQL.

Q1. Questions related to general information of legal document

Our KG can answer questions related to general information of a legal document, namely the name of the legal document, the enaction date, the person who enacts the legal document, etc. For example, the answer to the question *"Who enacts the Regional Regulation of Jambi Province No. 11 of 2008?"* is Zulkifli Nurdin. Table 2 shows the query and its results, which match the desired answer. Note that the label of `LegalDocument` instances is standardized and we ignore typos that might occur in the SPARQL query.

Q2. Questions on related legal document relationships

Our KG can answer questions related to the relationship of a legal document to other documents, such as legal basis, implementation, repealing, and amendment. For example, the answer to the question *"What are the documents related to the Regional*

Table 3. SPARQL query and the answer of question *"What are the documents related to The Regional Regulation of Maluku Province 2/2008?"*

```
SELECT distinct (coalesce(?label, ?ans)
                 as ?answer)
WHERE {
  ?LegalDocument a lexid-s:
      LegalDocument ;
     lexid-s:hasLegalBasis
     | lexid-s:implements
     | lexid-s:amends
     | lexid-s:repeals ?ans ;
     rdfs:label "Peraturan Daerah
         Provinsi Maluku Nomor 12 Tahun
          2008"^^xsd:string .
  ?ans a lexid-s:LegalDocument .
  OPTIONAL{
    ?ans rdfs:label ?label .
  }
}
LIMIT 5
```

answer
"Undang-Undang Republik Indonesia Nomor 8 Tahun 1981"
"Undang-Undang Republik Indonesia Nomor 17 Tahun 2003"
"Undang-Undang Republik Indonesia Nomor 33 Tahun 2004"
"Peraturan Pemerintah Republik Indonesia Nomor 58 Tahun 2005"
"Undang-Undang Republik Indonesia Nomor 32 Tahun 2004"

Regulation of Maluku Province 2/2008?". The expected answers of this question actually consist of 11 acts, 6 government regulations, and 3 provincial regulations. They are:

- Act 20/1958, 8/1981, 21/1992, 6/1996, 18/1997, 23/1997, 17/2003, 31/2004, 10/2004, 32/2004, and 33/2004,
- Government Regulation 64/1957, 142/2000, 66/2001, 54/2002, 58/2005, and 38/2007,
- Maluku Province Regulation 05/1987, 03/2007, and 15/2004.

Table 3 shows the query and its results (limited to 5 randomly selected answers), all of which are among the expected answers.

Q3. Questions related to the structure of legal document

Our KG can answer questions related to the body structure of legal document, such as chapters or articles enacted in the document. For example, the answer to the question *"What are the chapters contained in the Regional Regulation of Maluku Province No. 12 of 2008?"*. The answer to that question are the eleven chapters contained in the regulation. Table 4 shows the SPARQL query used to answer the question and its result (limited to five randomly selected results), which are indeed among the chapters of the aforementioned regulation. Note that the chapter names are printer in capital letters as originally given by the document.

Table 4. SPARQL query and the answer of question *"What are the regulation chapters had by The Regional Regulation of Maluku Province No. 12 of 2008?"*

```
SELECT distinct (concat(?contentLabel,
    ": ", ?contentName) as ?answer)
WHERE {
  ?LegalDocument a lexid-s:
      LegalDocument ;
      lexid-s:hasContent ?
          topLevelContent ;
      rdfs:label  "Peraturan Daerah
          Provinsi Maluku Nomor 12
          Tahun 2008"^^xsd:string.
  ?topLevelContent lexid-s:hasPart* ?
      content .
  ?content a lexid-s:Chapter ;
      rdfs:label ?contentLabel ;
      lexid-s:name ?contentName .
}
LIMIT 5
```

answer
"BAB XI: P E N Y I D I K A N"
"BAB III: GOLONGAN RETRIBUSI, CARA MENGUKUR TINGKAT PENGGUNAAN JASA"
"BAB II: NAMA, OBJEK, DAN SUBJEK RETRIBUSI"
"BAB VI: PENDAFTARAN, PENETAPAN DAN PEMUNGUTAN RETRIBUSI"
"BAB VII: P E N G A W A S A N"

Q4. Questions related to the text content of legal clauses

We show how our KG can answer questions related to legal clauses, such as the legal clause of an article or a section of a legal document. For example, question textit"What are the values of Article 12 of The Regional Regulation of Maluku Province No. 12 of 2018?". Article 12 of Maluku Province No. 12 of 2018 has three sections, consisting of one legal clause each. Therefore, the question has three expected answers (in Bahasa Indonesia) as follows:

"*(1) Dalam melaksanakan produksi dan perdagangan hasil perikanan setiap pengusaha wajib memenuhi segala persyaratan yang ditetapkan berdasarkan Peraturan Daerah ini.*"
"*(2) Apabila dari laporan petugas pengambil contoh ternyata hasil produk perikanan tidak memenuhi syarat, maka terhadap hasil-hasil tersebut dikenakan penertiban menurut petunjuk teknis yang ditetapkan oleh Kepala Dinas Perikanan dan Kelautan Provinsi Maluku.*"
"*(3) Pengusaha berkewajiban memberikan kesempatan seluas-luasnya kepada pengawas untuk melaksanakan tugasnya berdasarkan Peraturan Daerah ini.*"

Table 5 shows the query and the given results matching the expected answers.

Q5. Questions about the current state of legal documents concerning amendment and repealing

We show how our KG can answer questions related to the current state of a legal document with respect to the amendment or repealing status of the document, as well as content addition, deletion or modification of the legal document. For example, the question *"What are the values of Article 6 of The Regional Regulation of Purworejo Residence 13/2004 after amendment?"*. The Regional Regulation of Purworejo Residence 13/2004 has been amended by the Regional Regulation of Purworejo Residence 18/2008 in which Article 6 has been modified. In the current version, that article consists of two sections. Therefore, the question has two expected answers (in Indonesian) as follows:
"*(1) Kewenangan penerbitan izin dan rekomendasi ada pada Bupati.*"
"*(2) Kewenangan sebagaimana dimaksud pada ayat (1), dapat dilimpahkan kepada Kepala Dinas.*"

Table 6 shows how to query answers the question, matching the expected results. The query is quite complicated involving a subquery. Essentially, the sub-query computes all contents of a legal document, both the original version and after the amendment. The output consists of the year and number of document, the content label, and the legal clause. Then the results are filtered by the desired content, and we take the top one answer sorted by the latest version of the document.

Q6. Semantic content questions

Finally, we give an example of how our KG answers questions focusing on the semantic content

Table 5. SPARQL query and the answer of question *"What are the values of Article 12 of The Regional Regulation of Maluku Province No. 12 of 2018?"*

```
SELECT distinct
        (concat (coalesce(?sectionName,
             ""),
                 " ", ?ans) as ?answer)
WHERE {
  ?LegalDocument a lexid-s:
      LegalDocument ;
      lexid-s:hasContent ?
          topLevelContent ;
      rdfs:label  "Peraturan Provinsi
          Maluku Nomor 12 Tahun 2008"^^
          xsd:string.
  ?topLevelContent lexid-s:hasPart* ?
      article .
  ?article a lexid-s:Article ;
      rdfs:label ?label ;
      rdfs:label "Pasal 12"^^xsd:string
          .
  {
    {
      ?article lexid-s:hasPart ?section
          .
      ?section a lexid-s:Section ;
          lexid-s:name ?sectionName;
          dct:description ?ans .
    }
    UNION
    {
      ?article dct:description ?ans .
    }
  }
}
```

answer
"(1) Dalam melaksanakan produksi dan perdagangan hasil perikanan setiap pengusaha wajib memenuhi segala persyaratan yang ditetapkan berdasarkan Peraturan Daerah ini."
"(2) Apabila dari laporan petugas pengambil contoh ternyata hasil produk perikanan tidak memenuhi syarat, maka terhadap hasil-hasil tersebut dikenakan penertiban menurut petunjuk teknis yang ditetapkan oleh Kepala Dinas Perikanan dan Kelautan Provinsi Maluku."
"(3) Pengusaha berkewajiban memberikan kesempatan seluas-luasnya kepada pengawas untuk melaksanakan tugasnya berdasarkan Peraturan Daerah ini."

of legal clauses. For example, consider the question *"Apa saja kegiatan yang dapat menggunakan Bantuan Pendanaan PTN Badan Hukum?"* (What type of activities can use funding assistance given to state universities of incorporated legal entity?), which is an example of legal semantic question. We can find the answer to the question in Article 5 of

Table 6. SPARQL query and the answer of question *"What are the values of Article 6 of The Regional Regulation of Purworejo Residence 13/2004 after amendment?"*

```
SELECT distinct (lcase (group_concat(distinct ?value; separator = "\\n")) as ?
    answer)
WHERE
{
  {
    SELECT distinct
         ?year ?number ?article
         (lcase(concat(?larticle," ", coalesce(?lsection, ""))) as ?lcontent)
         ?value
    {
      ?document a lexid-s:LegalDocument ;
                rdfs:label "Peraturan Daerah Kabupaten Nomor 18 Tahun 2008" .
      ?document (lexid-s:hasContent|lexid-s:hasPart)* ?parent .
      {
        {
          ?document lexid-s:amendedBy ?amendment .
          ?amendment lexid-s:hasContent ?articleI ;
                     lexid-s:regulationYear ?year ;
                     lexid-s:regulationNumber ?number .
          ?articleI lexid-s:modifies ?modification .
          ?modification lexid-s:hasModificationTarget ?parent ;
                        lexid-s:hasModificationContent ?content .
          ?content lexid-s:hasPart* ?article.
        }
        UNION
        {
          ?document lexid-s:regulationYear ?year ;
                    lexid-s:regulationNumber ?number .
          ?parent lexid-s:hasPart* ?article .
        }
      }
      ?article a lexid-s:Article ;
              rdfs:label ?larticle .
      {
        {
          ?article lexid-s:hasPart ?section .
          ?section a lexid-s:Section ;
                  rdfs:label ?lsection ;
                  dct:description ?value .
        }
        UNION
        {
          ?article dct:description ?value .
        }
      }
    }
  }
  FILTER(regex(?lcontent, "pasal 6"))
}
GROUP BY ?year ?number ?article
ORDER BY desc(?year) ?desc(?number)
LIMIT 1
```

answer
"kewenangan penerbitan izin dan rekomendasi ada pada bupati. kewenangan sebagaimana dimaksud pada ayat (1), dapat dilimpahkan kepada kepala dinas."

Table 7. SPARQL query and the answer of question *“Apa saja kegiatan yang dapat menggunakan Bantuan Pendanaan PTN Badan Hukum?”*

```
SELECT distinct
      ( concat(?lqActType,
             " ",
             replace( group_concat(distinct
                         lcase(concat(replace(?conjQObject, "And", "; dan "),
                                       replace(?lqObject, "\\W+$", "")
                                       )
                                )
                         separator=""),
                     "^; dan ",
                     ""),
               "."
             ) as ?answer )
{
  ?source lexid-s:hasRule ?norm .
  ?norm lexid-s:hasAct ?act .
  ?act  lexid-s:hasSubject ?subject ;
        lexid-s:hasActType ?actType ;
        lexid-s:hasQualifier ?qualifier .
  ?subject rdfs:label "Bantuan Pendanaan PTN Badan Hukum" .
  ?actType rdfs:label "digunakan" .
  ?qualifier lexid-s:hasQualifierType ?qtype ;
             lexid-s:hasQualifierValue ?qvalue .
  ?qtype rdfs:label "untuk" .
  ?qvalue lexid-s:hasActType ?qActType ;
          lexid-s:hasObject ?qObject .
  ?qActType rdfs:label ?lqActType .
  ?qObject lexid-s:hasElement ?qObjects ;
             rdfs:label ?conjQObject .
  ?qObjects dct:description ?lqObject .
}
GROUP BY ?lqActType
```

answer
”mendanai biaya tenaga kependidikan; dan biaya dosen; dan biaya pengembangan; dan biaya investasi; dan biaya operasional.”

the Government Regulation No. 26/2015, which is: *“mendanai:*

a. biaya operasional;
b. biaya dosen;
c. biaya tenaga kependidikan;
d. biaya investasi; dan
e. biaya pengembangan.”

Table 7 shows the query answering the question together with the results. Note that the structure of LexID ontology means that the answer to the question actually corresponds to a non-trivial subgraph of the KG. This is a consequence of our decision to have a fine-grained logical representation down to the word/phrase level, which is motivated by a possible future use of the KG as a basis for automated legal reasoning. This query searches for a qualifier of an act of a norm. The subject and the type of the act are *“Bantuan Pendanaan PTN Badan Hukum”* and *‘digunakan’*, while the qualifier type is *’untuk’*. The value of the query is an act, which has an action type and objects bound by a coordinating conjunction (’and’). The output modifier inside the SELECT command consists of a series of rather complicated operations to collect all the information in that subgraph into a single string.

6. Evaluation

We evaluate LexID on the ability to answer the questions given. There are 1,152 question tests which consist of Q1-Q6 question types, shown in the Table. These questions have been verified by someone who can read a legal document but is not a legal expert. In this evaluation, we use the macro average of the F1 score as an evaluation metric, one of the most common metrics used in the QA system [23]. Our KG got a value of 0.91 for the macro average of the F1 score, in for this evaluation. The result shows that the ability to answer the given questions of our KG is good enough.

Table 8. The quality of LexID based on the ability to answer the use case question. The arterisk mark (*) shows that the question type need to be verified by the legal experts.

Question type	**Number of questions**	**F1 score**
Q1	200	0.96
Q2	202	0.92
Q3	200	0.94
Q4	200	0.89
Q5*	200	0.90
Q6*	150	0.84
Total	**1152**	**0.91**

7. Conclusion

In this paper, we have presented LexID KG and ontology, which is the first knowledge graph and ontology representing Indonesian legal documents that contains both metadata representation at the document level as well as a fine-grained semantic representation of the legal information down to the word/phrase level. We have also shown how the KG was constructed from legal documents using an essentially rule-based approach. The described use cases indicate that the KG and ontology can be employed in a variety of knowledge retrieval scenarios. Based on the evaluation result, our ontology can answer the given questions with a score of 0.91 on the macro average of the F1 score.

Future work includes extending the KG and ontology to cover other legal products produced by the government, evaluate the accuracy for each steps, and better involve the legal expert. Our approach for the KG construction in principle relies on POS tagging of legal clauses. As a future work, expressivity of the KG can then be enriched if one makes use of linguistic feature extraction such as NER, e.g., to enable categorization of legal clauses based on topical taxonomy. Meanwhile, yet another future direction of this work is to employ the KG and ontology as the basis of automated legal reasoning, since both already contains fine-grained representation of legal clauses. Finally, from an application perspective, a future work includes the development of Indonesian legal semantic search powered by the LexID KG and ontology.

References

[1] Presiden Republik Indonesia, *Undang-Undang Negara Republik Indonesia Nomor 12 Tahun 2011 Tentang Pembentukan Undang-Undang*. Lembaran Negara Republik Indonesia, 2011. [Online]. Available: https://peraturan.go.id/peraturan/view.html?id=11e44c4e8cfb29e09e4b313231343538

[2] A. Nurhilmiyah, "Critical review of the existence of legal fiction principles in indonesian law," *International Conference of ASEAN Perspective and Policy*, vol. 1, no. 1, pp. 36–41, 2018. [Online]. Available: https://jurnal.pancabudi.ac.id/index.php/ICAP/article/view/265

[3] M. Farida, A. F. Muhlizi, Chairijah, I. Syamsul, S. AR, I. Padmanegara, H. Hastuty, L. Wulandari, and T. R. Silaban, *Laporan Kompendium Bidang Hukum Perundang-Undangan*. Pusat Penelitian dan Pengembangan Sistem Hukum Nasional, Departemen Hukum dan HAM RI, 2008. [Online]. Available: https://www.bphn.go.id/data/documents/kompendium_perundang2an.pdf

[4] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, and A. Zimmermann, *Knowledge Graphs*, ser. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool, 2021, no. 22. [Online]. Available: https://kgbook.org/

[5] H. Chen, G. Cao, J. Chen, and J. Ding, "A practical framework for evaluating the quality of knowledge graph," in *Knowledge Graph and Semantic Computing: Knowledge Computing and Language Understanding - 4th China Conference, CCKS 2019, Hangzhou, China, August 24-27, 2019, Revised Selected Papers*, ser. Communications in Computer and Information Science, X. Zhu, B. Qin, X. Zhu, M. Liu, and L. Qian, Eds., vol. 1134. Springer, 2019, pp. 111–122. [Online]. Available: https://doi.org/10.1007/978-981-15-1956-7_10

[6] L. T. McCarty, "A language for legal discourse I: basic features," in *Proceedings of the Second International Conference on Artificial Intelligence and Law, ICAIL '89, Vancouver, BC, Canada, June 13-16, 1989*, J. C. Smith and R. T. Franson, Eds. ACM, 1989, pp. 180–189. [Online]. Available: https://doi.org/10.1145/74014.74037

[7] R. van Kralingen, "A conceptual frame-based ontology for the law," in *Proceedings of the First International Workshop on Legal Ontologies, (LEGONT '97)*, 1997.

[8] J. Breuker and R. Hoekstra, "Epistemology and ontology in core ontologies: FO-Law and LRI-Core, two core ontologies for law," in *Proceedings of the EKAW*04 Workshop on Core Ontologies in Ontology Engineering : Northamptonshire (UK), October 8, 2004*, ser. CEUR Workshop Proceedings, A. Gangemi and S. Borgo, Eds., vol. 118. CEUR-WS.org, 2004. [Online]. Available: http://ceur-ws.org/Vol-118/paper2.pdf

[9] R. Hoekstra, J. Breuker, M. D. Bello, and A. Boer, "The LKIF core ontology of basic legal concepts," in *Proceedings of the 2nd Workshop on Legal Ontologies and Artificial Intelligence Techniques June 4th, 2007, Stanford University, Stanford, CA, USA*, ser. CEUR Workshop Proceedings, P. Casanovas, M. A. Biasiotti, E. Francesconi, and M. Sagri, Eds., vol. 321. CEUR-WS.org, 2007, pp. 43–63. [Online]. Available: http://ceur-ws.org/Vol-321/paper3.pdf

[10] E. Filtz, S. Kirrane, and A. Polleres, "The linked legal data landscape: Linking legal data across different countries," *Artificial Intelligence and Law*, vol. 29, no. 4, pp. 485–539, 2021. [Online]. Available: https://doi.org/10.1007/s10506-021-09282-8

[11] T. Francart, J. Dann, R. Pappalardo, C. Malagon, and M. Pellegrino, "The european legislation identifier," in *Knowledge of the Law in the Big Data Age, Conference 'Law via the Internet 2018', Florence, Italy, 11-12 October 2018*, ser. Frontiers in Artificial Intelligence and Applications, G. Peruginelli and S. Faro, Eds., vol. 317. IOS Press, 2018, pp. 137–148. [Online]. Available: https://doi.org/10.3233/FAIA190016

[12] M. van Opijnen, "European case law identifier: Indispensable asset for legal information retrieval," *From Information to Knowledge*, vol. 236, no. 2-3, pp. 91–103, 2011. [Online]. Available: https://ssrn.com/abstract=2046160

[13] A. Oksanen, M. Tamper, J. Tuominen, E. Mäkelä, A. Hietanen, and E. Hyvönen, "Semantic finlex: Transforming, publishing, and using finnish legislation and case law as linked open data on the web," in *Knowledge of the Law in the Big Data Age, Conference 'Law via the Internet 2018', Florence, Italy, 11-12 October 2018*, ser. Frontiers in Artificial Intelligence and Applications, G. Peruginelli and S. Faro, Eds., vol. 317. IOS Press, 2018, pp. 212–228. [Online]. Available: https://doi.org/10.3233/FAIA190023

[14] M. Abdurahman, F. Darari, H. Lesmana, M. Hartopo, I. Rhesa, and B. C. L. Tobing, "Lex2KG: automatic conversion of legal documents to knowledge graph," in *2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9631310

[15] R. Cyganiak, D. Wood, and M. Lanthaler, Eds., *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation, 25 February 2014. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/

[16] G. Schreiber and Y. Raimond, Eds., *RDF 1.1 Primer*. W3C Working Group Note, 25 February 2014. [Online]. Available: https://www.w3.org/TR/rdf11-primer/

[17] T. Berners-Lee, "Linked data," 2006. [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html

[18] A. Krisnadhi, N. Karima, P. Hitzler, R. Amini, V. Rodríguez-Doncel, and K. Janowicz, "Ontology design patterns for linked data publishing," in *Ontology Engineering with Ontology Design Patterns - Foundations and Applications*, ser. Studies on the Semantic Web, P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, Eds. IOS Press, 2016, vol. 25, pp. 201–232. [Online]. Available: https://doi.org/10.3233/978-1-61499-676-7-201

[19] S. Harris and A. Seaborne, Eds., *SPARQL 1.1 Query Language*. W3C Recommendation, 21 March 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/

[20] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web*, vol. 8, no. 3, pp. 489–508, 2017. [Online]. Available: https://doi.org/10.3233/SW-160218

[21] J. Breuker, P. Casanovas, M. C. A. Klein, and E. Francesconi, "The flood, the channels and the dykes: Managing legal information in a globalized and digital world," in *Law, Ontologies and the Semantic Web - Channelling the Legal Information Flood*, ser. Frontiers in Artificial Intelligence and Applications, J. Breuker, P. Casanovas, M. C. A. Klein, and E. Francesconi, Eds., vol. 188. IOS Press, 2009, pp. 3–18. [Online]. Available: https://doi.org/10.3233/978-1-58603-942-4-3

[22] A. Gangemi, M. Sagri, and D. Tis-

cornia, “A constructive framework for legal ontologies,” in *Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications*, ser. Lecture Notes in Computer Science, V. R. Benjamins, P. Casanovas, J. Breuker, and A. Gangemi, Eds. Springer, 2003, vol. 3369, pp. 97–124. [Online]. Available: https://doi.org/10.1007/978-3-540-32253-5_7

[23] D. Jurafsky and J. H. Martin, *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*, ser. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. [Online]. Available: https://www.worldcat.org/oclc/315913020

Appendix A.
List of properties in the LexID ontology

In this appendix, we present figures and tables of a property listing from Table A1-Table A5, instantiation of the LawAmendment class from Fig. A.1-Fig. A.3, IRI scheme on Table A6, Parsing Rule from Table A7-TableA8, and sample output for several stages of the process from Table A9, Fig. A.5, and Fig. A.6.

Table A1. List of properties in LexID ontology, all in the `lexid-s` namespace, unless specified otherwise.

Property	**Subject type**	**Object type**	**Description**
`considers`	`LegalDocument`	`xsd:string`	A consideration matter when creating the legal document.
`dct:description`	`Article` or `Section` or `Item`	The text content of the article, section, or item.	
`name`	`LegalDocument` or `Chapter` or `Part` or `Paragraph`	`xsd:string`	The title of the legal document or name of the legal text group, e.g., chapter name, part name, etc.
`owl:sameAs`	`Person` or `Office` or `City`	Wikidata instances	A Wikidata item/instance aligned to this person, office, or city.
`hasCreator`	`LegalDocument`	`Office`	The office that creates the legal document, e.g., Minister of Finance, etc.
`hasDictum`	`LegalDocument`	`xsd:string`	A statement of the outlines of legal document.
`hasEnactionDate`	`LegalDocument`	`xsd:dateTime`	The enactment date of the legal document.
`hasEnactionLocation`	`LegalDocument`	`City`	The city where the legal document is enacted.
`hasEnactionOffice`	`LegalDocument`	`Office`	The office of the person who enacted the legal document.
`hasEnactionOfficial`	`LegalDocument`	`Person`	The person who enacted the legal document.
`hasPromulgationDate`	`LegalDocument`	`xsd:dateTime`	The promulgation date of the legal document.
`hasPromulgationLocation`	`LegalDocument`	`City`	The city where the legal document is promulgated.
`hasPromulgationOffice`	`LegalDocument`	`Office`	The office of the person who promulgated the legal document.
`hasPromulgationOfficial`	`LegalDocument`	`Person`	The person who promulgated the legal document.
`hasPromulgationPlace`	`LegalDocument`	`PlaceOfPromulgation`	Show where the legal document is placed in after promulgation.
`hasRegulationNumber`	`LegalDocument`	`xsd:string`	The legal document's number, e.g., the act number, the regulation number, etc.
`hasRegulationYear`	`LegalDocument`	`xsd:int`	The year when the legal document was created.
`rdfs:label`	`owl:Thing`	`xsd:string`	The (string) label of the entity.

Table A2. List of relationships between two legal document in LexID ontology (defined in `lexid-s` namespace).

Property	**Subject type**	**Object type**	**Description**
`amendedBy`	`LegalDocument`	`LegalDocument`	The subject legal document is amended by the object legal document.
`amends`	`LegalDocument`	`LegalDocument`	`amends` is the inverse of `amendedBy`.
`implementedBy`	`LegalDocument` or `Article` or `Section`	`LegalDocument`	the subject legal document is implemented by the object legal document.
`implements`	`LegalDocument`	`LegalDocument` or `Article` or `Section`	`implements` is the inverse of `implementedBy`.
`hasLegalBasis`	`LegalDocument`	`LegalDocument`	The subject legal document has the object legal document as its legal basis.
`legalBasisOf`	`LegalDocument`	`LegalDocument`	`legalBasisOf` is the inverse of `hasLegalBasis`.
`repealedBy`	`LegalDocument`	`LegalDocument` or `Article` or `Section`	The subject legal document is repealed by the object legal document.
`repeals`	`LegalDocument` or `Article`	`LegalDocument`	`repeals` is the inverse of `repealedBy`.

Table A3. List of properties of the content of Legal Document in LexID ontology (defined in `lexid-s` namespace).

Property	**Subject type**	**Object type**	**Description**
`hasContent`	`LegalDocument`	`Chapter` or `Article`	A legal document can has Chapters or Articles as its top level content.
`hasPart`	`Chapter`	`Part` or `Paragraph` or `Article`	A Chapter can has Parts, Paragraphs, or Articles as its parts.
	`Part`	`Paragraph` or `Article`	A Part can s Paragraphs or Articles as its parts.
	`Paragraph`	`Article`	A Paragraph can has articles as its parts.
	`Article`	`Section`	An Article can has sections as its parts.
`isContentOf`	`Chapter` or `Article`	`LegalDocument`	`isContentOf` is inverse of `hasContent`.
`isPartOf`	`Section`	`Article`	`isPartOf` is the inverse of `hasPart`, i.e., it goes in the opposite direction of `hasPart`.
	`Article`	`Paragraph` or `Part` or `Chapter`	
	`Paragraph`	`Part` or `Chapter`	
	`Part`	`Chapter`	

Table A4. List of amendment properties in LexID ontology (defined in `lexid-s` namespace).

Property	**Subject type**	**Object type**	**Description**
`adds`	`Article`	`LawAddition`	The article in the amendment document that adds a content to the target legal document.
`hasAdditionContent`	`Addition`	`LegalDocumentContent`	The content of the addition.
`hasAddtionTarget`	`Addition`	`LegalDocumentContent`	The position (i.e., content segment in the target legal document) at which the added content must be added.
`deletes`	`Article`	`LegalDocumentContent`	The article of amendment document that deletes a content in the amended legal document.
`modifies`	`Article`	`LawModification`	The article of the amendment legal document that modifies a content of the target legal document.
`hasModificationContent`	`Modification`	`LegalDocumentContent`	The modification content.
`hasModificationTarget`	`Modification`	`LegalDocumentContent`	The original content of the amended legal document that is to be modified.

Table A5. List of semantic properties in LexID ontology (defined in `lexid-s` namespace).

Property	**Subject type**	**Object type**	**Description**
`hasAct`	`Norm` or `Concept`	`RuleAct`	The act ruled by Norm.
`hasActType`	`RuleAct`	`Concept`	The type of an act.
`hasElement`	`CompoundExpression`	`RuleAct` or `Concept` or `LegalDocument` or `LegalDocumentContent`	The elements which have coordinating conjunction expression.
`hasCondition`	`Norm` or `Concept`	`RuleAct`	The condition that must be met by norm or concept.
`hasModality`	`Norm`	`Concept`	The modality of the norm.
`hasObject`	`RuleAct`	`Concept`	The object of an act.
`hasQualifier`	`RuleAct`	`Concept`	The qualifier entity describing a qualifier of an act.
`hasQualifierType`	`Concept`	`Concept`	The type of a qualifier.
`hasQualifierValue`	`Concept`	`Concept` or `RuleAct`	The value of the qualifier.
`hasRule`	`Article` or `Section`	`Norm` or `Concept`	The rule of the article of section content.
`hasSubject`	`Norm` or `RuleAct` or `Concept`	`Concept` or `CompoundExpression`	The subject of the instance.
`isRuleOf`	`Norm` or `Concept`	`Article` or `Section`	`isRuleOf` is the inverse of `hasRule` .
`owl:sameAs`	`Concept` or `CompoundExpression`	`Concept` or `CompoundExpression`	The subject has the oubject as an alias.
`refersTo`	`Concept` or `CompoundExpression`	`LegalDocument` or `LegalDocumentContent` or `CompoundExpression`	An instance of `LegalDocument`, `LegalDocumentContent`, or `CompoundExpression` referred to by the given concept or compound expression.

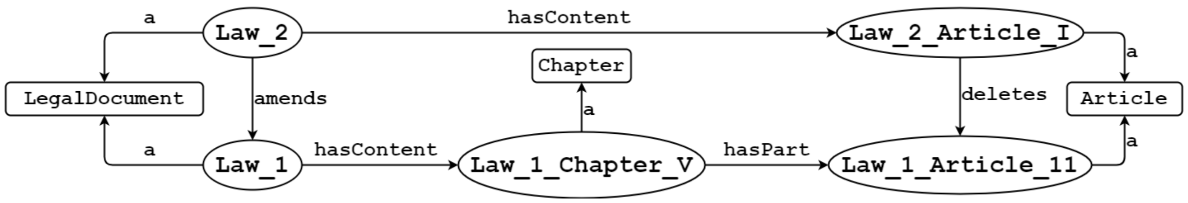


Figure A.1. Instantiation of `Deletion`

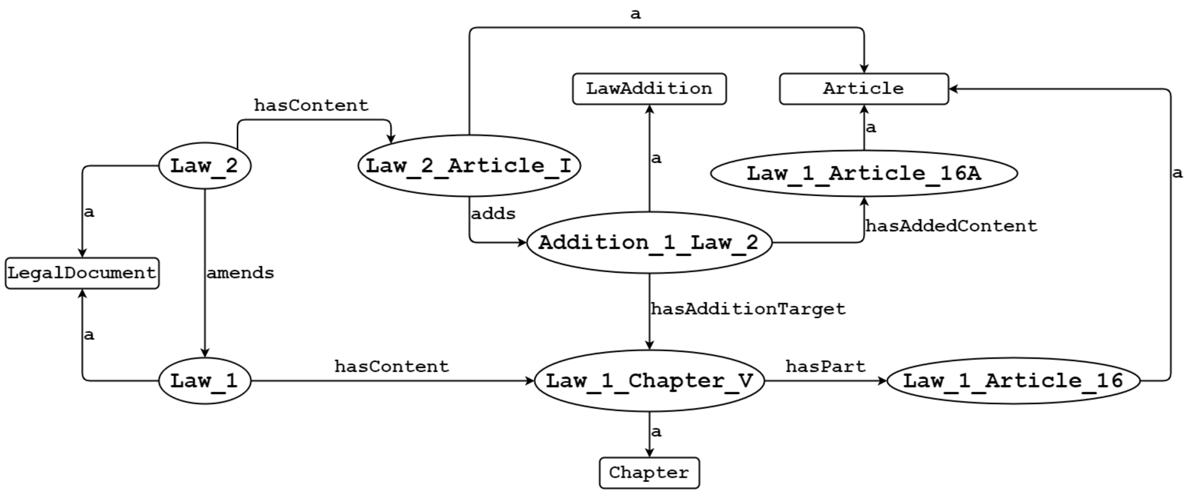


Figure A.2. Instantiation of `Addition`

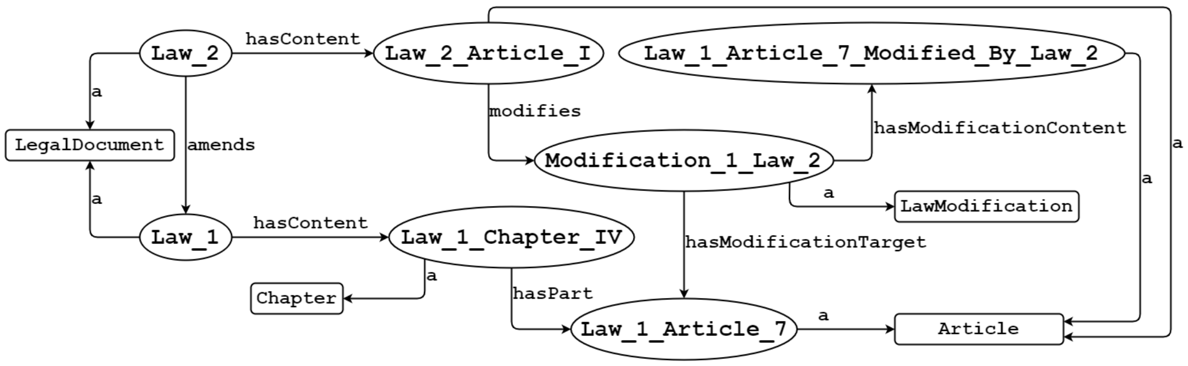


Figure A.3. Instantiation of `Modification`

Table A6. Instance IRI naming scheme (defined in the `lexid` namespace). Instances of class `Item` and `Concept` have multiple naming schemes used in different situations.

Class	**IRI pattern for the instances of the class with example**
`LegalDocument`	`{TYPE}_{YEAR}_{NUMBER}`, e.g., `lexid:Permen_Agama_2019_17`
`Chapter`	`{LegalDocument}_Chapter_{CHAPTER ID}`, e.g., `lexid:Permen_Agama_2019_17_Chapter_III`
`Part`	`{LegalDocument}_Part_{CHAPTER ID}_{PART ID}`, e.g., `lexid:Permen_Agama_2019_17_Part_III_Kesatu`
`Paragraph`	`{LegalDocument}_Paragraph_{CHAPTER ID}_{PART ID}_{PARAGRAPH ID}`, e.g., `lexid:Permen_Agama_2019_17_Paragraph_III_Kesatu_1`
`Article`	`{LegalDocument}_Article_{ARTICLE ID}`, e.g., `lexid:Permen_Agama_2019_17_Article_13`
`Section`	`{LegalDocument}_Section_{ARTICLE ID}_{SECTION ID}`, e.g., `lexid:Permen_Agama_2019_17_Section_13_1`
`Item`	`{Article}_Number_{ITEM ID}`, e.g., `lexid:Permen_Agama_2019_17_Article_1_Number_1` `{Section}_Number_{ITEM ID}`, e.g., `lexid:Permen_Agama_2019_17_Section_3_1_Number_2` `{Article}_Letter_{ITEM ID}`, e.g., `lexid:Permen_Agama_2019_17_Article_4_Letter_A` `{Section}_Letter_{ITEM ID}`, e.g., `lexid:Permen_Agama_2019_17_Section_13_2_Letter_B`
`LegalDocumentContent` for modification	`{LegalDocumentContent}_ModifiedBy_{LegalDocument}`, e.g., `lexid:PP_2018_7_Article_1_ModifiedBy_PP_2020_35`
`LawAddition`	`Addition_{ADDITION ID}_By_{LegalDocument}`, e.g., `lexid:Addition_1_By_Permen_Agama_2020_1`
`LawModification`	`Modification_{MODIFICATION ID}_By_{LegalDocument}` e.g., `lexid:Modification_1_By_Permen_Agama_2020_1`
`Norm`	`Norm_{NORM ID}_{Article or Section}`, e.g., `lexid:Norm_Permen_Agama_2019_17_Section_13_2`
`RuleAct`	`Act_{ACT ID}_{Article or Section or Item}`, e.g., `lexid:Act_1_Permen_Agama_2019_17_Section_13_1_Letter_A`
`Concept`	`Concept_{TEXT CONCEPT}`, e.g., `lexid:Concept_Menteri` `Concept_{TEXT CONCEPT}_{LegalDocument}`, e.g., `lexid:Concept_Menteri_Permen_Agama_2019_17` `Concept_{CONCEPT ID}_{Article or Section or Item}`, e.g., `lexid:Concept_1_Permen_Agama_2019_17_Article_14`
`AndExpression`	`And_{AND ID}_{Article or Section or Item}`, e.g., `lexid:And_1_Permen_Agama_2019_17_Article_1_Number_3`
`OrExpression`	`Or_{OR ID}_{Article or Section or Item}`, e.g., `lexid:Or_1_Permen_Agama_2019_17_Section_30_1_Number_2`
`XorExpression`	`Xor_{XOR ID}_{Article or Section or Item}`, e.g., `lexid:Xor_1_Permen_Agama_2019_17_Article_15_Letter_C`
`Person`	`{TEXT OF THE PERSON NAME}`, e.g., `lexid:Joko_Widodo`
`Position`	`{TEXT OF THE OFFICE}`, e.g., `lexid:Menteri_Agama_Republik_Indonesia`
`City`	`{TEXT OF THE CITY}`, e.g., `lexid:Depok`

Table A7. Parsing rules for surface legal information extraction. Rules are typically based on location (title, preamble, etc., of the document), case format, and/or the appearance of certain words/phrases (given in quotes). Values of each field can be a string, or a list of them, or a (nested) dictionary.

Field	**Indicator/Value description**
Surface information taken from the title part of the document.	
document_type	Uppercase; starts with "PERATURAN", "UNDANG-UNDANG", "KEPUTUSAN" or "INSTRUKSI"; appears before "NOMOR".
document_number	Between "NOMOR" and "TAHUN".
document_year	Uppercase; between "TAHUN" and "TENTANG" .
document_name	Uppercase; between "TENTANG" and "DENGAN RAHMAT TUHAN YANG MAHA ESA".
amended_document	If the value of document_name starts with "PERUBAHAN ATAS" or "PERUBAHAN KEDUA ATAS", etc., the value of amended_document is the value of document_name after the word "ATAS".
document_creator	Uppercase; after "DENGAN RAHMAT TUHAN YANG MAHA ESA"; before "Menimbang".
Surface information taken from the preamble part of the document.	
consideration_matters	A list of strings with elements taken from the text after "Menimbang: " and before "Mengingat: " or "Memperhatikan", split by a semicolon ";".
implemented_document	A list of implemented documents with elements taken from the consideration matter that contains phrase "bahwa untuk melaksanakan {*implemented document*}'. In most cases, {*implemented document*} may refer to the name of another legal document or a single article or section of another legal document. However, we allow for the possibility of referencing to multiple implemented documents.
legal_basis	A list of strings with elements taken from the text after "Mengingat: " or "Memperhatikan: " and before the text "Memutuskan: ", split by a semicolon ";".
dictum	After "Memutuskan"; before "BAB" or "Pasal".
Surface information taken from the body part of the document.	
document_structure	A nested dictionary of document structures (chapters, articles, etc.) organized hierarchically. See the explanation in Section 4.2.
values	A dictionary with keys taken from article/section/item identifiers (used also as keys in the nested dictionary of document_structure) and values from the corresponding legal clause text appearing after the article/section/item.
repealed_document	A dictionary with keys taken from the legal document names occurring in the text "Pada saat {*document type*} ini berlaku {*repealed documents*} dicabut dan dinyatakan tidak berlaku". Here, {*repealed documents*} describes a comma- or semicolon-separated list of names of repealed legal documents.
Surface information taken from the closing part of the document.	
promulgation_place	In the text began by "Agar setiap orang mengetahuinya, ", the value of promulgation_place is the title-formatted text between the text "Lembaran" or "Berita" and dot punctuation ".".
enaction_location	After the phrase "Ditetapkan di"; before the phrase "pada tanggal".
enaction_date	After "pada tanggal" that appears after the value of enaction_location; appears in the date format: "{dd} {Month} {yyyy}"; the extracted value is reformatted to ISO format ("{yyyy}-{mm}-{dd}").
enaction_office	In uppercase; between the value of enaction_date and the comma ","; in its own line.
enaction_official	In uppercase; after the text "ttd." that appears after the value of enaction_office; appears in its own line.
promulgation_location	After the phrase "Diundangkan di"; before the phrase "pada tanggal".
promulgation_date	After the phrase "pada tanggal" that appears after the value of promulgation_location; appears in the date format "{dd} {Month} {yyyy}"; the extracted value is reformatted to ISO format ("{yyyy}-{mm}-{dd}").
promulgation_office	In upppercase; after the value of promulgation_date; before the next comma ",";.
promulgation_official	In uppercase; after the text "ttd." that appears after the value of promulgation_office; appears in its own line.

Table A8. Parsing rules for body segment.

Segment	Indicator
chapter	In new line, starts with "BAB" followed by chapter number in roman characters ('I', 'V', 'X', 'L', 'C', 'D', 'M')
part	In new line, starts with "Bagian" followed by part number in text format ('kesatu', 'kedua', 'ketiga', etc.).
paragraph	In new line, starts with "Paragraph" followed by paragraph number ('1', '2', '3', etc.).
article	In new line, starts with "Pasal" following by article number ('1', '2', '3', etc.).
section	In new line, starts with number in the round parentheses ('(1)', '(2)', '(3)', etc.).
item	In new line, Identified by a number or lowercase letter following by dot punctuation ('1.', '2.', 'a.', 'b.', etc.).

Table A9. Example result of surface information extraction

Field	Example
document_type	PERATURAN MENTERI AGAMA
document_number	17
document_year	2019
document_name	SEKOLAH TINGGI AGAMA KATOLIK NEGERI
amended_document	PERATURAN MENTERI AGAMA NOMOR 30 TAHUN 2018
document_creator	MENTERI AGAMA REPUBLIK INDONESIA
considerans	bahwa untuk mewujudkan penyelenggaraan pendidikan tinggi dan pengelolaan perguruan tinggi yang baik pada sekolah tinggi agama katolik negeri pontianak, perlu dibentuk statuta, …
implemented_document	Peraturan Menteri Kesehatan Nomor 30 Tahun 2019, …
legal_basis	Undang-Undang Nomor 20 Tahun 2000, …
dictum	Menetapkan : PERATURAN MENTERI AGAMA TENTANG STATUTA SEKOLAH AGAMA KATOLIK NEGERI PONTIANAK
document_structure	see Figure A.4
values	…, {id : SECTION_13_1, value : Sekolah Tinggi menjunjung tinggi kebebasan akademik, kebebeasan mimbar akademik, dan otonomi keilmuan}, …
repealed_document	…, Article_60:[Peraturan Menteri Kesehatan Nomor 30 Tahun 2019, …], …
promulgation_place	Berita Negara
enaction_location	Jakarta
enaction_date	2019-09-12
enaction_position	MENTERI AGAMA REPUBLIK INDONESIA
enaction_official	LUKMAN HAKIM SAIFUDDIN
promulgation_location	Jakarta
promulgation_date	2019-09-12
promulgation_official	WIDODO EKATJAHJANA
promulgation_office	DIREKTUR JENDERAL PERATURAN PERUNDANG-UNDANGAN KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA REPUBLIK INDONESIA

```
...
Chapter_III: {
  name: ‘PENYELENGGARAAN TRIDHARMA PERGURUAN TINGGI’
  parts: {
    Part_III_Kesatu: {
    name: ‘Pendidikan’
    paragraphs: {
      Paragraph_III_Kesatu_1: {
        name: ‘Kebebasan Akademik, Kebebasan Mimbar Akademik, dan Otonomi
            Keilmuan’
        articles: {
          Article_13: {
            name:‘Pasal 13’
            sections: {
              Section_13_1: {
                name:‘(1)’
                }, ...
              }
            }, ...
          }
        }, ...
      }
    }, ...
  }
}, ...
```

Figure A.4. Example of nested dictionary of legal document structure. This particular example corresponds to “Bab III” in Figure 1. The corresponding entity name in the dictionary (“Chapter III”) is supposed to include the legal document name according to Table A6, but omitted here to shorten the description.

```
lexid:Permen_Agama_2019_17 a lexid-s:MinisterialRegulation .
lexid:Permen_Agama_2019_17 rdfs:label "Peraturan Menteri Agama Republik Indonesia
     Nomor 17 Tahun 2019" .
lexid:Permen_Agama_2019_17 lexid-s:name "STATUTA SEKOLAH TINGGI AGAMA KATOLIK
    NEGERI PONTIANAK" .
lexid:Permen_Agama_2019_17 lexid-s:hasRegulationNumber "17" .
lexid:Permen_Agama_2019_17 lexid-s:hasRegulationYear "2019"^^xsd:int .
lexid:Permen_Agama_2019_17 lexid-s:considers "bahwa untuk mewujudkan
    penyelenggaraan pendidikan tinggi dan pengelolaan perguruan tinggi yang baik
    pada sekolah tinggi agama katolik negeri pontianak perlu dibentuk statuta"^^
    xsd:string .
lexid:Permen_Agama_2019_17 lexid-s:considers "bahwa berdasarkan penimbangan
    sebagaimana dimaksud dalam huruf a, perlu menetapkan peraturan menteri agama
    tentang statuta sekolah tinggi agama katolik negeri pontianak"^^xsd:string .
lexid:Permen_Agama_2019_17 lexid-s:hasDictum "Menetapkan: PERATURAN MENTERI AGAMA
     TENTANG STATUTA SEKOLAH TINGGI AGAMA KATOLIK NEGERI PONTIANAK."^^xsd:string .
lexid:Permen_Agama_2019_17 lexid-s:hasCreator lexid:Menteri_Agama_Republik_
    Indonesia .
lexid:Permen_Agama_2019_17 lexid-s:hasEnactionOfficial lexid:Lukman_Hakim_
    Saifuddin .
lexid:Permen_Agama_2019_17 lexid-s:hasEnactionOffice lexid:Menteri_Agama_Republik
    _Indonesia .
lexid:Permen_Agama_2019_17 lexid-s:hasEnactionDate lexid:"2019-09-12"^^xsd:date .
lexid:Permen_Agama_2019_17 lexid-s:hasEnactionLocation lexid:Jakarta .
lexid:Permen_Agama_2019_17 lexid-s:hasPromulgationOfficial lexid:Widodo_
    Ekatjahjana .
lexid:Permen_Agama_2019_17 lexid-s:hasPromulgationOffice lexid:Direktur_Jenderal_
    Peraturan_Perundang_Undangan_Kementerian_Hukum_Dan_Hak_Asasi_Manusia .
lexid:Permen_Agama_2019_17 lexid-s:hasPromulgationDate lexid:"2019-09-12"^^xsd:
    date .
lexid:Permen_Agama_2019_17 lexid-s:hasPromulgationLocation lexid:Jakarta .
lexid:Permen_Agama_2019_17 lexid-s:hasPromulgationPlace lexid:Berita_Negara .
lexid:Lukman_Hakim_Saifuddin a lexid-s:Person .
lexid:Widodo_Ekatjahjana a lexid-s:Person .
lexid:Menteri_Agama_Republik_Indonesia a lexid-s:Office .
lexid:Menteri_Hukum_Dan_Hak_Asasi_Manusia a lexid-s:Office .
lexid:Jakarta a lexid-s:City .
lexid:Berita_Negara a lexid-s:PlaceOfPromulgation
lexid:Permen_Agama_2019_17_Article_13_1 a lexid-s:Article .
lexid:Permen_Agama_2019_17_Article_13_1 rdfs:label "Ayat 1" .
lexid:Permen_Agama_2019_17_Article_13_1 dct:description "Sekolah Tinggi
    menjunjung tinggi kebebasan akademik, kebebasan mimbar akademik, dan otonomi
    keilmuan." .
lexid:Jakarta owl:sameAs wd:Q281134 .
```

Figure A.5. Snapshot of metadata description of Regulation of the Minister of Religion 17/2019, in the Turtle syntax. The property a is equivalent to `rdf:type`

```
lexid:Permen_Agama_2019_17 lexid-s:hasContent lexid:Permen_Agama_2019_17_Chapter_
    III .
lexid:Permen_Agama_2019_17_Chapter_III lexid-s:isContentOf lexid:Permen_Agama
    _2019_17 .
lexid:Permen_Agama_2019_17_Chapter_III lexid-s:hasPart lexid:Permen_Agama
    _2019_17_Part_III_Kesatu .
lexid:Permen_Agama_2019_17_Part_III_Kesatu . lexid-s:hasPart lexid:Permen_Agama
    _2019_17_Paragraph_III_Kesatu_1 .
lexid:Permen_Agama_2019_17_Paragraph_III_Kesatu_1 lexid-s:hasPart lexid:Permen_
    Agama_2019_17_Article_13 .
lexid:Permen_Agama_2019_17_Article_13 lexid-s:hasPart lexid:Permen_Agama_2019_17_
    Section_13_1 .
lexid:Permen_Agama_2019_17_Part_III_Kesatu lexid-s:isPartOf lexid:Permen_Agama
    _2019_17_Chapter_III.
lexid:Permen_Agama_2019_17_Paragraph_III_Kesatu_1 lexid-s:isPartOf lexid:Permen_
    Agama_2019_17_Paragraph_III_Kesatu_1 .
lexid:Permen_Agama_2019_17_Article_13 lexid-s:isPartOf lexid:Permen_Agama
    _2019_17_Paragraph_III_Kesatu_1 .
lexid:Permen_Agama_2019_17_Section_13_1 lexid-s:isPartOf lexid:Permen_Agama
    _2019_17_Article_13 .
lexid:Permen_Agama_2019_17_Chapter_III lexid-s:name "PENYELENGGARAAN TRIDHARMA
    PERGURUAN TINGGI"^^xsd:string .
lexid:Permen_Agama_2019_17_Part_III_Kesatu lexid-s:name "Pendidikan"^^xsd:string
    .
lexid:Permen_Agama_2019_17_Paragraph_III_Kesatu_1 lexid-s:name "Kebebasan
    Akademik, Kebebasan Mimbar Akademik, dan Otonomi Keilmuan"^^xsd:string .
```

Figure A.6. Example triples representing legal content structure.