Khanam *et al. Hum. Cent. Comput. Inf. Sci.* (2019) 9:23
https://doi.org/10.1186/s13673-019-0184-7

Human-centric Computing and Information Sciences

RESEARCH Open Access

Comprehensive structured knowledge base system construction with natural language presentation

Shirin Akther Khanam*, Fei Liu and Yi-Ping Phoebe Chen

*Correspondence: 18739964@students.latrobe.edu.au
Department of Computer Science and Information Technology, La Trobe, Melbourne 3086, Australia

Abstract

Constructing an ontology-based machine-readable knowledge base system from different sources with minimum human intervention, also known as ontology-based machine-readable knowledge base construction (OMRKBC), has been a long-term outstanding problem. One of the issues is how to build a large-scale OMRKBC process with appropriate structural information. To address this issue, we propose Natural Language Independent Knowledge Representation (NLIKR), a method which regards each word as a concept which should be defined by its relations with other concepts. Using NLIKR, we propose a framework for the OMRKBC process to automatically develop a comprehensive ontology-based machine-readable knowledge base system (OMRKBS) using well-built structural information. Firstly, as part of this framework, we propose formulas to discover concepts and their relations in the OMRKBS. Secondly, the challenges in obtaining rich structured information are resolved through the development of algorithms and rules. Finally, rich structured information is built in the OMRKBS. OMRKBC allows the efficient search of words and supports word queries with a specific attribute. We conduct experiments and analyze the results of relational information extraction, with the results showing that OMRKBS had an accuracy of 84% which was higher than the other knowledge base systems, namely ConceptNet, DBpedia and WordNet.

Keywords: Machine readable, Concept, Knowledge base system, Ontology construction

Introduction

Machine readable knowledge bases are used to store datasets so that these datasets can be accessible through systems. Machine-readable knowledge base construction (MRKBC) involves the automated extraction and integration of data from different sources and generating meaningful information with interoperable knowledge [1]. There is a large body of research on the automatic extraction of information for MRKBC. Initially, the research focused on syntactic information extraction [2, 3], but more recently, the extraction of lexical semantic information has received more interest from the research community [4, 5]. Knowledge base systems (KBS) which use traditional databases are not effective due to the limited operational and analytical workload and

latency for retrieval [6]. On the other hand, ontologies which provide descriptions of terms important to a specific domain [5] are often used as a resource and have become an alternative to KBS in applications where elements are defined using the relations between concepts [7]. The mechanism of building an ontology-based machine-readable knowledge base system, also known as ontology-based machine-readable knowledge base construction (OMRKBC) is gaining more attention from the research community. While developing this process, most research studies include defining the ontological elements in a machine-readable way [6, 8], providing descriptions of concepts using the relations between concepts [4] and a more enriched meaning [9, 10]. The attributes or descriptions are from sources that are publicly available but are difficult to obtain and structure into a single KBS [8]. There are several publicly available knowledge bases that are extremely reliable and commonly used such as DBpedia [1], ConceptNet [11], FrameNet [12] and WordNet [13]. Reusing these reliable knowledge bases is one way to facilitate the assignment of meaning to the terms of a domain [14]. However, the construction of ontologies is time-consuming and requires a thorough knowledge of the domain [15]. Furthermore, building an appropriate structure that represents information about terms is not a trivial task [16]. Research in the knowledge base literature has proposed several approaches to improve the structure of information on terms using ontologies. Researchers have focused on how to rapidly improve the quality of data through the structure in such a system [17–19]. Additionally, some approaches are restricted to a single domain, hence they are not applicable to other domains.

The main objective of OMRKBC is to obtain knowledge about each term from different sources through appropriate structured information and by representing the information to be queried in a meaningful and logical way. When terms and definitions are mapped to an ontology, they are often richly structured with different relations, attributes and simple relationships between concepts. Well-structured information or definitions support the efficient access of data from our OMRKBS which returns meaningful results. Before such a system can be used, an ontology needs to be created based on the existing data. For this purpose, first, we manually build a base ontology from two sources: BioPortal [20] and CRISP [21]. Then, we automatically build OMRKBS based on this base ontology from three reliable KBS: DBpedia [1], ConceptNet [11] and WordNet [13]. This research focuses on automating OMRKBC to obtain high-quality data and increase its effectiveness. We present a method to obtain a base ontology with important concepts. Once the important concepts are established in a base ontology, they can be used to define more complex concepts automatically from sources.

More broadly, this paper proposes NLIKR, a scheme for an ontology-based KBS. This scheme represents each English word as a concept in KBS. A word or concept is defined by its properties (i.e. its relationships with other concepts). The characteristics of a concept are indicated by its relationship with other concepts. As a result, a concept definition can go beyond human language since every word is a concept and is defined by another concept. For example, '*water has no colour*' is one feature of water, the association between '*water*' and other concepts (i.e. '*no*', '*colour*') in the feature represents the properties of '*water*' such as <*water, no, colour*>. In the associations, '*colour*', '*no*' are all concepts. A concept inherits the properties of its super concepts. For instance, '*water*' is a sub-concept of liquid. Therefore, water shares the characteristics

of liquid, such as '*having no fixed shape*'. Our research develops a program for an ontology-based KBS where the definitions or features of concepts are structured so they can be entered into the ontology using the NLIKR scheme.

We observe that the process of OMRKBC is iterative: enriching the knowledge base by importing concepts, instances and relations and defining concepts from various sources. This motivated us to develop a program to automatically import data from different sources. In one part of the program, we import instance datasets which are available in CSV format from DBpedia. However, several problematic issues were identified while importing CSV instances into an ontology, such as its time-consuming nature and it consumes a large amount of space. We propose several algorithms and techniques to resolve these issues. The program performs the following operation to import instances by resolving the issues. First, a pre-processing algorithm is executed to process the large data file of instances and then a mapping algorithm is executed to automatically create the mapping expression to embed the instances in OMRKBS. Finally, a program loads the instances with a mapping expression and embeds the instances in the system using OWL API. In the other part, we import a definition for each concept in OMRKBS. First, this program pre-processes a definition to turn the long text into features using the OpenIE [22] and some rules. Then, the program discovers each word in the text as a concept in the system and creates a mapping expression to embed the features. Finally, the features are implanted using a mapping expression in OMRKBS.

Our primary improvements to the program are defining each concept with a description, features and instances through appropriately structured information. These features and instances of concepts are richly structured due to the advantages obtained by using NLIKR. This advantage implies that the features of a concept are structured in such way that each word in the structure of a feature is a concept and all concepts in the structure are linked as stated in the feature. We identified individual or unique features from the definition. Then, we embedded each feature in the system by its interrelationships with other concepts, relations and attributes as these features would be inherited to subclasses of the concept and the concept itself. These individual features with relations and/or attributes that are embedded in the system are called rich structured information (RSI). Consequently, each feature is machine interpretable since machines can discover each concept and find the interrelationship of concepts through the structure of features. Next, we concentrated on the retrieval and presentation of information on the concept being queried using simple SPARQL [23] queries. In doing so, the results of the query are effective since the results are RSI (rich structured information) which makes the information specific, meaningful and sensible.

The rest of the paper is organized as follows. "Related work" section briefly discusses the previous research and background. The next section presents the framework of the OMRKBC system. "Building the OMRKBC system" section details the procedure of the OMRKBC system. "System output" section discusses how to retrieve and represent data from OMRKBC system. Different types of KBSs sourced by our scheme are also briefly discussed in the following section. Our experiment evaluation is presented in "Experiments and a comparison of the results" section. Finally, we conclude the paper and suggests directions for further research.

Related work

Recently, several approaches that reuse existing knowledge bases to automate ontology construction from unstructured text have been proposed [24–26]. The drawbacks of these approaches include labour costs to construct the dictionary, its domain-specific nature and the limited number of patterns. Several approaches to ontology-based knowledge bases have been proposed to reformulate knowledge representation in ontologies [27–30]. However, semantic searches in knowledge bases still face difficulties, such as the lack of a detailed methodology that guides the ontology learning process from text. Portage [31] supports plugins to import datasets from various sources to construct an ontology, however they are costly to assemble and continuous human effort is needed to keep them up to date.

Automatically constructing a KBS from sources is an important and challenging task. A large body of research exists on automatically obtaining large and quality (but textual) information from Wikipedia. The DBpedia [1] extracts structured information from Wikipedia covering many specific domains and general world knowledge [12]. But the extracted knowledge is mostly limited to named entities or concepts with proper names, such as cities, persons, species, movies, organizations etc. The linguistic relation between such concepts that are more relevant for ontology mappings is absent in DBpedia. YAGO [32] is identical to DBpedia in that each article in Wikipedia becomes an entity in YAGO. YAGO mainly extracts a smaller number of relations between concepts. Nevertheless, YAGO does not interrelate concepts if WordNet does not contain the concepts. BabelNet [10] is similar projects that collect crowd-sourced knowledge from similar sources. In these KBS, a large, structured, multilingual taxonomy is created from a combination of Wikipedia's structured knowledge and WordNet [13]. However, a large amount of information is still being hidden in the text of the Wikipedia articles which is not covered in DBpedia, YAGO or BabelNet. The automatic extraction of semantic concept relations from raw text in KBC, even for concepts that are not yet listed in an existing repository such as WordNet, is a still challenging issue.

Numerous research efforts aim at extracting knowledge from text corpora but research on the exact purpose of commonsense knowledge (commonsense knowledge presents facts or individual features about the concept, such as 'Lemons are sour') which is machine-readable, is comparatively rare [33]. Automatically inferring missing facts from existing ones has thus become an increasingly important task. Cyc [34] is an AI platform with human reasoning, knowledge and logic on an enterprise scale. To reason about text using Cyc, mapping the text into its proprietary logical representation is required using its own language Cyc. However, this mapping process is quite complex because the inherent ambiguity in natural language must be resolved to produce the unambiguous logical formulation required by Cyc. Wordnet [13] is an original and prominent linguistic resource. Words can point to one or several synsets and synsets can be referenced by one or several words in WordNet. However, WordNet focuses on the formal taxonomies of words. In contrast, ConceptNet [11] which has been created from reliable sources, is a freely available large-scale commonsense knowledge base that focuses on a richer set of semantic relations between compound concepts and supports many practical textual-reasoning tasks over real-world documents. ConceptNet can best be seen as a semantic resource whose scope of contents is general world knowledge in the same vein as Cyc.

These KBs store common-sense facts in a machine-processable way and more recent work puts a focus on human interaction such as building question answering systems [35, 36]. However, facts can exhibit their properties in multiple aspects and fact expression has lost some properties or attributes through these KBSs. Moreover, not all the words in fact expression are interrelated in these KBSs, rather they present as a whole statement in KBSs. Therefore, these KBSs are not fully machine interpretable.

Currently, knowledge base construction solutions have focused on obtaining rich structure information from text [17, 18, 37–40]. These KBCs already support a broad range of downstream applications such as information retrieval, question answering and medical diagnosis. However, the essence of the information (individual features) remains latent in knowledge representation, where relations and attributes are expressed via combinations of textual and structural information. Moreover, the entities extracted by these systems have not been integrated into a single homogenous ontology.

In this paper, we design an OMRKBC process that defines concepts automatically with definitions and instances from reliable sources to build a comprehensive OMRKBS. Our approach acknowledges the facility of three reliable KBSs: Dapedia, ConcpetNet and WordNet and integrates various types of knowledge such as features and instances from these resources into OMRKBS through rich structured information that helps to define the object from various perspectives. Concepts are linked with attributes and relations in the rich structured information. The features of the concepts are built through rich structured information so that the system can return logical, meaningful and informative results to the user's query. We construct an ontology as a whole KBS, not as a domain, which facilitates the process of defining words and represents the query data in an informative way.

The framework of OMRKBC

We propose a framework to build the OMRKBC process efficiently. First, we create the base ontology manually from existing ontologies such as CRISP [21]. Then, we extract information about concepts from DBpedia, WordNet and ConcpetNet and design a program to build the OMRKBC system with this information. A description of how the OMRKBC system is built is given in "Building the OMRKBC system" section and we represent the search results using efficient queries from our KBS in "System output" section. Figure 1 shows an overview of the framework of OMRKBC. The main purpose of OMRKBC is to define the concepts of the base ontology automatically from various types of structured information such as descriptions, instances and relations. In DBpedia and ConceptNet, such information is available in CSV format. We extract information from these large sources and turn this information into a rich knowledge base. For this, we propose a program to build the OMRKBC process in three phases: extracting resources, addressing the challenges and embedding information in OMRKBS. Each phase is defined as follows:

Extracting resources The abstract (DBpedia provides a short abstract for each article and we used this abstract as definition in OMRKBC) and instances corresponding to concepts are extracted from DBpedia. Also, we extract relations and their corresponding data associated with concepts from ConceptNet. Some descriptions of concepts are extracted from CRISP.

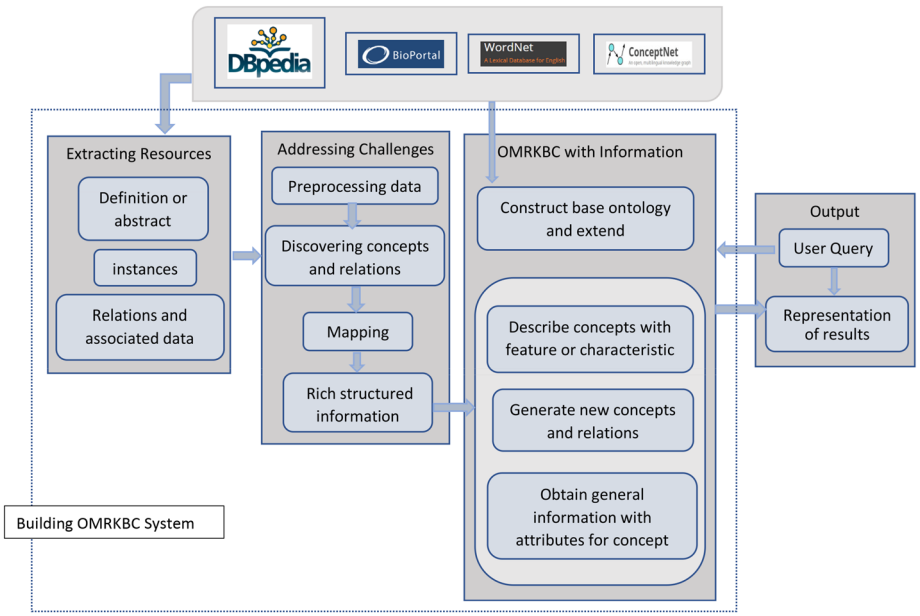

Fig. 1 The proposed framework of OMRKBC

Addressing the challenges In this process, the abstract or a description of a concept is turned into a set of individual features, and instances are converted into general information with attributes. We call this rich structure information. We focus on three challenges in relation to processing the information into RSI (rich structure information). Firstly, data must be pre-processed before being converting into RSI. Then, each word is discovered or allocated as a concept and possible groups words/phrases are discovered as relations in the OMRKBC system. Thirdly, data are mapped to convert into RSI. Finally, the well-structured information is ready to be entered into the ontology.

OMRKBC with information We design a program to build the RSI in OMRKBC. Individual features or characteristics of concepts and general information on concepts associated with attributes are embedded in RSI. Therefore, rich structured individual features and general information with attributes are built in OMRKBC. After importing the short abstract or the description, the ontology is enriched with new concepts, relations or attributes.

Example 1 Take as an example the president '*Donald Trump*' which is described as follows: "Donald John Trump (born June 14, 1946) is the 45th and current president of the United States. Before entering politics, he was a businessman and television personality". This text as a description will be built in OMRKBC through the following structure.

Structured information input: *<Donald Trump, businessman> <Donald Trump, television personality> <Donald Tramp, enter politics> <Donald Tramp, current president of US> <Donald Tramp, 45th president of US>*

Example 2 Donald Trump's birthplace and spouse name are USA and Ivana Zelnickova respectively. The birthplace and spouse name are referred to as attributes. This information is imported through the following structure.

Structured information input: <*Donald Trump, birthplace, USA*> <*Donald Trump, spouse name, Ivana Zelnickova* >

Building the OMRKBC system

We describe how OMRKBC is built. Firstly, we construct the base ontology manually. Then, we introduce methods to discover the concepts and relations. Next, we develop a standard procedure to define the concepts through RSI.

Constructing the base label ontology

Concepts are classified and stored in a hierarchical structure in an ontology. Three major domains: *'existence'*, *'science'*, and *'part of speech (POS)'* are the top of the structure in OMRKBC. These top three domains will be the basic class labels in the ontology, which means all the concepts will be assigned under these three classes. These basic class labels are built in the ontology manually. We illustrate these three class labels as follows.

'*existence*' is one root class of the hierarchical structure which is divided into '*physicalExistence*', '*abstractExistence*', '*entity*'. '*physicalExistence*' can be '*lifeExistence*' and '*nonLifeExistence*'. '*entity*' is something that exists apart from the other things, having neither an abstract or physical existence, having its own independent existence (e.g. '*weather*'). The '*attributes*' and '*relation*' class are added in the 'abstractExistence' class. Important phrases (e.g. '*perform in*', '*capable of*') are added in the relation class and important attributes (e.g. colour, size) are added in the attributes class. The '*generalAttributes*' class, which is a subclass of attributes, contains general properties of the class and the corresponding instances. Another top class is '*science*' which has eight domains. These domains are related to eight major science disciplines: '*behaviour*' or '*social science*', '*biology*', '*chemical*', '*physical*', '*food*', '*medicine*', '*diseases*', '*technology*' that are mostly imported from CRISP [21]. These domains contain concept which are related to their topics. '*parts of speech*' is one more top class, and some general words are added here such as verbs, prepositions, adjectives, adverbs and articles.

An ontology with these basic classes is called the base ontology. Some important domains are extracted from BioPortal [21], EVS [41] and DBpedia [1] repository. For example, various types of important attributes (e.g. '*shape*', '*depth*', '*speed*') from the '*attributes*' domain of a thesaurus ontology in EVS and the '*organization*', '*place*', '*creative work*', '*entity*' and '*action*' domains from the ontologies (e.g. schema, entity) in BioPortal and DBpedia are extracted and then these domains are placed under the base ontology. Figure 2 shows segments of the '*existence*' and '*science*' domains. We extend the base ontology to enrich the domain so that various types of concepts can be assigned under the base ontology. Figure 3 shows the segments of relation and attributes that are discovered by OMRKBC system.

Example 3 Take as an example the word ''*water*' which is described as follows: "H_2O; tasteless, colourless, odourless compound present in all tissues, and the most universal

Fig. 2 Segment of science and existence domains. Concepts marked with the symbol ⊖ are subclasses of the concepts marked with the symbol ⊕. The concepts denoted by the colour orange are discovered by OMRKBS

of solvents. The density of water is 1". We found some individual features of water from this sentence: '<*water, H20*> <*water, no, taste*> <*water, no, colour*> <*water, no, odour*> < *water, compound, present in, organic, tissue*> <*water, solvent*> <*water, density, 1*>. Figure 4 shows how "*water*' can be defined by the proposed ontology.

Discovering concepts

In OMRKBC, each word is assigned as a concept and each concept can be defined by its relationships with other concepts. This section explains how each word is discovered or allocated as a concept in an ontology. First, the existence of the word is checked in the OMRKBC system. A concept is discovered when the word exists as a concept in the ontology. Otherwise, the word should be assigned into the ontology. Assigning a word as a concept is as follows:

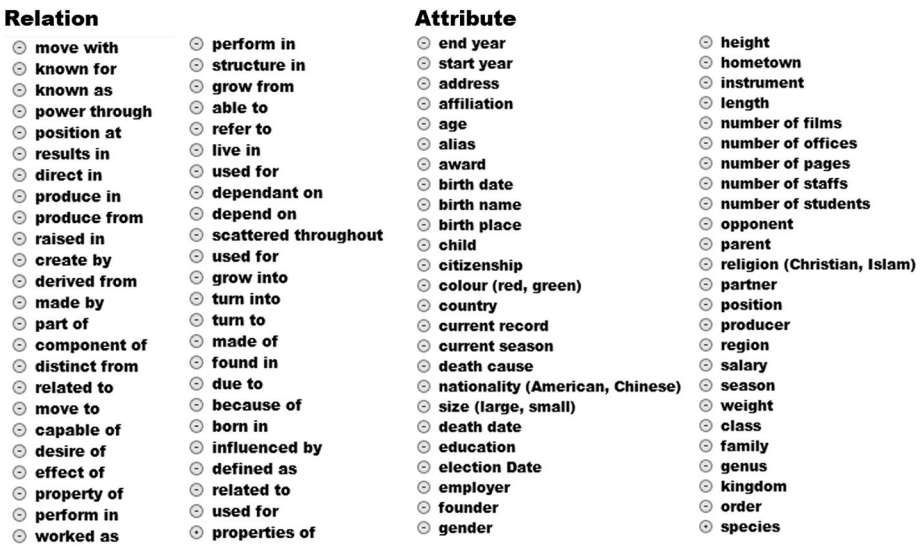

Fig. 3 Segment of relation and attribute domains

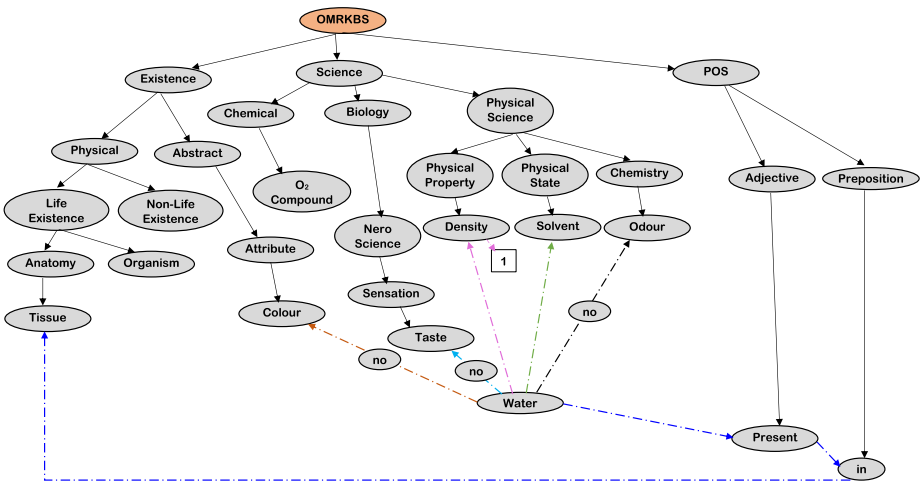

Fig. 4 An example of a class '*water*' defined by the proposed ontology. The words in the grey circles are concepts and the root is denoted by the orange circle. The different coloured dashed arrows indicate the relationship between concepts according to the features of the concept '*water*' i.e. <*water, no, taste*> <*water, no, colour*> while the solid arrows indicate the subclasses

First, the word's synonyms are found using WordNet. When the synonyms are found in the ontology, the word is added as same class as the synonym class. For example, the synonym of '*undertaken*' using WordNet is '*take*' which exists in our ontology under the class of '*action*' as a verb. '*undertaken*' is added to the '*action*' class. If a synonym cannot be found in our ontology, the ontology should be checked to find the hypernym of the word. If a hypernym is found, the word is added under the hypernym class. For instance, when '*action*' is a hypernym of '*perform*' in WordNet and '*action*' exists in OMRKBC, '*perform*' is added under the '*action*' class. However, when a word does not exist in the ontology or it cannot be related to a concept using WordNet, then this word needs to be

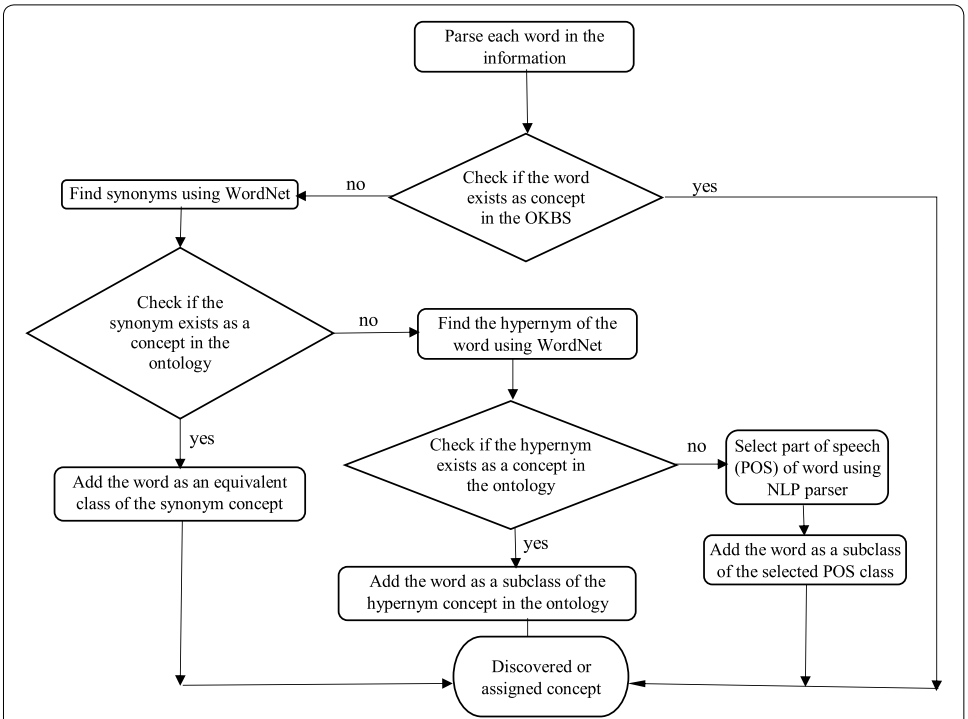

Fig. 5 Flowchart to discover a concept in OMRKBC

tagged as a part of speech and this word is added as an axiom under the part of speech class. For instance, '*fresh*' cannot be found in our ontology. So, '*fresh*' is tagged as an adjective and this word is added under the '*adjective*' class in our ontology. Verbs which are considered as actions are assigned to the '*action*' class. We used the Stanford NLP parser [42] to tag the POS. Figure 5 shows the flowchart to discover a concept. We use this method to discover concepts from various sources later.

Discovering relations

There are particular groups of words which are used often in sentences. These phrase words are discovered as concepts in an ontology. We call these groups of words '*relation*' since they can be used to link words in sentences. The relation is governed by a few rules when parsing the information to discover the concept.

Rule 1 When a word which is a verb (V) or abstract noun (ABSN) or common noun (CN) or an adjective (ADJ) is followed by a preposition (P) in the information, the two consecutive words are counted as a relation. Example: power through (V, P), leader of (CN, P), good for (ADJ, P), respect for (ABSN, P)

Rule 2 When a word is a verb but is acting as an adjective (VADJ) and is followed by a noun in the information, the two consecutive words are counted as a relation. Example: washing machine (VADJ, N).

Rule 3 When a word is an adjective and is followed by a verb in the information, the two consecutive words are counted as relation. Example: dry cleaning (ADJ, V).

Finally, these two consecutive words are joined together as one word (e.g. '*powerThrough, leaderOf, goodFor, respectFor*') and are assigned as a concept in the subclass

	A	B	C	D	E	F	G	H	I	J	K
1	**President**	**birthDate**	**birthPlace**	**spouse**	**age**	**country**	**children**	**politicalParty**	**serviceStartYear**	**serviceEndYear**	**Occupation**
2	Barack Obama	August 4, 1961	Honolulu, USA	Michelle Robi	57	USA	2	Democratic	January 20, 2009	January 20, 2017	
3	Donald Trump	June 14, 1946	New York City	Ivana Zelnickc	72	USA	5	Republican	January 20, 2017		Politician | businessman
4	Justin Trudeau	December 25, 19	Ottawa, Canada	Sophie Grégoi	47	Canada		Liberal	November 4, 2015		Teacher | politician
5	Moon Jae-in	24 January 1953	Geoje, South Korea	Kim Jung-sook	66	South Kore	2	Democratic	May 10, 2017		

Fig. 6 Example of instances of '*president*' domain in DBpedia CSV format

under the class 'relation'. Instantly, each word in the relation is discovered or assigned as a concept using the method outlined in Discovering concepts. The reason for calling the relation word a concept is because the concept can be defined with other concepts. Next, when a concept is discovered as a subclass of a class in the '*attributes*' domain, the attribute is added to the concept. For example, the '*American*' concept is subclass of the '*nationality*' concept in '*attributes*'. When '*American*' is discovered in the structure, the '*American*' concept is added to the '*nationality*' class in the structure using ':' (e.g. *nationality: American*).

OMRKBC with instances from DBpedia

Importing instances from the spreadsheet data of DBpedia in OMRKBS (IISDBS) is one part of the proposed OMRKBC process. We provide the background on the functional procedure of IISDBS. We also discuss the challenges of the functional procedure which motivate us to design a program with algorithms and techniques for IISDBS.

Extracting instances

The largest DBpedia KBS which is extracted from the English edition of Wikipedia consists of over 400 domains. Each domain has various properties known as attributes. The core DBpedia data in tabular form are available in CSV format in http://web.informatik.uni-mannheim.de/DBpediaAsTables/. Each CSV file contains instances of one concept and corresponding instances of properties or attributes.

Example 4 As shown in Fig. 6, the first column in the CSV file of the '*president*' domain contains the name of presidents as instances. In this file, '*president*' has more than 90 properties i.e. '*birthdate*', '*birthplace*', '*spouse*'... etc. and these property fields contain instances corresponding to each president's name.

Functional procedure

A functional procedure from the Protégé project available at https://github.com/protegeproject/cellfie-plugin imports spreadsheet data into the ontology in three steps using Protégé [31]. Firstly, the contents of the Excel file are loaded using the Cellfile plugin. Cellfile is a plugin which supports the creation of OWL ontologies from spreadsheets through a flexible mapping expression which maps spreadsheet content to OWL ontologies. Next, a simple mapping rule or expression for the class declaration axiom is created. Finally, axioms are imported into the ontology. We adopt the functional procedure to import instances from DBpedia data into our ontology.

Addressing the challenges of IISDBS

There are several challenges when implementing this procedure. We focus on three challenges in building the IISDBS. The first challenge is pre-processing the extracted CSV data so that these data can be imported into OMRKBS efficiently. The next challenge is to discover or allocate each word in the data as a concept in the ontology. The last challenge is mapping the data to embed instances in OMRKBS. We develop algorithms to address the three challenges which are discussed in the following:

(a) Pre-processing the data Protégé [31] or OWL API [43] only support Excel files with .xlsx extensions when importing spreadsheet data into an ontology. A CSV file contains many invalid characters which are not supported when being imported into an ontology, which results in many NULL values which consume a lot of space in an ontology. Also, large files take a long time to process and sometimes the process is terminated, which is another challenge. A lot of work has to be done manually before CSV data can be imported into an ontology.

Algorithm 1: Pre-process CSV data

```
input  : String FileName: The location of CSV file and file name
output: Excel file generation with formatted data
initialization
eachRow: contain the row of csv file ;
begin
    while eachRow in FileName do
        A[]← contain cell content of each row;
        for i ← 0 to A.length do
            B← A[i];
            B← removeAll_invalid_char(B)
            B← replaceAll_null(B)
            A[i] ← B;
        end
    end
end
```

Therefore, CSV data should be pre-processed before being imported into our ontology. For this reason, we propose an algorithm for pre-processing the data from a CSV file. This algorithm converts all CSV files into Excel files and the file size is reduced by almost 68.4% for each file. All invalid characters are removed from the CSV files and all null values are replaced with empty values. After this, the file size reduced by 93%. Excel files are split after each 3000 rows, resulting in thousands of Excel files which are only 716 KB in size for each file. The overviews of the algorithm for pre-processing the spreadsheet data is shown in Algorithm 1. Now, the system can load and process each file and embed the instances in OMRKBS.

(b) Discovering concepts The names of all attributes or properties are concepts in our ontology. The reason for adding attributes as concepts instead of object properties is because an attribute as a concept can be related to other concepts in the proposed ontology. Discovering each property's name as a concept is one of the important

points in the IISDBS procedure. This stage confirms that the names of all the properties in the CSV file are discovered or allocated as concepts. The names of the properties in a domain which are presented in the first row of the Excel file are imported as concepts into the ontology. The properties of the concept i.e. '*birthdate*', '*birthplace*' are imported as concepts under the '*generalAttributes*' class. When importing the data into the ontology, it is important to check whether this concept already exists or not. Suppose the property name i.e. '*spouse*' already exists in the ontology, this property name will not be imported into the ontology twice. Now each word in the property names is discovered or assigned using the method described in "Discovering concepts" section. For example, '*birthdate*' contains two words birth and date. These words are assigned or discovered.

(c) Mapping Mapping spreadsheet content to OWL ontologies is a great challenge in the process of IISDBS. There are more than 400 domains in DBpedia. Writing mapping expressions for each domain with properties is time consuming. Furthermore, some domains contain more than 700 properties. Manually writing mapping rules for a large number of properties in a domain is a tedious task. We notice that the mapping expressions [44] are the same for all domains except the properties and column names. We consider that these property names and columns are variables. Example 4 shows a part of the CSV file of the president domain where the president names exist in the first column '*A*' i.e. '*Donald Trump*' and the other columns i.e. '*B*', '*C*' and '*D*' contain data on the corresponding properties i.e. '*birthdate*', '*birthplace*', '*spouse*'.

The mapping expression can be written automatically for each domain with only the property name P_1, P_2, P_3 ('*birthdate*', '*birthplace*', '*spouse*') ...corresponding to the column name (*'A', 'B', 'C'*...) needing to be changed. Otherwise, all terms in the expression are the same. Since all column names corresponding to the property names are listed in the file, we can write a fact expression programmatically. Therefore, a pseudocode is developed to create a mapping expression to map the spreadsheet data to the ontology. Firstly, the property names of a domain which are presented in the first row are extracted from the Excel file. After this, an array list is used to store the property names P_1, P_2, P_3 . . . (i.e. '*birthdate*', '*birthplace*', '*spouse*'). Also, a function '*getNameFromNumber*' is devised to generate the column names (i.e. *'A', 'B', 'C'*) which correspond to the instances of a property or domain and return the column name in Col_{name} in Algorithm 2. For instance, '*birthdate*' is a property of the president domain and instances of '*birthdate*' are listed in column '*B* (see Fig. 5). The function will return the column name (e.g. '*B*') of the property name ('*birthdate*'). If properties exist as object properties in an ontology, we can write the mapping rule using the fact expression using a variable of the property name and the corresponding column name as follows. Here, instances of Concept are imported from column '*A*' and corresponding properties P_1, P_2 . . . information is imported from *'B','C'*...

$Individual{:}@A^*$
$Types{:}C_{oncept}$
$Facts{:}P_1@B^*, P_2@C^*, P_3@D^*, \ldots.$

However, properties exist as concepts not as object properties in our ontology. So, before using a fact expression in the mapping rule, we create individuals or instances for properties or concepts using the mapping rule as follows.

$$\begin{aligned} mapping\_rule_1 = {} & Individual{:}@B^*(N_m = P_1\#) \\ & Types{:}P_1 \\ & Individual{:}@C^*(N_m = P_2\#) \\ & Types{:}P_2 \end{aligned}$$

Algorithm 2: Create Mapping Expression

input : String FileName: The location of CSV file and file name
output: Excel file generation with formatted data

Function `getNameFromNumber(k)`:
  i ⟵ k % 26; letter ⟵ CharacterValue(65 + i); //return character value of i
  j ⟵ integerVal(k / 26);
  if $(j > 0)$ **then return** ConcatString(getNameFromNumber(j- 1), letter) ;
  else return letter ;
Function `CreateMappingExpression(`$FileName$, $DomainName$`)`:
  initialization
  $P[]$:contain properties name in File;
  num: contain number of column in file ;
  Col_{name}: contain the letter of column name;
  $mapping\_rule\ []$: contain mapping expression as string; M_p :"mm.prepend";
  begin
    $mapping\_rule_2$ ← "Individual: @A*
    Types: $C_{concept}$ Facts: ";
    P[] ← readFirstRowFile(FileName) //contain property name
    num ← count(P);
    for $i \leftarrow 0$ **to** num **do**
      Col_{name}= getNameFromNumber(i); // return the letter of column name
      if $(i = 1)$ **then** $mapping\_rule_1$ ← ConcatString (Individual: @, Col_{name},*(, N_m,=, P [1], #) Types:, P[1],)
      $mapping\_rule_2$ ← ConcatString (hasPropertyValue @, Col_{name}, * (, M_p, (, P[1], #))) ;
      else $mapping\_rule_1$ ← ConcatString(Individual:@, Col_{name},*(, N_m, =, P [i], #) Types:, P[i],)
      $mapping\_rule_2$ ← ConcatString ($mapping\_rule_2$, hasPropertyValue@, Col_{name}, *(, M_p (, P[i], #))) ;
    end
    return mapping_rule [];
  end
return

For example, '*birthdate*', '*birthplace*' are concepts in the ontology and the '*B*' and '*C*' columns in the CSV file contain all the instances of '*birthdate*', '*birthplace*' respectively. The instances of these properties are imported using the following mapping expression $mapping\_rule_1$. The instances of properties or attributes P_1, P_2 are imported from column *'B','C'*.... N_m contains the string 'mm:namespace'. We used the N_m variable which allows the ontology to have specific reference to properties. We used this reference to identify the instances of the attributes. After all the individuals corresponding to the concepts in the ontology are added, we relate the instances or individual concepts as properties to the domain or concept in the second phase. For instance, '*birthdate*', '*birthplace*' contain data on the corresponding president's name

in the CSV file. Instances of properties are related to the Concept i.e. '*president*' using the following mapping expression $mapping\_rule_2$.

$$\begin{aligned} mapping\_rule_2 = {} & Individual{:}@A^* \\ & Types{:}C_{concept} \\ & Facts:\ hasPropertyValue@B^*(M_p(P_1\#'')), hasPropertyValue \\ & @C^*(M_p(P_2\#)) \ldots \end{aligned}$$

Here, column '*A*' contains all the instances of the concept (i.e. president names) and '*hasPropertyValue*' is an object property which is used only to relate the instances of the other concepts or attributes $P_1, P_2 \ldots$ with instances of the main concept C_{oncept} i.e. the president's name. C_{oncept} variable contain the name of main concept ('*president*'). The instances of $P_1, P_2 \ldots$ lie in the CSV data in columns '*B*', '*C*'...respectively. M_p contains 'mm.prepend' which is used to prepend the properties' names with each instance.

In cases where the property name already exists as a concept, then this property name is declared to be an equivalent class as the existing concept. For example, "*occupation*' attribute is an equivalent class to "*occupation*'. Now, when the instances of a class already exist as a concept in the OMRKBS, the existing classes are added as types of instances through the 'types' properties. For instance, businessman is an instance of "*occupation*' and "*businessman*' also exists as a class under the "*occupation*' class. So, the concept "*businessman*' is added as type to the "*businessman*' instances through the 'types' property. Also, we see in Example 4 that some instances (e.g. the "*occupation*' field) contain multiple values separated by '|' where each value is another instance. We split the instances or axiom by '|' and consider each split value (e.g. businessman, politician) as the instance corresponding to the concept (e.g. "*occupation*'). We add each split value separately corresponding to the concept in OMRKBC. After that, we relate these instances of concepts to the main concept to be defined (e.g. "*president*'). Algorithm 2 shows the pseudocode for creating the mapping rules automatically.

OMRKBC program

In this section, we explore how the IISDBS process is executed into a program efficiently. This is the main program where CSV content is imported into the ontology. Mapping master [44] is a source library which can be used to transform the content of spreadsheets to OWL ontologies. We use this library with OWL API in Java to convert the spreadsheet into ontologies [43]. The three algorithms proposed to address the challenges in IISDBS are called by the program in order. Primarily, CSV data are pre-processed, and large files are split into multiple files after being pre-processed. This code executes tasks for each file through a loop. First, each Excel file is loaded into the program. Next, the domain properties are discovered as concepts in the ontology. After this, the mapping expression algorithm is called and the mapped master expression is returned to the node which represents the expression. Then, the data is looped as specified by the Mapping Master expression.

Finally, the OWL axioms rendered by the Mapping Master expression are added to the source ontology. Algorithm 3 shows the pseudocode for importing Excel data into the ontology using OWL API. In line 8, we can see that a Mapping Master expression

is rendered over a range of cells in a sheet. A Mapping Master parser is created for the expression in line 10. The parser parses and returns a node representing the expression in line 11. In line 12, the cells are looped as specified by the Mapping Master expression. Line 14 shows that a Mapping Master expression is rendered in the context of a location in a spreadsheet. The OWL axioms are added which are rendered by the Mapping Master expression in line 20. The system takes an average of 20.1 min to embed the instances of each file of concepts after resolving the challenges. On the contrary, the large file could not be loaded and mapped into the system before resolving the challenges. In "Experiments and a comparison of the results" section, we discuss the details of the space reduction and the time consumed for this program.

Algorithm 3: Import Spreadsheet Data

input : String FileName: The Excel file name and domainNm: domain Name
output: generate axiom from excelfile and import into ontology

Initialization
ontology: contain an OWL ontology;
ontologySource: contain an ontology source
spreadsheetSource: contain spreadsheet data source
owlRendering: contain set of OWL axioms
mappingRule: contain mapping expression as string
mmExpression: contain node representing the expression
columnNumber: contain number of column in file
rowNumber: contain number of row in file
mapping_rule: contain mapping expression as string ;
begin
    ontologySource ← createOntologysource(ontology)
    spreadsheetSource ← createOntologysource(FileName)
    mapping_rule ← CreateMappingExpression(FileName, domainNm) /*call create mapping expression algorithm */
    mmExpression ← CreateMappingMasterExpression(mappingrule)
    parser: Mapping Master parser
    parser ← CreateMappingMasterParser(mmExpression)
    mmExpressionNode ← ParseNodeForExpression(parser)
end

OMRKBC with ConceptNet data

This section explores how ConceptNet data are built in the OMRKBS. We discuss the challenges involved in importing data from ConceptNet into OMRKBC and discuss the solution. Then, we present a program to build the ConceptNet data in OMRKBC.

Extracting the data

ConceptNet provides seven large CSV files as datasets which can be downloaded from https://github.com/commonsense/conceptnet5/wiki/Downloads. The important fields or columns in the CSV files are 'relation', 'node at the start' and 'node at the end'. To better understand the ConceptNet dataset in CSV format, Example 5 is given 'relation', 'node at the start' and 'node at the end' are expressed by the edge: '*/r/CapableOf*', '*/c/en/president*' and '*/c/en/govern_a_nation respectively*'.

Example 5 As an example, the 'president is capable of governing a nation' appears in the ConceptNet dataset as follows: */r/ CapableOf /c/en/president /c/en/govern_a_nation /ctx/all "weight": 1.0*

Addressing the challenges of building the ConceptNet data in OMRKBC

We concentrate on importing the features of the concept associated with the relations from ConceptNet. The challenges in converting the data into RSI are discussed and resolved.

(a) Pre-processing the data The ConceptNet CSV files are too large to open. Therefore, all the CSV files are imported into the MySQL database where relation, start node and end node' are three fields in the table of the ConceptNet database. The data of 'start node' associate the 'relation' with 'end node'. We can see from Example 5 that 'start node' contains '*president*', 'relation' contains '*capableOf*' and 'end node' contains '*govern a nation*' for the sentence '*president is capable of governing a nation*'. The information on each concept is queried with each relation where the concept is matched with the 'start node' or 'end node' field in the dataset. For instance, data are queried about a concept i.e. '*president* where the 'start node' or 'end node' field is like '%president%' and 'relation' is like '*capableOf*'. The query will return all data lying between '*capableOf*' and '*president*' which means the query will return all things an president is capable of (i.e. *governing a nation*).

(b) Discovering concepts and relations This stage confirms that all the relations of ConceptNet are built in OMRKBC. Each concept is discovered here using the method described in "Discovering concepts" section. First, ConceptNet uses some important relations to represent concepts and these relations appear in the relation field of the dataset. They are '*capable of*, '*used for*' etc. All relations are assigned under the '*relation*' class. After this, each word in the relation is discovered or assigned if the relation contains more than one word. Next, we split all the words in the statement or description (i.e. '*governing a nation*') which results from the query and each word in the statement is discovered or assigned as a concept. In cases where any word group in a statement is identified as a relation according to the rule given in "Discovering relations" section, these word groups are assigned as concepts under the '*relation*' class.

(c) Mapping The associations between statements and relations corresponding to a domain (i.e. '*president*') are mapped in the OMRKBS. A mapping expression is given to build relations with the domain as shown in Example 5.

(*govern* and *a* and *nation*) and *capableOf* is a superclass of *president*

A synonym is also a relation in ConceptNet. The synonym of a word is expressed the same as other relations in OMRKBC. For example, in ConceptNet, the synonyms of '*president*' are '*head of state*'. This is shown by the following expression.

(*head* and *of* and *state*) and *synonym* is a superclass of *president*

OMRKBC program

We design a program to import the features of the concepts associated with the relations from ConceptNet. We use the procedures proposed to resolve the three challenges in converting the data into RSI. Then, the RSI is built in the OMRKBS using the proposed program.

OMRKBC with a description of concept

Defining a concept with a description is an important part of the proposed OMRKBC process. We identify the challenges in converting a description to RSI and provide the formula to address the challenges. In this section, we explore how concepts are defined by a short abstract or description through RSI. Also, we define the instances with a description and the relation with meanings.

Extracting the description

The DBpedia dataset provides a short abstract for each article and can be downloaded from (http://wiki.dbpedia.org/data-set-36). This abstract can be used as a definition or description of the concept. A short abstract from DBpedia is shown in Examples 6–8. Next, we extract the meaning of the concept as text from WordNet. Then, concepts are imported with a description or annotation from the existing ontologies (i.e. CRISP, Schema) while constructing the base ontology. The meaning of the relation is retrieved from the Oxford Dictionary [45] using API, which is available at https://developer.oxforddictionaries.com and the relation is defined with the meaning. We take three example from Examples 6 to 8 to illustrate the OMRKBC with definition and they are '*politician*' domain, concept '*president*' which is subclass of '*politician* and '*Donald Trump*' which is instances of '*president*'.

Example 6 For instance, a short abstract of the "*politician*' domain is written in the dataset as http://dbpedia.org/resource/politician "A politician is a person active in party politics, or a person holding or seeking office in government. In democratic countries, politicians seek elective positions within a government through elections. In non-democratic countries, they employ other means of reaching power through appointment, bribery, revolutions and intrigues. Politicians propose, support and create laws or policies that govern the land and, by extension, its people. Broadly speaking, a politician can be anyone who seeks to achieve political power in any bureaucratic institution".

Example 7 A short abstract of the '*president*' concept is given as "A president is the leader of a country or a division or part of a country, typically a republic, a democracy, or a dictatorship. Among other things, President today is a common title for the heads of state of most republics, whether presidential republics, semi-presidential republics or parliamentary republics".

Example 8 Take for example instance '*Donald Trump*' of '*president*'. The short abstract of '*Donald Trump*' is "Donald Trump is an American businessman, author, television producer, politician, and the Republican Party nominee for President of the United States in the 2016 election. He is the chairman and president of The Trump Organization, which is the principal holding company for his real estate ventures and other business interests. During his career, Trump has built office towers, hotels, casinos, golf courses, an urban development project in Manhattan, and other branded facilities worldwide. He was a businessman and television personality. Trump was born and raised in the New York City".

Addressing challenges in building a description in OMRKBC

Our challenge is to learn how to process the text in the description into RSI. As discussed, there are three types of challenges which must be addressed: pre-processing content, mapping information and embedding information.

(a) Preprocessing the content Short abstracts or descriptions must be pre-processed because sentences in the short abstract may be too complex or too long to relate words in the sentence with concepts. As shown in Examples 6–8, the sentence is so complex that it is difficult to relate the words in the sentences directly with the concepts in the ontology. Therefore, the text in an abstract is reformed into RSI in three steps. Firstly, each complex or long sentence is split into several simple clauses. Open information extraction (Open IE) [40] which is part of the Stanford NLP parser [42] extracts simple clauses from sentences. Each simple sentence appears to be an individual feature of the concept and is presented as (subject; property; object). Table 1 shows examples that how sentences which are taken from Example 6–8 are formatted after splitting.

Secondly, we remove some sentences from a list of simple sentences before converting the structured input to reduce redundancy. First, we only take one sentence which contains the subject, object and predicate related to the concept to be defined. If a synonym or an equivalent of the concept exists as a subject or object, we consider the sentence also. Next, if there is more than one sentence which looks similar or almost similar, the most complete sentence is included in the structure. For example, between two statements: <*politician, active person, in party politics*> <*politician, is, person active*>, the complete statement is <*politician, active person, in party politics*>. We remove the other similar statement.

Finally, the rest of the simple sentences are converted into structured information. Subject is the concept to be defined, and (predicate, object) are the characterization of the concept. The structures of the simple sentences are constituted from (predicate, object). Each predicate and object are parsed from each sentence and are turned into the structured input using a few rules. The structured input is presented with a series of arguments and each argument is separated with ",". We discuss the rules with examples as follows.

Rule 1: the verb to be in the predicate acts as simple present and will not be included in the structure.

Rule 2: when a clause in the predicate or object contains 'of' or '*by*', the clause is split into three parts: the first part is the words following '*by*'/'*of*', the next part is '*of*'/'*by*' itself, and the last part is the words preceding '*of*'/'*by*'. Each part is an argument in the structure.

Rule 3: when a clause in the predicate or object can be declared as a relation using the method in "Discovering relations", the relation word is an argument.

Rule 4: if a verb is not included as a relation as in Rule 3, the verb is included in the structure in base form.

Rule 5: Adverbs located before verbs are removed in the structure.

Rule 6: Adjectives of the subject or object are removed in the structure.

Rule 7: When the structure contains only the object with no prepositions (e.g. '*of*') and a concept (e.g. '*businessman*') in the structure is discovered as a subclass of the attributes class (e.g. '*occupation*'), the attribute class is included as an argument in the structure also.

Table 1 Example of how sentences are formatted after splitting

Politician
<politician, create; laws> <create, policies>
<politician, propose, laws> <propose, policies>
<politician, seek elective positions within; a government>
<politician, holding office in; a government>
<politician, seek political power; to achieve, in any bureaucratic institution>
<politician, reaching power through, bribery>
<politician, reaching power through, revolutions>
<politician, reaching power through, intrigues>
<politician, seek elective positions at; times>
<politician, is active person, in party politics>
<politician, is, person active>
President
<president, is the leader of, a country>
<president, common title for, the heads of state of most republics>
<president, is, common title>
<president, is the leader of, division or part of a country> v
<president, is, leader>
<president, is leader of, republic>
<president, is leader of, democracy>
<president, is leader of, dictatorship>
Donald trump
<Donald Trump, is, the chairman, of The Trump Organization>
<Donald Trump, be president of, The Trump Organization>
<Donald Trump, be the Republican Party nominee in, the 2016 election>
<Donald Trump is American businessman>
<Donald Trump is businessman>
<Donald Trump is American>
<Donald Trump is chairman>
<Donald Trump is chairman of Trump Organization>
<Donald Trump has built facilities During his career>
<Donald Trump has built hotels>
<Donald Trump has built facilities>
<Donald Trump development project is in Manhattan>
<Donald Trump has built development project>
<Donald Trump has built golf courses>
<Donald Trump has built casinos>
<Donald Trump has built office towers>
<Donald Trump has built development project in Manhattan>
<Donald Trump has built branded facilities>
<Donald Trump, was, television personality>
<Donald Trump, born in New York City>
<Donald Trump, raised in New York City>

Sentences are taken from Examples 6–8. All these examples have moved to another concept, as indicated by double line

The features which were generated from the sentences in Examples 6–8 by splitting are structured with the rules shown in Table 2. Each sentence in the description has been formatted into structural information so that the structure of the sentences can be mapped easily to build RSI into the ontology.

Table 2 Examples of how rules structure the sentences about concepts

	Rule
Politician	
<create; laws> <create, policies>	No rule
<propose, laws> <propose, policies>	No rule
<seek elective, positionsWithin, a government>	Rule 3
<reach, powerThrough, bribery> <reach, powerThrough, revolutions> <reach, powerThrough, intrigues>	Rule 4
<seek political power to, achieveIn, bureaucratic institution>	Rule 3 and Rule 6
<active person, in party politics>	Rule 1
<hold office, in government>	Rule 4
President	
<leaderOf, a country>	Rule 1 and Rule 3
<title for, heads of state, of, republics>	Rule 1 and Rule 2 and Rule 6
<leaderOf, division or part, of, a country>	Rule 1 and Rule 3 and Rule 2
<leader>	Rule 1
<leaderOf, republic> <leaderOf, democracy> <leaderOf, dictatorship>	Rule 1 and Rule 3
Donald Trump	
<chairmanOf, Trump organization> <presidentOf, Trump organization>	Rule 1 and Rule 3
<republican party nominee, in the 2016 election>	Rule 1
<American businessman>	Rule 1
<occupation, businessman>	Rule 1 and Rule 7
<nationality, American>	Rule 1 and Rule 7
<built, facilities, during career> <built, branded facilities> <built, hotels> <built, development project, in Manhattan> <built, golf courses> < built, casinos> < built, office towers>	Rule 1
<television personality>	Rule 1
<bornIn, New York City> <raisedIn, New York City>	Rule 3

The features of concept are taken from Table 1. All these examples have moved to another concept, as indicated by the dark grey colour

(b) Discovering concepts and relations Here, each word is discovered using the method discussed in "Discovering concepts" section. After extracting the structure of sentences in the abstract, each word in the structure is discovered as a concept. Instantly, possible relations and attributes in the structure are identified according to the rule given in "Discovering relations" section.

(c) Mapping Each sentence in the description about a concept or instance are built into the ontology according to the structure of the sentences. For this, the relationship among the arguments in the structure are mapped using the following two phases. First, the relationships among the words in the arguments and the relationships among the arguments in the structure are made by joining the words with an 'and' expression. Then, this relation is declared as a superclass of concept or types of instances.

Example 9 As an illustration, we take some sentence structures about '*politician*' from Table 2 and map them using the following expression.

create and *law* is a superclass of *president*
reach and *powerThrough* and *bribery* is superclass of *president*

Example 10 As another example, 'Donald Trump is an instance in OMRKBC and the expression for mapping is as follows:

bornIn and *New York City* is types of *Donald Trump*
nationality and *American* is types of *Donald Trump*

The reason for declaring features as super-classes is because these features or characteristics are inherited by the concept and the subclasses of the concept. The features of the instances will be derived through property type.

OMRKBC program

We designed a program to define the concept with a description through RSI. In this program, the text in the description or abstract is processed into structural information by resolving the challenges and the structural information is built in the OMRKBS. In conclusion, when any new word is added as a concept in the ontology, this new concept can be defined with the description and the data and instances and synonyms from DBpedia and ConceptNet. Thus, we can develop an independent ontology-based KBS.

System output

In this section, we show how information about a word is queried and represented in the OMRKBS. We use the SPARQL [43] language for the query and format the proposed ontology in the RDF format. We introduce three types of searches in the system and represent the information according to the search. In the following, we describe these three types of searches: concept search, instance search and process queries. Generally, when a word is searched in the system to retrieve information on this word, we call the word a '*query word*'. We declared few PREFIXs to reference IRIs where nsf prefix is source of OMRKBC.

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX nsf: <http://www.semanticweb.org/shirinkhan/2017/>

Concept search

A concept search is executed when a '*query word*' exists as a concept in an ontology. First, we introduce the queries to return the information about the concept. After this, the resulting data for a concept is represented. There are two types of queries by which to extract information about the concept. These queries extract features or characteristics, instances and general attributes. Table 3 shows the queries for the '*president*' concept and the result of the queries. The query on the first row retrieved the features of the

Table 3 Queries for the concept search of '*president*'

No	Query
1	select ?superclass where nsf:politician rdfs:subClassOf ?superclass.
2	SELECT ?entity ? type WHERE { ?entity rdf:type ?type. ?type rdfs:subClassOf* nsf:president.}

concept and the second row retrieved the instances of the concept. In this section, we explore how information is presented with these queries.

(a) Feature representation: Since features are considered to be a superclass of a concept, the extraction of a feature becomes easier. A simple SPARQL query syntax is written to retrieve all the superclasses of a concept which represent the characteristics or features of a concept in an ontology. The query syntax and results of a query are in the first row of Table 3. All the features of a concept are combined from the resulting data and are then represented as the definition of the concept. The presentation of the resulting information replaces the 'and'/'or' expression' with ',' in the original result.

Example 11 We give examples of the representation of features as the definition of concept '*president*' as follows:

President	
Leader	*Capable of*:
Title for heads of state of republics	Give a speech
Leader of	Lie
Country	Duck the issue
Division of a country	Govern a population
Part of a country	
Republic, democracy, dictatorship	

Because each feature appears individually, and the feature may be associated with attributes and relations in the result, the presentation of the results becomes more specific and meaningful. In Example 11, we understand from the result: who is the leader of country. Also, some important relations such as '*leaderOf*' helps to separate some characteristics which specify the result. We consider a relation to be important if the relation has a connection with more than two features.

(b) Instance representations: The query in the second row of Table 3 returns the instances of the concept. Also, if the instances of a subclasses of a concept exist in an ontology, this query returns the instances of the subclasses with the subclasses name. We place ',' between the names of the instances and the subclasses in the representation of the resulting data. Appending instances of the concept makes the output more informative. In conclusion, when a concept search is performed, the results of these two types of queries are represented in a bind.

Example 12 For example, '*president*' concept returns the list of president names in the following format.

List of presidents:
Barack Obama
Donald Trump
Justin Trudeau
Moon Jae-in
..
..

Table 4 Query for the instance search (i.e. '*Donald Trump*')

Query
select distinct ?value ?generalAttributes ?Gtype ?GClass where {
nsf:'Donald Trump' ?property ?value.
nsf:'Donald Trump' rdf:type ?GClass.
FILTER(?GClass != owl:Class&& ?GClass != owl:NamedIndividual)
?value rdf:type ?GeneralAttributes.
OPTIONAL
?value rdfs:seeAlso ?label.
?label rdf:type ? Gtype. FILTER(? Gtype!= owl:Class&& ? Gtype!= owl:NamedIndividual)
filter (?property not in (rdf:type))
FILTER(? GeneralAttributes!= owl:Class&& ? GeneralAttributes!= owl:NamedIndividual)}

Instance search

When the '*query word*' exists as an instance of a concept in OMRKBC, the system starts the instance search. This search returns the concept of the instance and the instances of the attributes corresponding to the instance. Table 4 shows the query for this search and Example 13 shows how the information on the instance '*Donald Trump*' is represented.

Example 13 For example, '*Donald Trump*' is queried in the system and the query returns information about '*Donald Trump*' as shown in Table 4. The information means that '*Donald Trump*' is an instance of the 'president' class. This president's birthplace is the USA and the USA is also a concept 'country' which is mentioned through 'type.'' The results for the instances '*Donald Trump*' are presented as follows. Also, the instance will have the same feature of concept. For example, '*Donald Trump*' is an instance of the '*president*' concept. Therefore, '*Donald Trump*' has all the characteristics of a president. The features of a president which are inherited by instances are retrieved using a concept search. Thus, it is possible to vary the search scope in OMRKBS.

Donald Trump	
President	Birth Date: June 14, 1946
American businessman	Birthplace: USA
Chairman of Trump Organization	Spouse: Ivana Zelnickova
President of Trump Organization	Nationality: American
Republican party nominee in the 2016 election	Occupation: businessman
Television personality	Start Year: 1976
Born in New York City	Political Party: Republican
Raised in New York City	Age: 72
Built	Service Start Year: January 20, 2017
Facilities during career	
Branded facilities	
Hotels	
Development project in Manhattan	
Golf courses	
Casinos	
Office towers	

Table 5 The query for an example question ('*politician*', '*policies*')

Query
SELECT ?subClassOf WHERE {
nsf:politician (rdfs:subClassOf| owl:equivalentClass) * ?subClassOf.
?subClassOf owl:intersectionOf | owl:unionOf ?subClassOf_name.
?subClassOf_name rdf:rest*/rdf:first* ?name.
optional{
?name owl:intersectionOf | owl:unionOf ?subsubClassOf_name.
?subsubClassOf_name rdf:rest*/rdf:first* ?subname.}
filter(nsf:policies in (?name) | | nsf: policies in (?subname))}

Process queries

Process queries allow the user to ask a question to find information about a word which has a specific property or attribute. This question is asked with two arguments where the first argument usually represents the main word that the user wants to know, and the second argument represents the property or attribute of a main word. Each argument is separated by ',' in the question. The result of this search shows the relationship between the two arguments. Table 5 shows the query for a question ('*politician*', "*policies*') with the result.

First, the main word can be the concept in the question. Also, the property or attribute can be a concept which describes the main concept with other concepts. This search returns all the features of the main concept related to the property or attribute. It also filters all the inherited features of the main concept associated with the property concept.

Example 14 For example, '*politician*' is the main concept and '*policies*' is one property concept of '*policies*'. When the question ('*politician*', '*policies*') is asked in the system, the results show all the features of the politician related to policies (e.g. politician create policies).

Example 15 For example, '*president*' is a subclass of '*politician*' and '*politician*' has been defined in the ontology as <*politician create policies*>. When ('*president*', '*policies*') is queried, the results show <*president create policies*> as '*president*' inherits the superclass features of '*politician*'.

Secondly, suppose a user wants information on an instance associated with a specific property. In this case, the main word is an instance and the property is a concept in the question. This query returns instances of a property associated with a main word instance.

Example 16 For example, ('*Donald Trump*', '*occupation*') or ('*Donald Trump*', '*birthdate*') is queried in the system to retrieve information on the occupation or birthdate of '*Donald Trump*', hence the query will return the result (Donald Trump's occupation: businessman and birthdate: June 14, 1946)

In short, a concept search finds the definition of a concept or word and an instance search returns information about an instance. Finally, a question can be asked about a word associated with a property, and the system will return data related to the word and the property.

Comparison of the characteristics of OMRKBS with other KBSs

There are several fundamental qualities that facilitate efficient searches in the OMRKBS. First, every concept is defined by relating other concepts in the OMRKBS rather than by using annotation. The importance of the object and data properties is not significant, since each property is declared as a concept in OMRKBS. Then, OMRKBC supports various relations and attributes. Next, individual features are richly structured with relations and attributes and are called superclasses of a concept in OMRKBS. The features are inherited by the concept and subclasses of the concept. Also, OMRKBS provides general information about concepts or instances (e.g. '*birthdate*', '*spouse*'). These fundamental qualities enable different types of searches in OMRKBS and assist in returning specific, meaningful, and logical information for the search.

In addition, as we see from Examples 14 to 16, OMRKBS enables questions to be asked with two arguments to understand the relation between these two arguments. The DBpedia system supports instance searches (e.g. '*Donald Trump*', '*occupation*') but not concept searches ('*politician*', '*policies*'). But other KBSs such as WordNet or ConceptNet do not support this type of query. Also, the individual features as defined will be inherited by the subclasses in response to the question. WordNet and our system have this functionality. '*president*' inherits all the features of '*politician*' and answer the question ('*president*', '*policies*') that <*president create policies*>. Therefore, the system can answer logical questions. Table 6 compares the characteristics of OMRKBS with the other KBSs.

Experiments and a comparison of the results

We propose a standard implementation of our framework in Java for Windows, version 10 and the experiments are performed on a PC with quad-core CPU (4 GHz) and 16 GB of RAM. The resources that were used in the development of OMRKBC are DBpedia, ConcpetNet 5 and WordNet. We used OWL API to import axioms as a subclass, superclass or class in OMRKBC. The program uses the JWNL, which is an interface to the WordNet dataset. The synonymous terms and the considered hypernyms are retrieved from WordNet using this interface. We use the Mapping master source library to transform the content of the spreadsheets to OWL ontologies. We used Open IE and Stanford NLP parser library [42] sources to split the sentences. We connected the MySQL database with the MySQL JDBC driver to retrieve the ConcedptNet data. We used the SPARQL query language to retrieve the data from our KBS. To evaluate OMRKBC, we created a KBS in English for the domain '*agent*' ('*person, artist, athlete, journalist*, etc), *place* (*city, country, region, country*), *work, game* (*soccer, cricket, golf*, etc), *organization*, *animal*, *educational institution* (*university*), *event, album, chemical*. We grouped the results by dataset and analysed the outcome of the structural information from different sources. The datasets are ConceptNet 5, WordNet 3.0 and DBpedia. To compare the results with our KBS, we used the abstract and instances with the properties from

Table 6 Comparison of the characteristics of OMRKBS with the existing KBSs

Characteristic	DBPedia	WordNet	ConceptNet	OMRKBC
Individual features are presented	X	X	✓	✓
Each word is a concept	X	X	X	✓
No specific data property	X	X	X	✓
Limited object properties	X	X	X	✓
Provides general information on an instance or concept	✓	X	X	✓
Inherits the superclass features	X	✓	X	✓
Individual feature is presented with attributes	X	X	X	✓
Individual feature is presented with relations	X	X	✓	✓
Allows a question to be asked about a concept with a property	✓	X	✓	✓

DBpedia, and the relation statement '*capable of*, '*used for*' and '*type of*' from ConceptNet, and the meaning, hypernyms and synonyms from WordNet.

First, we conduct an experiment to execute the program for IISDBS over the existing datasets. As part of the experiment for this program, we evaluate the space reduction of the file over the pre-processing algorithm and time consumption of the program of IISDBS. We can see from Table 7 that the average file reduction is 68.4% from the actual size after Excel conversion and 93.5% after the null and invalid values are removed over the pre-processing algorithm. The Excel files size is only 716 KB after each file is split. We embedded instances of one file for each domain. The execution times of the IISDBS program for each domain are shown in Fig. 7. We can see from Fig. 7 that the average execution time of each file is 20.14 min. This implies that the system can process the small size file and embed the data in the system after pre-processing the large file and creating an automatic mapping expression whereas the large file could not be loaded and mapped into the system before resolving these issues.

Next, we evaluate the accuracy of the process queries, relation discovery and overall accuracy of OMRKBS over the selected datasets. An evaluation team of 15 PhD students, all experienced in the field of information retrieval, was formed to assess the accuracy of the process queries, relation discovery and overall accuracy of OMRKBS. We can see from Table 8 that the mean accuracy of the process queries is 93%. The accuracy of relation discovery is measured by the number of relations which are discovered correctly in system and the results shows a 100% accuracy. Finally, the evaluation team was asked to determine how well each statement in the results is expressed when a word is queried in the system. When a statement presents ambiguous, unrelated or unclear individual features, this information is regarded as being poorly expressed. On the other hand, when a statement presents relevant and correct individual features with attributes and relations, this information is regarded as being well expressed. The accuracy of the word query is calculated using the number of well-expressed features among the results. The evaluation team calculated the accuracy of the concept search and the instance search using Eq. (1). The results from each domain were averaged for the two searches as shown in Fig. 8. We calculate the overall accuracy of OMRKBS by the mean accuracy of these two searches using Eq. (2) and present the overall accuracy in Fig. 8. Observe that the accuracy of the instance search is slightly lower than the concept search. Interestingly,

Table 7 This table shows the file size reduction of each domain after each step of the pre-processing algorithm

Domain	Original size	Size after excel conversion	Size after null and invalid values removed	File size after split (KB)
Agent	13 GB	5 GB (60%)	260 MB (98%)	800
Animal	250 MB	61 MB (75%)	23 MB (91%)	820
Chemical	15 MB	3.1 MB (79%)	1.7 MB (89%)	705
Event	134 MB	44 MB (68%)	8.1 MB (94%)	718
University	32 MB	7.5 MB (77%)	3.4 MB (89%)	690
Game	1 MB	188 KB (82%)	132 KB (87%)	660
Album	327 MB	116 MB (65%)	13 MB (96%)	650
Organization	1.32 GB	564 MB (57%)	46 MB (96%)	680
Place	4.43 GB	1.8 GB (60%)	125 MB (97%)	720
Work	1.7 GB	663 MB (61%)	27 MB (98%)	710
		68.4%	93.5%	716

The second column shows the original size of the file

Columns 3, 4 and 5 show the file size reduction after the program converts the file to Excel, removes the null and invalid values, and splits the file. The percentage of file reduction is given in parentheses. The bottom line indicates the average percentage of the file reduction of each step

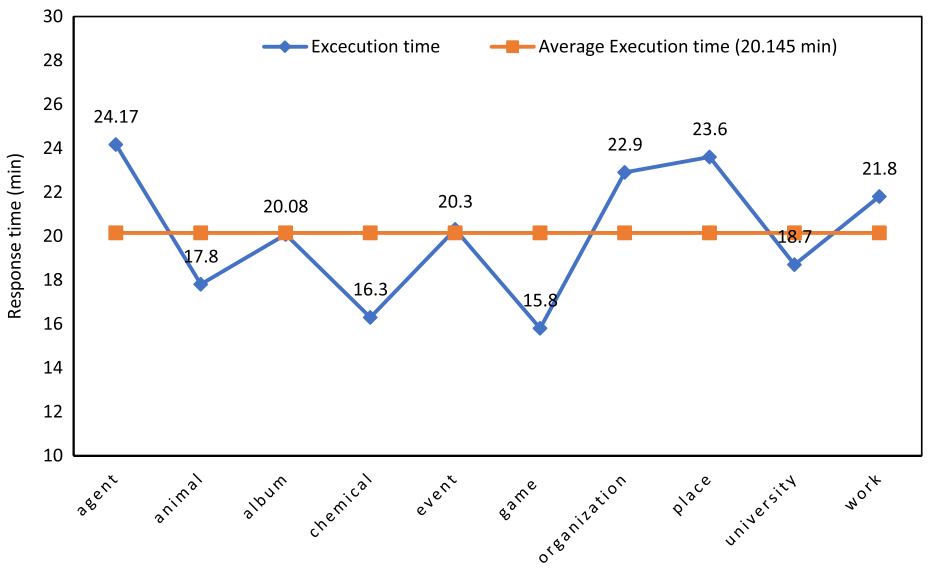

Fig. 7 The time consumption to execute the IISDBS program

the event and organization domains have a higher overall accuracy whereas the chemical domain has the lowest accuracy.

$$Accuracy = \frac{|relevant\ results \cap retrieved\ results|}{|retrieved\ results|} \tag{1}$$

$$Overall\ Accuracy = \frac{|Concept\ search\ accuracy + Instance\ search\ accuracy|}{2} \tag{2}$$

Table 8 Evaluation of the accuracy of the process queries and relation discovery

Domain	Process queries (%)	Relation discovery (%)
Agent	89	100
Animal	93	100
Chemical	96	100
Event	97	100
University	93	100
Game	94	100
Album	94	100
Organization	91	100
Place	90	100
Work	94	100
Overall	93	100

The bottom line shows the mean accuracy of the process queries and relation discovery in OMRKBS

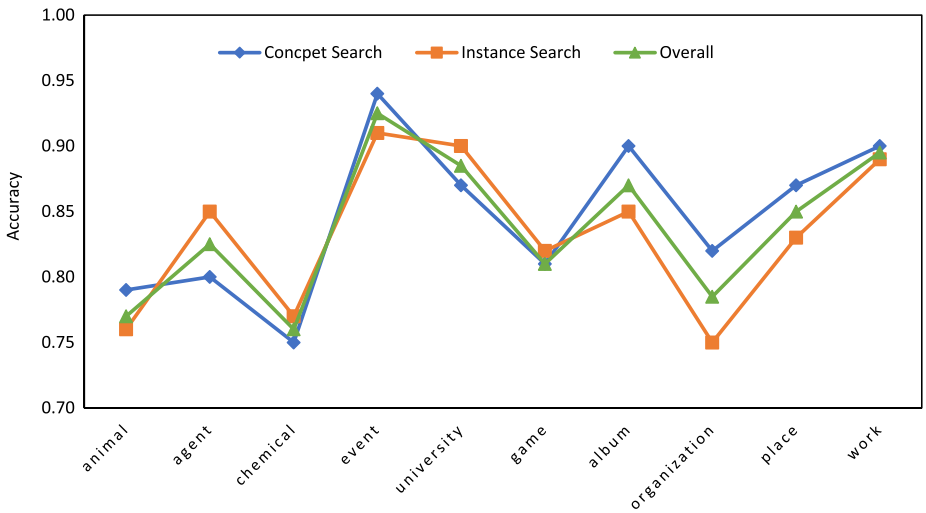

Fig. 8 Evaluation of accuracy of the concept search and instance search of the important domains in OMRKBS. The green line shows the overall accuracy of OMRKBS

Now the accuracy of the KBSs: DBpedia, ConceptNet, WordNet and OMRKBS is evaluated using the same dataset. Figure 9 gives a breakdown of the results. In OMRKBS, each word is a concept, and each concept is defined with other concepts rather than with statement or a description. Therefore, it is feasible to represent the features of the concepts with attributes, relations and related concepts in specific ways. OMRKBS supports many relations (i.e.'*born in*', '*capable of*') and attributes ('*nationality*', '*occupation*'). In DBpedia and WordNet, the definitions (e.g. abstracts, meanings) are descriptive and presented in text format whereas ConceptNet and OMRKBC provide more specific or individual features with relations and attributes. ConceptNet has a higher accuracy of 77% in terms of the appearance of individual

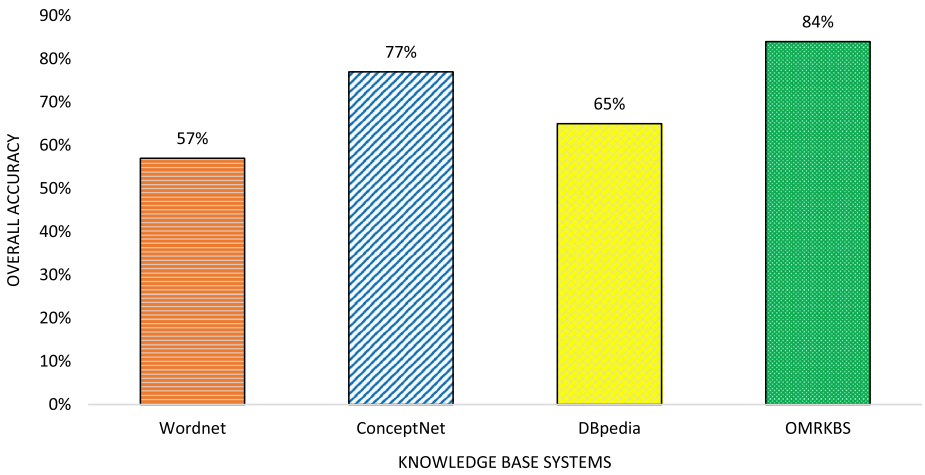

Fig. 9 Comparison of the accuracy of OMRKBS with the existing KBSs over the same dataset

features than DBpedia and WordNet as ConceptNet gives specific information with relations.

However, OMRKBC is one repository where concepts are interconnected with relations and attributes in various ways. Therefore, individual features are informative and meaningful. For example, when an instance such as president name ('*Donald Trump*') is queried, OMRKBC shows the president's general information such as '*birthdate*', '*birthplace*' etc., whereas ConceptNet mostly does not provide general information. Although DBpedia provides general information with attributes, it does not present individual features with relations (e.g. <*Donald Trump, chairman of, Trump Organization*>). Some properties of an object are lost or invisible when defining an object in ConceptNet. In contrast, concepts or instances can be represented with attributes in OMRKBC. For instance, *Donald Trump* is businessman which is represented in our ontology <*Donald Trump, occupation, businessman*>. ConceptNet may mention '*Donald Trump*' is businessman but the property of businessman is hidden. In this sense, the representation of data in OMRKBS is more meaningful and tangible. OMRKBS has higher accuracy (84%) than the other KBSs for individual features. This result suggests that the individual features of concepts are well-structured in OMRKBS.

Conclusion

This paper discussed the process of building an OMRKBS over the last few years and several issues relating to OMRKBC. We described the NLIKR scheme in which each concept, denoted by an English word or phrase, is defined by its relations with other concepts and its position in the concept space. We identified a key challenge, this being to convert the data into RSI using a classical technique to map information into the structure. However, the classical techniques are not effective on large datasets, hence, we used the NLIKR scheme to translate the information in OMRKBS. We applied rules, algorithms and techniques to transform the data into RSI. As a result, the information is well structured with attributes and relations. This improves the effectiveness of the

query results and has higher accuracy compared to the other KBSs. Though OMRKBC is not a fully independent KBS, it is partially developed and focuses on a specific domain. However, it is a proposed method, where the development of a complete large knowledge base repository is possible.

Acknowlegements
Not applicable.

Authors' contributions
SAK carried out design of the proposed framework and run the experiment to collect the data, did the analysis of the results. FL proposed NLIKR scheme which is part of OMRKBC scheme. YPPC and FL mainly managed and supervised this paper. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 5 November 2018 Accepted: 23 May 2019
Published online: 10 June 2019

References

1. Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, van Kleef P, Auer S, Bizer C (2015) DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia. Sem Web J 6(2):167–195
2. Benferhat S, Dubois D, Prade H (1997) Some syntactic approaches to the handling of inconsistent knowledge bases: a comparative study part 1: the flat case. Studia Logica 58–1:17–45
3. Hasan KS, Ng V (2014) Automatic keyphrase extraction: a survey of the state of the art. In: Proceedings of the 52nd annual meeting of the association for computational linguistics, pp 1262–1273
4. Najmi E, Hashmi K, Malik Z, Rezgui A, Khan HU (2014) Conceptonto: an upper ontology based on conceptnet. In: 11th ACS/IEEE international conference on computer systems and applications (AICCSA), Doha, pp 366–372
5. Zghal HB, Moreno A (2014) system for information retrieval in a medical digital library based on modular ontologies and query reformulation. Multimedia Tools Appl 72–3:2393–2412
6. Gorskis H, Aleksejeva L, Polaka I (2016) Database analysis for ontology learning. Procedia Comput Sci 102:113–120
7. Nakhla Z, Nouira K (2017) Automatic approach to enrich databases using ontology: application in medical domain. Procedia Comput Sci 12:387–396
8. Copestake A (1990) An approach to building the hierarchical element of a lexical knowledge base from a machine readable dictionary. In: Proceedings of the first international workshop on inheritance in natural language processing, Tilburg, The Netherlands, pp 19–29
9. Ji H, Grishman R (2011) Knowledge base population: successful approaches and challenges. In: Proceedings of the 49th annual meeting of the association for computational linguistics, Human Language Technologies, pp 1148–1158
10. Navigli R, Ponzetto SP (2012) Babelnet the automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artif Intell 193:217–250
11. Speer R, Chin J, Havasi C (2017) Conceptnet 5.5: an open multilingual graph of general knowledge. In: Proceedings of the AAAI conference on artificial intelligence (AAAI), pp 4444–4451
12. Boas HC (2017) Computational Resources: FrameNet and Constructicon. In: Dancygier B ed. Cambridge handbooks in language and linguistics. Cambridge University Press, pp 549–573. https://doi.org/10.1017/9781316339732.035
13. Fellbaum C (2012) The encyclopedia of applied linguistics. Wordnet. American Cancer Society, Dordrecht
14. Wilson MD (1988) Mrc psycholinguistic database: machine usable dictionary (version 2.00). Behav Res Methods Instrum Comput 20–1:6–11
15. Sanchez D, Moreno A (2004) Recent advances in artificial intelligence research and development. Creating ontologies from web document. IOS Press, New York
16. Riloff E (1993) Automatically constructing a dictionary for information extraction tasks. In: Proceedings of the 11th national conference on artificial intelligence. AAAI Press, Washington, D.C, pp 811–816
17. Wu S, Hsiao L, Cheng X, Hancock B, Rekatsinas T, Levis P, R C (2018) Fonduer: knowledge base construction from richly formatted data. In: Proceedings of the 2018 international conference on management of data (SIGMOD '18), pp 1301–1316
18. Sa CD, Ratner A et al (2017) Incremental knowledge base construction using deepdive. VLDB J 26:81–105
19. Glauber R, Claro DB (2018) A systematic mapping study on open information extraction. Expert Syst Appl 112:372–387. https://doi.org/10.1016/j.eswa.2018.06.046
20. Noy NF, Shah NH, Whetzel PL, Dai B, Dorf M, Griffith M, Rubin DL, Storey MA, Chute CG (2009) Bioportal: ontologies and integrated data resources at the click of a mouse. Nucleic Acids Res 37:170–173

21. Ah B, Lp B, Lc P, Lc B, Dl S (1996) Taking a bite out of crisp strategies on using and conducting searches in the computer retrieval of information on scientific projects database. Comput Nurs 14–4:218–24
22. Martinez-Rodriguez Jose L, Ivan Lopez-Arevalo ABR-A (2018) Openie-based approach for knowledge graph construction from text. Expert Syst Appl 113:339–355
23. Kollia I, Glimm B, Horrocks I (2011) Sparql query answering over owl ontologies. In: Proceedings of the 8th extended semantic web conference on the semantic web: research and applications (ESWC), vol. part 1, pp 382–396
24. Doing-Harris K, Livnat Y, Meystre S (2015) Automated concept and relationship extraction for the semi-automated ontology management (seam) system. J Biomed Sem 6(1):15
25. Alobaidi M, Malik KM, Sabra S (2018) Linked open data-based framework for automatic biomedical ontology generation. BMC Bioinform 19(1):319
26. Qawasmeh O, Lefrançois M, Zimmermann A, Maret P (2018) Improved categorization of computer-assisted ontology construction systems: focus on bootstrapping capabilities
27. Bast H, Buchhold B, Haussmann E (2016) Semantic search on text and knowledge bases. Found Trends® Inform Retrieval 10:119–271
28. Khanam SA, Youn HY (2016) A web service discovery scheme based on structural and semantic similarity. J Inform Sci Eng 32–1:153–176
29. Jaana K (2005) Ontology as a search-tool: a study of real users' query formulation with and without conceptual support. In: Advances in information retrieval
30. Amato F, Moscato V, Picariello A, Sperlí G (2017) Kira: a system for knowledge-based access to multimedia art collections. In: 2017 IEEE 11th international conference on semantic computing (ICSC), pp 338–343
31. Musen AM, Team P (2015) The protégé project: a look back and a look forward. AI Matters 1–4:4–12
32. Thomas R, Fabian S, Johannes H, Joanna B, Erdal K, Gerhard W (2016) Yago: a multilingual knowledge base from wikipedia, wordnet, and geonames. In: The semantic web–ISWC 2016. Springer, Cham, pp 177–185
33. Jastrzebski S, Bahdanau D, Hosseini S, Noukhovitch M, Bengio Y, Cheung JCK (2018) Commonsense mining as knowledge base completion? A study on the impact of novelty. CoRR arXiv:abs/1804.09259
34. Lenat DB (1995) Cyc: a large-scale investment in knowledge infrastructure. Commun ACM 38(11):33–38
35. Trinh TH, Le QV (2018) A simple method for commonsense reasoning. CoRR arXiv:abs/1806.02847
36. Young T, Cambria E, Chaturvedi I, Zhou H, Biswas S, Huang M (2018) Augmenting end-to-end dialogue systems with commonsense knowledge. AAAI
37. Manning CD, Surdeanu M, Bauer J, Finkel J, Inc P, Bethard SJ, Mcclosky D (2014) The stanford corenlp natural language processing toolkit. In: In ACL, system demonstrations
38. Goldman RS (2018) Structural aspects of constructing meaning from text. In: Kamil PBM, Pearson PD, Barr R eds, M.LHandbook of Reading Research, pp 311–335
39. Al-Zaidy RA, Giles CL (2018) Extracting semantic relations for scholarly knowledge base construction. In: 2018 IEEE 12th international conference on semantic computing (ICSC). Laguna Hills, pp 56–63
40. Upadhyay P, Bindal A, Kumar M, Ramanath M (2018) Construction and applications of teknowbase: a knowledge base of computer science concepts. In: Companion proceedings of the the web conference 2018 (WWW), pp 1023–1030
41. Coronado DS, Haber MW, Sioutos N, Wright LW (2004) Nci thesaurus: using science-based terminology to integrate cancer research results. Medinfo 107:33–37
42. Manning DC, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D (2014) The stanford corenlp natural language processing toolkit. In: Proceedings of the 52nd annual meeting of the association for computational linguistics: system demonstrations, pp 55–60
43. Horridge M, Bechhofer S (2011) The owl api: a Java API for owl ontologies. Semantic Web 2–1:11–21
44. O'Connor MJ, Halaschek-Wiener C, Musen MA (2010) M2: a language for mapping spreadsheets to owl. In: OWLED
45. Bailey RW (2004) The meaning of everything: the story of the Oxford english dictionary (review). In: Kamil PBM, Pearson PD, Barr R, eds. Dictionaries, pp 169–174

Publisher's Note