IEICE TRANS. INF. & SYST., VOL.E101–D, NO.1 JANUARY 2018

PAPER Special Section on Semantic Web and Linked Data

An Automatic Knowledge Graph Creation Framework from Natural Language Text

Natthawut KERTKEIDKACHORN[†a)], *Nonmember and* **Ryutaro ICHISE**[†,††,†††], *Senior Member*

SUMMARY Knowledge graphs (KG) play a crucial role in many modern applications. However, constructing a KG from natural language text is challenging due to the complex structure of the text. Recently, many approaches have been proposed to transform natural language text to triples to obtain KGs. Such approaches have not yet provided efficient results for mapping extracted elements of triples, especially the predicate, to their equivalent elements in a KG. Predicate mapping is essential because it can reduce the heterogeneity of the data and increase the searchability over a KG. In this article, we propose T2KG, an automatic KG creation framework for natural language text, to more effectively map natural language text to predicates. In our framework, a hybrid combination of a rule-based approach and a similarity-based approach is presented for mapping a predicate to its corresponding predicate in a KG. Based on experimental results, the hybrid approach can identify more similar predicate pairs than a baseline method in the predicate mapping task. An experiment on KG creation is also conducted to investigate the performance of the T2KG. The experimental results show that the T2KG also outperforms the baseline in KG creation. Although KG creation is conducted in open domains, in which prior knowledge is not provided, the T2KG still achieves an F1 score of approximately 50% when generating triples in the KG creation task. In addition, an empirical study on knowledge population using various text sources is conducted, and the results indicate the T2KG could be used to obtain knowledge that is not currently available from DBpedia.

key words: *knowledge graph, knowledge discovery, knowledge extraction, linked data*

1. Introduction

A knowledge graph (KG) stores knowledge in the form of a graph, in which a node represents an entity and an edge denotes the relationship between entities. Well-known examples of KGs are DBpedia [1], Freebase [2] and YAGO [3]. These KGs play an important role in applications such as answering queries, browsing knowledge and data visualization. New knowledge is constantly becoming available; however, it is usually in the form of natural language text, which is not straightforward to transfer to a KG. Furthermore, the rate of publication of natural language text is increasing dramatically [4].

Recently, many approaches have been proposed to extract knowledge as a triple (subject, predicate, object) from text [4]–[10]. Although these approaches perform well for extracting triples from text, they still have limitations regarding the mapping of the elements of a triple, especially the predicate, to its corresponding element in a KG. Mapping elements of a triple to its identity in a KG is necessary because it can reduce the heterogeneity of the data and increase the searchability over a KG. Many studies have focused on mapping entities, which are usually the subject or object of triples, to their counterpart in a KG [9], [11], [12]. Mapping the predicate is usually not considered, although a study [10] introduced a procedure for mapping the predicate of a triple extracted from natural language text to an identical predicate in a KG. This method involved a simple rule-based approach. As a result, it could efficiently deal with rule generation due to the sparsity of text.

In this article, we introduce the T2KG: an automatic framework for KG creation from natural language text. In the T2KG, we propose a hybrid approach that combines a rule-based approach and a similarity-based approach for mapping the predicate of a triple extracted from text to its corresponding predicate in an existing KG. In the similarity-based approach, we introduce a novel vector-based similarity metric for computing the similarity between the elements of triples to overcome the sparsity problem. We conduct experiments to evaluate the performance of each stage of the T2KG framework in a KG creation task. In addition, we also conduct an empirical study using the T2KG in a knowledge population task. We note that the difference between the knowledge population task and the knowledge creation task is that in the knowledge population task a given KG is populated with triples, while the KG creation task considers the construction of the KG.

The rest of this paper is organized as follows. Section 2 gives a brief survey of the related work on KG creation. In Sect. 3, the details of our T2KG framework are presented. The experiments and their results are reported in Sect. 4. Finally, this work is concluded in Sect. 5.

2. Related Work

The creation of a KG consists of three stages: 1) knowledge extraction, 2) entity mapping and 3) data integration. Previous approaches for constructing KGs can be roughly divided into three groups depending on which stage they focus on.

The first group of methods focuses on knowledge extraction from natural language text [5]–[7]. NELL [5] is a never-ending system that reads the worldwide web by ex-

Manuscript received February 16, 2017.
Manuscript revised June 14, 2017.
Manuscript publicized September 15, 2017.

[†]The authors are with SOKENDAI (The Graduate University for Advanced Studies), Tokyo, 101–8430 Japan.
[††]The author is with National Institute of Informatics, Tokyo, 101–8430 Japan.
[†††]The author is with National Institute of Advanced Industrial Science and Technology, Tokyo, 100–8921 Japan.
a) E-mail: natthawut@nii.ac.jp

DOI: 10.1587/transinf.2017SWP0006

tracting triples using a bootstrapping approach to learn new constraints. ReVerb [6] and OLLIE [7] are open information systems that extract a triple from a sentence by using syntactic and lexical patterns. Although these approaches can extract triples from natural language text, they do not consider entity mapping. As a result, there may be some ambiguity in the extracted entities.

The second group of approaches also investigate knowledge extraction and entity mapping [4], [8], [9]. In some studies [4], [8], a triple is extracted from natural language text by using natural language processing (NLP) techniques. Then, a triple is stored as a resource description framework (RDF) triple using its own ontology. LODifier [9] uses deep semantic analysis and named entity recognition with a coreference resolution system to acquire a triple, and generates an RDF triple using the WordNet representation without considering other ontologies. Even though these approaches reduce the ambiguity of an extracted entity, they do not integrate all of the elements of a triple into existing KGs.

The third group of methods considers all three stages described above when creating a KG [10]. Exner et al. use a semantic role labeling method with state-of-the-art NLP techniques to extract a triple from Wikipedia and then apply a rule-based approach to integrate the RDF triple into the ontology of the KG [10]. Although this approach can integrate a predicate into the ontology of a KG, due to the sparsity of natural language text, bootstrapping training data to generate rules might not cover all possible patterns, and consequently, some rules may be missing.

To overcome this problem, we introduce T2KG, an automatic KG creation framework for natural language text. In T2KG, a hybrid approach that combines the rule-based approach and the similarity-based approach is introduced, using a vector-based similarity metric to identify corresponding predicates. Some preliminary results for the T2KG have been previously reported [13]. In this article, further details of the framework, experimental settings and the experimental results are described and discussed. In addition, we introduce new experimental results for the T2KG framework in a knowledge population task.

3. Knowledge Graph Creation

In this section, the architecture of the T2KG framework is described. The T2KG takes natural language text as input and produces a KG as output. As shown in Fig. 1, the T2KG consists of five components: 1) entity mapping, 2) coreference resolution, 3) triple extraction, 4) triple integration, and 5) predicate mapping. The entity mapping component links an entity in natural language text to its corresponding entity in the KG. The coreference resolution component detects coreferring chains of entities, in which entities and pronouns in natural language text referring to the same entity are grouped together. The triple extraction component extracts a relation triple from natural language text by using the open information extraction technique. The triple

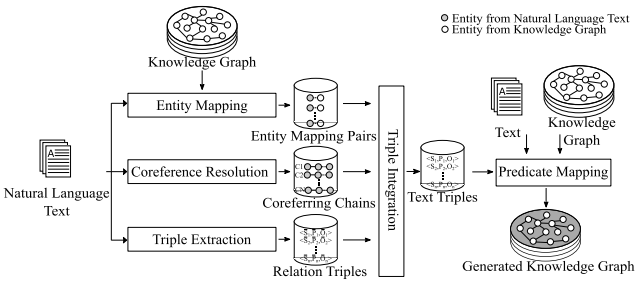


Fig. 1 Architecture of the T2KG framework.

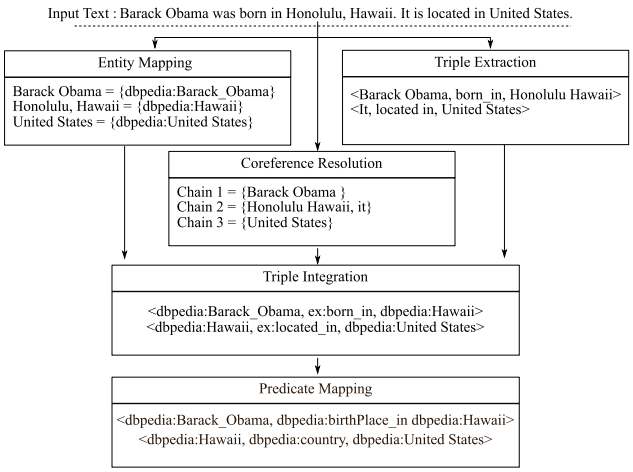


Fig. 2 Example of data flow in the T2KG framework.

integration component generates a text triple by integrating the results from the entity mapping component, the coreference resolution component and the triple extraction component. The predicate mapping component maps a predicate of a text triple to a predefined predicate in other KGs. The details of each component are presented below.

3.1 Entity Mapping

The aim of the entity mapping component is to map an entity in natural language text to a uniform resource identifier (URI). In the entity mapping component, entities are recognized from natural language text to create a set of extracted entities. If an extracted entity can be mapped to an identical entity in a KG, the URI of such an entity in that KG can be used as a representative for the extracted entity. Otherwise, a new URI is assigned to the entity. For example, consider "dbepdia:United_States" as a URI in the KG. If the entity "United States" in natural language text is mapped to "dbepdia:United_States", the same URI is used. On the other hand, if the entity does not exist in the KG, a new URI, e.g., "ex:Barron_Trump", is assigned to the entity "Barron Trump".

To further illustrate the flow of the T2KG framework, an example is given in Fig. 2. In Fig. 2, the example sentence is "Barack Obama was born in Honolulu, Hawaii. It is located in the United States.", the expected results of the entity

mapping component are a set of mapping entities for each entity, e.g., *Barack Obama* = {*dbpedia*: *Barack_Obama*}.

3.2 Coreference Resolution

The aim of the coreference resolution component is to detect coreferring chains of entities in natural language text and to group them together. This is an essential component because natural language text usually contains abbreviations, pronouns, and different expressions that refer to the same entities. With the coreference resolution component, an entity and its different expressions can be grouped so that actions for identical entities using different expressions can be identified. To discover chains of coreferring entities, a coreference resolver for the coreference resolution task is used [14], [15]. An example is shown in Fig. 2. The expected results of the coreference resolution component are coreferring chains of entities. Based on the input in the example, the coreferring chain is $C2$ = {*Honolulu Hawaii*, *it*}.

3.3 Triple Extraction

The aim of the triple extraction component is to extract relation triples from natural language text. This is a key step to acquire knowledge from natural language text. According to linguistic theory [16], the meaning of an arbitrary sentence can be interpreted by considering a set of relations and its associated arguments. Consequently, a relation triple is defined as a triple describing a relation and its associated arguments in an arbitrary sentence. In our scenario, the relation is a predicate of a triple and its associated arguments are the subject of a triple and the object of a triple.

To extract a relation triple from natural language text, any open information extraction technique can be used. Open information extraction techniques are used to extract information from an arbitrary sentence using pattern templates, and then to convert such information into a relation triple. For example, as depicted in Fig. 2, the example of the relation triple from the triple extraction component is <*Barack Obama*, *born in*, *Honolulu Hawaii*>, where "born in" is a relation, which is the predicate of the triple, and "Barack Obama" and "Honolulu Hawaii" are its arguments, which are the subject and the object of the triple, respectively.

3.4 Triple Integration

The aim of the triple integration component is to generate text triples using outputs from the entity mapping component, the coreference resolution component and the triple extraction component.

In the triple extraction component, we can extract relation triples from natural language text. However, entity mapping and coreference resolution among the entities of such triples are not performed. As a result, ambiguity in the triple occurs and links among entities in the KG are not established. Consequently, the relation triple must be transformed to conform to the standards of the KG. This is achieved by integrating the results from three components and transforming them with the following processes.

First, identical entities are grouped using coreferring chains from the coreference resolution component. Second, a representative for the group of coreferring entities is selected by the voting algorithm. Because entities in the same group might have multiple representations, the most commonly occurring representation, after excluding pronouns, for the group is chosen as the group representative. Third, all entities belonging to the group in the relation triples are replaced by the representative of their groups. Fourth, the relation of a relation triple is straightforwardly transformed into a predicate by assigning a new URI. Finally, if an object of a relation triple is not an entity, it is left as literal. After performing these processes, text triples are extracted from natural language text.

Figure 2 shows an example of this component. The triple integration component generates the text triple, e.g., <*dbpedia*:*Barack_Obama*, *ex*: *born_in*, *dbpedia*:*Hawai*>. However, the predicate of the triple, *ex*: *born_in*, is not mapped to any predicate in the KG in this component.

3.5 Predicate Mapping

The aim of the predicate mapping component is to map a predicate of a text triple onto an identical predicate in the KG. Previous studies mainly focus on predicate linking between KG triples [17]–[19]. This mapping cannot be applied in a straightforward manner for our task, since we aim to map the predicate of a text triple extracted from a sentence to its identical predicate in a KG triple. Although there are many studies on distance supervised learning for identifying relationships between entities in sentences [7], [20], they do not directly consider the mapping between a text predicate and a KG predicate. The most relevant study for our task is that by Exner and Nugues [10].

In the above study [10], a rule-based approach was proposed for mapping the predicate of a triple onto an identical predicate in a KG. However, the results depended on the generated rules. Because of the sparsity of natural language text in open domains, generated rules cannot cover all possible patterns. As a result, their approach is not sufficiently general for discovering new rules that have not previously appeared. Therefore, reasonable recall cannot be realized. To deal with heterogeneous vocabularies and to alleviate the sparsity of natural language text, a hybrid combination of a rule-based approach and a similarity-based approach using the vector-based similarity metric is proposed in the present study.

In our hybrid approach, rules for mapping a predicate in a similar way to [10] are learned, and then a similarity-based approach using the vector-based similarity metric is applied for unseen rules to determine identical predicates, as depicted in Fig. 3. First, text triples are enriched by the triple enrichment module. This module enriches a text triple and a KG triple by their data types and classes, and then in-

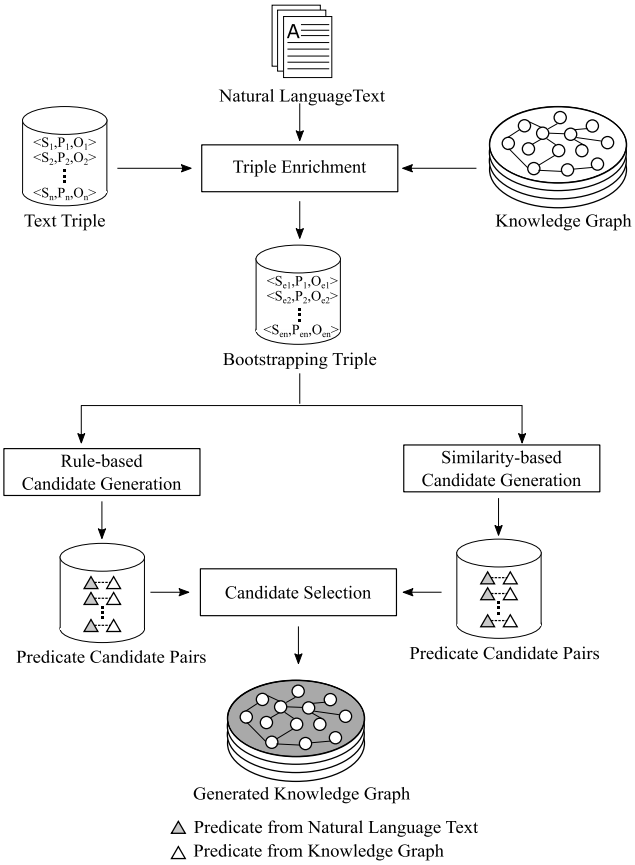


Fig. 3 Diagram of the predicate mapping component.

tegrates and normalizes the text triples, the KG triples and the enriched triples to create bootstrapped triples for use in the later modules. Second, the rule-based candidate generation module then uses the bootstrapped triples to create rules and then generates predicate-candidate pairs for the text predicate. Third, the similarity-based candidate generation module uses the bootstrapped triples to embed the elements of triples as vector representations, and then these vectors are used to compute the similarity between a text predicate and a KG predicate to generate predicate candidate pairs. Finally, the candidate selection module is used to select the most suitable mapping candidate. A candidate is then selected, completing the KG creation process. An example of this component is shown in Fig. 2. As can be seen, *ex*: *born_in* is mapped to *dbpedia*: *birthPlace*, and the triple <*dbpedia*: *Barack_Obama*, *dbpedia*: *birthPlace*, *dbpedia*: *Hawaii*> is assigned to the generated KG. The details of each module are as follows.

3.5.1 Triple Enrichment

The triple enrichment module enriches text triples and KG triples and then integrates them to obtain the bootstrapped triples for the later modules. Text triples and KG triples are enriched with their classes and data types. The enrichment process is performed only on the subject (domain) and object (range) elements of a triple.

To enrich a triple, the subject and the object of the triple are bound to their corresponding class. For KG triples and text triples, whose subject or object is mapped to a KG entity, the subject and the object of the triple are bound using the vocabulary rdf:type. For example, given DBpedia as the KG, and <*dbpedia*: *Barack_Obama*, *dbpedia*: *birthPlace*, *dbpedia*: *Hawaii*> as the triple, the enriched result is <*dbpedia*: *Person*, *dbpedia*: *birthPlace*, *dbpedia*: *Location*>. For the text triples, whose subject or object cannot be mapped to a KG entity, the name entity recognition (NER) system is used to retrieve the class of the subject and the object of the triple. Then, the class is mapped to the KG class using string matching as a workaround. For example, the NER class, *Person*, is mapped to the KG class, *dbpedia*: *Person*.

As well as the class of the entities, the data type is also considered. In the T2KG, we use the URI, string, number and date as data types. The data types of the subject and object of the triple are converted using a simple parser. If a subject or an object of a triple can parse the date, the date type is used. If a subject or an object of a triple contain only a number, the number type is used. If a subject or an object of a triple are a URI, the URI type is used. Otherwise, the string type is used. For example, given the triple <*dbpedia*: *Barack_Obama*, *ex*: *born_in*, *dbpedia*: *Hawaii*>, the result is <*URI*, *ex*: *born_in*, *URI*>. All of the generated triples, called bootstrapped triples, are used as the output of this module.

3.5.2 Rule-Based Candidate Generation

The rule-based candidate generation module extracts rules and uses them to determine predicate-candidate pairs. In this module, a similar strategy as in [10] is implemented to create the following rules for mapping the predicate. First, if the subject and the object of the text triple are similar to the subject and the object of the KG triple, respectively, it is assumed that the predicate of the text triple and the predicate of the KG triple are equivalent. Note that, if the predicate of the text triple matches many predicates of the KG, the predicate of the KG that appears the most often with the text predicate is selected. Second, the class of the subject and the class of the object are used as constraints in the mapping. For example, <*Person*, *ex*: *born_in*, *Location*> is mapped to *dbpedia*: *birthPlace*, when using DBpedia as the KG. Even though this approach uses bootstrapped triples to generate reliable rules, the number of rules is very limited due to the small number of bootstrapped triples and the sparsity of text, and thus, some rules are missing. To avoid such problems, a similarity-based approach using the vector-based similarity metric is applied in the T2KG framework.

3.5.3 Similarity-Based Candidate Generation

The similarity-based candidate generation module gener-

ates predicate candidate pairs based on the similarity between predicates. Generally, a string-based similarity metric is used for the entity mapping or the predicate mapping task [18], [21], [22]. Due to the heterogeneous vocabularies in the open information extraction task, the string-based similarity metric can fail to correctly represent the similarity between predicates. To cope with heterogeneous vocabularies, each vocabulary should be learned and represented at a deeper level than just their textual string. We therefore propose a novel vector-based similarity metric for computing the similarity between elements of triples.

Mikolov et al. proposed vector representations of words that can capture both syntactic and semantic patterns in [23]. Inspired by this vector representation of words, we present elements of triples in the vector space using other elements in the same triple. The idea is that elements of triples that have a similar context should be embedded more closely with each other in the vector space than dissimilar elements. The objective function is then formulated as follows.

$$L(\theta) = \arg\max_{\theta} \sum_{(e,c)\in BT} \Big(\log \sigma(\bar{v}_c \cdot v_e) + \sum_{(neg,c)\in BT'_{(e,c)}} \log \sigma(-\bar{v}_c \cdot v_{neg}) \Big) \quad (1)$$

where $\sigma(x) = 1/(1+\exp(-x))$, e is an element of a triple, c is another element of the same triple, BT is a set of bootstrapped triples from the triple enrichment module, $BT'_{(e,c)}$ are randomly generated negative triples of the element of the triple e in the context c, which are not contained in BT, neg is a negative example of the element of a triple in the context c, $\bar{v}_c$ is an average of the vector representations of words in the context c, $v_e, v_c, v_{neg} \in \theta$ and v_e, v_c and v_{neg} are vector representations of elements of triples e, c and neg respectively.

After acquiring a vector representation for each element of the triples, the similarity between a predicate of a text triple and a predicate of a KG triple is computed to generate a list of predicate candidate pairs, ranked by their similarity score. In our approach, the similarity score is defined as follows.

$$Sim(P_{\hat{T}}, P_{KG}) = \delta\left(\frac{\overrightarrow{P_{\hat{T}}} \cdot \overrightarrow{P_{KG}}}{|\overrightarrow{P_{\hat{T}}}||\overrightarrow{P_{KG}}|}\right) + (1-\delta)\left(\frac{context(P_{KG}) \cdot (\overrightarrow{S_{\hat{T}}} - \overrightarrow{O_{\hat{T}}})}{|context(P_{KG})||\overrightarrow{S_{\hat{T}}} - \overrightarrow{O_{\hat{T}}}|}\right) \quad (2)$$

$$context(P_{KG}) = \frac{\sum_{n=1}^{N} (\overrightarrow{S_{P_{KG_n}}} - \overrightarrow{O_{P_{KG_n}}})}{N} \quad (3)$$

where $S_{\hat{T}}$, $P_{\hat{T}}$, $O_{\hat{T}}$ are the subject, the predicate and the object of the triple $\hat{T}$, respectively, $\hat{T}$ is a text triple, P_{KG} is a predicate in KG, $S_{P_{KG_n}}$ and $O_{P_{KG_n}}$ are the n^{th} pairs of subjects and objects, respectively, corresponding to the predicate P_{KG} in the KG ($<S_{P_{KG_n}}, P_{KG}, O_{P_{KG_n}}> \in KG$), N is the number of triples in the KG, whose predicates are P_{KG}, and δ is the weight parameter between the predicate similarity and the context similarity. The basis for these scores is that the similarity between predicates can be measured directly by the cosine similarity of the vector, as reflected in the first term of Eq. (2). However, the predicate may vary depending on the context. Consequently, in the second term of Eq. (2), the similarity between contexts is also computed to validate the suitability of the predicate with its context. This assumption is based on the fact that the more suitable the context, the more likely the predicates can be mapped. Because the first and the second terms in Eq. (2) are different, the weight parameter is introduced to adjust the salient aspect between the predicate similarity and the context similarity. Equation (3) is proposed to compute the average vector representation of the context of P_{KG}.

3.5.4 Candidate Selection

The candidate selection module selects the mapping for the predicate of the text triple. In this module, priority is given to the predicate-candidate pair that is generated by the rule-based candidate generation module. If such a predicate-candidate pair does not exist, the predicate-candidate pair generated by the similarity-based candidate generation module is then considered. If the similarity of the predicate-candidate pair is greater than the threshold θ, the predicate pair is mapped to the candidate. Otherwise, the new URI of the text triple, e.g., $ex:\ born\_in$, is assigned as the predicate. The output of the candidate selection module is the generated KG, in which some entities and predicates are linked to other KGs.

4. Experiment

4.1 Experimental Setup

Three experiments are designed to evaluate: 1) the performance of the hybrid approach in the T2KG framework, 2) the overall performance of the T2KG framework for the KG creation task and 3) the performance of the T2KG when populating an existing KG with new knowledge.

In the T2KG, each component is implemented and its parameters are configured as follows. In the entity mapping component, DBpedia Spotlight [12] is used to map entities. If an entity cannot be mapped to any of the DBpedia entities, a new namespace "ex:" is adopted as the prefix of the entity to create a new URI. This namespace can also be applied to an unmapped predicate. In the coreference resolution component, the Stanford NLP tool [14], [15] is used as the coreference resolver. In the triple extraction component, OLLIE [7], a state-of-the-art tool for open information extraction, is applied to extract relation triples from natural language text. In the triple enrichment module, the Stanford NLP tool is also used as the NER system. In the similarity-based candidate generation module, word2vec[†] is used for

[†]https://code.google.com/archive/p/word2vec/

the training vector representations of the elements of triples. The default parameter settings for the word2vec are used. To collect the corpus for creating vector representations of the elements of triples, text triples and KG triples are constructed. To gather the text triples, 120,000 Wikipedia articles are randomly selected and then the pre-processing step is applied. In the pre-processing step HTML markups, wiki marks and hyperlink annotations are removed. Duplicates of sentences are discarded. After that the pre-processed sentences are passed to the T2KG framework to create text triples. For KG triples, the whole DBpedia [1] is used.

In the similarity-based candidate generation module, there are two hyperparameters: the weight δ and the threshold θ. To optimize these two parameters, we automatically create identical predicate pairs for the training data using the matching strategy for predicates of text triples and the predicates of the KG. The matching strategy for constructing the training data is conducted as follows. If the subject of the text triple and the subject of the DBpedia triple are the same and the object of the text triple and the object of the DBpedia triple are the same, we assume that the predicate of the text triple and the predicate of the DBpedia triple are the same. Figure 4 shows an example of the matching between the predicate of a text triple and the predicate of a KG triple for generating the training data. As shown in this figure, the subject and the object of the text triple and the DBpedia triple are equivalent. Consequently, the predicates, "ex:born_in" and "dbpedia:birthPlace", are assumed to be equivalent. In the training data construction, the targets of the mapping predicate are 2,800 DBpedia ontology properties. Although this method can be used to find many of the matching pairs, it is possible that the validity of these pairs may be uncertain due to multiple matches. Multiple matches occur when two or more predicates share the same subject-object pair. For example, "ex:bear_in" might be mapped to both "dbpedia:birthPlace" and "dbpedia:deathPlace" if the same person (subject) was born and died in the same place (object). Consequently, multiple matching can lead to ambiguity in the dataset. To alleviate this problem, we simply remove text triples when a predicate has multiple matches. According to the above matching strategy, the number of mapped predicate pairs remaining is approximately 43,800. After that, we use the grid search algorithm to find suitable δ and θ parameters from the training dataset. The interval for the parameter search is [0.00, 1.00], and the step size is 0.01. The hyperparameters that performed best in the training data are used in the experiment.

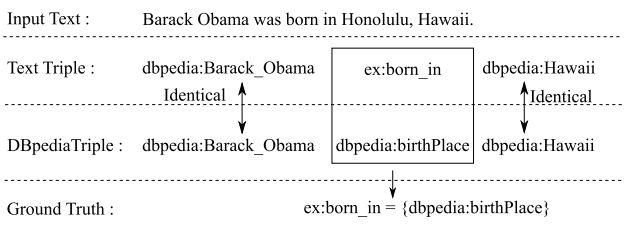


Fig. 4 Example of the matching strategy between a text predicate and a KG predicate.

4.2 Experiment 1

The aim of this experiment is to evaluate the performance of our hybrid approach for the predicate mapping task. To investigate the accuracy of our approach, the rule-based approach from [10] is used as the baseline for comparison.

In the experiment, we manually create the benchmark dataset by randomly selecting 300 text triples and then asking an expert to create links between them and the DBpedia predicates. The dataset is available for download from [†].

In this experiment, given a predicate text, the algorithms return the DBpedia predicate having the highest similarity according to their respective methods. The micro/macro precision, recall and F1 score are then used to measure whether the DBpedia predicate and the predicate text are correctly matched or not. The macro evaluation averages the performance for each predicate type across the dataset, while the micro evaluation aggregates the performance of all predicate types in the dataset.

Table 1 shows the results of the rule-based approach compared with our hybrid approach. The experimental results indicate that the hybrid approach can improve the recall by 10.60%, 9.67% and F score by 4.72%, 4.41% in the macro/micro evaluation respectively. The recall results show that the discoverability of the hybrid approach is higher than the baseline, confirming that the hybrid approach including the similarity-based approach performs better at correctly matching predicates.

To further investigate the performance of the hybrid approach, we illustrate the failure of the rule-based approach. Although a text predicate is observed during the rule construction process, the subject-object pairs of the predicate do not cover all of the observed combinations. The rules for mapping some predicates of a text triple are missing. For example, given the text triple <dbpedia:Granai_airstrike, occurs in, dbpeida: Granai>, the required rule of this triple is "dbpedia:Event, occur_in, dbpedia:Thing". Although there are some learned rules for the text predicate "occur_in", e.g. the rule "dbpedia:Event, occur_in, dbpedia:Place", they are not an exact match with the required rule of this triple. The rule-based approach could not deal with text triples that do not match any rules. In contrast, the similarity-based approach in the hybrid approach allows the direct computation of the similarity between the representations of predicates. Based upon the above example, the text predicate "occur

Table 1 The results of our approach in the predicate mapping task on the benchmark dataset comparing with the baseline (Experiment 1).

Approach	Macro			Micro		
	Precision	Recall	F1	Precision	Recall	F1
Rule-based [10]	**0.7217**	0.5600	0.6306	**0.7693**	0.6400	0.6987
Our approach	0.6902	**0.6660**	**0.6778**	0.7491	**0.7367**	**0.7428**

[†]https://ri-www.nii.ac.jp/VSim/dataset.zip

in" can be mapped to dbpedia:place[†]. The hybrid approach therefore can alleviate the problems caused by the limitations of pattern-matching in the rule-based approach.

4.3 Experiment 2

The aim of this experiment is to evaluate the performance of the T2KG framework in the text-based KG creation task. Unlike Experiment 1, this experiment aims to evaluate the correctness of the results at the triple level. Because no gold standard exists for evaluating the results of generating triples from text, we conduct the evaluation by manually establishing a small set of triples from the text to be used as the gold standard. To create this gold standard, we randomly select 100 sentences from Wikipedia articles and then manually extract and map triples to the DBpedia triples. Note that to fairly evaluate the performance of the T2KG framework, the 120,000 Wikipedia articles for training the similarity-based candidate generation module are excluded from the random selection process.

To evaluate the results, 100 sentences are input to the T2KG framework. The precision, recall and F1 scores are then used as the evaluation metrics to evaluate the accuracy of the triples. Also, we measure the number of mapped triples to show the mapping ability of the framework. In this experiment, the T2KG framework without the similarity-based candidate generation module is used as the baseline. This baseline is similar to that used by Exner and Nugues, [10].

The results are listed in Table 2, and show the amount of discovered knowledge that can be integrated into the existing KG. It can be seen that the T2KG framework performs better than the baseline. Furthermore, the precision, recall and F1 score for the T2KG framework are also higher than those for the baseline.

The errors arising in each component of the system are also investigated. There are four main error sources in the framework: entity mapping, coreference resolution, triple extraction and predicate mapping. We therefore calculate the proportion of errors arising from each of these four components. The results show that 35.21% of the errors are caused by triple extraction, 23.00% by predicate mapping, 21.60% by coreference resolution and 20.19% by entity mapping. The reason the largest source of errors is triple extraction is because the task in this study is an open domain task, in which no schema or prior knowledge is provided. The errors in triple extraction mostly occur when extracting triples from a complex sentence, where a relation and its arguments are not clearly identified.

Errors in the elements of the generated triples are also inspected. We find that the largest number of errors is 38.18% caused by predicates, 36.97% by objects and 24.85% by subjects. The reason predicates resulted in so many errors is that the triple extraction step cannot perfectly extract predicates from natural language text due to the complexity of the text in open domains. Nonetheless, although the KG creation in our study is conducted in open domains, the T2KG system still reproduces approximately 50% of the quality and quantity of generated triples required for creating the KG.

Table 2 Performance of T2KG framework for constructing KG comparing with the baseline (Experiment 2).

Approach	Precision	Recall	F1	# Mapped Triples
Baseline	0.4444	0.5231	0.4806	135
T2KG	**0.4620**	**0.5615**	**0.5069**	**140**

[†]http://dbpedia.org/ontology/place

4.4 Experiment 3

The aim of this experiment is to empirically investigate the performance of the T2KG framework for populating an existing KG with new knowledge extracted from text. In this experiment, two datasets, the gold standard and an online article, are used. The gold standard dataset contains 100 sentences, which are randomly selected from Wikipedia. This dataset is the same as that used in Experiment 2. The online dataset contains articles obtained from the Internet. To create this dataset, we randomly trawl articles on websites in various domains including news, movies, books and travel. Both datasets are then passed to the T2KG method to create the KG. Since this experiment investigates population of an existing KG with knowledge from text, the existing KG is used as the scope for this process. DBpedia is used as the existing KG. Consequently, only triples whose subject and predicate can be mapped to DBpedia, (mapped triples) are considered. After acquiring the mapped triples, the correctness of these triples is manually checked to evaluate their quality, and the number of new knowledge items (not originally in DBpedia) is measured. To evaluate the results, we report the summary statistics for the dataset, the number of triples extracted from the text, the number of extracted triples that could be mapped to DBpedia, the number of correctly mapped triples, and the number of new knowledge items. In this experiment DBpedia 3.9 (2016/04) is used.

The experimental results are presented in Table 3, which shows that the T2KG framework can successfully populate DBpedia with extracted triples. Although some of this information is already contained in DBpedia, new information is also obtained with our framework. Furthermore, based on our observations, the results for the online dataset contain more unmapped triples than the gold standard dataset. This occurs because many entities in the online dataset cannot be mapped to DBpedia. For example, some entities are from movies which were released after the version of DBpedia used in this experiment. As a result, many triples and new data are discarded.

5. Conclusion

This paper presents the T2KG, an automatic KG creation framework for natural language text. In the T2KG, a hybrid

Table 3 The result of the knowledge population of DBpedia.

Dataset	# Articles	# Sentences	# Triples	# Mapped Triples	# Correct Triples	# New Knowledge
Gold Standard	-	100	222	140	62	34
Online Articles	60	1624	3605	499	116	76

approach for mapping predicates is introduced using a novel vector-based similarity metric. The experimental results indicate that the hybrid approach improves recall significantly for mapping a predicate to a KG. Furthermore, the experimental results demonstrate that the T2KG framework can successfully generate a KG from natural language text. Although KG creation in this study is conducted in open domains, the T2KG system still reproduces 50% of the available information in terms of both the quality and quantity of triples generated for the KG. In addition, a study on knowledge population, and the results indicate that T2KG can successfully populate DBpedia with new knowledge from text.

Based on error analyses, the main pitfall of the framework is the triple extraction component when applied to an open information extraction system. This component not only degrades the performance across the whole framework but also introduces errors in the predicate mapping task. In the predicate mapping task, both the rule-based approach and the hybrid approach perform poorly when applied to open information to extract predicates containing many composite words. For example, the text triple <dbpedia: Gustav_Klimt, be_an_important_influence_on dbpedia: Egon_Schiele>. In this example, our method cannot find the rule and the representation of the predicate for computing the similarity due to many composite words in the text predicate. Therefore, both approaches fail to map such complex text triples. Fortunately, open information extraction is still an active area of research. In the future, we aim to improve the implementation of the T2KG framework using a later version of the open information extraction system.

Acknowledgements

This work was partially supported by NEDO (New Energy and Industrial Technology Development Organization).

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," The Semantic Web, Lecture Notes in Computer Science, vol.4825, pp.722–735, Springer, Berlin, Heidelberg, 2007.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," Proc. 2008 International Conference on Management of Data, pp.1247–1250, ACM, 2008.

[3] J. Hoffart, F.M. Suchanek, K. Berberich, and G. Weikum, "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia," Artificial Intelligence, vol.194, pp.28–61, 2013.

[4] V. Kríž, B. Hladká, M. Nečaskỳ, and T. Knap, "Data extraction using NLP techniques and its transformation to linked data," Proc. 13th Mexican International Conference on Artificial Intelligence, Lecture Notes in Computer Science, vol.8856, pp.113–124, Springer, Cham, 2014.

[5] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr., and T.M. Mitchell, "Toward an architecture for never-ending language learning.," Proc. Twenty-Fourth AAAI Conference on Artificial Intelligence, pp.1306–1313, 2010.

[6] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," Proc. Conference on Empirical Methods in Natural Language Processing, pp.1535–1545, ACL, 2011.

[7] M. Schmitz, R. Bart, S. Soderland, and O. Etzioni, "Open language learning for information extraction," Proc. Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp.523–534, ACL, 2012.

[8] R. Cattoni, F. Corcoglioniti, C. Girardi, B. Magnini, L. Serafini, and R. Zanoli, "The KnowledgeStore: An entity-based storage system," Proc. Eighth International Conference on Language Resources and Evaluation, pp.3639–3646, 2012.

[9] I. Augenstein, S. Padó, and S. Rudolph, "LODifier: Generating linked data from unstructured text," The Semantic Web: Research and Applications, Lecture Notes in Computer Science, vol.7295, pp.210–224, Springer, Berlin, Heidelberg, 2012.

[10] P. Exner and P. Nugues, "Entity extraction: From unstructured text to DBpedia RDF triples," Proc. Web of Linked Entities Workshop, pp.58–69, CEUR-WS, 2012.

[11] L. Ratinov, D. Roth, D. Downey, and M. Anderson, "Local and global algorithms for disambiguation to Wikipedia," Proc. 49th Annual Meeting of the Association for Computational Linguistics, pp.1375–1384, 2011.

[12] P.N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "DBpedia spotlight: Shedding light on the web of documents," Proc. 7th International Conference on Semantic Systems, pp.1–8, ACM, 2011.

[13] N. Kertkeidkachorn and R. Ichise, "T2KG: An end-to-end system for creating knowledge graph from unstructured text," Proc. AAAI-17 Workshop on Knowledge-Based Techniques for Problem Solving and Reasoning, pp.743–749, 2017.

[14] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky, "Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task," Proc. 15th Conference on Computational Natural Language Learning: Shared Task, pp.28–34, ACL, 2011.

[15] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning, "A multi-pass sieve for coreference resolution," Proc. Conference on Empirical Methods in Natural Language Processing, pp.492–501, ACL, 2010.

[16] C.J. Fillmore, "Frame semantics and the nature of language," Annals of the New York Academy of Sciences, vol.280, no.1, pp.20–32, 1976.

[17] Z. Abedjan and F. Naumann, "Synonym analysis for predicate expansion," Extended Semantic Web Conference, Lecture Notes in Computer Science, vol.7882, pp.140–154, Springer, Berlin, Heidelberg, 2013.

[18] L. Zhao and R. Ichise, "Ontology integration for linked data," Journal on Data Semantics, vol.3, no.4, pp.237–254, 2014.

[19] Z. Zhang, A.L. Gentile, E. Blomqvist, I. Augenstein, and F. Ciravegna, "Statistical knowledge patterns: Identifying synonymous relations in large linked datasets," International Semantic Web Conference, Lecture Notes in Computer Science, vol.8218, pp.703–719, Springer, Berlin, Heidelberg, 2013.

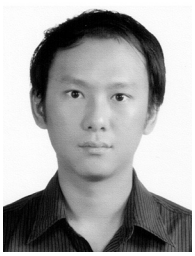
[20] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervi-

sion for relation extraction without labeled data," Proc. Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2, ACL '09, Stroudsburg, PA, USA, pp.1003–1011, ACL, 2009.

[21] N. Kertkeidkachorn, R. Ichise, A. Suchato, and P. Punyabukkana, "An automatic instance expansion framework for mapping instances to linked data resources," Proc. Joint International Semantic Technology Conference, pp.380–395, 2013.

[22] D. Gerber, S. Hellmann, L. Bühmann, T. Soru, R. Usbeck, and A.-C.N. Ngomo, "Real-time RDF extraction from unstructured data streams," The Semantic Web Conference 2013, Lecture Notes in Computer Science, vol.8218, pp.135–150, Springer, Berlin, Heidelberg, 2013.

[23] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," Proc. Advances in Neural Information Processing Systems, pp.3111–3119, 2013.



Natthawut Kertkeidkachorn received B.Eng. and M.Eng. degrees in computer engineering from Chulalongkorn University, Bangkok, Thailand in 2011 and 2013 respectively. He is currently a Ph.D. candidate at Sokendai (The Graduate University for Advanced Studies) in Japan. His research interests include the Semantic Web, machine learning and natural language processing.

Ryutaro Ichise received a Ph.D. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2000. From 2001 to 2002, he was a visiting scholar at Stanford University. He is currently an associate professor in the Principles of Informatics Research Division at the National Institute of Informatics in Japan. His research interests include the Semantic Web, machine learning, and data mining.