The AAAI-17 Workshop on
Knowledge-Based Techniques for Problem Solving and Reasoning
WS-17-12

T2KG: An End-to-End System for Creating Knowledge Graph from Unstructured Text

Natthawut Kertkeidkachorn,[1,2] **Ryutaro Ichise**[1,2,3]
[1]Department of Informatics, Sokendai (The Graduate University for Advanced Studies)
[2]National Institute of Informatics, Tokyo, Japan
[3]National Institute of Advanced Industrial Science and Technology,Tokyo, Japan
natthawut@nii.ac.jp, ichise@nii.ac.jp

Abstract

Knowledge Graph (KG) plays a crucial role in many modern applications. Nevertheless, constructing KG from unstructured text is a challenging problem due to its nature. Consequently, many approaches propose to transform unstructured text to structured text in order to create a KG. Such approaches cannot yet provide reasonable results for mapping an extracted predicate to its identical predicate in another KG. Predicate mapping is an essential procedure because it can reduce the heterogeneity problem and increase searchability over a KG. In this paper, we propose T2KG system, an end-to-end system with keeping such problem into consideration. In the system, a hybrid combination of a rule-based approach and a similarity-based approach is presented for mapping a predicate to its identical predicate in a KG. Based on preliminary experimental results, the hybrid approach improves the recall by 10.02% and the F-measure by 6.56% without reducing the precision in the predicate mapping task. Furthermore, although the KG creation is conducted in open domains, the system still achieves approximately 50% of F-measure for generating triples in the KG creation task.

Introduction

A knowledge graph (KG) is a graph-structured knowledge base that stores knowledge in the form of the relation between entities. An example KG is DBpedia(Auer et al. 2007). The KG plays an important role in various applications, e.g., question answering, browsing knowledge and data visualization. However, most of the published data is unstructured data and the trend of publishing such data is dramatically growing faster than publishing structured data (Kriz et al. 2014). Consequently, a large amount of data cannot be straightforwardly transformed into a KG and so is left as unstructured data.

Recently, many approaches have proposed transforming unstructured text to structured text in order to create a KG (Carlson et al. 2010; Fader, Soderland, and Etzioni 2011; Schmitz et al. 2012; Cattoni et al. 2012; Augenstein, Pado, and Rudolph 2012; Kriz et al. 2014; Exner and Nugues 2012). Although those studies perform well for extracting triples (subject, predicate, object) from unstructured text, they still have a limitation regarding mapping a predicate of a triple extracted from unstructured text to its identical predicate in the KG. Generally, many studies (Augenstein, Pado, and Rudolph 2012; Ratinov et al. 2011; Mendes et al. 2011) focus on mapping only an entity, which is usually a subject or an object of a triple, to its identical entity in a KG. Mapping a whole predicate to its identical predicate is usually ignored. Mapping a predicate to its identical predicate in a KG is an essential procedure because it can reduce the heterogeneity problem and increase the searchability over a KG. Although one study (Exner and Nugues 2012) introduced mapping a predicate of a triple extracted from unstructured text to an identical predicate in a KG, the approach uses the simple rule-based approach. As a result, it cannot efficiently deal with the limitation of rule generation due to the sparsity of unstructured text.

In this paper, we introduce T2KG: an end-to-end system for creating a KG from unstructured text. In T2KG, we propose a hybrid approach that combines a rule-based approach and a similarity-based approach for mapping a predicate of a triple extracted from unstructured text to its identical predicate in an existing KG. The existing KG is used as control knowledge when creating a new KG. In the similarity-based approach, we present a novel vector-based similarity metric for computing the similarity between the elements of triples to overcome the sparsity problem.

The rest of this paper is organized as follows. Section 2 gives a brief survey and related work regarding KG creation. In Section 3, the detail of T2KG is presented. Experiments and results are conducted in Section 4. Eventually, this work is concluded in section 5.

Related Work

KG construction generally considers the following three tasks: 1) knowledge extraction, 2) entity mapping and 3) data integration. Based on these tasks, previous approaches can be roughly divided into three groups.

The first group (Carlson et al. 2010; Fader, Soderland, and Etzioni 2011; Schmitz et al. 2012) mainly focuses on knowledge extraction from unstructured text. NELL (Carlson et al. 2010) is a never-ending system that learns to read the web. To extract triples in NELL, bootstrap constraints are used to learn new constraints. ReVerb (Fader, Soderland, and Etzioni 2011) and OLLIE (Schmitz et al. 2012) are open information systems that extract a triple from a sentence by using

syntactic and lexical patterns. Although these approaches successfully extract triples from unstructured text, they still do not consider entity mapping. As a result, ambiguity of an extracted entity might occur.

The second group (Cattoni et al. 2012; Augenstein, Pado, and Rudolph 2012; Kriz et al. 2014) also investigates knowledge extraction and entity mapping. In some studies (Cattoni et al. 2012; Kriz et al. 2014), a triple is extracted from unstructured text by using Natural Language Processing (NLP) techniques. Then, a triple is stored as an RDF triple by using its own ontology. LODifier (Augenstein, Pado, and Rudolph 2012) uses a deep semantic analysis system and a named entity recognition system with a coreference resolution system to acquire a triple and generates an RDF triple by using WordNet representation without considering other ontologies. Even though these approaches can resolve the ambiguity of an extracted entity, all elements of a triple are not yet integrated into other KGs.

The third group (Exner and Nugues 2012) considers all aspects for creating a KG. Exner et al. use a semantic role labeling method with a state-of-the-art NLP to extract a triple from Wikipedia and then applies a rule-based approach to integrate an RDF triple into the ontology of the KG (Exner and Nugues 2012). Although this approach (Exner and Nugues 2012) can integrate a predicate into the ontology of a KG, it still has the following severe limitation. Due to the sparsity of unstructured text, bootstrapping training data for generating rules might not cover all possible patterns; consequently, some rules are missing. To overcome this problem, we introduce the hybrid approach that combines the rule-based approach and the similarity-based approach by using a vector-based similarity metric to identify the same predicate in the T2KG system.

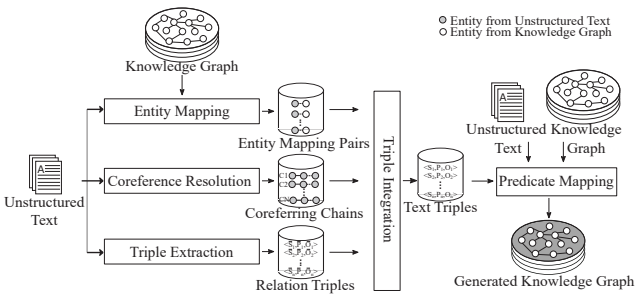

Figure 1: Architecture of the T2KG system

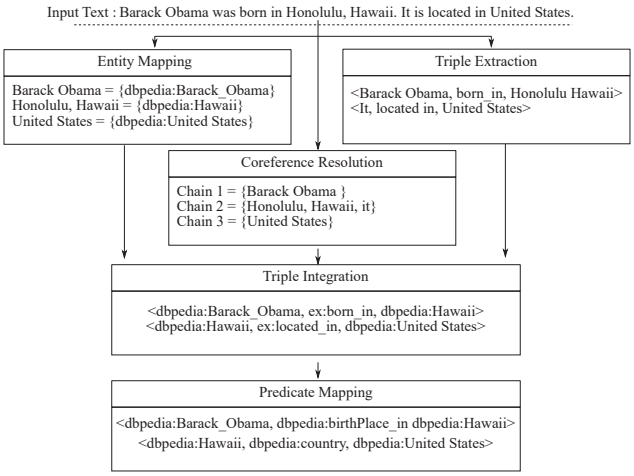

Figure 2: Example of the data flow in the T2KG system

Knowledge Graph Creation

In this section, the architecture of the T2KG system is described. T2KG is designed to take unstructured text as input and produce a KG as output. As shown in Figure 1, T2KG has 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 unstructured text to its corresponding entity in the KG. The Coreference Resolution component detects coreferring chains of entities in unstructured text. The Triple Extraction component extracts a relation triple from unstructured text by using the open information extraction technique. The Triple 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 as follows.

Entity Mapping

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

To further illustrate the flow of the T2KG system, an example is given in Figure 2. In Figure 2, the given sentence is "Barack Obama was born in Honolulu, Hawaii. It is located in United States. ", the expected results of the Entity Mapping component are a set of mapping entities for each entity, e.g., $Barack\ Obama = \{dbpedia\colon Barack\_Obama\}$.

Coreference Resolution

The aim of the Coreference Resolution component is to detect coreferring chains of entities in unstructured text and to group such entities. This is also an essential component because unstructured text usually contains abbreviations, pronouns and different expressions of entities that point to the same entities. With the Coreference Resolu-

tion component, an entity and its different expressions can be grouped so that actions of identical entities in different expressions can be captured. To discover the chains of coreferring entities, a coreference resolver is used. An example is shown in Figure 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 $C2 = \{Honolulu\ Hawaii,\ it\}$.

Triple Extraction

The aim of the Triple Extraction component is to extract relation triples from unstructured text. This is a key step to acquire knowledge from unstructured text. According to linguistic theory (Fillmore 1976), 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 a subject of a triple and an object of a triple.

To extract a relation triple from unstructured text, any open information extraction technique can be used. An open information extraction technique is used to extract information in an arbitrary sentence by using pattern templates and then to convert such information into a relation triple. In an open information extraction technique, a relation and its associated arguments in a sentence are identified without using either prior domain knowledge or a predefined vocabulary. For example, as depicted in Figure 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.

Triple Integration

The aim of the Triple Integration component is to generate text triples by 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 unstructured 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 interlinking to entities in the KG is not established. Consequently, transformation of a relation triple that conforms to the standard of KB is required. Therefore, to deal with such problems, the results from three components are integrated and transformed by the following processes.

First, identical entities are grouping by 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 various representations, the majority excluding pronouns in 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 its group. 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 unstructured text.

Figure 2 shows our example of this component. The Triple Integration component generates the text triple, e.g., $< dbpedia\colon Barack\_Obama, ex\colon born\_in, dbpedia\colon Hawai >$. However, the predicate of the triple, $ex\colon born\_in$, is still not mapped to any predicate in the KG.

Predicate Mapping

The aim of the Predicate Mapping component is to map a predicate of a text triple to an identical predicate in the KG. In the study (Exner and Nugues 2012), the rule-based approach is proposed for mapping a predicate of a triple to an identical predicate in a KG. However, the study greatly depends on the generated rules. Because of the sparsity of unstructured text in open domains, generated rules cannot cover all possible patterns. As a result, the study does not generalize enough to discover new rules that have not appeared before. Therefore, reasonable recall cannot be realized. To deal with heterogeneous vocabularies and to alleviate the sparsity of unstructured text, a hybrid combination of a rule-based approach and a similarity-based approach using the vector-based similarity metric is proposed in this study.

In our hybrid approach, bootstrapping triples are used to learn rules for mapping a predicate in a way similar to that of the reference study (Exner and Nugues 2012), and then the similarity-based approach using the vector-based similarity metric is applied for unseen rules to determine the identical predicates, as depicted in Figure 3. First, a text triple is enriched by the Triple Enrichment module. This module enriches a text triple and a KG triple by their data types and classes, and then integrates and normalizes the text triples, the KG triples and the enriched triples to create the bootstrapping triples for the later modules. Second, the Rule-based Candidate Generation module uses the bootstrapping triples for creating rules and then generates predicate candidate pairs for the text predicate. Third, the Similarity-based Candidate Generation module uses the bootstrapping triples for embedding the elements of triples as vector representations, and then such vectors are used to compute the similarity between a text predicate and a KG predicate in order to generate predicate candidate pairs. Eventually, the Candidate Selection module selects the most suitable mapping candidate. As a result, a candidate is selected and the KG creation process is completed. An example of the component is shown in Figure 2. As shown in the figure, $ex\colon born\_in$ is mapped to $dbpedia\colon birthPlace$, and the triple $< dbpedia\colon Barack\_Obama, dbpedia\colon birthPlace, dbpedia\colon Hawaii >$ is created as a triple for the generated KG. The details of each module are as follows.

Triple Enrichment The Triple Enrichment module enriches text triples and KG triples and then integrates all triples in order to create the bootstrapping triples for the later modules. Text triples and KG triples are enriched by their classes and data types. The enrichment process is performed

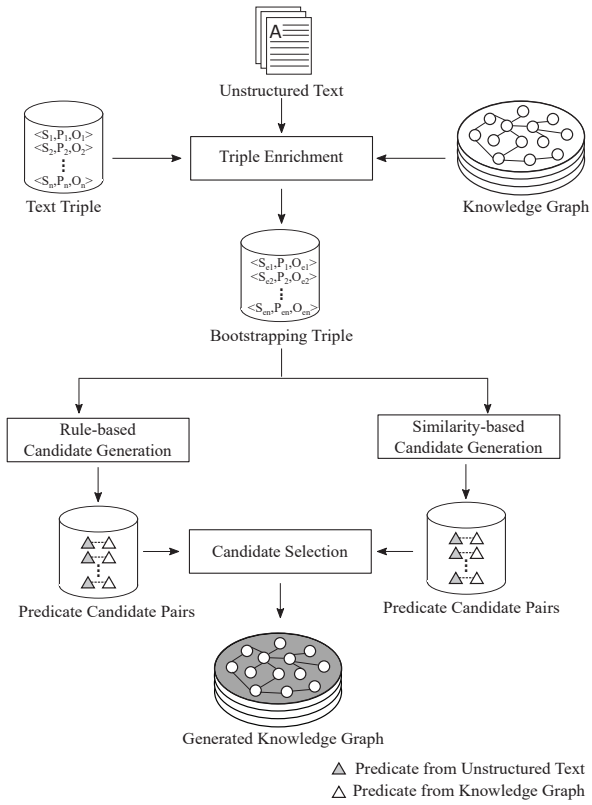

Figure 3: Diagram of the Predicate Mapping component

only on a subject (domain) and an object (range) 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 by using rdf:type. For example, given DBpedia as the KG and the triple, $<$ $dbpedia\colon Barack\_Obama$, $dbpedia\colon birthPlace$, $dbpedia\ Hawaii >$, the enriched result is $<$ $dbpedia\colon Person$, $dbpedia\colon birthPlace$, $dbpedia\colon Location >$. 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 map to KG class by using string matching as a workaround. For example, the NER class, $Person$, is mapped to KG class, $dbpedia\colon Person$.

Apart from the class of entities, the data type is also considered. In T2KG, we use the URI, string, number and date as data types. The data types of a subject and an object of the triple are converted by 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 contains only a number, the number type is used. If a subject or an object of a triple is a URI, the URI type is used. Otherwise, the string type is used. For example, given the triple $<$ $dbpedia\colon Barack\_Obama, ex\colon born\_in, dbpedia\colon Haw-aii >$, the result is $<$ URI, $ex\colon born\_in$, $URI >$. All generated triples, called bootstrapping triples, are used as the output of this module.

Rule-based Candidate Generation The Rule-based Candidate Generation module extracts rules and uses these rules to produce predicate candidate pairs. In this module, the strategy in the reference study (Exner and Nugues 2012) is implemented to create rules for mapping the predicate, as follows. 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 identical. Second, the class of the subject and the class of the object are used as a constraint for mapping. For example, a finding rule can be $<$ $Person$, $ex\colon born\_in$, $Location$ $>$ is mapped to $dbpedia$: $birthPlace$ (using DBpedia as the KG). Even though this approach uses bootstrapping triples to generate reliable rules, the number of rules is very limited due to the small number of bootstrapping triples and the sparsity of unstructured text. Therefore, some rules are missing. To avoid such problems, the similarity-based approach using the vector-based similarity metric is applied in the T2KG system.

Similarity-based Candidate Generation The Similarity-based Candidate Generation module generates predicate candidate pairs by using the similarity between predicates. Generally, a string-based similarity metric is used for the entity mapping or the predicate mapping task (Kertkeidkachorn et al. 2013; Gerber et al. 2013). Due to the heterogeneous vocabularies in the open information extraction task, the string-based similarity metric can fail to identify the similarity between predicates. To cope with heterogeneous vocabularies, each vocabulary should be learned and represented at a deeper level than just their surface form. We therefore propose the novel vector-based similarity metric for computing the similarity between elements of triples.

Based on a review, we found that Mikolov et al. proposed vector representations of words that can capture both syntactic and sematic patterns (Mikolov et al. 2013). Inspired by vector representations of words (Mikolov et al. 2013), we present elements of triples in the vector space by using other elements in the same triple. The objective function is formulated as follows.

$$L(\theta) = \arg\max_{\theta} \sum_{(e,c)\in T} \log \sigma(v_e \cdot v_c) + \sum_{(e,c)\in T'} \log \sigma(v_e \cdot v_c) \quad (1)$$

where $\sigma(x) = 1/(1 + exp(-x))$, e is an element of a triple, c is other elements of the same triple, T is a set of bootstrapping triples from the triple enrichment module, T' is randomly generated triples (negative triple), which do not existed in T, $v_e, v_c \in \theta$ and v_e, v_c are vector representations of elements of triples e and c respectively.

After acquiring vector representations of 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.

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

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

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} pair of a subject and an object, respectively, corresponding to predicate P_{KG} in KG ($< S_{P_{KG_n}}, P_{KG}, O_{P_{KG_n}} > \in KG$), N is the number of triples in KG, whose predicate is P_{KG}, and δ is a weight parameter between the predicate similarity and the context similarity. The basis behind these equations 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 might be varied by its 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 they can map. Because the first and the second terms in Eq. 2 are different, the weight parameter is introduced for adjusting the salient aspect between the predicate similarity and the context similarity. Eq. 3 is proposed to compute an average vector representation of the context of P_{KG}.

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, which 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 considered. If the similarity of the predicate candidate pair is greater than a threshold, 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 candidate selection is the generated KG, in which both the entities and the predicates are linked to other KGs.

Experiment

Experimental Setup

The experiments are designed to evaluate the performance of the hybrid approach in the T2KG system, and the performance of the T2KG system.

In the experiments, 120,000 Wikipedia articles are randomly selected and then pre-processing is applied to establish unstructured text as input for the system. In the preprocessing, HTML markups, wiki marks and any hyperlink annotations are removed. Duplicated sentences are reduced to one sentence. Apart from the unstructured text, another input is the KG. In our experiments, DBpedia (Auer et al. 2007) is set as the KG in the T2KG system. Note that any KG can be used in the T2KG system.

In the T2KG system, each component is implemented and its parameters are configured as follows. In the Entity Mapping component, DBpedia Spotlight (Mendes et al. 2011) is used to map entities. If an entity cannot map to any DBpedia entities, a new namespace "ex: "is adopted as the prefix of the entity for creating a new URI. This namespace also applies for an unmapped predicate. In the Coreference Resolution component, the Stanford NLP tool (Lee et al. 2011; Raghunathan et al. 2010) is used as a coreference resolver. In the Triple Extraction component, OLLIE (Schmitz et al. 2012), which is one of the state-of-the-art for open information extraction, is applied to extract relation triples from unstructured 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 the training vector representations of the elements of triples. The parameters of word2vec are set by default. The weight parameter δ in the similarity score computation is set at 0.6. In the Candidate Selection module, the mapping threshold of the similarity is set at 0.25.

Experiment 1

The aim of this experiment is to evaluate the performance of our hybrid approach for the predicate mapping task. To investigate the contribution of our approach, the rule-based approach in the study (Exner and Nugues 2012) is used as the baseline for comparison.

To conduct the experiment, the ground truth is automatically constructed. The predicate mapping between the predicates of text triples and the predicates in DBpedia is performed. The strategy for constructing the ground truth is 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 demonstrates an example for constructing the ground truth. As shown in this figure, the subject and the object of the text triple and the DBpedia triple are identical. Consequently, the predicates, "ex:born_in" and "dbpedia:birthPlace", are assumed to be identical. In the ground truth construction, the targets of the mapping predicate are 2,800 DBpedia ontology properties. Although this method can help to establish a lot of ground truth, it is possible that the ground truth might be ambiguous due to multiple matching. Multiple matching is that two or more predicates share the same subject-object pair. For example, "ex:bear_in" might be forcedly 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 of the dataset. To alleviate this problem, we simply remove triples, of which the subject-object pair appears more than once. According to the ground truth construction, the number of remaining mapped predicate pairs is 43,800.

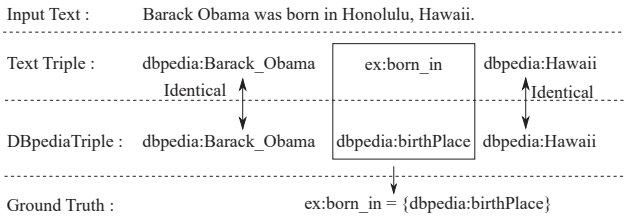

Figure 4: Example of Ground Truth Construction

Table 1: Results of Experiment 1

Approach	Precision	Recall	F-Measure
Rule-based (Exner et al.)	0.54135	0.36324	0.43473
Hybrid	0.54377	0.46340	0.50036

In this experiment, given a predicate text, an approach returns the DBpedia predicate having the highest similarity. Precision, recall and F-Measure then use to measure whether the DBpedia predicate and the predicate text are correctly match or not. To evaluate our similarity metric, 10-fold cross-validation is performed.

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 precision by 0.24%, recall by 10.02% and F-measure by 6.56%. Because the results show that the discoverability of the hybrid approach outperforms the baseline, it conforms to our hypothesis that the hybrid approach including the similarity-based approach contributes to the discovery of identical predicates. The hybrid approach therefore can alleviate the problem caused by the limitation of patterns in constructing the rule-based approach.

Experiment 2

The aim of this experiment is to evaluate the quality and quantity of generated triples for the KG creation task. Because no gold standard exists for evaluating the results of generating triples from unstructured text, we conduct the evaluation by manual establishing a small set of the gold standard of triple extracted from unstructured text. To create the gold standard, we randomly select 100 sentences from Wikipedia articles and then manually extract and map triples to DBpedia triples.

Based on the results, T2KG could extract 1.76 triples per sentence on average. The generated triples realized precision of 49.39%, recall of 52.26% and F-measure of 50.78%.

To more deeply analyze the results, the errors in each component of the system are investigated. The system consists mainly of four components that can provide errors: Entity Mapping, Coreference Resolution, Triple Extraction and Predicate Mapping. We therefore calculate the ratio of errors based on those four components. The results show that 31.94% of the errors are caused by Triple Extraction, 26.85% by Predicate Mapping, 20.83% by Entity Mapping and 20.37% by Coreference Resolution. The largest source of errors is Triple Extraction because the task in this study is the 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 their arguments are not clearly identified.

Furthermore, errors in the elements of generated triples are inspected. We find that the largest number of errors is 41.94% caused by predicates, 34.68% by objects and 23.39% by subjects. Based on the error analysis, the majority of errors are caused by the predicates of generated triples. The reason is that Triple Extraction can not perfectly extract predicates from unstructured text due to the complexity of the text in open domains. Nonetheless, although KG creation in our study is conducted in open domains, the T2KG system still achieves approximately 50% in both quality and quantity of generated triples for creating the KG.

Conclusion

This paper presents T2KG, a system for automatic knowledge graph creation from unstructured text. The experimental results demonstrate that the T2KG system can successfully generate a KG from unstructured text. Although KG creation in this study is conducted in open domains, the T2KG system still achieves approximately 50% in both quality and quantity of triples generated for creating the KG. Furthermore, the hybrid approach for mapping a predicate is introduced. In the hybrid approach, the novel vector-based similarity metric is proposed. The experimental results indicate that the hybrid approach improves both the precision and the recall for mapping a predicate to a KG.

Based on error analyses, the performance can still be improved. In future work, we aim to improve the implementation of the T2KG system, in particular the Triple Extraction component. Furthermore, because our approach does not adhere to any data resources, we also intend to conduct experiments on other data resources.

Acknowledgements

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

References

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. *DBpedia: A nucleus for a web of open data*. Springer.

Augenstein, I.; Pado, S.; and Rudolph, S. 2012. Lodifier: Generating linked data from unstructured text. In *The Semantic Web: Research and Applications*. Springer. 210–224.

Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka Jr, E. R.; and Mitchell, T. M. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of AAAI*.

Cattoni, R.; Corcoglioniti, F.; Girardi, C.; Magnini, B.; Serafini, L.; and Zanoli, R. 2012. The KnowledgeStore: an

Entity-Based Storage System. In *Proceedings of LREC*, 3639–3646.

Exner, P., and Nugues, P. 2012. Entity extraction: From unstructured text to dbpedia rdf triples. In *The Web of Linked Entities Workshop*, 58–69. CEUR-WS.

Fader, A.; Soderland, S.; and Etzioni, O. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1535–1545. ACL.

Fillmore, C. J. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences* 280(1):20–32.

Gerber, D.; Hellmann, S.; Buhmann, L.; Soru, T.; Usbeck, R.; and Ngomo, A.-C. N. 2013. Real-time RDF extraction from unstructured data streams. In *Proceedings of The Semantic Web–ISWC 2013*, 135–150. Springer.

Kertkeidkachorn, N.; Ichise, R.; Suchato, A.; and Punyabukkana, P. 2013. An automatic instance expansion framework for mapping instances to linked data resources. In *Joint International Semantic Technology Conference*, 380–395.

Kriz, V.; Hladka, B.; Necasky, M.; and Knap, T. 2014. Data Extraction Using NLP Techniques and Its Transformation to Linked Data. In *Proceedings of 13th Mexican International Conference on Artificial Intelligence*, 113–124. Springer.

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

Mendes, P. N.; Jakob, M.; Garcia-Silva, A.; and Bizer, C. 2011. DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, 1–8. ACM.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in neural information processing systems*, 3111–3119.

Raghunathan, K.; Lee, H.; Rangarajan, S.; Chambers, N.; Surdeanu, M.; Jurafsky, D.; and Manning, C. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 492–501. ACL.

Ratinov, L.; Roth, D.; Downey, D.; and Anderson, M. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of ACL*.

Schmitz, M.; Bart, R.; Soderland, S.; and Etzioni, O. 2012. Open language learning for information extraction. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 523–534. ACL.