Knowledge Graph Population from Natural Language Text

Natthawut Kertkeidkachorn

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

Knowledge Graph Population from Natural Language Text

Author:
Natthawut Kertkeidkachorn

Supervisor:
Assoc. Prof. Ryutaro Ichise

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

September 2017

A dissertation submitted to Department of Informatics,
School of Multidisciplinary Sciences,
SOKENDAI (The Graduate University for Advanced Studies),
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Advisory Committee

1. Assoc. Prof. Ryutaro Ichise	SOKENDAI
2. Prof. Ken Satoh	SOKENDAI
3. Prof. Hideaki Takeda	SOKENDAI
4. Assoc. Prof. Yusuke Miyao	SOKENDAI
5. Prof. Akiko Aizawa	The University of Tokyo

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Ryutaro Ichise for the continuous support of my doctoral study and research, for his patience, encouragement, and immense knowledge. Without his guidance and support, this thesis would not have been possible. I greatly appreciate all his contributions of times, ideas and supervision to make my research experience productive and stimulating.

I would like to thank advisory committee members: Prof. Ken Satoh, Prof. Hideaki Takeda, Prof. Yusuke Miyao and Prof. Akiko Aizawa, for not only their insightful and constructive comments and encouragement, but also for the fruitful suggestions that inspired me to broaden my research from various perspectives.

I appreciate my lab members, Khai, Mizanur, Ebisu, Lihua, Munne as well as all internship students, and anyone who encouraged me and provided thoughtful comments on my research.

Mostly importantly, I would like to deeply express my heart-felt gratitude to my family members, who have been a constant source of love, concern, encouragement and continuous support throughout my years of studying, researching, and writing this thesis. None of this would have not been possible without them.

Lastly, I would like to thank SOKENDAI (The Graduate University for Advanced Studies), National Institute of Informatics and Ministry of Education, Culture, Sports, Science and Technology for the educational and financial support during my doctoral studies. I appreciate all the incredible opportunities and memorable experience they offered so far.

SOKENDAI (THE GRADUATE UNIVERSITY FOR ADVANCED STUDIES)

Abstract

School of Multidisciplinary Sciences
Department of Informatics

Doctor of Philosophy

Knowledge Graph Population from Natural Language Text

by Natthawut Kertkeidkachorn

Knowledge Graph (KG) plays a crucial role in many modern applications as prior knowledge. In recent years, there are many existing KGs such as DBpedia, Freebase, YAGO and etc. Nevertheless, it is well-known that a KG is incomplete. Also, new knowledge emerges every day. Consequently, the KG becomes more and more incomplete over time. It is therefore necessary to populate new knowledge to the KG in order to fill missing knowledge. Considering the growth of the data, we found that the volumes of natural language text, are massively exploding from various data resources. Based on this reason, most of new knowledge has been published as natural language text. Nevertheless, natural language text has been treated as string, which cannot interpret any semantics due to the schemaless problem. Moreover, due to the complex structure of language, it is not feasible for a machine to understand knowledge in natural language text. Furthermore, publishers usually publish natural language text by using their vocabulary. It leads to the heterogeneous problem, where an identical thing is represented by many representations. As a result, a large amount of knowledge in natural language text cannot directly transfer to KGs and so is left as natural language text.

Recently, there are many approaches for constructing a KG from natural language text in order to transfer knowledge to the KG. However, constructing the KG from natural language text usually builds its KG separately without integrating extracted knowledge to other existing KGs. Integrating extracted knowledge is an essential procedure because it reduces the heterogeneous problem and increases searchability over KGs. In this dissertation, we therefore aim to propose T2KG: the framework for automatically constructing/populating a KG from natural language text, where extracted knowledge is integrated to an existing KG. To integrate extracted knowledge to the existing KG, two major tasks, 1) entity linking and 2) predicate linking, are taken into account. In the framework, two sub-frameworks, namely HMiLDs and HRSim, are also proposed for dealing with the entity linking task and the predicate linking task respectively.

Linking entities to KGs becomes a challenge problem because of the continuous growth of KGs. Due to a large number of KGs, we could not know which KG contains an identical entity. As a result, some entities could not be linked to KGs. To the best of our knowledge, linking entity to multiple KGs is not addressed yet. In the entity linking task, we therefore proposed a Heuristic expansion framework for Mapping Instances to KG data sets (HMiLDs). The main idea of HMiLDs is to directly map entities to one particular KG and then gradually expand a search space to other KGs for discovering identical entities. Due to a large amount of entities in KGs, an expansion strategy and a heuristic function for limiting the expanding search space are designed into the framework. In experiments, HMiLDs could successfully map entities to the KGs by increasing the coverage up to 90%. Moreover, experimental results also indicated that the heuristic function of HMiLDs could efficiently limit the expansion space to a reasonable space by reducing the number of candidate pairs without affecting any performances.

Predicate linking is used to identify the predicate in a KG that exactly corresponds to an extracted predicate; this is to avoid the heterogeneity problem when populating a KG. Although there have been a few studies that considered linking predicates, most of them have relied on statistical knowledge patterns, which are not able to generate the possible patterns. In the predicate linking task, we therefore proposed a Hybrid combination of Rule-based approach and Similarity-based approach (HRSim). In HRSim, we also proposed a novel distributed representation of the elements in triples and show how this can be used to compute the similarity between predicates in order to find links that would not appear in statistical patterns. The experimental results show that our distributed representation-based similarity metric outperforms other traditional similarity metrics. Also, leveraging distributed representation-based similarity metric could help to discover and identify identical KG predicates for text predicates. As a result, our approach could alleviate the problem caused by the limitation of statistical knowledge patterns due to the sparsity of text and improve the discoverability for the predicate linking task.

Finally, we introduced T2KG: the framework for populating knowledge from natural text to existing KGs. In T2KG, entity linking and predicate linking are considered when populating knowledge to existing KGs. The intuition of T2KG is to extract knowledge as triples by an open information extraction system and then integrate the knowledge into the existing KGs by performing entity linking and predicate linking in order. The experimental results show that T2KG outperforms the traditional KG construction. Although the KG population is conducted in open domains, in which any prior knowledge is not given, T2KG still achieves approximately 50% of F1 score for generating triples in the KG population task. In addition, the empirical study on the knowledge population using various text sources is conducted. The experimental results indicate T2KG could succeed to discover new knowledge that does not exist in DBpedia.

Contents

List of Figures

List of Tables

To my family, teachers, and friends.

Chapter 1

Introduction

In this chapter (Chapter 1), we present the background about Knowledge Graph and the Linked Data concept in Section 1.1. Then, we introduce the motivation of this dissertation and discuss about the research statement and remained problems in the Section 1.2. Next we present the contributions of this dissertation in the Section 1.3. Finally, in Section 1.4, we provide the outline of the rest of the dissertation.

1.1 Background

In recent years, an amount of available data is dramatically increasing due to the growth of the Internet. Since the invention of the World Wide Web by Tim Berners-Lee [1], the data on the web has been generated and published continuously. The availability of the data on the web becomes more and more extremely rich. As a result, we can acquire fruitful knowledge from the data on the web. Such knowledge can obviously assist both a human and a machine to make a decision. For example, a human uses the knowledge to comprehend his/her interest as the prerequisite background or a search engine uses the knowledge as the prior knowledge to retrieve a relevant document. In order to collect and utilize the knowledge efficiently and effectively, a suitable technology is required. Traditionally, a knowledge base is the technology used to manipulate and store knowledge by a computer system. Currently, a modern knowledge base has become popularly known as Knowledge Graph [2].

Knowledge Graph (KG) is a structure knowledge base, which stores knowledge in the form of real-world entities and their relationships. The term Knowledge Graph becomes widely known because of the release of the Google' s Knowledge Graph in 2012 [3]. Recently, the KG term gradually gains the attention from many researchers, especially in the semantic web community, because the main concept behind KG is the Linked Data concept [4], which is the major technology for the semantic web. The Linked Data concept was introduced by Tim Berners-Lee to provide a standard method of publishing and connecting the data on the web [5]. The idea of Linked data comes with the growth of the data on the web since the web technology also evolved by the time. In the past, Web 1.0 as the static web provided the read only content, while more recent, Web 2.0 as the dynamic web allowed users to interact and engage with the activity on the web directly [6]. Currently, huge efforts have been used to put forward from Web 2.0 to Web 3.0. In Web 3.0 or the semantic web, the technology not only allows humans to consume and provide the content similar to Web 2.0 but also enables the machines to do as well [7]. The principles of Linked Data [5] are defined as follows:

- Things should be represented by Uniform Resource Identifier (URI).
- Things should use HTTP URI so that people or an agent can dereference the name of such things .
- The data should be published under Resource Description Framework (RDF), specifically in the form of a triple (Subject, Predicate, Object) .
- Links between things should be provided to enhance discoverability among things.

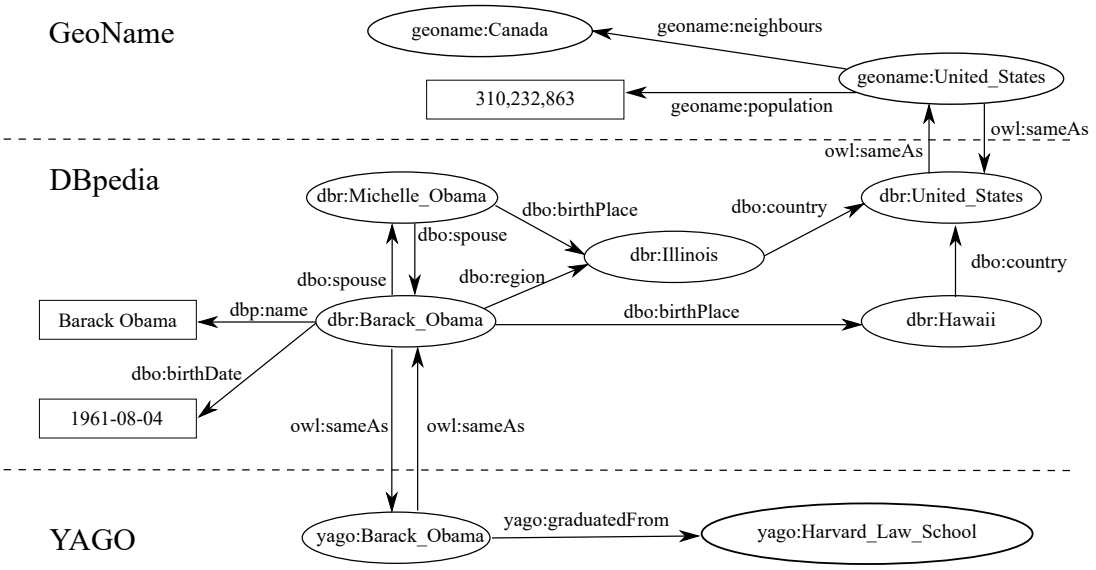


FIGURE 1.1: An Example of knowledge in Knowledge Graphs

Based upon these principles, knowledge in KGs has been connected with other knowledge as the web of the data.

Since Linked Data enables connection between several KGs, a machine could use knowledge in a KG by querying over structured links. For example, if a thing in the KG A connects to its identical thing in the KG B, the information of such thing in the KG A could be enriched by its connected links. As a result, another perspective of knowledge will be discovered. In Figure 1.1, an example of knowledge in KGs is illustrated. In the example, knowledge in three KGs: GeoName, DBpedia and YAGO, are connected. Consequently, we could query over KGs to find what we are looking for. For example, we could find the school where "Barack Obama" graduated from, by querying the entity "dbr:Barack_Obama" in DBPedia. Althouhg DBpedia do not contain the answer, by travsing in the structure links between KGs, we can find the answer "yago:Harvard_Law_School" in YAGO. This demonstration explicitly shows that KGs are extremely rich knowledge resources.

Recently, immense efforts have been put to construct wide-ranging KGs [2]. Currently, there is the ongoing project, which aims to construct KGs, named the Linked Open Data (LOD) cloud [8]. The LOD cloud is a set of KGs that haven been openly published and provided links among them. The LOD cloud project started in 2007 and over the past years its growth is dramatic. Recently, there are more than a thousand of KGs, contain more than a billion of knowledge in the form of RDF triples, (subject, predicate, object)[9]. KGs in the LOD cloud are categorized into nine domains: cross-domain, geography, government, life science, linguistic, media, publications, social networking

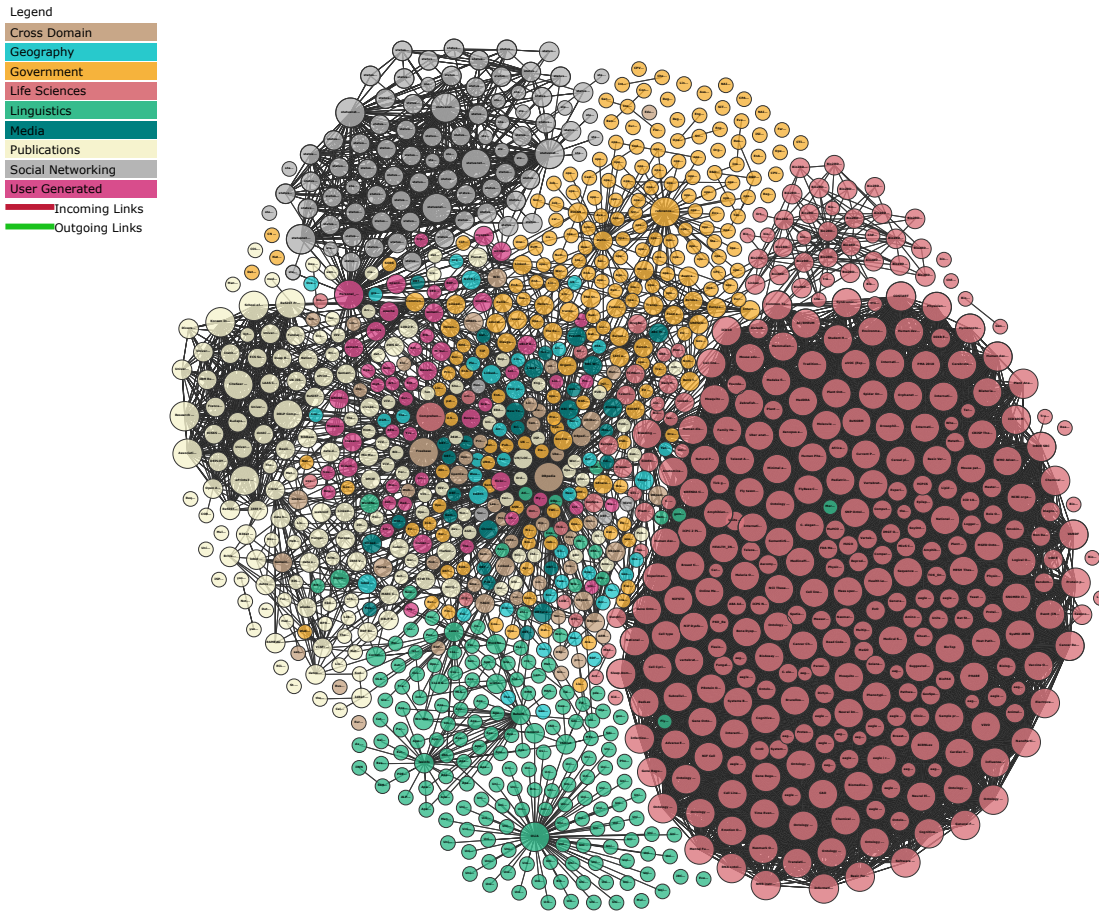


FIGURE 1.2: The Knowledge Graphs in Linked Open Data Cloud Project [8]

and user-generated content as shown in Figure 1.2. Some of well-known KGs in the LOD cloud are DBpedia [10], Freebase [11] and YAGO [12].

Such KGs play an important role in many advanced applications such as a question and answering system, a quiz generation system, a search engine and etc [13]. For example, question and answering systems [14, 15], have used KGs as the prior knowledge to answer a specific question given by a user. A quiz generation system uses the knowledge in KG to formulate a question and choice for a user to answer [16]. Search engines [3, 17] use KGs to understand the concept of search keywords to transform a text-based search engine into a semantic-based search engine, which can semantically understand a user' s query. Apart from the open-domain KGs, several specific domain KGs such as Bio2RDF[18] and Neurocommons [19] have been used to support decisions in the life science domain applications. As a result, KGs become the prominent resource for many modern artificial intelligent systems.

1.2 Motivation

Since KGs play an important role in many modern applications as prior knowledge, many researchers, especially in areas of the semantic web and the natural language processing, pay a huge attention to construct and populate knowledge into KGs. In recent years, there are existing KGs such as DBpedia, Freebase, YAGO and etc. However, new knowledge regularly emerges everyday. Consequently, the current KGs gradually become obsolete and some knowledge in KGs may not be useful in the future. Considering the fact about president of the United State in 2017 as an example, if KGs are not updated, the knowledge about the president of the United State provided by KGs is "Barack Obama". Nevertheless, in 2017 a new president of United State "Donald Trump" is elected. The knowledge about the president of the United State in KGs becomes out of date. Consequently, when we search for the president of the United State, we could retrieve "Barack Obama" as the result, which becomes inappropriate in the current context. Another example is that suppose a new movie is going to release very soon, KGs might not be able provide information about such movie because knowledge about such information is missing. As a result, such KGs become not useful. Therefore, it is necessary to populate new knowledge to existing KGs in order to keep the existing KGs up to date.

Generally, most of new knowledge have been published as natural language text on the web and such trend is dramatically growing faster than the growth of KGs [20]. Since such natural language text has been published on the web, we can access to them easily. Nevertheless, natural language text has been traditionally treated as string, which cannot be explicitly interpret any meanings or do not contain schemas or any links to any KGs. Moreover, due to language complexity, which relates to the structure of the language, it is not feasible for a machine to understand knowledge in natural language text directly. Furthermore, a publisher usually publishes natural language text by using his/her own vocabulary. It leads to the heterogeneous problem, where an identical real-world thing could be represented by many representations. Based on these reasons, a large amount of natural language text cannot be straightforwardly transformed into a KGs and so is left as natural language text.

Furthermore, although a human can directly consumes natural language text, a machine cannot make much use of such knowledge in form of natural language text. The main reason is that a machine cannot understand a concept or a meaning in natural language text. Consequently, a machine loses an opportunity to use rich knowledge resources, which is left as natural language text. Due to advantage of KGs, it is therefore essential to transform natural language text to KGs so that a machine can also utilize the knowledge more efficiently and effectively.

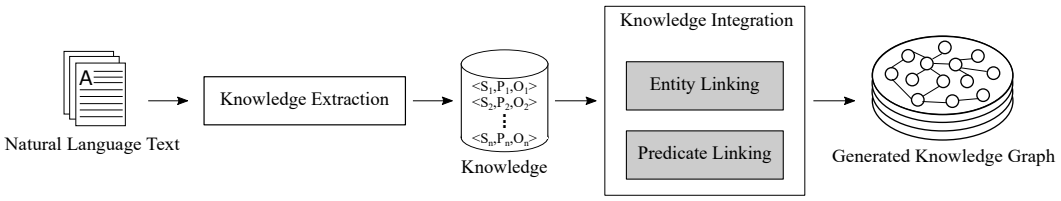


FIGURE 1.3: The General Work Flow of the Knowledge Graph Population

The main interest of this research therefore is to populate knowledge from natural language text to existing KGs. The big picture of the KG population is depicted in Figure 1.3. In the work flow, knowledge is extracted from natural language text and then integrates to existing KGs. In this research, we mainly focus on the knowledge integration part, including entity linking and predicate linking, as well as the KG population. Those three problems are described as follows.

Knowledge Graph Population is to extend knowledge in existing KGs by extracting new knowledge from natural language text. However, due to the complexity of natural language text and heterogeneity of vocabularies, populating KG becomes one of the challenging problem for the semantic web community. Generally, natural language text that is interested in this dissertation is the open domain. Although there are many approach proposed to extract knowledge from natural language text, the integration, Entity Linking and Predicate Linking, still have not been solved.

- **Entity Linking**

  Entity Linking is to find a link between identical entities. When extracting an entity from natural language text, the extracted entity might be ambiguous. For example, the *Washington* entity could refer to a person or a place due to the context that the entity appear in. In contrast with an entity from natural language text, an entity in KGs is ideally unique for each real-world object because the KG entity is represented by URI. If we could map the extracted entity to the entity in KG, it will eliminate the ambiguity of the entity due to an identity of URI. Furthermore, if the entity from natural language text could be mapped to the entity in the KG, it means that the entity could be enriched in two ways. In one way, the entity in the KG is enriched by information from the extracted entity. In other way, the entity from natural language text is also enriched by knowledge from KGs.

  Although there are many studies proposed the method for linking between entities, linking an entity to KGs in our problem is different from the standard entity linking. Linking entity to KGs becomes a challenging problem because of the

continuous growth of KGs. When the LOD cloud project started in 2007, there are only 12 KGs, while currently there are more than 1,000 KGs [8]. Due to a large number of KGs, we could not know which KGs contains an identical entity for the extracted entity from natural language text. As a result, some entities could not be linked to KGs. To link an extracted entity to KGs, we should consider multiple KGs so that an identical entity could be discovered. To the best of our knowledge, entity linking to multiple KGs is not addressed yet.

- **Predicate Linking**

  Predicate Linking is to find a link between identical predicate. When extracting a relation between entities from natural language text, the relation, which is the predicate in this context, might be heterogeneous. For example, the relation "birth place" may represented by "was born in" or "the place of birth" and etc. Linking predicate to its identical predicate in KGs plays an crucial role for utilizing KGs, because it could reduce the heterogeneous problem and could increase searchability over KGs.

  However, linking predicate from natural language text to KGs predicate differs from the simple predicate linking because there is no any schemas provided for the predicate extracted from natural language text. It also leads to the sparsity of text, in which the necessary patterns is missing. As a result, the traditional approach, which used the statistical patterns, could not achieved the reasonable results. Therefore, predicate linking without schemas still remains unsolved yet.

In this research, this KG population takes the knowledge integration issue into account. The knowledge integration is to connect a new knowledge to existing knowledge. Consequently, the knowledge integration aims to increase the compatibility of knowledge in existing KGs and to reduce the heterogeneous problem. If the knowledge integration discards, the new knowledge is populated as an isolated KG. Although we can acquire new knowledge in the new KG, the existing KGs yet are obsolete. Therefore, the knowledge integration should taken into consideration so that the existing KGs can still remain up-to-date.

1.3 Contribution

To address and solve the research problems above, we propose one main framework and two sub-frameworks. The problem and its corresponding framework are listed in Table 1.1. The contribution of each component are presented in the following.

TABLE 1.1: Research Problems and Corresponding Frameworks

Research Problems	Corresponding Frameworks
Entity Linking	HMiLDs
Predicate Linking	HRSim
Knowledge Graph Population	T2KG

- **HMiLDs** [21, 22]

  HMiLDs is a heuristic framework for mapping entity to multiple KGs. The basic idea of HMiLDs is to directly map entities to one particular KG and then gradually expand a search space for discovering identical entities to other KGs in order to find other identical entities. Due to a large amount of entities in KGs, an expansion strategy and a heuristic function for limiting the expanding search space are designed into the framework. The framework could successfully map entities to the KG by increasing the coverage. The heuristic function in the framework could efficiently limit the expansion space to a reasonable space. Based upon the limited expansion space, the framework could effectively reduce the number of candidate pairs without affecting any performances.

- **HRSim** [23–25]

  HRSim is the framework for mapping predicate to a KG. The idea of HRSim is to combine a rule-based approach and a similarity-based approach as the hybrid system for mapping predicate. The rule-based approach is a highly accurate approach but required massive of training data to construct rules. In contrast, the similarity-based does not required much data but the performance of the similarity-based might not be good enough when textual string is significantly different. We propose the hybrid combination between a rule-based approach and a similarity-based approach to gain advantages of each approach. Also, the similarity-based approach greatly relies on similarity measure. In the similarity-based approach, we propose a novel vector-based similarity measure. Our novel vector-based similarity measure outperformances other standard similarity measures. Furthermore, HRSim could perform better than the rule-based approach in the predicate linking task.

- **T2KG** [23, 26]

  T2KG is the framework for automatic populating knowledge from natural language text to KGs, where entity mapping and predicate mapping are taken into account. 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 achieves approximately 50% in both quality and quantity of triples generated for creating the KG. In addition, the empirical study on the knowledge population from text is investigated. The experimental results indicate that T2KG can successfully populate new knowledge from text to DBpedia.

In conclusion, we propose T2KG: the framework for automatic generating Knowledge Graph from natural language text, where entity mapping and predicate mapping are taken into account. In the framework, two sub-frameworks, namely HMiLDs and HRSim, are proposed for dealing with entity mapping and predicate mapping respectively. A Heuristic expansion framework for Mapping Instances to Knowledge Graph data sets (HMiLDs) is the framework for mapping entity to multiple Knowledge Graph resources, while a Hybrid combination of Rule-based approach and Similarity-based approach (HRSim) is the framework for mapping predicate to predicates in existing Knowledge Graph.

1.4 Outline

In this dissertation, there are 7 chapters. The remaining of our dissertation is organized as follows.

Chapter 2: Fundamentals and Related Work

This chapter presents the fundamentals about KG, knowledge representation and world assumptions. Then, the related work on KG construction, KG completion and KG integration are reviewed and discussed respectively. Later, we further discussed and surveyed on KG integration topics including, entity linking, predicate linking as well as the similarity measurement that frequently uses in the KG integration task. Finally, we discussed and identified the remain issue before concluding the chapter.

Chapter 3: Entity Linking - HMiLDs

This chapter presents the details of the HMiLDs for the entity linking in multiple KGs. In the chapter, the objective of the entity linking problem is further described. Then, the details of components of the HMiLDs framework are given. Next, the experiments are conducted to investigate each component of the HMiLDs framework as well as the whole framework. Finally, we discuss and summarize the HMiLDs framework.

Chapter 4: Predicate Linking - HRSim

This chapter presents the predicate linking task for populating knowledge to an existing KG. In the chapter, the objective of the predicate linking task is given. Then, the details of the HRSim framework for solving this predicate linking problem are introduced. Next, the experiments are conducted and the results are reported. Eventually, we conclude this chapter with the summary of the HRSim framework.

Chapter 5: Knowledge Graph Population - T2KG

This chapter introduces T2KG: the framework for automatic generating Knowledge Graph from natural language text, where entity mapping and predicate mapping are taken into account. In this chapter, the details of the T2KG framework are presented. Next, we report the comparison with the baseline approach and the empirical experiments on how well the new knowledge can be populated to existing KGs. Finally, we summarize the T2KG framework.

Chapter 6: Discussion

This chapter discusses the achievement what we accomplished by the proposed framework, including HMiLDs, HRSim and T2KG. Then, the scope, the limitation and the discussion of each framework are presented respectively.

Chapter 7: Conclusion

In the last chapter, we summarize the dissertation and discuss about the future direction of the research on knowledge graph population.

Chapter 2

Fundamentals and Related Work

In this Chapter, we presented fundamentals of KG and related terms as well as their formal definition in Section 2.1.1. Then, the representation of KG and the assumptions of the KG were discussed in Section 2.1.2 and Section 2.1.3 respectively. In Section 2.2, we reviewed and discussed the KG construction approaches and methods of each approach were presented in the following sub-sections. After that, the KG completion was discussed in the Section 2.3. Next, the survey on KG integration was reviewed and discussed in Section 2.4. In the following sub-section (Section 2.4.2), KG resolution, including entity linking and predicate linking, and several similarity measurements that widely used for KG integration were presented. Finally, We discussed the remained issues and summary this chapter in Section 2.5 and Section 2.6 respectively.

2.1 Knowledge Graph

Many studies use the term of Knowledge Graph in many ways [2]. Although, in the previous chapter, we presented the background and necessity of Knowledge Graph and many related technical terms, their definitions are still not clearly clarified yet. We are therefore going to formally define the terms and give some further backgrounds and fundamentals for Knowledge Graph as follows.

2.1.1 Knowledge Graph Definition

Knowledge Graph is a graph-based knowledge base, which model knowledge between entities and relations [13]. Its definition can be formalized in the following definition.

Definition 2.1. Knowledge Graph

A Knowledge Graph $KG = (V, E)$, where V is a set of vertices and E is a set of edges with a label. A vertex or a node in KG is an entity, while a labelled edge in KG is a relation.

- **Entity** is a vertex or a node in KG, which represents a unique real-world object. Note that, a description of the entity, referred as literal, can also be a node in KG.
- **Relation** is an edge with the label in KG, which express relationship between entities or between an entity and its description.

2.1.2 Knowledge Graph Representation

Knowledge Graph is represented by the Linked Data concept [13]. In the Linked Data concept, the Resource Description Framework (RDF) is a standard model for publishing Linked Data [5]. It uses to describe the information and their relations and also make data become interchangeable [5]. The specification of RDF is introduced by W3C Recommendation (RDF 1.1 Concepts and Abstract Syntax) [27]. In the RDF specification, the key concept is a RDF graph, which is a collection of RDF triples. A RDF triple consists of a subject, a predicate and an object. A subject and an object are treat as a vertex, while a predicate is considered as a relation. The direction of the edge is used to identify which vertex is the subject and which vertex is the object. The edge of a RDF triple points out from the subject and points into the object. An RDF triple therefore can be viewed as a directed graph, which composes of two vertices and one directed edge, as shown in Figure 2.1.

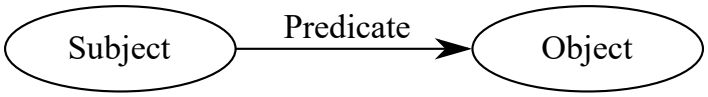


FIGURE 2.1: RDF Graph Representation [27]

Furthermore, the RDF specification [27] defines three kinds of resource representations: Internationalized Resource Identifier, literal and blank node. Such representations are used to describe an element of a RDF triple. Note that, based upon our observation on many KGs, a blank node is usually ignored because the blank node does not provide any meaning and makes representation of a KG become more complicated. Therefore, in our study, the blank node is not covered.

Internationalized Resource Identifier (IRI) is a unicode string that comply the RFC standard and it uses to identify a resource as a unique identifier so that the resource could be referred [27]. A unique resource identifier (URI) and a unique resource locator (URL) are subset of IRI. Also, IRI also is a generalization of URI [9] since it supports wider range of the encoding characters [27]. The examples of IRI are as follows.

Organization A

`http://www.organizationA.org/resource/Global_Business`
`http://www.organizationA.org/resource/Smith`
`http://www.organizationA.org/term#employedDate`

Organization B

`http://www.organizationB.org/resource/Smith`

In these examples, the real-world entity, the person named "Smith", is represented by the IRI `http://www.organizationA.org/resource/Smith`. In the RDF specification [27], W3C recommended to use IRI, which begin with "http:" to identify the entity so that a unique entity can be identify. For example, "Smith" who works for the Organization A can be denoted by `http://www.organizationA.org/resource/Smith`, while the other "Smith" who works for the Organization B can be represented by `http://www.organizationB.org/resource/Smith`. Note that although a IRI begin with "http:", it is not necessary to have the actual resource on the networks or the Internet. In our study, we mainly use URI, which is a part of IRI, to represent the resource as the same manner of many research on KGs. Therefore, the remainder of the paper uses the term URI to represent the resource identifier for an entity or a relation.

Due to the long length of a URI, RDF allow to shorten a URI by defining a prefix for the particular namespace. From the examples above, we show to define prefixes as follows.

Prefix

```
@prefix  orgA-res:  http://www.organizationA.org/resource/
@prefix  orgA-term: http://www.organizationA.org/term#
@prefix  orgB-res:  http://www.organizationB.org/resource/
```

Based on above prefixes, we can shorten above example URIs as follows.

Organization A

```
orgA-res:Global_Business
orgA-res:Smith
orgA-term:employedDate
```

Organization B

```
orgB-res:Smith
```

Literal is a string used to identify a value. There are two types of literal: untyped and typed. On one hand, untyped literal is a string without identify type of the literal; in consequence, the untyped literal can be any string, e.g. "Smith"", "May 2017", "1.23" and "+81-1-2345-6000". On the other hand, typed literal is a string with the datatype can be identified. The datatype are string types such as language number, digit, date, etc. Since there are many types, it is necessary to identify the dataype of typed literal. In order to identify the datatype of typed literal, the extension of the markup are used to specific as shown the following examples.

`"John Smith"@en`	represents John Smith as in English
`"2017-06-01"^^xsd:date`	represents the date on 1^{st} June 2017
`"12345"^^xsd:integer`	represents 12345 as an integer

Due to the characteristic of a RDF triple as shown in Figure 2.1, a resource representation that can be used to describe each element of a RDF triple, is therefore depended upon the position, which is a subject or a predicate or an object, of the element.

- **Subject :** The subject can be represented only by a URI. Since the subject is the entity. it need to be identified as a unique resource, which is only represented by URI.
- **Predicate :** The predicate is also expressed by a URI. As shown in Figure 2.1, a predicate is a edge with its label. Different relations therefore can be expressed by different types of labelled edges.

TABLE 2.1: Examples of RDF triples

subject	predicate	object
orgA-res:Smith	orgA-term:employedDate	"2017-06-01"^^xsd:date
orgA-res:Smith	orgA-term:name	"John Smith"
orgA-res:Smith	orgA-term:department	orgA-res:Global_Business
orgA-res:Smith	rdf:type	dbo:Person
orgA-res:Smith	owl:sameAs	orgB-res:Smith
orgB-res:Alice	dbo:spouse	orgB-res:Smith

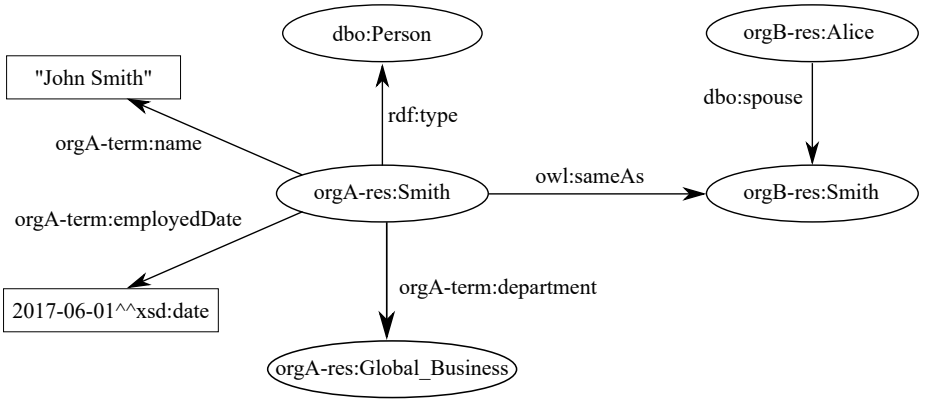


FIGURE 2.2: An Example of RDF Graph

- **Object :** The object can be a URI or literal. In contrast with the subject, an object is a information that fulfil the relation with its subject. Therefore, object allows the representation as URI or Literal. If it is URI, it express the relation between entities, subject and object. In case of literal, it describe the detail for subject, known as its description.

To demonstrate RDF triples and their corresponding RDF graph, an example of RDF triples are listed in Table 2.1. Based on the RDF triples in Table 2.1, KGs can derived as shown in Figure 2.2. Note that, in Figure 2.2 a ellipse uses to represent a URI as an entity, while a rectangle uses to denote and literal. A label on an edge uses to represent relation in the form of URI.

2.1.3 Knowledge Graph Assumptions

Based on the survey[13], the interpretation assumptions are required so that knowledge in KGs can be understood. In KGs, the relationships between entities and their description are store as knowledge, formally known as fact triples. It is obvious that a KG is incomplete. Therefore, non-existing triples in KGs have to be defined. Generally, there are two different assumptions: closed world assumption and open word assumption, which use to interpret the meaning of non-existing triples.

- **Closed World Assumption :** Closed world assumption simply assumes that non-existing triples imply relationships between entities are not existed. In other word, it assumes that knowledge in a KG is already completed. Although this assumption reduce complexity of the incomplete problem of a KG, its usage is extremely limited. Considering an extreme case shown in Table 2.1, there is no relation between `orgB-res:Alice` and `dbo:Person`. Based upon this assumption, it conclude that "Alice is not a person". Based on closed world assumption, it is correct; however, in fact, it is still not possible to conclude in this stage because of the lack of knowledge.
- **Open World Assumption :** Open world assumption supposes that non-existing triples cannot be interpreted and be treated as unknown. Considering the same example with the closed world assumption, there is no relation between `orgB-res:Alice` and `dbo:Person`. Under the open world assumption, we cannot conclude "Alice is not a person". Instead of that, we conclude that "Alice may be or may be not a person".

In RDF triples and the semantic web, open world assumption are generally made so that we can fulfil the knowledge with the new acquired one. However, close world assumption frequently use for learning the model for complete the knowledge graph such as negative sampling method [13].

2.2 Knowledge Graph Construction

Knowledge Graph Construction is to collect knowledge and build a KG from such knowledge. The approach for knowledge graph can roughly categorize into three approaches: 1) the manual approach, 2) the semi-automatic approach and 3) the automatic approach. In the manual approach, a KG is manually created by mainly using a human effort. In the semi-automatic approach, human efforts are put to craft rules or patterns so that

a KG is automatically constructed. In the automatic approach, a KG is automatically generated by using various of techniques, which reduces the human intervention. The further details and well-known projects for each approach are listed as follows.

2.2.1 Manual Approach

In the manual approach, triples are manually gathered and integrated in order to build a KG. There are two popular methods: curated method and collaborative method, for the manual approach. In the curated method, a KG is built by a closed group of experts while in the collaborative method, a KG is crafted by an open community. Each method is presented in the following sub-sections.

2.2.1.1 Curated Method

The curated method usually aims to build a specialized KG for some specific purpose because this method require a huge effort from human to build a KG. Furthermore, in the curated method a KG is built by using a closed group of experts. Such group of experts collect knowledge from a specific data resource and comply such knowledge to produce the KG. Example projects for the curated method is present in the following list.

- **Cyc/OpenCyc [28]**

  Cyc/OpenCyc is one of pioneer projects for constructing KG, former known as knowledge base. The aim of the project is to establish, collect and assemble ontology and ontology of common sense knowledge in daily life so that an intelligent agent or system can utilize such knowledge in Cyc/OpenCyc for reasoning in target applications. In this project more than million of axiom have been manually collected by a Cycorp Inc., which is a company investing to create a large-scale knowledge base [28]. Here, we presented some usage of knowledge. Some examples of knowledge in this project are "every tree is a plant" and "every plant dies eventually" [29]. Such knowledge, refereed as rules, can be used for inferencing and reasoning in order to discover a new knowledge. For example, if we known that cherry blossom is a tree, based on the example rules, we can entail that "cherry blossom dies eventually".

- **WordNet [30]**

  The WordNet project aims to construct a KG, which is also known as lexical resource. In WordNet, a collection of lexicon and their semantic relationship are

stored as knowledge. Fundamentally, there are six main semantic relationships between lexicons in the KG as follows [30].

- **Synonymy** is the relationship which identify the identical semantic relationship between lexicons. For example the lexicon "good" and the lexicon "well" holds the synonymy relation because their meanings are identical.
- **Antonymy** is the relationship that describe the opposite semantic relationship between lexicons. For example the lexicon "good" and the lexicon "bad" holds the antonymy relation because their meanings are opposite.
- **Hyponymy and Hypernymy** are the relation that identify subset and superset between lexicons respectively. Sometimes, these relationships refer as is-A relation. For example "Dog" is a hypernym of "Animal", because every dog is a kind of animal. Here, Hyponymy and Hypernymy are considered as a semantic relation (is-A) because they are inverse of each other.
- **Meronymy and Holonymy** are the relation that describe the partOf and is-partOf relationship between lexicons respectively. For example, given the lexicon "'door' is a meronym of "house", it can be interpreted as "door is a part of house". Consequently, this relation frequently refer as partOf relation. Here, Meronymy and Holonymy are considered as one semantic relation because they are inverse of each other.
- **Troponymy** is a relation that present co-occurrence between lexicons. For example, "to bite" is a troponym of "to eat" since the activity "to bite" is doing "to eat" in some manner.
- **Entailment** is a relation that infer the one lexicon cause the result to the other one. For example, "to cry" is entailed by "to tears flow" because when crying, your tears must flow.

WordNet is the linguistic domain lexicon resource, which contain many relationship between lexicons. Therefore, the construction process needs a group of experts in the linguistic domain to collect and build this KG so that the quality can be controlled. Recently, WordNet becomes the most valuable KG resource for the linguistic community. Therefore, it is widely used in many Natural Language Processing (NLP) applications [31].

- **UMLS [32]**

The Unified Medical Language System (UMLS) project is also a specialized domain KG. It aims to construct the KG regarding the biomedical-related domain. Also, the project developed by a expert group at US National Library of Medicine [32]. In the UMLS project, more than ten million of triples relationship among almost

million concept are stored in the repository. Both ontology and taxonomy are integrated from various sources by the creator group.

As shown in the above projects [28, 30, 32], the curated method mainly purpose for the specific target for specific purpose. Moreover, the limitation of this method is that it requires the curator in the group to collect, manipulate and update the knowledge in a KG. Although the curated method can create the small portion of knowledge for the specific purpose, it consume a lot of resource such as time, human efforts. Specifically, in the large-scale project like Cyc/OpenCyc. The time estimation for complete the project is more than 350 man-years (as estimated in 1986) [33]. However, the new knowledge emerge very. As a result, we can not accurately estimate time.

2.2.1.2 Collaborative Method

The collaborative method aims to manually build a KG by people similar to the curated method. However, instead of a closed group of experts, the collaborative method allows an open community to collect knowledge and create a KG. This collaborative method is known as crowdsourcing [34]. In crowdsourcing, online communities play a significant role in the KG construction process. Conventionally, an existing platform is published online so that Internet users, who are interest in contributing to the project, can access and help the community to create a wide-range of KGs. Two examples of the collaborative method project are presented as follows.

- **Freebase [11]**

  Freebase is one of primary large-scale collaborative KGs, which allows any Internet users to manipulate knowledge in the project directly. Basically, the Freebase project acquired knowledge from several online data sources for constructing knowledge in the structured format. Two main data sources are harvested from the online structure data resources and user generated content. The online structure data is a data that available online in the structure form such as Wikipedia. A user generated content source is the source, where any Internet users create the content and added them to Freebase. Nowadays Freebase compose of million knowledge because it started in 2007; however, the project was announced to shutdown recently [35]. Currently, Freebase remains still accessible; but, the project was set as read-only. This means that any new knowledge or content cannot be added to the project and any users cannot contribute to the project. Fortunately, knowledge of this project is migrated to Wikidata, which is also the collaborative with the same manner as freebase. The detail of Wikidata is presented in the following project.

- **Wikidata [36]**

  Wikidata is an open KG project that enables Internet users to add, remove and update knowledge in the same manner when the Freebase project was active. Wikidata as the sister project of Wikipedia stores the knowledge in the form of structure data and provide API to online community so that the communities can get involved with the project [36]. This idea is similar to Wikipedia project, where any individual user can provide or manipulate content on a Wikipedia page. Many users or parties can collaborate together to create knowledge in Wikidata project as the crowdsourcing. Also, the Wikidata project is an open access project, which mean that any users can freely collaborate with the community. Thanks to crowdsourcing aspect, more than 25 million knowledge are included into the Wikidata project.

Comparing with the curated method, the scalability of the collaborative method are far better because of open communities. Although, the collaborative method can create very rich KG, the correctness of KG is still an issue. Any individual can directly manipulate the knowledge graph. Thus, we cannot ensure the quality of the knowledge of the KG. Furthermore, there is no quality measurement in the Wikidata [36]. Moreover, the scalability in the manual approach could be partly solved by the collaborative method. Still, knowledge emerge everyday. Therefore, we needs the approach that can done automatically or use less human effort in order to deal with the practical situation in the big data era.

2.2.2 Semi-automatic Approach

In the semi-automatic approach, hand-crafted rules or regular expression rules are manually defined and then such rules are used to automatically extract knowledge from structure data in order to generate triples of a KG. Well-known KG projects, which apply the semi-automatic approach are DBpedia, YAGO and Freebase. The details of each project are in the following list.

- **DBpedia [10]**

  DBpedia is the project to construct a KG by extracting knowledge from structure content, specifically Wikipedia. Wikipedia is a free online encyclopaedia platform that allows any individual users to collaborate each other for creating the web content. In Wikipedia, a user can add update and remove content on the project directly. Also, one of the advantage of Wikipedia is that a number of hyperlink among pages are fruitful. Currently, there are many content on Wikipedia since

FIGURE 2.3: The sample screenshot of the Wikipedia page [37]

Wikipedia implement on the crowdsourcing method to gather the content. In Figure 2.3, the snapshot of the example of the Wikipedia page is presented. As shown in the figure, a Wikipedia page consists of two main parts: 1) description text and 2) infobox. The description text is text which given finer details about the pages, while the infobox provided the significant information of the page in the well-defined structure format.

DBpedia, as categorized in the semi-automatic approach, mainly extract knowledge from an infobox and some structure markup of the Wikipedia such as abstract or links. The process of the extraction is straightforward. In the process, an entity and a relation are extracted. For an entity, the URI representing the entity of the page is construct under the namespace of DBpedia (`dbr:`). For example, considering the example Wikipedia entity "Barack_Obama" in Figure 2.3, the URI representation that corresponded to this entity is `dbr:Barack_Obama`. For a relation, properties are extracted under the namespace of DBpedia (`dbo:` or `dbp:`) in the same manner of entities from the infobox of the Wikipedia page. For example, "Born" in the infobox in the example is extracted and then by applying the rule, "Born" is mapped to the property `dbo:birthDate` or `dbo:birthPlace` of DBpedia. Note that, the difference between `dbo:` and `dbp:` is that `dbp:` is a direct map from the infobox without integrating into DBpedia ontology, while `dbo:` resolves the integration problem.

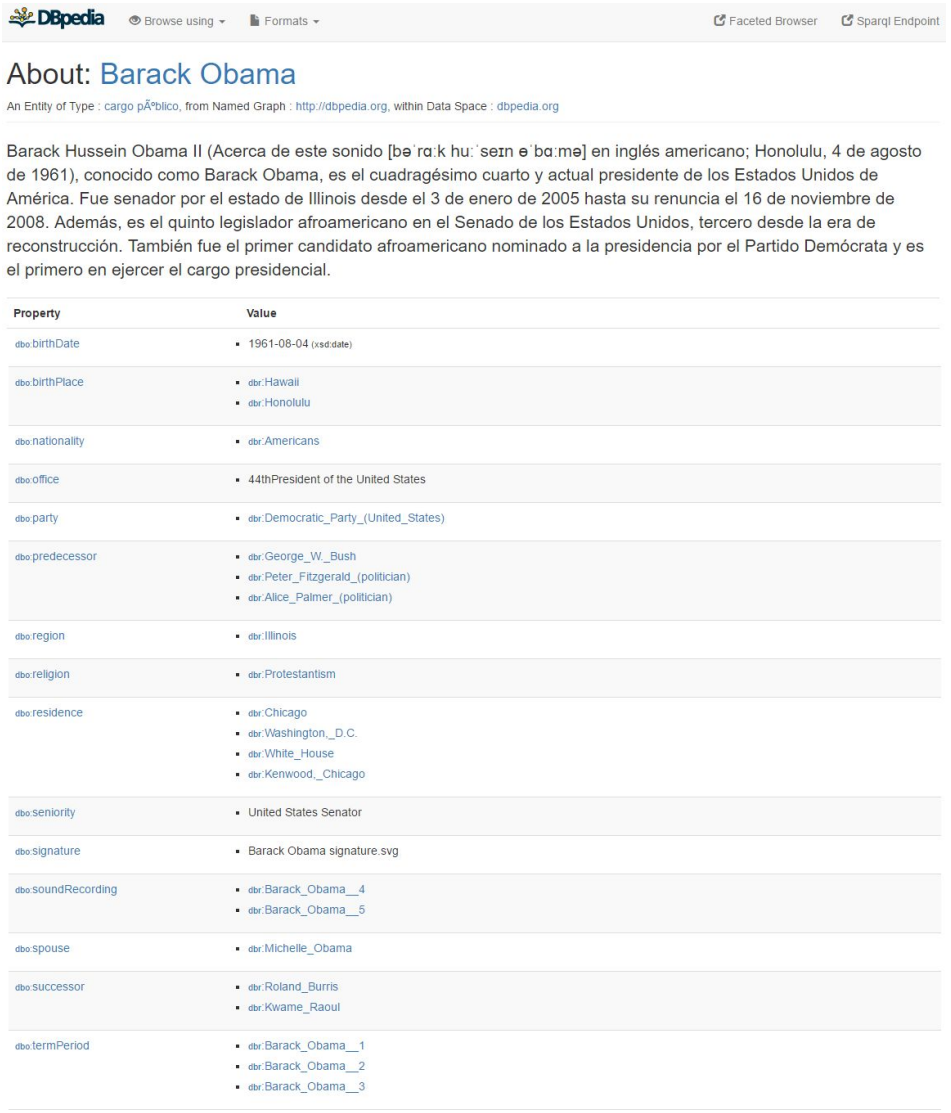


FIGURE 2.4: The sample screenshot of the DBpedia page `dbr:Barack_Obama` [38]

Apart from the infobox, the extractor of DBpedia also extract the content in the HTML markup format such as abstract of the Wikipedia page, the Wikipedia links to other pages, etc. As a result, the DBpedia entity can be created. In Figure, the snapshot of DBpedia `dbr:Barack_Obama`, which corresponds to Wikipedia page "Barack_Obama" is illustrated in Figure 2.4.

Nowadays, DBpedia becomes one of the most important KGs. Because of the fruitfulness of entities in DBpedia, this KG therefore gains more and more attention from many KG publishers as well as researcher communities, e.g. semantic web community, NLP community. Furthermore, due to the quality of DBpedia and wide-range of language, which DBpedia provided, it becomes the multilingual KG with the high quality [39]. Consequently, many KGs frequently connect their knowledge, including entities and relationships, to DBpedia. As a result, DBpedia becomes the hub of KGs [40].

- **YAGO [12]**

  YAGO is a project similar to DBpedia. It is extract structure knowledge from Wikipedia, e.g infobox, category, redirected, and Wordnet, e.g. synset, and Geoname [41] in order to create a KG. The project reports that YAGO contains more than 1 million entity and 5 million fact connecting such entities. Furthermore, in the YAGO project, interlinking links between DBpedia and YAGO are provided. Specifically, YAGO provides links to DBpedia ontology [10] and SUMO ontology [42]. In the study [12], the empirical evaluation of the correctness of YAGO is conducted. The result show that YAGO achieved an accuracy of 95%, which is highly reasonable in the KG construction.

- **Freebase [11]**

  We discussed Freebase in aspect of manual KG construction in previous Section 2.2.1.2. In fact, Freebase also extracted knowledge from the structure resources such as DBpedia. The idea is similar to DBpedia and YAGO. Freebase derived the information in the info box of Wikipedia to create the fact triple. Note that, such create triple are allowed to manually manipulate by an user. Therefore, Freebase project is the hybrid combination of the semi-automatic approach and the manual approach.

Although the semi-automatic approach can construct effectively, some information or knowledge still left on the natural language text. For example, considering the Figure 2.3 and 2.4, the Wikipedia page described the entity "Barack_Obama" contains much more knowledge than DBpedia provided. This characteristic occurs because DBpedia do not directly extract information from the unstructured text, specifically natural language text. Consequently, many knowledge was left as text.

2.2.3 Automatic Approach

In the automatic approach, a KG is directly built from natural language text. An automatic approach extracts knowledge from natural language text and then creates the KG by such knowledge. Generally, The process of the automatic approach could viewed as shown in Figure 1.3. In the automatic approach, there are two main method: 1) schema-based method and 2) schemaless-based method. The schema-based method populates knowledge as defined in the predefine ontology or vocabulary. In contrast with the schema-based method, the schemaless-based method extracts the knowledge directly from the natural language text without a predefine vocabulary or an ontology. The details of the schema-based method and the schemaless-based method are as follows.

2.2.3.1 Schema-based Method

The schema-based method aims to build a KG by using a predefine ontology and a finite set of vocabularies as control constraints. In this method, a set of entities and a set of relations are defined; specifically relations are fixed by a set of predefine vocabularies. When extracting knowledge from the natural language text, this method focuses on extracting the knowledge that corresponding to the given vocabularies. As a result, a set of vocabularies plays significant role in this method. The details of systems that categorized into the schema-based method are presented as follows.

- **Never-Ending Language Learning (NELL)** [43]

  NELL is a never-ending learning project, which aims to extract knowledge from the web [43]. The main idea of NELL is to create a set of triples with its ontology by gathering information on the Internet. The main concept of NELL is that NELL will accumulate knowledge over time and due to knowledge acquisition, it become better and better to learn a new knowledge. In NELL, predefined ontology is defined and some bootstrapping triples together with a set of constraints, including domains and ranges of a relation and mutual-exclusion condition, are given to NELL as the bootstrapping learning data. Such bootstrapping learning data are used to learn constraints for extracting a new belief. One of the prominent feature of NELL is that it uses multiple extractors and validators to learn and verify a new knowledge. This strategy called "couple learning'' . Specifically, NELL uses one extractor to support or argue other extractor in order to populate new knowledge from a set of beliefs. NELL project started since 2010 and has been continuously running since then. Currently NELL contains more than 50 million candidate beliefs and more than 3 million beliefs, with high confident as knowledge.

- **LODifier** [44]

  LODifier is a project to generate a KG from unstructured text. In LODifier, Discourse Representation Structures (DRS) [45] that represents the meaning of a sentence from unstructured text is extracted by the statistical parser C&C and the semantics construction toolkit Boxer [46]. Also, many NLP systems, including NER system, Coreference Resolution system and Word Sense Disambiguation, are applied and the results are mapped to RDF triples. Furthermore, RDF WordNet [47] are used as the predefine ontology in order to directly map result, which is a vocabulary from the synset, to a RDF triple without considering other KGs.

- **KnowledgeStore [48]**

  KnowledgeStore is a general-purpose system to extract store and manage knowledge. To achieve the goal the system builds on the state of the art NLP applications, e.g. Tagging system and Coreference Resolution system. The architecture of this system consists of four layers: 1) resource layer, 2) mention layer, 3) entity layer and 4) context layer. Each layer of the system designs to deal with resource, mention, and entity and context. The resource is where an entity is acquired. Mention is the specific object, which considered in the text, while an entity is a unique object, which the mention map to. This means that different mention can map to the identical entity. The context describes the entity based on specific context such as time, location, etc. Based on this architecture, the KnowledgeStore can well manipulate store knowledge; however, integration of knowledge to other KGs still does not take into account. As a result, the usage of the knowledge is very limited to the local KG, e.g. Trentino [48].

- **Kriz et al. [20]**

  Kriz et al.[20] propose transformation of unstructured text to a RDF triple by using NLP to extract triple and then uses its own ontology to represent an extracted triple. In their approach, the syntactic structure of a document is exploited by NLP applications and then the predefine ontology in the study [49] is used as the schema when populating knowledge. Even though this approach could extract the entity and the target relation, such entity still do not integrate to other KGs.

- **Knowledge Vault [50]**

  Knowledge Vault is a project to build a large-scale probabilistic KG based on the combination of content extracted from the web documents. In Knowledge Vault, a predefined ontology, including entity type and predicate, is given as the fixed schema similar to other approaches. The main diffrence between Knowledge Vault and other systems is that the noisy during the extraction process is considered; in consequence, Knowledge Vault become more robust. To extract knowledge, several extractors are used to gather knowledge from various types of the data, e,g natural language text, tree (DOM), table, etc. In here, we mainly focused on natural language text. To extract knowledge in natural langue text, the following processes are performed. Firstly, entities over all documents are recognized. Secondly, each entity are resolved and linked to KG by using the NLP suit tool [51]. Thirdly, the supervise learning technique, named distant supervision [52], is used to learn the relationship between entities based on the seed triples to find the relationship between entities.

Given the predefine vocabularies, it avoid the heterogeneous problem when populating knowledge; however, the acquire knowledge is very limit due to the condition of predefine vocabulary. To reduce these limitation, the other method, schemaless-based method, is proposed to extract knowledge. The details of the schemaless-based method are described in the following section.

2.2.3.2 Schemaless-based Method

The schemaless-based method, also known as the Open Information Extraction task [53], aims to build a KG without requiring pre-specified ontology and vocabulary. This method therefore have to automatically identify arbitrary relations and extract such relations. In this method, the lexicon plays an important role in the extraction process because the extraction is performed at lexical level. At the lexical level, there is no require any schema or vocabularies. As a result, structure of sentences, obtained by a parsing system, significantly helps this method to extract triples. In the following, the systems in the schemaless-based methods are presented.

- **TextRunner [53]**

  TextRunner is a scalable open IE system, which extract triples from text and assign probability to each triple. In the TextRunner system, there are three main components: 1) extractor component 2) Self-Supervised Classifier component and 3) assessor component. The extractor component, namely Single-Pass extractor, read through the entire documents and separately process each sentence in the document to produce the extraction results; however this component requires seed triples, as supervision. The Self-Supervised Classifier component module is developed to classify whether extracted triples are trustable or not and then trustable triples are given back to the extractor component as seed triples. The assessor component validates and judges the probability score for each extraction result.

- **Reverb [54]**

  ReVerb is an open information extraction system that extracts a triple from a given sentence by using syntactic patterns and lexical constraints. In ReVerb, a relation phrase is identified by using syntactic patterns and lexical constraints. For the syntactic patterns, the prior knowledge regarding to the language are provided such as “’ phrase relation must start with verb and end with the preposition’’ . Such syntactic pattern used to identify useful syntactic to extract triples. On the other hand, the lexical constraints help to generalize the extracted results. For example, some relation between entity might be extract with the long phrase

of relation, meaning that it is too specific. To avoid such problems the lexical constraints is applied. Then, entities, noun phrase that correspond to relation phrase are assigned. Finally, the confident score for the extraction triple is given and is adjusted.

- **OLLIE [55]**

  OLLIE system is an open information extraction system, which has been improved from ReVerb [54]. The OLLIE system works in similar manner to ReVerb; however, the OLLIE system can extract more finer details in the local context of a sentence. ReVerb constraints mainly focused on the main verb of the sentence. As a result, some latent relation is missing. In OLLIE, patterns are not limited to a main verb of a sentence or a local context of a sentence. For example, "The President of United State Donald Trump announce the new regulation". ReVerb focuses only the relation "announce", while OLLIE can extract the "President of United State" relation.

- **Exner et al. [56]**

  Exner et al. proposed the pipeline system to take natural language, specifically Wikipedia articles as input and yielded the KG triple as the output. The idea of the system is to use the state-of-the-art natural language processing tools, e.g. a semantic role labeler (SRL) and name entity resolution and so on, to extract the relation from text. Then, the system links extracted entities to KG entities and determines the statistical pattern of the text predicate and the KG predicate based on each subject-object pair, and then forms a link between identical predicates. Since the study used a SRL tool to analyze the relation without the predefined vocabulary. Therefore, the system has been categorized into the schemaless-based method. Nevertheless, the knowledge integration with the predefined vocabulary is applied as well. Therefore, this approach could also be viewed as the schema-based method.

The schema-based method help us to populate knowledge for the existing KG; however, we can populate some of knowledge from text. In the schema-less-based method, lexicon term play an important role in the knowledge extraction process and it is not depended on any vocabulary; in consequence, the schema-less-based method can populate more knowledge but there are not much useful because of the heterogeneous problem. In this research, we aims to take both advantages of the schema-based and schema-less-based method in order to build the KG with the wide coverage.

TABLE 2.2: The Summary of the Knowledge Graph Construction projects

Approaches	Methods	Projects
Manual	Curated	Cyc/OpenCyc [28], WordNet [30], UMLS [32]
	Collaborative	Freebase [11], Wikidata [36]
Semi-Automatic	-	DBpedia [10], YAGO [12], Freebase [11]
Automatic	Schema-based	NELL [43], Exner et al. [56], KnowledgeStore [48], Kriz et al. [20], Knowledge Vault [50], LODifier [44]
	Schemaless-based	TextRunner [53], Reverb [54], OLLIE [55], Exner et al. [56]

In this section, we reviewed and surveyed on various approaches and methods in each approach for the KG construction task. To sum up our discussion so far, we present the summary of the approaches, the methods and their corresponding projects in Table 2.2.

2.3 Knowledge Graph Completion

KG completion is a task to fill a missing knowledge in KG. As we known that KG is incomplete, KG completion uses the current structure or knowledge in KG to find whether there are any other missing relation in KG or not. This task is also widely-known as Link Prediction [13] since the scenario in the KG completion is that the missing linkings between entities are predicted. In the KG completion task, there are many approaches [57–62] proposed so far. A traditional approach for KG completion is to use association rule to mine to rules for filling the knowledge. In contrast with the traditional approach, a modern approach use the embedding method to embed an entity in the KG so that the links between entities can be predicted. In the following list, AMIE [57], which is a traditional approach, and TransE [59], which is a modern approach, are presented since they are fundamental of these approaches.

- **AMIE** [57]

  AMIE is a rule-mining system, which extract logical rules, specifically Horn clauses [63]. The AMIE system design to work on Open World Assumption. If the logical rules do not contradict with association rules that discover by the system, a triple

cannot be identify whether it is correct or not. In order to create rules, an efficient association rule mining algorithm is proposed to deal with the large scale of KG. One major contribution is to simulate negative triples in KG. Since association rule mining algorithm requires the negative samples when learning rule, Closed World Assumption is applied in the association rule learning state. The example of learning rule is "isDirectedBY(movie, person)" [57]. After acquiring the rules such rules are used to populate a new triples which not existing in the current KG.

- **TransE** [59]

  TransE is a pioneer research project for KG embedding task. The KG embedding task is to represent each element of triples in KG into the continuous vector space as similar in the word representation [64]. After embedding elements in KG in the distributed representations, such representation can be used to predict the missing the relation in the KGs. This goal can be accomplished because the objective function of KG embedding is to try to minimize the error cause by a particular relation and two entities and maximize the non-existing relation. More Formally, given a triple (h, l, t), the main assumption of the KG embedding is that $h+l \approx r$. Therefore, the objective function of transE is to optimize Equation 2.1.

$$\underset{h,r,l,h',l',r'}{\text{minimize}} \qquad | \; d(h+l,r) \; - \; d(h'+l',r') \; | \tag{2.1}$$

  where (h, l, r) is an existing triple, (h', l', r') is a non-existing triple, $d(\cdot)$ is a distant metric, e.g, euclidean distance.

  Based upon the inspiration of the TransE method, many studies further investigate the KG embedding method to tackle with the KG completion task. As a result, the variation of the TransX models. e.g., TransH[60], TransR[61] and TransG[62], are proposed.

As discussed above, KG completion is to predict the missing relation between entities in a KG. Techniques used in this task is totally different from the KG construction. One prominent aspect is that the KG completion does not get involved with other natural language text when completing knowledge in a KG. This implies that the external knowledge resources have not been used. Furthermore, KG completion approaches simply find the missing link; however, for a non-existing entity is out of scope of this research topic.

2.4 Knowledge Graph Integration

In KG construction, each KG expresses entities and relationship by its own vocabularies. Consequently, even once knowledge is added to a KG, it cannot be used efficiently due to the problem of heterogeneity. In the heterogeneous problem, a publisher uses his/her vocabulary to describe things in his/her KG; in consequence, the inconsistency between identical things occurs. For example, one may use "placeOfBirth" to express the place where a person was born, while another may describe the same meaning with other way, e.g. "birthPalce". Resolving the heterogeneous problem can increase searchability over KGs. As a result, we can use KG more efficiently. In this following section, we discuss the KG resolution problem, including entity linking and predicate linking and then review the similarity measurements that usually use for solving the KG resolution problem.

2.4.1 Knowledge Graph Resolution

The KG resolution is to identify identical things between KGs and provide an identical link between such things of KGs. In the KG resolution task, there are two tasks: entity linking and predicate linking. The survey on these two tasks are reviewed and discussed in the following subsections respectively.

2.4.1.1 Entity Linking

Generally, entity linking, also known as instance matching, object co-reference resolution, object consolidation, duplicate record detection, or entity resolution, is the problem, which aims to discover two identical objects or records in the same data resource or between difference KG resources. Due to its necessities, many approaches are proposed. The proposed approaches could roughly divided into two categories: a Domain-Dependent category and a Domain-Independent category.

In the domain-dependent category, prior knowledge regarding an interested domain or a schema of a data resource is required. In this category, there are many proposed approaches such as Silk[65], AgreementMaker[66] and Zhishi.Links[67]. In the Silk approach, three steps are introduced in order to discover and manipulate the matching between different data resources. For the first step, a discovery engine computes identical links between different data resources. Then the second step is to fine-tune and evaluate the correctness of such links. The third step aims to manipulate the links when changing of data resources is applied. AgreementMaker is a resolution system for

matching both ontologies and entities. In AgreementMaker, three phases are performed respectively in order to match between entities. Firstly, potential candidate pairs of entities are selected by similarity between labels of entities in the candidate generation phase. Secondly, similarity between entity pairs are extracted during the disambiguation phase and then entity pairs are verified, whether they are correct match or not, in the matching phase. In Zhishi.Links, which is an enhanced version of Silk, some weighting schema is applied to improve the matching results between entity pairs. Even though approaches in the domain-dependent category yield reasonable results, they still require an end user to drive the prior knowledge.

In the domain-independent category, the prior knowledge regarding a considered domain or a schema of data resource is not necessary. Many approaches are introduced in this category. ObjectCoref [68] is a self-learning system, which detects identical objects by iteratively learning discriminative property. SERIMI [69] selectes high entropy predicates, which usually possess abilities to discriminate identical objects, in order to select entity pairs and then build a binary classifier to classify whether such entity pairs are correctly matched or not. SLINT [70] and SLINT+ [71] select an useful predicates between the data source and the target resource for generating potential candidate pairs of entities and then such candidate pairs are verified whether they are identical or not. Rong et al. later introduce an entity matching approach using similarity metrics [72]. Several types of the similarity metric are proposed to extract similarity features between candidate entities and then a binary classifier is employed to justify that candidate pairs are matched. Although those approaches success to identify identical entities without using prior knowledge, such approaches still execute entity matching on two given-specific data resources. Consequently, the immensity problem is still not considered yet.

2.4.1.2 Predicate Linking

Predicate linking, also known as ontology integration and synonym identification, is to map a predicate to its identical predicate. Most of studies [73–75] focus on predicate linking between KG triples. Abedjan et al. [73] proposed the association rule mining to learn associated patterns in KGs and used the patterns to discover identical predicate pairs. Zhao et al. [74] introduced the statistical graph patterns to group candidate predicates and used the string-based similarity approach to verify whether such candidates are identical. Zhang et al. [75] proposed using statistical knowledge patterns to identify identical predicates in the KG. Based upon the results, their approaches [73–75] could identify identical predicates between KGs. Nevertheless, it cannot be applied in a straightforward manner for our task, since we focus on forming a link between a predicate in the text triple and its matching counterpart in a KG triple. Generally, text

triples are ambiguous and sparse than KG triples. As a result, properties for building the statistical knowledge pattern between text triples and KG triples become missing.

The most applicable study for the predicate linking task in our study is Exner' s study [56]. Exner et al. uses the state-of-the-art natural language processing (NLP) tool to link entities of text triples to KG entities and determines the statistical pattern of the text predicate and the KG predicate based on each subject-object pair, and then forms a link between identical predicates. Although this approach [56] could avoid the ambiguity of an element in a text triple, generating statistical patterns might not cover all possible patterns; consequently, some statistical patterns are missing.

2.4.2 Similarity Measurement

Similarity measurement often uses to identify identical entities or relations in different KGs. The similarity measurement is necessary for KG integration task because the integration process is required a measurement metric, which determine the similarity between things so that intralinking and interlinking in KGs can be established. Without the similarity measurement, the identical relation between things in KGs cannot identify. In this section, we reviewed various similarity measurements. We categorized similarity measurement into three categories: 1) Word-based Similarity, 2) Document-based Similarity and 3) Vector-based Similarity. The details of each category are presented in the following subsections.

2.4.2.1 Word-based Similarity

The word-based similarity measurement is to compute the similarity at the word level. The word-based similarity use when comparing each property values [76]. For example given, the property value "Barack Obama" and "Obama", such similarity between words can compute by the word-based similarity metric. So far, several word-based similarity metrics have been proposed to compute the similarity between words. In this review, we mainly focused on the common one as presented in the following list.

- **Levenshtein Distance** [77]

  Levenshtein Distance is the word similarity that consider distance between words. In the Levenshtein Distance, each word is treat as a string or a sequence of characters. Then, the minimum cost needed to transform one string into the other string is computed as the similarity score. The operations for transforming string include insert, delete and substitute of one string character in the sequence. For

Levenshtein Distance, Edit distance is a special case, in which set all costs, insert cost, delete cost and substitute cost, to 1. This special case frequently use to measure words, which usually contains misspelling. The definition of Levenshtein Distance is defined as follows.

Definition 2.2. Given a string S_1, S_2, a set of operation sequences Op, which map the string S_1 to the target string S_1, $(op : S_1 \mapsto S_2)$ and a operation cost function, which map operation sequences to cost value, $(w : op \mapsto \mathbb{R})$, the Levenshtein Distance is a similarity $\delta : \mathbb{S} \times \mathbb{S} \mapsto [0, 1]$, where $\delta(S_1, S_2)$, is the least cost of the transformation sequence as the following Equation.

$$\delta(S_1, S_2) = \min_{op_1, op_2, op_3, .., op_n} \quad w_{op_i} \tag{2.2}$$

- **Jaro-Winkler** [78]

  Jaro-Winkler distance is the string similarity that measures the number of proximity of two string sequences and computes the similarity between these strings. The Jaro-Winkler is an improvement of Jaro distance in the study [79]. The Jaro distance usually used to compute the similarity between proper names since the proper names tend to contain similar mistake [79]. The definition of the Jaro distance is defined as follows.

  Definition 2.3. Given a string S_1, S_2, the Jaro distance is a similarity $\delta : \mathbb{S} \times \mathbb{S} \mapsto [0, 1]$, where $\delta(S_1, S_2)$, is the distance between two strings as the following Equation.

$$\delta(S_1, S_2) = \begin{cases} 0, & \text{if } \ m = 0 \\ \frac{1}{3} \times (\frac{|m|}{|S_1|} + \frac{|m|}{|S_2|} + \frac{m-t}{m}), & \text{otherwise} \end{cases} \tag{2.3}$$

  where, m is the number of string that matched and t is a half of the number of transition strings between S_1 and S_2.

  The Jaro-Winkler distance improved the Jaro distance by considering the weight of the prefix string that match at the beginning of the prefix strings. Consequently, the Jaro-Winkler distance can derive in the following definition.

  Definition 2.4. Given a string S_1, S_2, the Jaro-Winkler distance is a similarity $\delta : \mathbb{S} \times \mathbb{S} \mapsto [0, 1]$, where $\delta(S_1, S_2)$, is the distance between two strings as the following Equation.

$$\delta(S_1, S_2) = \delta(S_1, S_2) + (l_p(1 - \delta(S_1, S_2))) \tag{2.4}$$

  where l_p is a length of prefix strings.

- **N-gram** [80]

  N-gram similarity is a string similarity, which measures a number of common grams that two strings. In N-gram, n is a number of gram sizes, that is considered in the set. For example, given the word "apple", the set of grams of 2 and 3 grams are as follows: $2-gram = ap, pp, pl, le$ and $3-gram = app, ppl, ole$.

  Definition 2.5. Given a string S_1, S_2, the n-gram similarity is a similarity $\delta : \mathbb{S} \times \mathbb{S} \mapsto \mathbb{R}$, where $\delta(S_1, S_2)$, is the common n-grams between two strings as shown the following Equation.

$$\delta(S_1, S_2) = |ngram(S_1, n) \cap ngram(S_2, n)| \tag{2.5}$$

  where, $ngram(s, n)$ is the function returning a set of n-gram of the string s, of gram size equals n. Generally, the above n-gram function returns real value. In order to change $\delta : \mathbb{S} \times \mathbb{S} \mapsto \mathbb{R}$ to $\delta : \mathbb{S} \times \mathbb{S} \mapsto [0, 1]$, the normalization of the n-gram is presented in the following Equation.

$$\delta(S_1, S_2) = |\frac{|ngram(S_1, n) \cap ngram(S_2, n)|}{min(|S_1|, |S_2|) - n + 1} \tag{2.6}$$

  where n is a number of gram size in the setting.

- **WordNet Similarity** [81]

  WordNet Similarity is a word similarity, which capture the semantics relatedness between words. In the word similarity described above, they do not take the meaning of words into account. The textual form or surface of words as string is considered. Consequently, they fail to capture the semantic relation between words. For example, the word "good" and the word "well", they are semantic equivalent; however, applying above metrics cannot achieve good results of this words due to different surface forms. There are many WordNet-based similarity score. In this study, we focused on the primary one, which was proposed by Wu and Palmer [81]. In this method, the depth between concept structure of words are used to calculate the relatedness between words. The calculation process is done by the following Equation.

  Definition 2.6. Given a string S_1, S_2, the Wu-Palmer similarity utilize WordNet as the prior knowledge to computed relatedness between words by the function $\delta : \mathbb{S} \times \mathbb{S} \mapsto \mathbb{R}$, where $\delta(S_1, S_2)$, as follows.

$$\delta(S_1, S_2) = \frac{2 \times d(lcs(S_1, S_2))}{len(S_1, lcs(S_1, S_2)) + len(S_2, lcs(S_1, S_2)) + 2 \times d(lcs(S_1, S_2))} \tag{2.7}$$

where, $d(lcs(S_1, S_2))$ is the function returning a distance, referred as depth, between words in the WordNet structure

.

In the word similarity measure, the similarity between two words are considered; however, in KG integration, the similarity between words is not enough to identify similarity between knowledge. Observing knowledge, a compound of words as documents, word similarity is not effective to capture the global similarity. In the next section, the document-based similarity is reviewed.

2.4.2.2 Document-based Similarity

The document-based similarity measurement is to compute the similarity at the document level. The document-based similarity use when comparing overview of entities because each entity could be viewed as a collection of words or sentences. Currently, there are many available approaches for computing similarity between documents. The review below briefly presented the widely-used approaches.

- **TF-IDF Cosine Similarity** [82]

  TF-IDF Cosine Similarity is cosine similarity between bags of words of two documents. The term TF-IDF is stand for Term Frequency and Inverse Document Frequency. As Term Frequency, TF count occurrences of words over a document and create bags of words with their frequency. In contrast with TF, IDF count occurrences of words over all documents and create inverse of frequency of all documents. Therefore, TF-IDF consist of two assumptions. The first assumption is that two documents, which share more words, appear frequently, tend to be similar. The second assumption is that two documents, which share rare words tend to be similar. These two assumption used to weigh between generalization and specialization. The generalization is learned by the TF-term, which considered how many similar words share, while the specialization is obtained by the IDF-tern, which considers that not every words have the same weight, in particular rare words should gain more weight. The definition of TF-IDF cosine similarity is defined in the following definition.

  Definition 2.7. Given a collection of document string D_1, D_2, D_3,$\cdots$, D_n and each document represented by a bag of words,$W = w_1, w_2, w_3, \&, w_m$, where $w_j$is the word j in the vocabulary set, the TF-IDF function defined as $\delta : \mathbb{D} \times \mathbb{D} \mapsto [0, 1]$, where $\delta(D_i, D_j)$ can be computed in the following Equation.

$$\delta(D_i, D_j) = \frac{\sum_{i=1}^{N} w_{i,D_i} w_{i,D_j}}{\sqrt{\sum_{i=1}^{N} w_{i,D_i}^2} \sqrt{\sum_{i=1}^{N} w_{i,D_j}^2}} \tag{2.8}$$

$$w_{t,D_k} = tf_{t,D_k} \cdot log \frac{|D|}{|\forall x \in D | t \in x|} \tag{2.9}$$

where t is a term in set of bag of words, which has N words, tf_{t,D_k} is a number of occurrence of the term t in the document D_k, $|D|$ is a number of all documents. In Equation 2.9, the first represented the concept of TF term, while the second term present the idea of IDF term.

.

- **IDF Cosine Similarity** [72]

  IDF Cosine Similarity is cosine similarity between bags of words of two documents. This similarity can be considered as the special case of the TF-IDF, where all term, which appear more than 1 are standardized to 1. The idea of IDF cosine similarity is to observe the rare words directly since the rare words generally play an important role to identify the similarity between entity in KGs [72]. Therefore, IDF cosine similarity simply removes the TF term form Equation 2.8-2.9. The definition of IDF cosine similarity is defined in the following definition.

  Definition 2.8. Given a collection of document string D_1, D_2, D_3,$\cdots$, D_n and each document represented by a bag of words, $W = w_1, w_2, w_3, \&, w_m$, where $w_j$is the word j in the vocabulary set, the IDF function defined as $\delta : \mathbb{D} \times \mathbb{D} \mapsto [0, 1]$, where $\delta(D_i, D_j)$ can be computed in the following Equation.

$$\delta(D_i, D_j) = \frac{\sum_{i=1}^{N} w_{i,D_i} w_{i,D_j}}{\sqrt{\sum_{i=1}^{N} w_{i,D_i}^2} \sqrt{\sum_{i=1}^{N} w_{i,D_j}^2}} \tag{2.10}$$

$$w_{t,D_k} = \begin{cases} log \frac{|D|}{|\forall x \in D | t \in x|}, & \text{if } \ t \in D_k \\ 0, & \text{otherwise} \end{cases} \tag{2.11}$$

  where t is a term in set of bag of words, which has N words, tf_{t,D_k} is a number of occurrence of the term t in the document D_k, $|D|$ is a number of all documents.

  .

- **TopIDF Cosine Similarity** [72]

  TopIDF Cosine Similarity is cosine similarity between bags of words of two documents. The TopIDF similarity based on the concept of the IDF term, where the

highest in two documents are used to represent the similarity. Consequently, only IDF plays a role in this similarity without influence of TF term. Therefore, it could be viewed as the extreme case of IDF Cosine Similarity, where TF assumption totally removed. The definition of TopIDF cosine similarity is defined in the following definition.

Definition 2.9. Given a collection of document string D_1, D_2, D_3,$\cdots$, D_n and each document represented by a bag of words, $W = w_1, w_2, w_3, \&, w_m$, where $w_j$is the word j in the vocabulary set, the TopIDF function defined as $\delta : \mathbb{D} \times \mathbb{D} \mapsto [0, 1]$, where $\delta(D_i, D_j)$ can be computed in the following Equation.

$$\delta(D_i, D_j) = \frac{\sum_{t \in H_i \cap D_j} IDF(t) + \sum_{t \in H_j \cap D_i} IDF(t)}{\sum_{t \in H_i} IDF(t) + \sum_{t \in H_j} IDF(t)} \tag{2.12}$$

$$IDF(t, D) = log \frac{|D|}{|\forall x \in D | t \in x|} \tag{2.13}$$

where t is a term in set of bag of words, which has N words, $|D|$ is a number of all documents, H_i and H_j are set of words, that have the highest IDF score in the document D_i and D_j respectively.

.

- **Count Similarity** [72]

  Count similarity is used to compute the similarity by considering the links of the document. In KG integration, entities, which share more links tend to identical. Consequently, the count similarity count the similarity between document entities.

  Definition 2.10. Given a collection of document string D_1, D_2, D_3,$\cdots$, D_n and each document represented by a bag of link, $L_{D_i} = l_{D_{i1}}, l_{D_{i2}}, l_{D_{i3}}, \&, l_{D_{im}}$, where $l_j$is the word j in the document D_i, the count function defined as $\delta : \mathbb{D} \times \mathbb{D} \mapsto [0, 1]$, where $\delta(D_i, D_j)$ can be computed in the following Equation.

$$\delta(D_i, D_j) = \frac{1 - 2^{-|L_{D_i} \cap L_{D_j}|}}{1 - 2^{-\lceil \frac{|L_{D_j}| + |L_{D_j}|}{2} \rceil}} \tag{2.14}$$

  .

Several document-based similarity approaches have been studies for many decades. There are many studies using the document-based similarity to integrate the KG into homogeneous ontology. These approach can apply because each entity can be treated as a document.

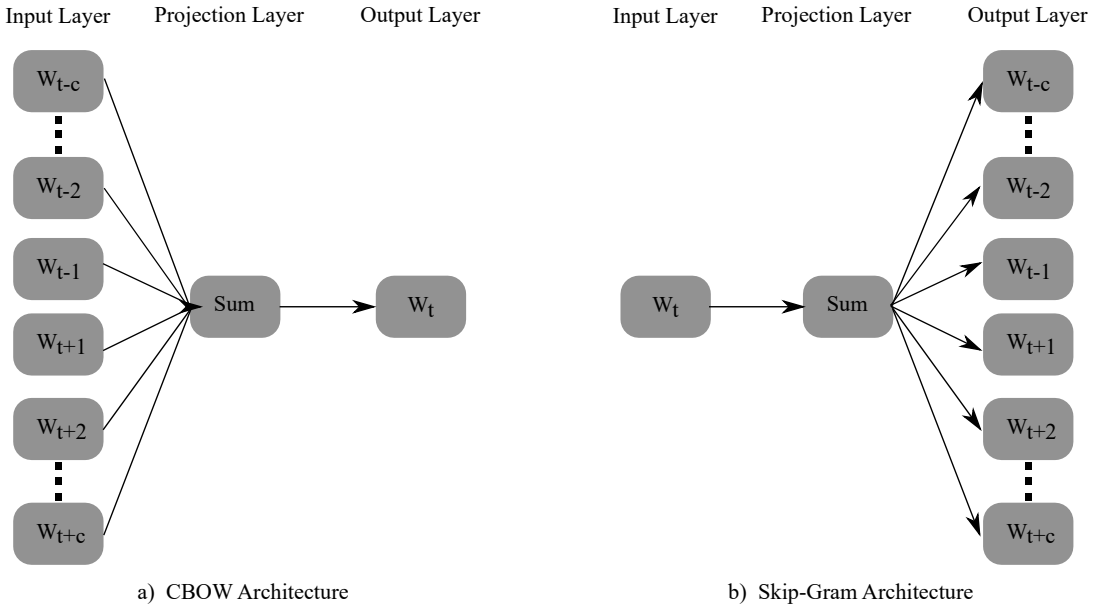


FIGURE 2.5: Two general architectures of neural network language models

2.4.2.3 Vector-based Similarity

The vector-based similarity approach is a modern similarity that is recently developed. The concept of the vector-based similarity approach is to represent things that are considered as the distributed vector representation in the continuous vector space. This approach is applicable for any levels of entity, e.g. word level, sentence level, document level, entity level, etc. The main concept behind learning the representation is neural network language model.

The neural network language model is introduced in the study [83]. The idea is to learn representations of words by using the neural network architectures. Typically, there are two general architectures: 1) Continuous Bag of Words (CBOW) model and 2) Skip-Gram model [64, 83, 84]. The graphical model of two architectures are depicted in Figure 2.5. As shown in Figure 2.5, the concept of the CBOW model is to predict the target word by the context words, while the idea of the Skip-Gram model is to predict the context words by the target word. More specifically, given a sequence of training words $w_1, w_2, w_3, .., w_N$ and a context window c, the learning function of the CBOW model is to maximize the following average log probability

$$\frac{1}{N}\sum_{t=1}^{N} log\ p(w_t|w_{t-c}, ..., w_{t-1}, w_{t+1}, ..., w_{t+c}) \tag{2.15}$$

where

$$p(w_t|w_{t-c},...,w_{t-1},w_{t+1},...,w_{t+c}) = \frac{exp(\bar{v} \cdot v_{w_t})}{\sum_{w=1}^{|V|} exp(\bar{v} \cdot v_w)}$$

Here, v_w is the distributed representation of the word w, $\bar{v}$ is an average distributed representation of words in the context c and V is the set of vocabulary of all words. Inversely, the Skip-Gram model is to maximize the following log probability

$$\frac{1}{N}\sum_{t=1}^{N}\sum_{-c\leq i\leq c, i\neq 0} log\ p(w_{t+i}|w_t) \tag{2.16}$$

where

$$p(w_{t+i}|w_t) = \frac{exp(v_{w_{t+i}} \cdot v_{w_t})}{\sum_{w=1}^{|V|} exp(v_w \cdot v_{w_t})}$$

Here, v_w is the distributed representation of the word w and V is the set of vocabulary of whole words.

Recently, many studies [64, 85, 86] proposed learning distributed representation of compound words. Mikolov et al. proposed the statistical-based approach for replacing compound words with unique tokens [64]. In their approach, the co-occurrence of words is used to determine whether such group of words is words in higher level such as phrase, sentence, document. In the study [85], the Doc2Vec approach was proposed. The concept of Doc2Vec is to learn distributed representation of sentences and paragraphs with the context words. Although this approach can learn the distributed representation of larger chunk of words, the predefined tokens are still required in the learning process. Consequently, both studies [64, 85] cannot directly estimate the distributed representations of new compound words by using distributed representation of words. Later, Dima et al. proposed a method to estimate distributed representations of the compound noun by using the deep neural network architecture [86]. The approach can interpret the distributed representations of compound words by their words. However, their architecture is limited to handle two consecutive words as a compound word. Consequently, it cannot deal with the dynamic length of compound words.

In contrast with the studies[64, 85, 86], the workaround for estimating the distributed representation of a compound word by using their individual words is to average distributed representations of individual words in the compound words. This method can directly estimate the distributed representation of the compound word and solve the dynamic length of compound words. However, as reported in the study [64], the meaning of

a compound word is not a simple composition of the individual words. Consequently, estimating the distributed representation of a compound word by average representations might not be a good representation.

2.5 Remain Issues

As shown in Figure 1.3, the general work flow of the KG population consists of two steps: 1) Knowledge Extraction and 2) Knowledge Integration. Our motivation focuses on KG population, in particular knowledge integration. In the previous section, we reviewed and discussed various related works. Here, we then discussed some limitations in the reviewed related works. In the following, the remained problems on knowledge integration including entity linking and predicate linking are discussed as well as the remained problem of the main goal of this research, KG population, is presented as follows.

- **Knowledge Integration**

  In the Knowledge Integration, there are many studies proposed to solve the entity mapping problem and the predicate linking problem. Still, some issues regarding such problems remains as follows.

  - **Entity Linking** Although there are many studies[43, 54, 55], in which S-P-O triples are successfully extracted from natural language text, they still do not consider entity linking. As a result, the ambiguity of an extracted entity might be occurred. For example, the $Washington$ entity could refer to a person or a place. An URI in Knowledge Graph resources is unique for each real-world object. If we could map the extracted entity to the entity in Knowledge Graph resources, it will eliminate the ambiguity of the entity due to an identity of an URI. Furthermore, if the entity from natural language text could be mapped to the entity in the Knowledge Graph resources, it means that the entity could be enriched in two ways. In one way, the entity in Knowledge Graph resources is enriched by information from the extracted triples. In other way, the entity from natural language text is also enriched by the information from Knowledge Graph resources.

    Linking entity to Knowledge Graph resources becomes a challenging problem because of the continuous growth of Knowledge Graph resources. When the Knowledge Graph resources project, namely the LOD project cloud, started in 2007, there are only 12 data resources, while currently there are more than

1,000 data resources[1]. Due to a large number of Knowledge Graph resources, we could not know which data resources contains an identical entity. As a result, some entities could not be mapped to Knowledge Graph resources. To the best of our knowledge, linking entity to multiple Knowledge Graph resources is not addressed yet.

- **Predicate Linking** Many studies [20, 44, 48, 56, 87] are proposed to generate Knowledge Graph from natural language text. Still, linking predicate to its identical predicate in Knowledge Graph resources is usually ignored. Linking predicate to its identical predicate in Knowledge Graph resources, plays an crucial role for utilizing Knowledge Graph, because it could reduce the heterogeneous problem and could increase search ability over Knowledge Graph resources. Although there are some studies [44, 56, 87], in which the predicate linking task is considered, they still have some drawbacks. In the study [44], linking predicate merely relies on RDF WordNet [2] instead of using other Knowledge Graph resources, especially DBpedia [10], which is the Knowledge Graph resource that much more gains attention from many researchers. Later, in the study [56], linking a predicate of a triple extracted from natural language text to an identical predicate in DBpedia therefore is introduced; however, the approach still limits using a simple rule-based approach, which could not be generalized to discover other identical predicates in other Knowledge Graph resources. Then, the study [87] introduces the string-based similarity for computing similarity between predicates. Nevertheless, the string-based similarity could not perform well when surface forms of strings are quietly different but their meaning is the similar. Based upon these reasons predicate linking is still many remained unsolved.

- **Knowledge Graph Population**

  Knowledge Graph Population is a task relating two tasks: 1) KG construction and 2) KG integration. As our discussion on KG construction, we categorize the KG construction approaches into three main categories: 1) The manual approach, 2) The semi-automatic approach and 3) the automatic approach. Based on the discussions, to handle with flow of knowledge in the big date era the promising approach is the automatic approach. There are two methods in this approach: schema-based method and schemaless-based method. The studies [20, 43, 44, 48, 50] showed that the schema-based method can populate the knowledge based on the schema; however, new knowledge sometimes is not followed by the schema. As a result, the knowledge that can be populated is very limited. In contrast with the

[1] http://lod-cloud.net/
[2] http://wordnet-rdf.princeton.edu/

previous studies, the schemaless-based method is the most promising. Although the studies [53–55] reported the reasonable result on extracting knowledge as triples from natural language text without predefine ontology, the usage of such knowledge is also very limited. The main limitation is that the heterogeneous problem where publishers use their own vocabulary to provide the knowledge in form of natural language text. This heterogeneous problem in the natural language text is severe than the heterogeneous problem in KG. Therefore, the KG population, which is KG constructing together with KG integration, is still not solved yet.

In our framework, two main issues regarding entity linking and predicate linking are our primary focus. Based upon our discussion, we therefore propose two frameworks, namely HMiLDs and HRSim, in Entity Linking and Predicate Linking for dealing with these two issues in the KG integration problem. Also, the main framework, T2KG, is proposed to populate new knowledge from natural language text to an existing KG. The outline of our framework are as follows.

- **HMiLDs** is the framework for linking entity to multiple Knowledge Graph resources. The basic idea of HMiLDs is to directly map entities to one particular Knowledge Graph and then gradually expand a search space for discovering identical entities to other Knowledge Graph resources in order to find other identical entities. The details of HMiLDs are presented in Chapter 3.
- **HRSim** is the framework for mapping predicate to Knowledge Graph resource. The idea of HRSim is to combine a rule-based approach and a similarity-based approach as the hybrid system for mapping predicate. The rule-based approach is a highly accurate approach but required massive of training data to construct rules. In contrast, the similarity-based does not required much data but the performance of the similarity-based might not be good enough when textual string is significantly different. We therefore propose the hybrid combination between a rule-based approach and a similarity-based approach to gain advantages of each approach. Also, the similarity-based approach greatly relies on similarity measure. In the similarity-based approach, we therefore propose a novel vector-based similarity measure. The details of HRSim are presented in Chapter 4.
- **T2KG** is an automatic framework for constructing KG from natural language. The idea of the framework is to make use of the schemaless-based approach to acquire wide-range knowledge; however, the integration method has to be applied to resolve the heterogeneous problem in the schemaless-based approach in the T2KG framework. The details of T2KG are presented in Chapter 5.

2.6 Summary

KG Population is one of the most challenging problem that gain attentions from many groups of communities and researchers. In order to populate knowledge to a KG, various approaches have been proposed. KG population is a task consisting of the KG extraction and KG integration. KG extraction and KG construction are very similar. We therefore introduced and discussed the related work on the KG construction and differentiated the KG construction and KG completion, which currently become the other hot issue. Since the KG completion usually work on the existing KG; in consequence, the natural language text and new entities as well as relation were not considered. Since the knowledge integration has played an important role in the KG population, the reviewed on KG integration, including entity linking and predicate linking were reviewed and discussed. Furthermore, several useful similarity measures were reviewed because the KG integration usually used the similarity measure as the indicator score when integrating knowledge. Eventually, we introduced the remained issues and what component we used to solve these issues.

Chapter 3

Entity Linking

In this Chapter (Chapter 3), we presented **HMiLDs**: : a **H**euristic expansion framework for **M**apping Ent**i**ties to **L**O**D** KG**s** (HMiLDs). Firstly, the overview of the entity linking task and the motivation were presented in Section 3.1. Secondly, the problem definition and technical terms were formally defined in Section 3.2. Thirdly, the methodology of the HMilDs framework was described in the following sections (Section 3.3). Fourthly, the experiments were conducted and reported in Section 3.4. Finally, this chapter concluded in Section 3.5.

3.1 Overview

Interlinking among knowledge is the key procedure to utilize information more wisely. For example, if an entity in the KG A connects to the identical entity in the KG B, the information of the entity in KG A could be enriched by its connected entities in both the KG A and the KG B. As a result, another perspective of knowledge will be discovered. Therefore, Linked Data was created to provide a simply concept of publishing and connecting such data. The aim of Linked Data is to construct a collection of KGs, which known as the web of Data. Currently, there is the ongoing project, which aims to construct KG based on the Linked Data concept, named the Linked Open Data (LOD) cloud [8]. In the LOD cloud, there are more than million entities and links with many relations. Consequently, it is well-known that any entities linked to the LOD could be enriched by other entities in the LOD cloud. Therefore, the aim of the research in this chapter is to discover and map entities to their identical entities in the LOD cloud.

Linking entities to the LOD cloud becomes a challenging problem because of the continuous growth of LOD data sets. When the LOD project cloud started in 2007, there are only 12 data sets, while currently there are more than 1000 data sets [8]. In Figure 3.1, we visualize the growth of the LOD cloud through the time since the project started in 2007. Due to a large number of LOD data sets, we could not know which data set contains an identical entity. As a result, some source entities could not be mapped to the LOD cloud. To the best of our knowledge, linking entities to more than a data set is not addressed yet.

In this chpater, we introduce HMiLDs: a **H**euristic expansion framework for **M**apping Enti**i**ties to **LOD** KG**s** (HMiLDs). The basic idea of HMiLDs is to directly link entities to one particular data set and then gradually expand a search space for discovering identical entities to other LOD data sets in order to find other identical entities. Due to a large amounts of entities in the LOD cloud, an expansion strategy and a heuristic function for limiting the expanding search space are designed into the framework. In the following sections, the definition of linking entities to the LOD cloud is described and then the details of HMiLDs are reported.

3.2 Definition and Problem

To clarify the problem for linking entities to the LOD cloud and some technical terms in the chapter, their definitions are given in this section.

An entity represents a real-world thing. Let a, e, s, x and y are entities.

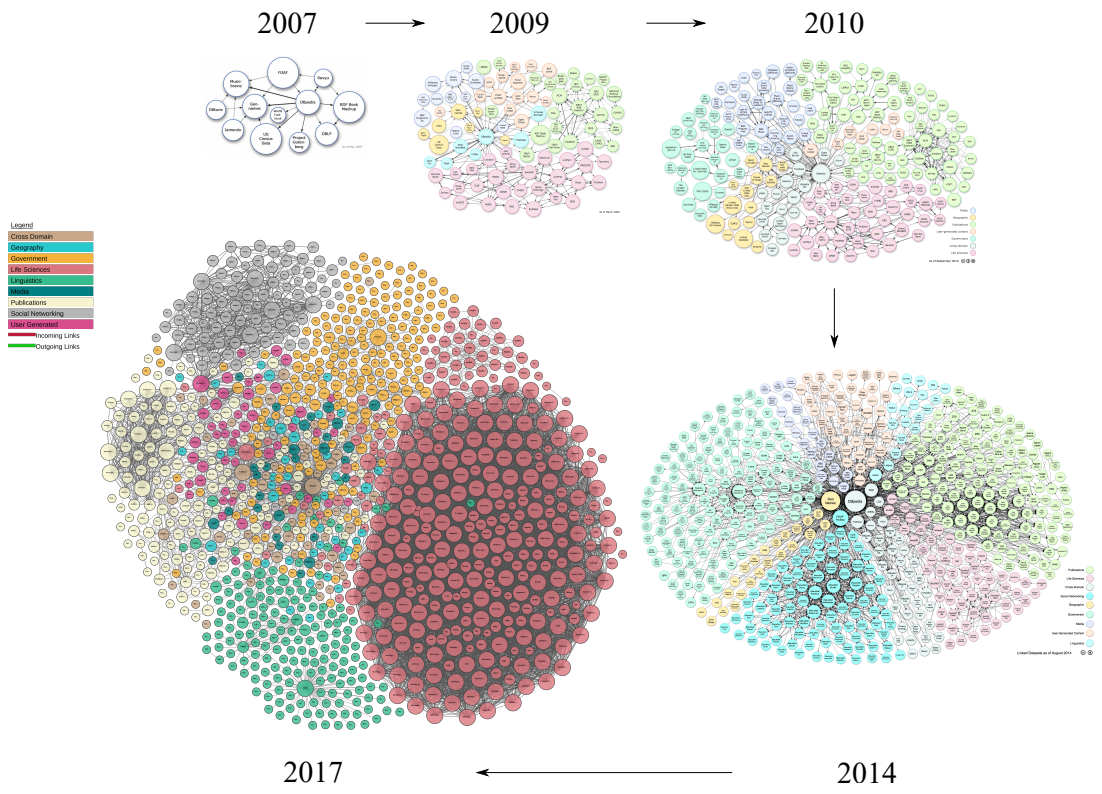


FIGURE 3.1: The Growth of the LOD cloud since 2007 [8]

Definition 3.1. Identical Entity : $x \equiv y$ denoted that x and y are identical if and only if they are referred to the same real-world object.

Definition 3.2. Source Entity : s is an source entity if $s \in S$, where S is a source data set. A source data set is a data set that aims to map to the LOD cloud.

Definition 3.3. Anchor Entity : a is an anchor entity if $a \in A$, where A is an anchor data set. An anchor data set is a base data set in the LOD cloud where a source data set is mapped to at first. Any LOD data set could be selected as an anchor data set. However, a data set strongly connected to other LOD data sets is preferred. If the anchor data set tightly connect to other LOD data sets, it will provide useful links to expand the search space to another data set.

Definition 3.4. Adjacent Entity : y is the adjacent entity of x, when the following set is not empty: $\{(o_x, s_y) \mid < s_x, p_x, o_x > \in t(x), < s_y, p_y, o_y > \in t(y) \; and \; o_x \equiv s_y\}$, where $t(x)$ and $t(y)$ are a set of triples of entity x and y respectively and $< s_x, p_x, o_x >$ is a triple of $t(x)$ and $< s_y, p_y, o_y >$ is a triple of $t(y)$.

Definition 3.5. Expanded Entity : e is an expanded entity, if $e \in E$, where E is an expanded data set. An expanded data set is a LOD data set that could be reached from the anchor data set.

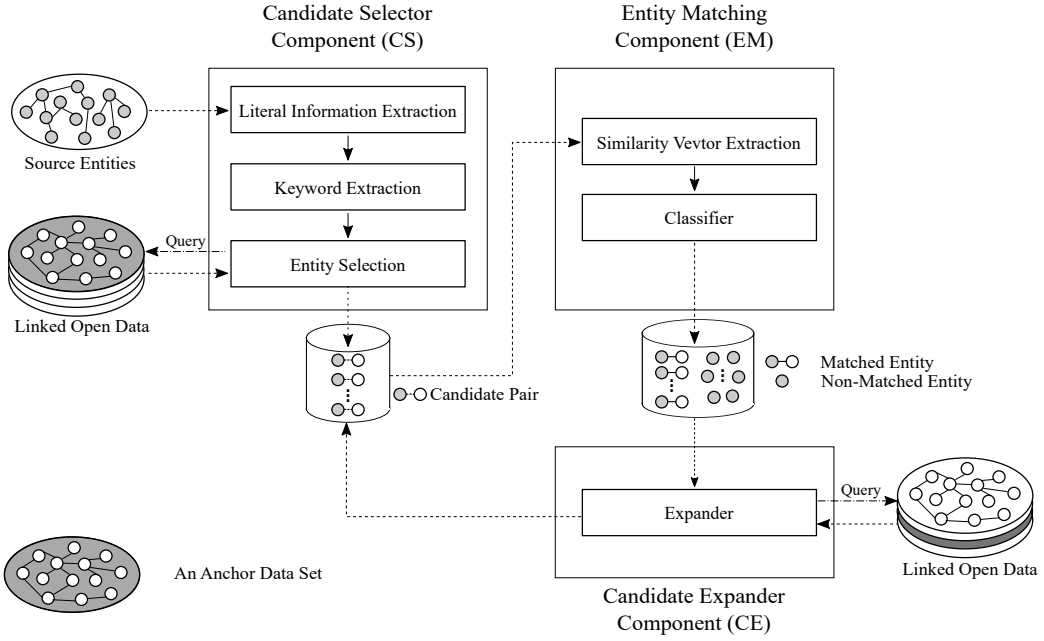


FIGURE 3.2: The Diagram of HMiLDs

Definition 3.6. Lininking entities to the LOD cloud : Lininking entities to the LOD cloud is not similar to conventional linking entities, which aims to map entities to a data set. Since the LOD contains tremendous amounts of data sets, linking entities to the LOD cloud aims to map entities to more than a data set or multiple LOD data sets. Given $D_1, D_2, D_3, ..., D_n$ and S, where D_i represents the LOD data set i, n is a number of LOD data sets and a source data set S, the definition of linking entities to the LOD cloud is to compute set $\omega = \{(x, y) \,|x \equiv y, x \in S, \exists i \,(y \in D_i)\}$.

3.3 Methodology - HMiLDs

In this section, the details of HMiLDs are presented. As depicted in Figure 3.2, HMiLDs consists of three components as follows: 1) the Candidate Selector component (CS), 2) the Entity Matching component (EM) and 3) the Candidate Expander component (CE). CS retrieves candidate entities from an anchor data set by using a set of keywords to generates a set of candidate pairs. EM verifies whether generated candidate pairs are correctly match or not by using a machine learning technique based on similarity vectors between entities. CE heuristically expands the search space from an anchor data set to another LOD data set in order to find other candidate entities for non-matched entities and re-generates a new candidate pair. The details of the components are as follows.

3.3.1 Candidate Selector

As shown in Figure 3.2, source entities are passed into CS. In CS, there are three modules: 1) the literal information extraction module, 2) the keyword extraction module and 3) the entity selection module. The literal information extraction module extracts description of a source entity. The keyword extraction module creates a set of keywords from description of the source entity. The entity selection module finds candidate entities in the anchor data set by using the set of keyword and then pairs the candidate entities with the source entity as candidate pairs. The result of CS is a set of candidate pairs.

3.3.1.1 Literal Information Extraction

The literal information module extracts description of the entity referred as literal information for each source entity. There are many studies [72, 88], extracting the literal information by considering rdfs:label and some other common properties [89]. Although the common properties are widely used to describe many LOD entities, due to heterogeneous problem, some entities might not contain those properties. Limiting extracting literal information to some properties might miss some useful information. Therefore, in the literal information extraction module, all properties are considered as literal information.

The literal information is divided into two types. Two types of literal information are a short literal string l_s and a long literal string l_l. l_s is a string, of which the length equals one phrase, whereas l_l is a string, of which the length is greater than one phrase. Considering the characteristic of l_s , l_s uses to represent specific information of an entity since a short string usually behave as a label of the entity or a concise description of the entity. In contrast with l_s , l_l generally describes greater detail of an entity. Therefore, l_s carries much more essential information than l_l. Nevertheless, in case that l_s of entities causes an ambiguity, l_l could help to disambiguate between entities. For example, given the entity, `dbr:Barack_ Obama`[1] with its predicate-object values as follows:

`dbr:Barack_Obama foaf:surname` "Obama"@en
`dbr:Barack_Obama rdfs:comment` "Barack Hussein Obama II (born August 4, 1961)...."@en

"Obama" is treated as l_s because its length equals one phrase. "Barack Hussein Obama II (born August 4, 1961) ...." should be considered as l_l because its length is greater than one phrase. In the example, The l_s "Obama" could not disambiguate between `dbr:Barack_Obama` and `dbr:Michelle_Obama`[2] because both entities are referred

[1] http://dbpedia.org/page/Barack_Obama
[2] http://dbpedia.org/page/Michelle_Obama

to "Obama". Nonetheless, the l_l "Barack Hussein Obama II (born August 4, 1961) ...." could help to distinguish between the entities. Therefore, HMiLDs does not only consider l_s but also consider l_l. Consequently, the results of this module are l_s and l_l of an entity.

3.3.1.2 Keyword Extraction

In the keyword extraction module, l_s and l_l are used to create a set of keywords. Since the characteristic between l_s and l_l is different, the different methods for extracting keywords are applied.

For l_s mostly contains critical information of an entity, omitting some words might not be able to represent an identity of entity. Each word in l_s therefore is selected as a keyword. Furthermore, the N-gram technique is also applied to capture co-occurrence words. For example, given l_s as "San Francisco", if we consider the words, "San" and "Francisco", separately, it might not be able to represent the exact meaning of "San Francisco". Consequently, the words generated by the N-gram technique is employed as keywords to cope with such characteristic.

For l_l, the same strategy as l_s cannot be applied because l_l comprises a lot of words. It therefore would be better to select some words. Name Entity Recognition (NER) technique [90] is employed in order to select keywords from l_l. Usually name entities such as person, location, organization name are highly relate to an entity. With the NER technique, a small set of keywords could be created. After acquiring keywords from l_s and l_l, a set of keywords is constructed by combing all keywords together.

3.3.1.3 Entity Selection

The entity selection module generates candidate pairs between a source entity and an anchor entity by using a set of keywords. Each keyword is used to retrieve anchor entities, which contain the same keyword. After that, the anchor entity are paired with the source entity as the candidate pair.

3.3.2 Entity Matching

In EM, each candidate pair from CS is verified whether it is a correct match or not by using a similarity vector. Consequently, the results of EM are match pairs and non-match entities. In EM, there are two modules: 1) the similarity vector extraction module and 2) the classifier module. In the similarity vector extraction module, a similarity vector

of a candidate pair is computed by various similarity metrics. The classifier module then classifies the candidate pair by using its similarity vector. When the classifier module classifies that the candidate pair is a correct match, the source entity of the candidate pair immediately map to its paired entity. Such relation could be utilized to access expanded entities later. Otherwise, the candidate pair is unpaired as a non-match entity.

3.3.2.1 Similarity Vector Extraction

In the similarity vector extraction module, a candidate pair is passed to various similarity metrics to compute a similarity vector between the entities. In the LOD cloud, usually entities are heterogeneous and some of them might be distorted and ambiguous [91]. As a result, limiting similarity metrics to a few metrics might not overcome such problems. In the similarity vector extraction module, many kinds of similarity metrics therefore are used in HMiLDs.

For the similarity metrics, the six similarity metrics: Term Frequency-Inverse Document Frequency cosine similarity (TF-IDF) [82], Jaro-Winkle similarity metric [78], Edit Distance similarity metric [91], Count similarity metric [72], IDF similarity metric [72] and TopIDF similarity metric [72], are used. These similarity metrics are conventional similarity metrics for representing the similarity between documents. In HMiLDs, an entity could be viewed as a document because we treat all literal information of entities as string. To create a document from an entity, all strings of an entity are appended together as a document. Consequently, we could apply these similarity metrics as the same manner of the conventional document similarity.

In addition, we also introduce two novel similarity metrics: the CommonKeyword similarity metric and the CandidateHits similarity metric.

The CommonKeyword similarity metric aims to capture the similarity between a set of keywords of entities. Based upon the characteristic of a set of keywords, entities sharing a lot of keywords have a high chance to be a match pair, because the entities of such candidate pairs intend to describe the same thing. Consequently, we assume that the more keywords entities of a candidate pair share, the more likely they are matched. To capture this characteristic, the CommonKeyword similarity metric is calculated by passing sets of keywords between entities of a candidate pair to the Jaccard Similarity [92].

The CandidateHits similarity metric is to represent how many times the candidate pairs are generated. During generating candidate pairs, some candidate pairs might be generated more than once. For example, different keywords might retrieve the same candidate

entity for generating the candidate pair. We therefore assume that the candidate pair, which is generated frequently, is likely to match, because the relation between such entity and the source entity is highly correlate. Based on this concept, the CandidateHits similarity metric is derived as shown in Eq. 3.1,

$$CandidateHits(c) = \frac{n(c)}{\sum_{i \in C} n(i)} \tag{3.1}$$

where $n(c)$ and $n(i)$ is how many times the candidate pair c and the candidate pair i are generated and C is a set of candidate pairs.

Apart from eight similarity metrics, we also consider types of literal information as a factor of combination of similarity metrics. Ignoring type of literal information might miss some description of an entity. We categorize literal information into three types: 1) l_s, 2) l_l and 3) combining l_s and l_l together. In HMiLDs, three documents regard types of literal information are extracted for each entity. Then, similarity metrics are applied for each document of the entity. However, a dimensional length of our feature vector is 22-dimensional feature vector. Since the CandidateHits similarity metric does not relate to a type of literal information of entities, a candidate pair applies this similarity metric only once.

3.3.2.2 Classifier

In the classifier module, a machine learning method is utilized to create a classifier C for determining whether the candidate pair is correct match or not. To create the classifier C, similarity vectors of candidate pairs and their label indicating the class of candidate pair are used. If the classifier C classifies the candidate pair as a correct match, the source entity of the candidate pair will be mapped to the LOD entity of the candidate pair. Otherwise, the candidate pair is unpaired as a non-match entity.

3.3.3 Candidate Expander

For non-match entities, CE expands the search space for finding other candidate entities in other LOD data sets to generate new candidate pairs. In CE, only the expander module is installed. The expander module enables HMiLDs to traverse through the LOD cloud by using link properties, in particular the owl:sameAs property, in order to discover a new candidate entity for re-generating a new candidate pair.

Generally, adjacent entities of an entity carries relevant information about the entity. We therefore assume that adjacent entities could be utilized to discover a new candidate

entity. The basic idea is to start with a non-matched entity and gradually expand the search space to its adjacent entities. Then, such adjacent entities continue to expand the search space to their adjacent entities. The expansion procedure is done repeatedly until a candidate entity of the non-matched entity could be discovered or there are no any adjacent entities to expand the search space.

Although we could gradually expand an source entity to an expanded entity by using adjacent entities and link properties, the expanded search space could grow up dramatically when expanding into higher depth level [21]. The depth level is measured by how many hops from the entity to the other entity via adjacent entities. According to this problem, a simple heuristic function is introduced by assuming that the search space should be expanded to an expanded entity if the expanded entities share at least a keyword with entities in traversed path of adjacent entities in the source data set [21]. Based on the preliminary experiment [21] with the maximum depth level set at 3, the expander module generates averagely 214,885 candidate pairs per an entity, while using the simple heuristic function the expander module establishes 203 candidate pairs per an entity. The heuristic function in the work [21] could reduce a number of candidate pairs. Still, the work [21] generates enormous useless candidate pairs, which have no chance to be a correct match pair. Therefore, a new heuristic function for generating candidate pairs is needed. The aim of the heuristic function is to limit the expanded search space so that a less number of candidate pairs and a less number of useless candidate pairs would be generated.

In HMiLDs, the keyword score is proposed as the new heuristic function. In the keyword score, two basic assumptions are made. First, the more entities share keywords, the more they are likely to match. This assumption is similar to the CommonKeyword similarity metric. Second, keywords from adjacent entities in the different depth level are distinct. Keywords of adjacent entities in the lower depth level could more highly related to the entity than keywords of adjacent entities in the higher depth level. Consequently, it could be assumed that the lower the depth level of keywords to the entity is, the more important the keyword is. Based upon two assumption above, a keyword score of an entity could be derived as shown in Eq. 3.2,

$$KeywordScore_s(e) = \frac{\sum_{i=0}^{d} w^i \cdot \frac{|keyword(e) \cap \cup_{x \in Adj(s)_i} keyword(x)|}{\sum_{x \in Adj(s)_i} |keyword(x)|}}{\sum_{i=0}^{d} w^i} \tag{3.2}$$

where e is an expanded entity, $keyword(x)$ is a function returned keywords of the entity x, d is the maximum depth level for expanding the search space, w is a parameter for weighting necessity of keywords at the different depth level and $Adj(s)_i$ is a set of

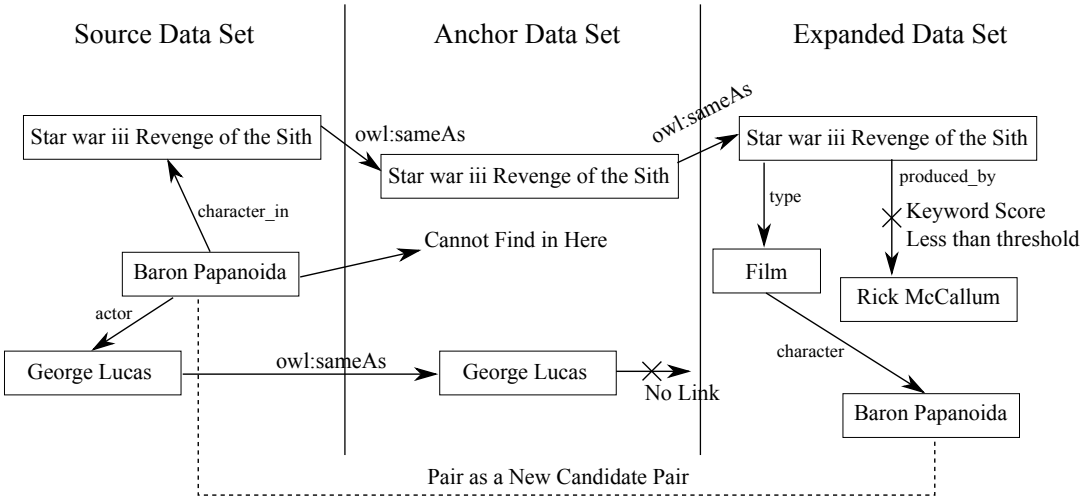


FIGURE 3.3: An Example of the Expanded Search Space from One Data Set to Other Data Sets

.

adjacent entities of s at the depth level i. For example, given the scenario in Figure 3.3, the entity "Baron Papanoida" in the source data set is s and the entity "Star war iii Revenge of the Sith" in the expanded data set is e. The entities, "Star war iii Revenge of the Sith" and "George Lucas", in the source data are entities in $Adj(s)_1$.

The aim of keyword score is to capture the correlation between the source entity s and the expanded entity e. If the keyword score is high, the correlation between entities will be high. We assume that the expander module should expand the search space to the high correlated entity because there is high probability to discover an identical entity.

In Algorithm 1, the algorithm of the expander module is expressed. Given a set of non-match entities (NI_s), the idea of the algorithm is to add a pair of a source entity and an adjacent entity, of which the keyword score is greater than the threshold δ to C. The algorithm starts by adding the source entity s to Q in line 4. Then, the entity in Q is iteratively dequeued as the entity p and then adjacent entities of the entity p is added to N in line 6-7. For each entity in N, which have $KeywordScore$ greater than the than the threshold δ, it is added to Q and is paired with the source entity s as the new candidate pair and is stored in C in line 8-13. Next, the entity p linked by the owl:sameAs property of entities in Q is added to Q in line 14. Eventually, the algorithm returns the set of new candidate pairs C in line 17.

In Figure 3.3, an example of the algorithm is illustrated. Given the source entity "Baron Papanoida", this entity does not match any entities in the anchor data set; in consequence, it is a non-matched entity. However, the entity "Baron Papanoida" has some adjacent entities: "Star war iii Revenge of the Sith" linking to "Baron Papanoida" by the character_in relation and "George Lucas" linking to "Baron Papanoida" by the actor

Algorithm 1 Expander Module for Generating New Candidate Pairs

Input: NI_s (Set of non-match entities)
Output: C (Set of Candidate pair)

$C \leftarrow \emptyset$
$Q \leftarrow \emptyset$ # Q is a queue of entities, which will explore and expand
foreach $s \in NI_s$ **do**
    $Q \leftarrow s$
    while $Q \neq \emptyset$ **do**
        $p \leftarrow Q.Dequeue()$ # Get the first entity of Q
        $N \leftarrow Adj(p)$ # Get the adjacent entities of entity p
        foreach $e \in N$ **do**
            if $KeywordScore_s(e) > \delta$ **then**
                $Q \leftarrow e$ # Add to Q for exploring next round
                $C \leftarrow (s, e) \cup C$ # Pair candidate between s and e
        $Q \leftarrow sameAsLink(p)$ # Get other entities linked to p
                                    # by owl:sameAs and store in Q
return C

relation, and they successfully map to the anchor data set. Utilizing the owl:sameAs property, the expander module could traverse through the expanded data set via the owl:sameAs property among the source entity, the anchor entity and the expanded entity. Then, the expander module expands and locally searches the expanded entity "Star war iii Revenge of the Sith" in order to discover a new candidate entity for the source entity "Baron Papanoida". Then, the new candidate pair between the source entity "Baron Papanoida" and the expanded entity "Baron Papanoida" could be acquired. In case that the keyword score of expanded entities is less than the threshold, the entity is removed from the expanded search space in order to limit the range of the search space. In Figure 3.3, the entity "Rick McCallum" is removed, since its keyword score is less than the threshold δ.

3.4 Experiments

3.4.1 Experimental Setup

To evaluate the framework, four experiments are conducted. The first experiment investigates contribution of our novel similarity metrics for EM. The second experiment evaluates the performance of CS and CE for linking entities to multiple LOD data sets. The third experiment investigates parameters for the heuristic function of HMiLDs. The forth experiment evaluates the performance of HMiLDs.

Entities from Ontology Alignment Evaluation Initiative 2012 (IM@OAEI 2012) track [93] are used as source entities. Then, the source entities are aligned to LOD entity by a domain expert in order to construct ground truth for experiments. All experiments are conducted by using this dataset. Due to a large number of LOD data sets, manual aligning source entities to all data sets is impossible. Therefore, entities in two prominent LOD data sets, DBpedia and Freebase, are selected to align to source entities for constructing ground truth. DBpedia [10] is also chosen as the anchor data set because DBpedia is well-known as the hub of the LOD cloud [10]. Based upon the preliminary experiment, we could find that only **90.36%** of our source entities could be manually mapped to DBpedia entities. This result shows the solid evidence conforming to our research statement, where only one data set might not be enough for linking entities. Freebase [11] is selected as the expanded data set because more than million links of DBpedia are connected to Freebase. Furthermore All of our source entities, which do not contain in DBpedia, could be found in Freebase.

In HMiLDs, there are many parameters. To conduct the experiments, the parameters are set as follows. In the keyword extraction module, N for the N-gram technique are set as 1, 2, 3 and n respectively, where n is the length of a considered string. For NER system, Stanford Named Entity Recognizer [94], is used. In the entity selection module, the DBpedia Lookup Service interface [95] is used to gather candidate entities from DBpedia. In the entity matching module, the Support Vector Machine (SVM) is used in this component as the classifier to classify a candidate pair as the same manner of the work [76]. To build the classifier, candidate pairs generated by HMiLDs are manually labeled to their corresponded class. Then, such label together with their similarity feature vectors are used to create parameters of the classifier.

3.4.2 Experiment 1

Experiment 1 is designed to investigate contribution of our two novel similarity metrics, CommonKeyword and CandidateHits, for EM. Furthermore, various similarity metrics are compared to find the suitable combination of the similarity metrics for the entity matching component. In the entity matching problem, many combination of a similarity metrics have been proposed [21, 72, 91, 96]. In the experiment, we compare combination of similarity metrics with combination of our similarity metrics. Combination of similarity metrics of our approach, including HMiLDs, HMiLDs without the CandidateHits similarity metric (HMiLDs-CH) and HMiLDs without the CommonKeyword similarity metric (HMiLDs-CK), and other studies [21, 72, 91, 96] are summarized as shown in Table 3.1.

TABLE 3.1: Summary of Similarity Metrics for each Method

Method	Similarity Metrics							
	TF IDF	Jaro Winkler	Edit Distance	Count [72]	IDF [72]	TopIDF [72]	Common Keyword	Candidate Hits
TF-IDF	✓							
TF-Jaro [96]	✓	✓						
RiMOM [91]	✓		✓					
Rong et al. [72]	✓		✓	✓	✓	✓		
Combined [21]	✓	✓	✓	✓	✓	✓		
HMiLDs-CH	✓	✓	✓	✓	✓	✓	✓	
HMiLDs-CK	✓	✓	✓	✓	✓	✓		✓
HMiLDs	✓	✓	✓	✓	✓	✓	✓	✓

TABLE 3.2: The Results of each Method

Method	Precision	Recall	F-Measure
TF-IDF	0.89204	0.76874	0.82276
TF-Jaro [96]	0.91825	0.79228	0.84760
RiMOM [91]	0.92389	0.82998	0.87256
Rong et al. [72]	**0.95569**	0.81925	0.88100
Combined [21]	0.94999	0.85782	0.90013
HMiLDs-CH	0.95243	0.87349	0.90970
HMiLDs-CK	0.94799	0.88237	0.91202
HMiLDs	0.95202	**0.91008**	**0.92949**

In the experiment, 10-fold cross-validation technique is applied to evaluate each combination of similarity metrics. Precision, Recall and F-Measure are employed to measure the performance of the results.

The experimental results are listed in Table 3.2. Considering the results in Table 3.2, we could notice that Rongś method provides the highest precision at 0.956, while HMiLDs gives the best recall and the best F-measure at 0.910 and 0.929 respectively. Although HMiLDs provides less precision than Rongś method, the highest result of the F-measure still acquired from HMiLDs.

In order to deeply investigate the results, HMiLDs is compared against other methods by the t-testing method. The significance level is set at 0.05. It turns out that although combination of our similarity metrics gives the lower precision result than Rongś method [72], the difference of the results is not significant. Nevertheless, the paired t-testś results of the recall and the F-Measure indicate that HMiLDs is significantly different from other methods excepting HMiLDs-CH and HMiLDs-CK, which include the CommonKeyword similarity metric or the CandidateHits similarity metric. To further

TABLE 3.3: The Results of Generating Candidate Pairs

Setting	Data Set	Selecting	Non-Selecting	Missing
Lookup [97]	DBpedia	84.30%	6.06%	9.64%
CS	DBpedia	88.71%	1.65%	9.64%
CS	Freebase	95.04%	4.96%	0.00%
CS + CE	DBpedia Freebase	97.80%	2.20%	0.00%

investigate contribution of each similarity metric, the CommonKeyword similarity and the CandidateHits similarity, the t-testś results of the recall and the F-Measure between HMiLDs-CH, HMiLDs-CK and other methods, excepting HMiLDs, are conducted. The t-test results turn out that for HMiLDs-CH there is no significance when comparing with Combined [13], while for HMiLDs-CK, there is no significance when comparing with Combined [21] and RiMoM [91]. Based upon this investigation, the CommonKeyword similarity metric together with the CandidateHits similarity metric could provide significant contribution for improving the recall result and the F-Measure result of EM while they do not affect the precision result.

3.4.3 Experiment 2

Experiment 2 is to investigate the contribution of CE for linking entities to many LOD data sets and the contribution of keywords in CS for discovering identical entities. In the experiment, to fairly evaluate CS and CE, all generated candidate pairs are manually verified whether they are correct match or not so that the effect of EM could be avoided.

There are four setting in the experiment. The first setting is the Lookup [97] with DBpedia as a baseline approach. The Lookup approach retrieves candidate entities by using only a label of an entity as a keyword. The second and the third settings are CS with two different data sets, DBpedia and Freebase respectively. The fourth setting is CS with CE (CS + CE). For CS + CE, the anchor data set is DBpedia and the expanded data set is Freebase. Three metrics: selecting, non-selecting and missing, are used to evaluate the components in the experiment. The selecting metric measures the percentage of existing identical entities between the source data set and the LOD data set retrieved by the component for generating candidate pairs. The non-selecting metric measures the percentage of existing identical entities between the source data set and the LOD data set, which could not be retrieved by the component for generating candidate pairs. The missing metric represents the percentage of non-existing identical entities between the source data set and the LOD data set.

Considering the results in Table 3.3, CS using the data set DBpedia outperforms Lookup. Even though both CS and Lookup utilize a set of keywords for retrieving an entity in DBpedia, the main difference is a method to obtain a set of keywords. Lookup selects only a label of an entity as a keyword, while our CS generates a set of keywords by using various strategies. Therefore, The result confirms that our keywords in CS are helpful to discover identical entities.

The results of the experiment are listed in Table 3.3. In the Table 3.3, two main contribution for CE are founded. Firstly, comparing the missing result of CS using DBpedia with CS + CE, the missing result reduces from 9.64% to 0.0%. This reduction significantly shows that CE could discover some missing entities in one data set by using expanded data set. This result conforms to the assumption of this research, where linking entities to one LOD data set is not enough. Secondly, considering the results between CS and CS + CE, the selecting result of CS + CE is greater than the selecting result of CS with any data sets. This results shows that CE could help to discover more identical entities because it can successfully map entities to many data sets. Based upon two contribution of CE, we conclude that CE greatly contributes to map entities to the LOD cloud.

3.4.4 Experiment 3

In Experiment 3, parameters in the heuristic function are studied to investigate the trade-off between the ability to limit the expanded search space and the ability to map entities to the LOD cloud.

Two parameters are studied in the experiment. The first parameter is the threshold δ for limiting the expanded search space. The threshold δ in the experiment is varied from 0.0 to 0.5 and increases each step by 0.05. The second parameter is the weighting w for computing the keyword score. In the experiment, the weight w at 0.0, 0.25, 0.5 and 1.0 are investigated respectively.

To fairly evaluate the performance and the goodness of the heuristic function in the framework, the work [21] is selected as the baseline for the experiment. The framework in the study [21] is mostly similar to HMiLDs in this article; however, the heuristic function of the expander module is different. In the experiment, the 10-fold cross validation technique is performed to evaluate the results. Based upon our preliminary experiment [21], we statically set the depth level d at 3, since it could be sufficient enough to reach candidate entities and does not allow the expander module to explore the large search space.

Three evaluation metrics: 1) the average number of generated candidates, 2) the average number of useless candidates and 3) the coverage percentage of linking entities to the LOD cloud, are used to evaluate the ability to map entities to the LOD cloud and the ability to limit the expanded search space in aspects of quantity and quality.

- *The average number of generated candidates* is used to evaluate the ability to limit the expanded search space in the aspect of quantity. The expanded search space directly relates to a number of generated candidate pairs. If the expanded search space is very large, a number of generated candidate pairs will also become greater. Consequently, the ability to limit the expanded search space in the aspect of quantity could be observed through an average number of generated candidate pairs.

- *The average number of useless candidates* is employed to measure the quality of the ability to limit the expanded search space. Although many useless candidate pairs are generated, such candidate pairs could not improve any coverage for linking entities to the LOD cloud. Therefore, the quality of the ability to limit the expanded search space could be observed via an average number of useless candidate pairs.

- *The coverage percentage of linking entities to the LOD cloud* is used to evaluate the ability to map entities to the LOD cloud. If the percentage of linking entities to the LOD cloud is high, it could be inferred that the ability to map entities to the LOD cloud is also high.

The results of the experiment are illustrated in Figure 3.4-3.6. In each figure, the x-axis is the threshold δ, while the y-axis is the result measured by each metric. Each line in the figure represents the result with the different weight configuration.

In the Figure 3.4-3.5, the results show that the threshold δ directly influences the ability to limit the expanded search space. At the threshold $\delta = 0.05$, the dramatic reduction of a number of generated candidate pairs and reduction of a number of useless candidate pairs could be observed. Therefore, the threshold δ directly plays an important role in the ability to limit the expanded search space.

Considering effect of the weight w in Figure 3.4-3.5, we could observe that the weight w contribute to the ability to limit the expanded search space at the threshold δ set at 0.05. When the weight w is reduced, the heuristic function tends to generate less candidate pairs and useless candidate pairs. This characteristic happens because when the weight w reduces, the priority of the keyword is given to the keyword that closes

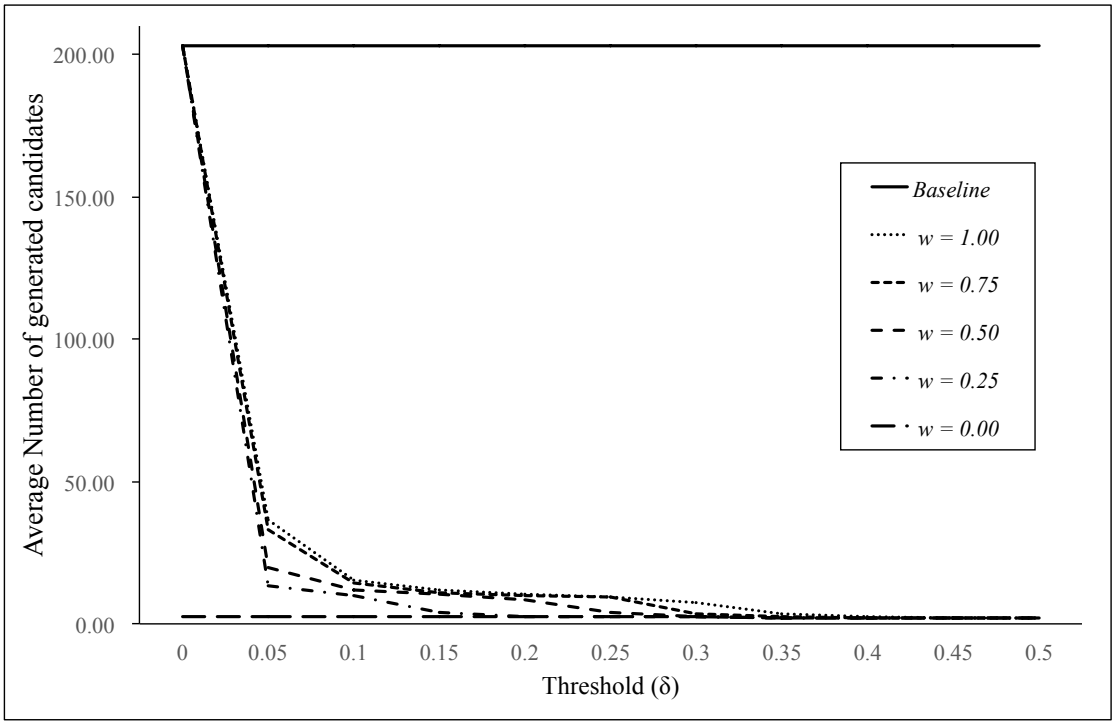


FIGURE 3.4: Average number of generated candidate pairs of the baseline and the framework with different weight w varied by the threshold δ

to the source entity. As a result, the heuristic function will eliminate useless candidate pairs.

In the Figure 3.6, when the threshold δ increases, the ability to map entities to the LOD cloud is decreased as we could observe from the reduction of the coverage number of linking source entities to the LOD cloud. For the weight w, considering the results in different weight w in Figure 3.6, the different weight w could differently provide the coverage number of linking source entities to the LOD cloud. However, the effect of the weight w could be governed by the threshold δ when the threshold δ is too large.

According to the experimental result, the threshold δ and the weight w contribute to the ability to limit the expanded search space and the ability to map entities to the LOD cloud. The best configuration of the weight w and the threshold δ for balancing between the ability to limit the expanded search space and the ability to map entities to the LOD cloud are achieved, when the weight w is set at 0.5 and the threshold δ is set at 0.05. With such configuration, HMiLDs could produce only 9.73% of candidate pairs of the baseline and 10.05% of useless candidate pairs of the baseline, whereas the highest coverage result of linking source entities to the LOD cloud at 90.36% is still reached as same as the baseline.

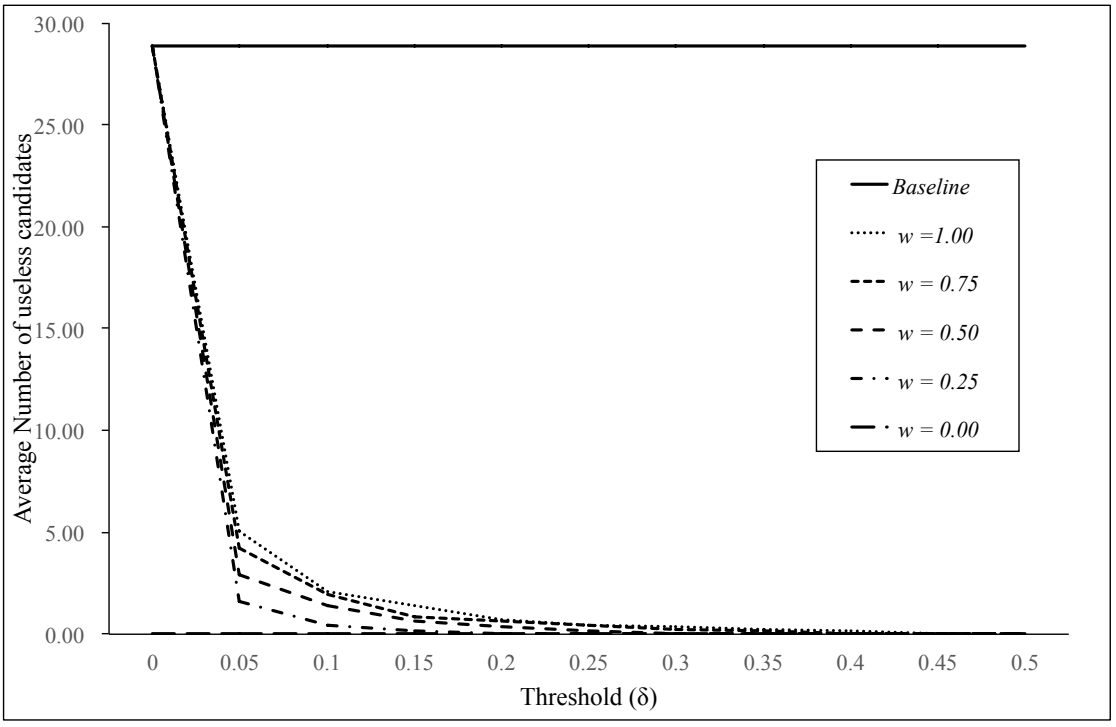


FIGURE 3.5: Average number of useless generated candidate pairs of the baseline and the framework with different weight w varied by the threshold δ

3.4.5 Experiment 4

In Experiment 4, the heuristic function of HMiLDs is investigated in three aspects. The first aspect is overall performance. The second aspect is the effectiveness to limit the expanded search space. The third aspect is as the efficiency for linking entities to the LOD cloud.

In the experiment, *Baseline* from the study [21] is used as the benchmark. The framework [21] is mostly similar to HMiLDs; however, the heuristic function of the expander module is different. In the experiment, the parameters of HMiLDs are set as follows. The weight w is set at 0.5. The threshold δ is set at 0.05. The depth d is set at 3. To evaluate the results the 10-fold cross validation technique is performed. All results are reported in Table 3.4 and 3.5.

In the first aspect, three standard metric: precision, recall and F-measure are employed to evaluate the performance of HMiLDs, when the heuristic function is installed. As shown in Table 3.4, HMiLDs provides the similar precision result, the similar F-Measure result and the same recall result, when comparing with *Baseline*. Consequently, in the first aspect regarding the performance, the heuristic function could not affect any performance. This result indicates that even though the expanded search space is reduced due to the heuristic function, HMiLDs could still obtain the similar performance as Baseline.

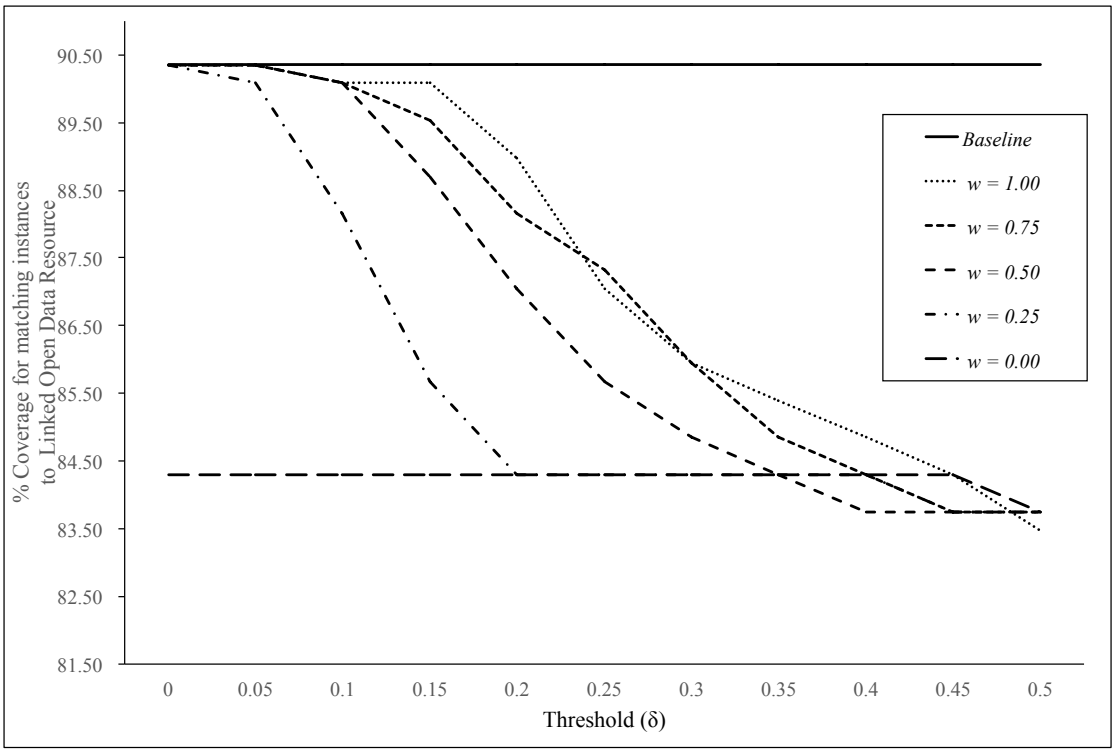


FIGURE 3.6: Coverage for linking entities to the LOD cloud of the baseline and the framework with different weight w varied by the threshold δ

In the second aspect, the average number of generated candidates and the average number of useless generated candidates are used to analyze this aspect. The expanded search space directly relates to a number of generated candidate. If the expanded search space is very large, a number of generated candidate pairs will be plentiful. Consequently, we could observe the effectiveness of the heuristic function for limiting the expanded search space via the average number of generated candidates and the average number of useless generated candidates. As shown in Table 3.5, the results indicate that our heuristic function helps to limit the expanded search space due to the reduction of an average number of generated candidate pairs. Furthermore, we could also observe that *Baseline* generates more useless candidate pairs than HMiLDs. Due to the reduction of generated candidate pairs and useless candidate pairs, it could be concluded that the heuristic function of HMiLDs could limit the expanded search space effectively.

In the third aspect, the percentage of coverage of linking entities to the LOD cloud (%Coverage) is measured to represent the efficiency of the heuristic function for linking entities to the LOD. Although HMiLDs could generate less generated candidate pairs and useless candidate pairs than *Baseline* in the second aspect, we also need to consider the efficiency of the heuristic function in the third aspect because the main purpose of the research is to map entities to the LOD cloud as many as possible. As shown in Table 3.5, HMiLDs provide the same percentage of coverage of linking entities to the

TABLE 3.4: The Results of the Framework Comparing with the Baseline

Approach	Precision	Recall	F-Measure
Baseline [21]	0.913	0.919	0.914
HMiLDs	0.917	0.919	0.917

TABLE 3.5: The Results of the Framework Comparing with the Baseline

Approach	Avg. number of Generated Candidates	Avg. number of useless generated candidates	% Coverage
Baseline [21]	203.03	28.87	90.36
HMiLDs	11.74	2.90	90.36

LOD cloud to *Baseline*. Therefore, the heuristic function of HMiLDs does not affect the percentage of coverage of linking entities to the LOD cloud even though the expanded search space is reduced.

Based upon the results in all aspects, HMiLDs outperforms *Baseline* in the second aspects and does not provide worse results than *Baseline* in any aspects. Therefore, This experiment indicates that the heuristic function of HMiLDs could effectively limit the expanded search space, while still maintains the efficiency to map entities to the LOD without affecting any performances.

3.5 Summary

Linking entities to the Linked Open Data (LOD) cloud plays an important role for enriching information of entities, since the LOD cloud contains abundant amounts of interlinked entities describing the entities. Consequently, many techniques have been introduced for linking entities to a LOD data set; however, most of them merely focus on tackling with the problem of heterogeneity. Unfortunately, the problem of the large number of LOD data sets has yet to be addressed. Owing to the number of LOD data sets, linking an entity to a LOD data set is not sufficient because an identical entity might not exist in that data set. In this article, we therefore introduce a heuristic expansion based framework for linking entities to LOD data sets. The key idea of the framework is to gradually expand the search space from one data set to another data set in order to discover identical entities. In experiments, the framework could successfully map entities to the LOD data sets by increasing the coverage to 90.36%. Experimental results also indicate that the heuristic function in the framework could efficiently limit the expansion space to a reasonable space. Based upon the limited expansion space, the framework could effectively reduce the number of candidate pairs to 9.73% of the baseline without affecting any performances.

Chapter 4

Predicate Linking

In this Chapter (4), we presented **HRSim**: A **H**ybrid combination of a **R**ule-based approach and a **Sim**ilarity-based approach was presented for the predicate linking task. In Section 4.1, the predicate linking and its motivation were described and discussed. Then, the formal definition and terms were specifically formalized in Section 4.2. In this Chapter, there are two methodologies: 1) HRSim and 2) the estimation of compositional vector. The methodology of HRSim and the estimation of compositional vector were presented in Section 4.3 and Section 4.4 respectively. Then, several experiments were conducted to evaluate those two methodologies in Section 4.5-4.6. Eventually, we concluded Chapter 4 in Section4.7.

4.1 Overview

A knowledge graph (KG) is a structured knowledge base that stores knowledge of the relation between entities or between an entity and its property. KGs play an important role in various applications, e.g., question answering, browsing knowledge, and data visualization; some well-known examples of KGs are DBpedia[10], Freebase[11], and YAGO[12]. Such KGs contain much useful knowledge, but it is obvious that new knowledge emerges every day. Unfortunately, most new knowledge is published in the form of natural language text, and it is not straightforward to transfer this to a KG. Furthermore, the rate of publication of natural language text is increasing dramatically [20]. As a result, a large amount of knowledge remains available only as text. Consequently, there is an urgent need for a way to populate a KG with knowledge obtained from text.

Recently, many open information extraction approaches have proposed the extraction of triples (subject, predicate, object) from text [43, 54, 55]. However, those studies have primarily focused on ways to extract the elements of triples, and they do not consider how this information can be linked to entities or predicates in other KGs. Consequently, even once knowledge is added to a KG, it cannot be used efficiently due to this problem of heterogeneity. Although there have been many approaches to forming links between a subject or an object of a triple and its identical entity in a KG, there are only a few studies [56, 73–75] that have considered linking the predicate to its counterpart in a KG. Furthermore, most proposed approaches to the predicate linking task use statistical knowledge patterns to identify the identical predicate in a KG. Although such an approach could establish many solid links between predicates, due to the sparsity of text, it is enormously difficult to determine all possible patterns for such links.

4.2 Definition and Problem

In this section, we formally define some particular terms, and we formalize the predicate linking task.

Definition 4.1. Triple A triple describes the relationship of a set of entities. The elements in a triple consist of the subject (S), predicate (P), and object (O), and the triple is denoted as (S, P, O).

Definition 4.2. Triples A set of text triples T_t is a collection of triples that have been extracted from text by any open information extraction system: $T_t = \{(S_{t_i}, P_{t_j}, O_{t_k})$ $| \exists i, j, k,\ 1 \leq i \leq |S_t|, 1 \leq j \leq |P_t|, 1 \leq k \leq |O_t|\} where |S_t|, |P_t|\ and\ |O_t|$ are the numbers

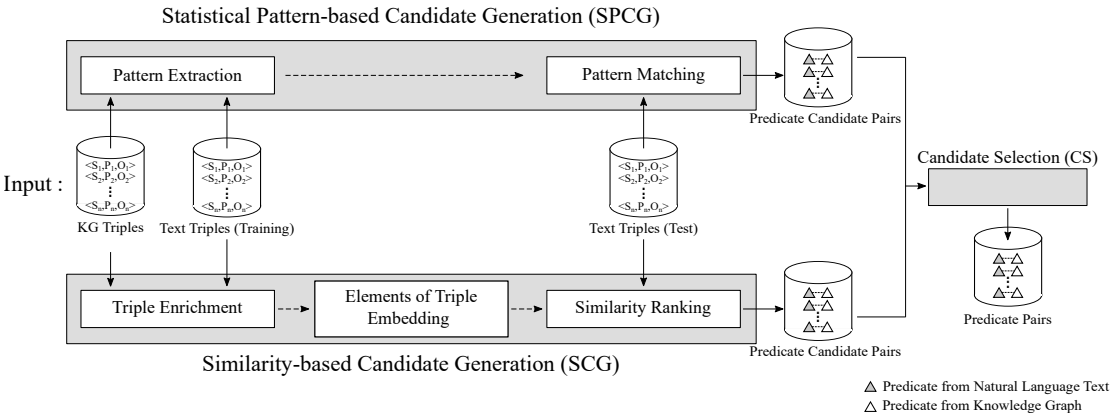


FIGURE 4.1: Architecture of our approach for predicate linking

of unique subjects, predicates, and objects, respectively, that have been extracted from text. For example, given the sentence "Barack Obama was born in Honolulu, Hawaii", an open information extraction system generated $(Barack\ Obama, was\ born\ in, Honolulu\ Hawaii)$ as the text triple.

Definition 4.3. A set of KG triples T_{KG} is a collection of triples in KG, $T_{KG} = \{(S_{KG_x}, P_{KG_y}, O_{KG_z}) \mid \exists x, y, z,\ 1 \leq x \leq |S_{KG}|, 1 \leq y \leq |P_{KG}|, 1 \leq z \leq |O_{KG}|\}$, where $|S_{KG}|, |P_{KG}|$ *and* $|O_{KG}|$ are the numbers of unique subjects, predicates, and objects in the KG. For example, given DBpedia as the KG, the sample KG triple is $(dbr^1\colon Barack\_Obama\ , dbo^2\colon birthPlace,\ dbr\colon Hawaii)$

Definition 4.4. Predicate Linking : Given a set of text triples T_t and a set of KG triples T_{KG} as the input, the predicate mapping task is to identify the P_{KG_y} that is equivalent to P_{t_j} denoted by $P_{t_j} \equiv P_{KG_y}$, given $\langle S_{t_i}, O_{t_k} \rangle$, which corresponds to P_{t_j}. For example, given the text triple (Barack Obama, was born in, Honolulu Hawaii), the predicate "was born in" should be linked to dbo:birthPlace in the context of $\langle Barack\ Obama,\ Honolulu\ Hawaii \rangle$, while given the text triple (Barack Obama, was born in, 1961), it should be linked to dbo:birthDate in the context of $\langle Barack\ Obama, 1961 \rangle$.

4.3 Methodology - HRSim

In this section, the architecture of our approach is described. Our approach takes a set of text triples T_t and a set of KG triples T_{KG} as input, and then as output, it provides a link between the predicate of each text triple and its identical predicate (if any) in the KG. As shown in Figure 4.1, our approach has three main components: (1) statistical pattern-based candidate generation (SPCG); (2) similarity-based candidate

[1]http://dbpedia.org/resource/
[2]http://dbpedia.org/ontology/

generation (SCG); and (3) candidate selection (CS). The SPCG component captures statistical knowledge patterns between text triples and KG triples and then uses these patterns to generate candidate pairs. The SCG component enriches the text triples and KG triples by using bootstrapping prior to embedding their elements in a distributed representations of elements; those vectors are used to compute the similarity between a text predicate and a KG predicate in order to generate predicate candidate pairs. The CS component selects the most suitable predicate candidate. As a result, a link is established between a text predicate and a KG triple that contains an identical predicate. The details of each component of our approach are given below.

4.3.1 Statistical Pattern-based Candidate Generation

The SPCG component extracts statistical patterns of the text predicates and KG predicates and then uses these patterns to link the predicates. As depicted in Figure 4.1, SPCG consists of two modules: pattern extraction and pattern matching. The details of each module are given below.

4.3.1.1 Pattern Extraction

The pattern extraction module extracts the statistical patterns of the text predicate and the KG predicate. The strategy is similar to that used by Exner et al. [56] to extract statistical patterns. We begin by mapping the subject and the object of the text triple to the KG by using an entity disambiguation system [98]. Then, if the subject and object of the text triple are similar to the subject and 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. Finally, the types of the subject and object are used as a constraint to generalize the statistical pattern. The types considered in our approach are the 43 top-level types of DBpedia. An example of pattern extraction is as follows. Given the (Barack Obama, was born in, Honolulu Hawaii) and DBpedia triples, the entity disambiguation system maps the given triple to (dbr:Barack_Obama, was born in, dbr:Hawaii). In the second step, we find the DBpredia triple (dbr:Barack_Obama, dbo:birthPlace, dbr:Hawaii), for which the subject and the object are similar to the triple in the first step. In the third step, dbr:Barack_Obama and dbr:Hawaii of the subject and object are generalized to person and place, and then we extract the statistical pattern that (PERSON, was born in, PLACE) is linked to dbo: birthPlace. Note that the same statistical pattern might link to a different KG predicate; in that case, the most frequently linked KG predicate for a given pattern is assumed to be the correct one.

TABLE 4.1: Examples of triple enrichment for enriching a text triple and a KG triple respectively

Text Triple	Enriched Triple
(Barack Obama, was born in, Hawaii)	(PERSON, was born in, PLACE)
KG Triple	**Enriched Triple**
(dbr:Obama, dbo:birthPlace, dbr:Hawaii)	(Barack Obama, dbo:birthPlace, Hawaii) (PERSON, dbo:birthPlace, PLACE)
(dbr:Obama, dbo:birthDate, 1961-08-04)	(Barack Obama, dbo:birthDate, 1961 08 04) (PERSON, dbo:birthDate, DATE)

4.3.1.2 Pattern Matching

The pattern matching module uses the statistical patterns and the text triple to generate the candidate predicate for the linking task. In order to utilize the statistical patterns, we first need to identify the type of the subject and object of the text triple in order to create a new pattern; this is done by any entity type classification system. The new pattern is compared with existing statistical patterns in an attempt to locate an identical pattern; if one is found, a link to the KG predicate is generated. For example, given the text triple (Shinzo Abe, was born in, Tokyo Japan), by using the entity type classification system, the pattern (PERSON, was born in, PLACE) is created. Then, using the statistical pattern generated by the pattern extraction module, the predicate "was born in" in the text triple (Shinzo Abe, was born in, Tokyo Japan) is linked to dbo:birthPlace; this forms the predicate candidate pair.

4.3.2 Similarity-based Candidate Generation

The SCG component uses the similarity of distributed representations to generate a predicate candidate pair comprising a predicate in a text triple and a predicate in the KG. As shown in Figure 4.1, SCG has three modules: (1) triple enrichment; (2) elements of triple embedding; and (3) similarity ranking. The details of each module are discussed below.

4.3.2.1 Triple Enrichment

The triple enrichment module enriches the text triples and the KG triples and then integrates all triples in order to create *bootstrapping triples* that will be used in a later

module for learning the distributed representations. In this module, only the subject (domain) and range (object) of each triple are enriched. To enrich each text triple, we use the entity type classification system. To enrich a KG triple, we use SPARQL to query the name and the type of the subject and the object of the triple by using the rdfs:[3]label and rdf:type, respectively. As the target for the enrichment, we consider 43 top-level DBpedia types and also include date and number type. Since there is a difference between the types obtained by the entity type classification system and those obtained by the rdf:type property, substring matching is used to match DBpedia types to those provided by the entity type classification system. Examples of enriched triples are shown in Table 4.1.

4.3.2.2 Elements of Triple Embedding

The elements of the triple embedding module uses the bootstrapping triples to learn the distributed vector representations of elements in triples. The idea of the learning is to embed an element of a triple into the continuous vector space by using the other elements in the same triple. In the learning process, distributed representations of other elements in a triple are used to predict the target element in the same triples. Although this learning process is similar to the CBOW architecture in the study [64], we use all other elements, which are not some elements in the traditional context window, to predict the target element in the same triple. Also, since this study focuses on the predicate linking, we concatenate a word sequence of a predicate to create one word for representing that predicate. Formally, given the bootstrapping triple i^{th} denoted by BT_i, e.g. (Barack Obama, was born in, Hawaii), in the set of bootstrapping triples BT, $1 \leq i \leq |BT|$, and a sequence of words of elements, $w_1, w_2, w_3, ..., w_n$ in BT_i (e.g. Barack, Obama, was_born_in, Hawaii), the model is to maximize the following objective function.

$$L = \sum_{i=1; w_e, w_{c_j} \in BT_i}^{|BT|} log\ p(w_e | w_{c_1}, w_{c_2}, w_{c_3}, ..., w_{c_n}) \tag{4.1}$$

where w_e is the target word of the element in the triple BT_i, w_{c_j} denotes the other elements in the same triple ($w_{c_j} \neq w_e,\ 1 \leq j \leq n$) and n is a number of elements in triple BT_i. The conditional probability $p(w_e | w_{c_1}, w_{c_2}, w_{c_3}, ..., w_{c_n})$ is computed by the following softmax function.

$$p(w_e | w_{c_1}, w_{c_2}, w_{c_3}, ..., w_{c_n}) = \frac{exp(\bar{v} \cdot v_{w_e})}{\sum_{u=1}^{|W|} exp(\bar{v} \cdot v_u))} \tag{4.2}$$

[3]https://www.w3.org/TR/rdf-schema/

where W is the set containing complete words of all elements in bootstrapping triples, v_{w_e} is the distributed representations of the element w_e, $\bar{v}$ is an average distributed representations of $w_{c_j}(v_{w_{c_j}}; w_{c_j} \neq w_e,\ 1 \leq j \leq n)$ and v_u is the distributed representations of the word u ($u \in W$).

Generally, the computational cost for summing over W is very expensive. Mikolov et al [64] therefore introduced *Negative Sampling* (NS) to transform the original objective function to the feasible computation one. Based on NS, $log\ p(w_e|w_{c_1}, w_{c_2}, w_{c_3}, ..., w_{c_n}$ in Eq. 4.1 is therefore derived as follows.

$$\log \sigma(\bar{v} \cdot v_{w_e}) \quad + \sum_{j=1; w_{ns} \in BT'_{neg_j}}^{|BT'_{neg}|} \log \sigma(-\bar{v} \cdot v_{w_{ns}}) \tag{4.3}$$

where $\sigma(x) = 1/(1 + exp(-x))$, BT'_{neg_j} is randomly generated triples (negative triples) and $v_{w_{ns}}$ is an distributed representations of w_{ns}, which is the negative sampling of w_e.

4.3.2.3 Similarity Ranking

The similarity ranking module computes the similarity between a text predicate and the KG predicates, and then the scores are used to form a ranked list of KG predicates. We propose two functions, $VSim\_P$ and $VSim$, for computing the similarity score between a text predicate and a KG predicate:

$$VSim\_P(P_{T_i}, P_{KG_j}) = \frac{\bar{P}_{T_i} \cdot \bar{P}_{KG_j}}{|\bar{P_{T_i}}||\bar{P_{KG_j}}|} \tag{4.4}$$

$$VSim(P_{T_i}, P_{KG_j}) = \delta(\frac{\bar{P_{T_i}} \cdot \bar{P_{KG_j}}}{|\bar{P_{T_i}}||\bar{P_{KG_j}}|}) + (1 - \delta)(\frac{SO(P_{KG_j}) \cdot (\bar{S_{P_{T_i}}} - \bar{O_{P_{T_i}}})}{|SO(P_{KG_j})||\bar{S_{P_{T_i}}} - \bar{O_{P_{T_i}}}|}) \tag{4.5}$$

$$SO(P_{KG_j}) = \frac{\sum_{n=1}^{N}(\bar{S}_{P_{KG_n}} - \bar{O}_{P_{KG_n}})}{N} \tag{4.6}$$

where $(S_{P_{T_i}}, P_{T_i}, O_{P_{T_i}})$ are the subject, the predicate and the object of the triple T_i, respectively, T_i is a text triple, P_{KG_j} is a predicate in KG, $S_{P_{KG_{j_n}}}$ and $O_{P_{KG_{j_n}}}$ are the n^{th} pair of a subject and an object, of which predicate is P_{KG_j} in KG, N is the number of triples in KG, whose predicate is P_{KG_j}, and δ is a parameter that determines the weighting between the predicate similarity and the context similarity.

The idea of $VSim\_P$ in Eq. 4.4 is that since the distributed vector representations of identical elements are closed in the same region of the vector space, the cosine between these vectors can be used to measure their similarity. However, the context $\langle subject,\ object \rangle$ might alter the meaning of the predicate and so it must be taken into account. We therefore generalize $VSim\_P$ to $VSim$. The $VSim$ similarity score function in Eq. 4.5 therefore consists of two terms: the first term directly computes the similarity between the predicates, while the second term computes the similarity between the contexts; this is done in order to validate the suitability of the predicate with its context. This is based on the assumption that the more the context is suitable, the more important it is that they be linked. Because the first and the second terms in Eq. (4.5) are different, a parameter is introduced to adjust the weight given to each. Eq. (4.6) computes the average vector representation of the context of P_{KG}.

4.3.3 Candidate Selection

The CS component selects a predicate from the KG that is a candidate for mapping to the predicate of a text triple. In this component, priority is given to the predicate candidate, which is generated by the SPCG component. If such a predicate candidate does not exist, the predicate candidate generated by the SCG component is considered. In the SCG component, the predicate candidate in KG, which provides the highest similarity score for the text predicate, is considered. If the similarity score between the candidate predicate and the text predicate is greater than the threshold θ, then the predicate of the text triple is linked to the candidate.

4.4 Methodology - Compositional Vector

Representations of words play a crucial role in many models of natural language processing tasks such as part of speech tagging [99] and parsing [100]. Conventionally, a word as an atomic unit of language is transformed to a vector by the one-hot encoder as its representation. However, such representations are presented in the high-dimensional space, which leads to the curse of the dimensionality [83]. To overcome the curse of the dimensionality, the concept of distributed representation was proposed [83]. Distributed representation of a word uses a dense real-value vector to represent the word in the low-dimensional vector space. Since the representation is projected to the low-dimensional vector space, this representation does not encounter with the curse of the dimensionality problem. Also, both synthetic and semantics of words are encoded into their representation [101].

Recently, many studies [64, 83, 84, 102] focus on modelling the distributed representation of words. One of the prominent approaches is word2vec [64, 84]. By leveraging the shallow neural network model trained over the unlabeled text corpus, words in the corpus are embedded as the distributed representations [64, 84]. Based upon the hypothesis of the distributional occurrence of context words, words sharing similar meaning occur in similar context words. Under this hypothesis, the neural network model can learn the distributed representations of words, which can capture both the function and the meaning of the words. However, to learn the distributed representations of words, each word in the text corpus is treated as an individual token. Consequently, the distributed representations of compound words, which typically consist of two or more tokens, could not be represented directly.

In order to cope with the problem of compound words, many approaches [64, 85] have been proposed. These approaches detect compound words by a statistical strategy and replace compound words by unique tokens. Such approaches can learn the actual representations of compound words; however, some representations of compound words may not be able to learn because we may not be able to detect all compound words. Furthermore, some detected compound words may appear infrequently; in consequence, the representations of such compound words cannot be learned. Estimating the distributed representation of a compound word by using its individual words is the promising solution, because we usually can learn the representations of the words in the compound word. For example, given the compound word "Hida Station", we may not be able to learn the actual representation of the compound word "Hida_Station". However, we can learn each representation of the words "Hida" and "Station", since these words, in particular "Station", appear frequently. Our assumption is that the representation of the compound word "Hida_Station" could be estimated by the representations of "Hida" and "Station". In this study, we want to estimate the distributed representations of compound words by using their individual word representations. Currently, the widely-used workaround for estimating the distributed representation of a compound word is to average vector representations of individual words in the compound word. Although many studies [103–105] utilized average distributed representations of words, the meaning of a compound word is not simple composition of the individual words [64].

In this section, we therefore introduce the recurrent neural network (RNN)-based approach for estimating distributed representations of compound words by their word representations. In the RNN-based approach, distributed representations of words in a compound word are given to the model in order to estimate the distributed representation of the compound word.

4.4.1 Learning Distributed Representation of Words

In the learning process, if a sequence of words contains compound words, we create a new sequence and replace any compound words in the sequence with a unique token. For example, given the sequence "San Francisco is located in United State of America", the new sequence "San_Francisco is located in United_State_of_America" is created. Both sequences are used to learn the representations of words. After learning the distributed representation of words, the distributed representation of the compound word, e.g. "San_Francisco", and individual words, e.g. "San" and "Francisco", are acquired. The distributed representations of compound words and individual words are given to our RNN-based approach so that our RNN-based approach can later learn to estimate the representations of compound words.

4.4.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are circular neural networks, which are typically used for modelling a sequence of arbitrary length. Formally, given a sequence of input $x_1, x_2, x_3, ..., x_n$, the RNN model learns to map the given sequence to a sequence of output $y_1, y_2, y_3, ..., y_n$, where each input and output at time t are corresponded. At any time t, the RNN model learns the current latent state with the input data at t and the previous latent state at time $t-1$. Then, the current latent state is used to predict the output. Based on this concept, the most basic RNN [106] is derived as the following equation:

$$h_t = f(W_{i,h}x_t + W_{h,h}h_{t-1} + b_h) \tag{4.7}$$

$$y_t = g(W_{h,y}h_t + b_y) \tag{4.8}$$

where x_t is the input vector at time t, h_t is the vector of hidden layer at time t, y_t is the prediction vector at time t, $W_{i,h}, W_{h,h}, W_{h,y}$ are parameter matrices, b_h, b_y are the bias parameters for the network and f, g are the activation function, e.g. sigmoids. Although the RNN model can deal with the sequential data, long-term dependencies cannot be captured due to the vanished and exploding gradient [107]. Recently, more advanced RNN models, e.g. long short-term memory (LSTM) [108] and gated recurrent unit (GRU) [109], are proposed to handle long-term dependencies. In this study, we focus on the GRU unit, which had been reported that it provides better performances than the LSTM unit in many datasets [109]. The GRU unit in our approach is descried

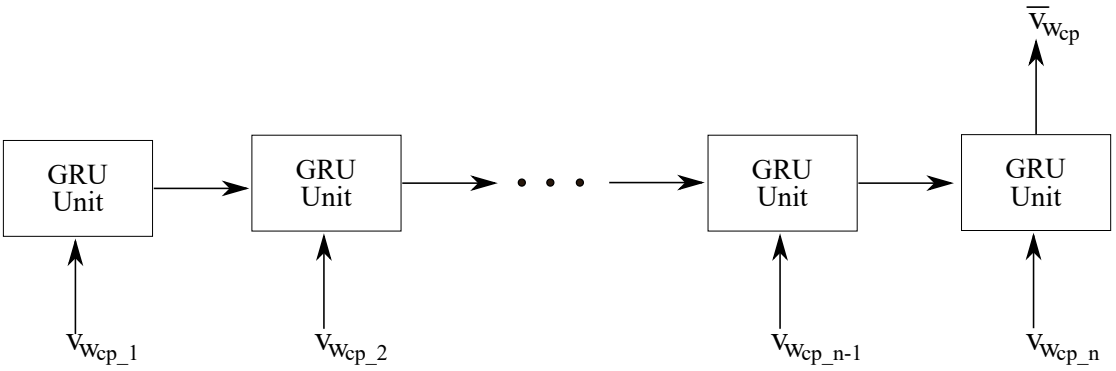


FIGURE 4.2: The RNN-based approach for estimating distributed representation of a compound word by its word sequence

as follows:

$$z_t = f(W_z x_t + U_z h_{t-1} + b_z) \tag{4.9}$$

$$r_t = f(W_r x_t + U_r h_{t-1} + b_r) \tag{4.10}$$

$$h_t = g(W_h x_t + U_h(r_t \circ s_{t-1}) + b_h) \tag{4.11}$$

$$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ h_t \tag{4.12}$$

where z_t is the update gate vector at time t, r_t is the reset gate vector at time t, h_t, h_{t-1} is the vector of hidden layer at time t and $t-1$, s_t is the output vector at time t, x_t is the input vector at time t, W and U are parameter matrices, b is the bias parameters, f and g are the activation function and $\circ$ represents the Hadamard product operation.

In our study, we only use the output from the last state as the distributed representation of a compound word as shown in Figure 4.2. Given a sequence of representations of the compound word w_{cp} $(v_{w_{cp\_1}}, v_{w_{cp\_2}w_{cp\_3}}, ..., v_{w_{cp\_n}})$, the RNN-based approach calculates the estimated distributed representation of the compound word w_{cp} from the last state as follows.

$$\bar{v}_{w_{cp}} = z_{Last} \circ s_{Last-1} + (1 - z_{Last}) \circ h_{Last} \tag{4.13}$$

where $\bar{v}_{w_{cp}}$ is the estimated representation of the compound word w_{cp}, z_{Last}, r_{Last} and h_{Last} are the update gate vector, the reset gate vector and the hidden layer vector at the last sequence respectively and h_{Last-1} is the hidden layer vector before the last state. In the learning process, the idea is that we want to minimize the error between the estimated representation and the actual representation of the compound word. Therefore, the objective of the whole network learning is to minimize the following function:

$$\mathcal{L}(v_{w_{cp}}, \bar{v}_{w_{cp}}) = \|v_{w_{cp}} - \bar{v}_{w_{cp}}\|_2^2 \tag{4.14}$$

where $v_{w_{cp}}$ is the actual representation of the compound word w_{cp}, learned by replacing the compound word w_{cp} in the sequence with the unique token (as described in Section 4.4.1), and $\bar{v}_{w_{cp}}$ is the estimated representation of the compound word w_{cp} by the RNN-based approach.

4.5 HRSim Experiments

In this section, we present experiments that we conducted to assess the performance of distributed representation of elements in triples and the performance of our approach for the linking predicates; for this task, we used three created benchmark datasets.

4.5.1 Experimental Setup

We performed two experiments: the ranking experiment and the classification experiment. In the ranking experiment, we compare the similarity ranking performance of the vector representation-based similarity with that of other similarity metrics. In the classification experiment, we evaluate the performance of our approach for the predicate linking task.

In the experiments, 120,000 Wikipedia articles were randomly selected and preprocessed to create text triples. In the preprocessing step, all formatting and any hyperlinks were removed. We then used a co-reference resolution system, the Stanford NLP tool [110], to convert the pronoun in each sentence to its corresponding proper name. Next, text triples were extracted from each sentence by OLLIE [55], which is a state-of-the-art open information extraction system. For the KG triples, we used DBpedia [10], a well-known KG. In this study, the targets of the predicate linking were the 2,800 DBpedia ontology properties.

Based upon the generated text triples, we created three benchmark datasets. The first two datasets (synthetic datasets) were automatically constructed, while the third (gold standard) was manually created by an expert. The synthetic datasets are used for a quantitative evaluation, while the gold standard is used for a qualitative evaluation.

To construct the synthetic datasets, the predicates of the text triples and those in DBpedia were automatically linked. The linking strategy is as follows: if both the subject and the object of a text triple are identical to their respective counterparts in a DBpedia triple, we link the predicate of the text triple to that of the DBpedia triple. Figure 4.3 shows an example for constructing these datasets. As shown in this figure, the subject and the object of the text triple are identical to the respective parts in the

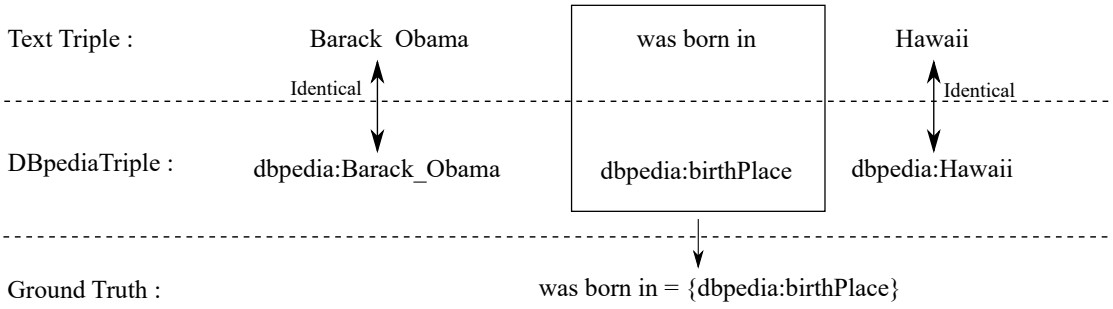


FIGURE 4.3: An example of data construction

DBpedia triple. Consequently, the predicates, "was born in" and dbpedia:birthPlace, are assumed to be identical. This process was performed using DBpedia Spotlight [98]. Although this strategy establishes many predicate pairs, some of them may be incorrect. For example, consider the text triples (Barack Obama, was born in, USA) and (Barack Obama, lives in, USA) and a KG triple dbr:Obama, dbo:birthPlace, dbr:USA, "was born in" is linked to dbo:birthplace, which is correct, but "lives in" is also linked to dbo:birthplace, which is incorrect. To avoid such problems, we automatically remove any data for which the same subject and object pair results in links to multiple predicates. Our synthetic dataset contains more than 100,000 text triples for which the predicates are each linked to a unique DBpedia predicate. Our synthetic dataset has two characteristics: predicates for which the links form a one-to-one relation between the predicates in the text and those in the KG, and those for which there is a one-to-many relation, where the same text predicate is linked to multiple KG predicates. An example of a one-to-many relation is that "was born in" could be linked to either dbo:birthPlace or dbo:birthDate, based on the context $\langle subject, object \rangle$ in the triple. Given the triple context $\langle BarackObama, HonoluluHawaii \rangle$ and $\langle BarackObama, 1961 \rangle$, "was born in" is linked to dbo:birthPlace or dbo:birthDate, respectively. Because of the difference in relations, we separated the synthetic data in two datasets. The first synthetic dataset, SYN(1-1), contains only those predicates that have a one-to-one relation, while the second synthetic dataset, SYN(1-N), has only those that have a one-to-many relation.

To construct the gold standard dataset, we randomly picked text triples and then asked an expert to create the links between the text predicate and the DBpedia predicate. We have made our three benchmark datasets[4] available for download in order to encourage the study of the predicate linking task, as described in this paper. Table 4.2 presents the statistics of the three benchmark datasets used in the experiments.

The implementation of our approach is as follows. In the pattern extraction module, DBpediaSpotlight[98] is used as the entity disambiguation system. Also, this system is used in the pattern matching module and the triple enrichment module to map the

[4]http://ri-www.nii.ac.jp/VSim/datasets.zip

TABLE 4.2: Statistic of the datasets in the experiments

Dataset	# Triples	Predicate		Subject	Object	
		# Text	# KG	# Entity	# Entity	# Literal
SYN (1-1)	33,391	30,732	322	17,021	7,317	4,061
SYN (1-N)	84,063	7,503	376	35,036	12,521	14,695
Gold Standard	300	186	91	295	239	47

entity (subject or object) of a text triple to an appropriate DBpedia entity. We then use rdf:type to query the type of entity from DBpedia. This method is used as the workaround of the entity type classification system. In the elements of triple embedding module, the word2vec library[5] is used for training distributed representations of the elements in triples. In the learning process, we consider a triple as a sequence of words from subject, predicate and object of the triple in order. Therefore, to use all elements of a triple the window size was set as the maximum length of the sequence in bootstrapping triples, while other parameters were set by their default values. In our approach, there are two hyperparameters: the weight δ and the threshold θ. These were set by using a grid search algorithm on the training dataset. The interval of the parameter searching was [0.00, 1.00], and the step interval was 0.01. The hyperparameters that performed best in training were used to test the result. In the experiments, we used a ten-fold cross-validation strategy for the datasets SYN(1-1) and SYN(1-N), and we used a combination of SYN, SYN(1-1), and SYN(1-N) to train the hyperparameter for the gold standard dataset.

4.5.2 Ranking Experiment

The ranking experiment investigated the accuracy of the ranking of the similarity of the distributed representation of predicates in triples by comparing the results with those of other metrics. In this experiment, we computed the similarity between a text predicate and a KG predicate, and then the closeness of the predicates was ranked based on this score (a higher similarity score resulted in a rank). More specifically, given the text triple $(S_{t_i}, P_{t_j}, O_{t_k})$ and the set of DBpedia predicates $\{P_{DB_1}, P_{DB_2}, P_{DB_3}, ..., P_{DB_n}\}$, we computed the similarity score between P_{t_j} and P_{DB_x}, $1 \leq x \leq n$. Then, the P_{DB_x} were ranked according to their similarity scores.

In our approach, we propose two distributed representation-based similarity metrics, $VSim\_P$ and $VSim$, for ranking the similarity between a text predicate and a KG

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

predicate. We compared them with three traditional string-based similarity metrics: (1) the edit distance-based similarity metric, which is widely used in the similarity ranking; (2) the N-gram-based similarity metric (N = 3, 5); and (3) the wordnet-based similarity metric [81]. Note that, since a DBpedia predicate is represented by a URI, we can use rdfs:label to retrieve the text form of the DBpedia predicates. However, $VSim\_P$ and $VSim$ can simply use the URIs of the DBpedia predicates to compute the similarity score.

The proportion of accurate links and the average rank were used as the evaluation metrics. We defined and measured two indices, Hit@1 and Hit@10. Hit@1 is the proportion of times that a text predicate is correctly linked to a KG predicate when their similarity is of the first rank; Hit@10 is the same as Hit@1, except for when the similarity rank is not larger than 10. The average rank (Mean Rank) is the rank averaged over text predicates correctly linked to KG predicates. We evaluated the results in two settings: the *Raw* setting measures the results obtained with the original ranking, and the $Filter$ setting measures the results after the original ranking has been filtered by using the domain and the range of the text predicate as a constraint, and then removing from the ranked list any KG predicates that are not in compliance with the constraint. For example, given the triple (Barack Obama, was born in, Hawaii), whose domain and range are person and place, respectively, the DBpedia predicate dbo:birthDate is removed from the ranked list because its range, which is date, does not conform to the constraint. The results of the ranking experiment are listed in Table 5.1.

The results of our distributed representation-based similarity metrics and those of three baseline metrics are listed in Table 5.1. As shown in the table, the results indicate that the proportion of the accuracy of $VSim\_P$ and $VSim$ outperformed the other similarity metrics in both the *Raw* and $Filter$ settings. These results conform to our hypothesis that the heterogeneous problem in the open information extraction task is severe for the traditional string-based similarity metrics. Due to the significant improvement, the distributed representation-based similarity metric is a promising technique for computing the similarity between the predicate of triples.

Furthermore, considering the results of $VSim$, we found that the result is significantly improved from the results of $VSim\_P$ even in the SYN(1-1) dataset. In the SYN(1-1) dataset, the context, $\langle subject,\ object \rangle$, is less influence to its predicate because each predicate individually aligns to the exact KG predicate. Consequently, the context in the triples also plays an important role in the predicate ranking task, even though the predicate itself is sufficient to identify its identical predicates.

TABLE 4.3: Comparison results between distributed representation of elements in triples with other similarity metric for the predicate ranking task

Dataset	Similarity Metric	Hit@1(Acc. %)		Hit@10(Acc. %)		Mean Rank	
		Raw	Filter	Raw	Filter	Raw	Filter
SYN (1-1)	Edit Distance	0.00	0.06	0.00	0.27	649.80	27.61
	N-gram (N=3)	0.01	0.07	0.02	0.31	607.66	28.42
	N-gram (N=5)	0.00	0.06	0.02	0.30	605.63	28.09
	WordNet	0.00	0.07	0.02	0.36	450.86	24.21
	VSim_P	0.04	0.31	0.30	0.69	167.05	11.13
	VSim	**0.29**	**0.38**	**0.54**	**0.80**	**112.17**	**8.04**
SYN (1-N)	Edit Distance	0.00	0.04	0.00	0.20	753.82	33.58
	N-gram (N=3)	0.01	0.06	0.05	0.36	544.93	26.87
	N-gram (N=5)	0.00	0.05	0.04	0.35	543.89	26.71
	WordNet	0.02	0.08	0.03	0.33	474.29	25.68
	VSim_P	0.04	0.16	0.15	0.64	249.17	12.42
	VSim	**0.15**	**0.25**	**0.38**	**0.79**	**205.67**	**7.87**
Gold Standard	Edit Distance	0.00	3.00	0.00	24.67	805.96	30.453
	N-gram (N=3)	7.33	20.67	18.00	50.33	476.86	20.58
	N-gram (N=5)	2.00	16.33	13.33	49.67	468.09	20.32
	WordNet	7.97	13.72	10.61	43.09	157.54	18.87
	VSim_P	25.67	55.67	55.67	87.67	58.39	4.23
	VSim	**35.33**	**56.00**	**59.00**	**91.00**	**50.13**	**3.75**

4.5.3 Classification Experiment

The classification experiment evaluated the performance of our approach in the predicate linking task. In the classification experiment, the linking between a text predicate and a KG predicate is testified whether the approach selects a KG predicate, which corresponding to a text predicate, or not. This is a classical multi-label classification problem: given the text triple $(S_{t_i}, P_{t_j}, O_{t_k})$, the approach selects $P_{DB_x} \in \{P_{DB_1}, P_{DB_2}, P_{DB_3}, ..., P_{DB_n}\}, 1 \leq x \leq n$ as the output.

In our approach, *VSim* is leveraged as the similarity metrics in the SCG component. The statistical knowledge pattern approach in the study [56], which is closely related to our approach, is selected as the baseline. In the experiment, the micro/macro evaluation of precision, recall and F1 are measured to evaluate our approach and the baseline. The macro evaluation averages the performance for each predicate across the dataset, while the micro evaluation aggregates the performance of all predicates in the dataset.

TABLE 4.4: The classification results of our approach the predicate linking task on three benchmark comparing with the baseline

Approach	Dataset	Macro			Micro		
		Precision	Recall	F1	Precision	Recall	F1
Baseline	SYN (1-1)	**1.0000**	0.0241	0.0471	**1.0000**	0.1130	0.2031
Our approach		0.1245	**0.1752**	**0.1456**	0.3720	**0.3228**	**0.3457**
Baseline	SYN (1-N)	**0.1207**	0.0181	0.0315	**0.5029**	0.4256	0.4610
Our approach		0.0592	**0.0585**	**0.0589**	0.4702	**0.4649**	**0.4675**
Baseline	Gold	**0.7217**	0.5600	0.6306	**0.7693**	0.6400	0.6987
Our approach	Standard	0.6902	**0.6660**	**0.6778**	0.7491	**0.7367**	**0.7428**

The micro/macro F1 are harmonic mean between the micro/macro precision and recall respectively. The micro/macro precision and recall are computed by Eq. 4.15 and Eq. 4.16

$$Precision_{micro} = \frac{\sum_{i=1}^{|P_{KG}|} |S_i \cap \hat{S}_i|}{\sum_{i=1}^{|P_{KG}|} |\hat{S}_i|}, \quad Recall_{micro} = \frac{\sum_{i=1}^{|P_{KG}|} |S_i \cap \hat{S}_i|}{\sum_{i=1}^{|P_{KG}|} |S_i|} \tag{4.15}$$

$$Precision_{macro} = \frac{1}{|P_{KG}|} \sum_{i=1}^{|P_{KG}|} \frac{|S_i \cap \hat{S}_i|}{|\hat{S}_i|}, \quad Recall_{macro} = \frac{1}{|P_{KG}|} \sum_{i=1}^{|P_{KG}|} \frac{|S_i \cap \hat{S}_i|}{|S_i|} \tag{4.16}$$

where P_{KG} is the set of predicates in KG, S_i is the set of predicates of text triples, which belong to P_{KG_i} and $\hat{S}_i$ is the set of predicates of text triples, which are predicted as P_{KG_i}.

Table 5.2 shows the results of our approach and that of the baseline. The experimental results indicate that in overall our approach, leveraging the distributed representation-based similarity, outperformed the statistical pattern approach. Although the precision results of the baseline are better than our approach due to the nature of statistic knowledge patterns, whose results are highly precise, the recall is relatively low when comparing with our approach. As a result, the F1 results of our approach can outperform the baseline in all datasets.

The low recall results in the baseline infers that we could not extract the majority of the statistical knowledge patterns due to the sparsity of natural language text, especially in the SYN datasets, the SYN(1-1) and the SYN (1-N) datasets. In the SYN datasets, the variation of predicate in text triples is enormous as shown in the Table 4.2. Although the statistical knowledge pattern-based approach could efficiently perform well on the

limited size of the predicate vocabularies as in the gold standard or between KGs, it still requires a huge amount of the training data to acquire enough knowledge patterns. This result conforms to our hypothesis that it is considerably difficult to extract all possible patterns. Therefore, our approach, leveraging the distributed representation-based similarity, could efficiently discover identical predicates in KG, which could not be achieved by the statistical knowledge patterns.

4.6 Compositional Vector Experiment

4.6.1 Experimental Setup

We start with describing the text corpus, which is used to learn the distributed representations of individual words and compound words. We then provide the details of the dataset for estimating distributed representations of compound words. After that, the implementation of the RNN-based approach and the network training are presented.

4.6.1.1 Corpus.

To learn the distributed representations of individual and compound words, all Wikipedia articles as sequences are used as mentioned in Section 4.4.1. Still, there is the problem regarding locating compound words in the sequence. In order to address this problem, we utilize tags provided by Wikipedia. We assume that a sequence of words in any tag is a compound word. For example, given the sentence [Barack Obama] was [the U.S. president], where [ ] denotes a tag, e.g. a hyperlink tag, we can form two sequences: 1) Barack Obama was the U.S. president and 2) Barack_Obama was the_U.S._president. The implementation of extracting tags and identifying compound words are followed Wiki2Vec[6].

Based upon the text corpus above, we create the pair between a compound word and its individual words as the **dataset** for experiments. From the above example, we can create the (Barack_Obama, Barack Obama) and (the_U.S._pres ident, the U.S. president). The statistic of the dataset is as follows. The number of pairs between compound words and their individual words is 16,369,076 pairs and the average length of the compound words is 3.06 tokens. This dataset is used as the input of the RNN-based approach.

[6] https://github.com/idio/wiki2vec

4.6.1.2 Implementation.

The implementation and the details of the network training are as follows. To learn the distributed representations of words and compound words, we use the python version of word2vec[7]. Most of the parameters are set by the default. Note that, in this setting all words that appear less than 25 times are ignored. However, since we use the text corpus, in which most sequences are duplicated, as the training set, we set the words that appear less than 50 times are discarded. Also, since the dimension of vector and the length of the window size directly affect representations of words, these two parameters are varied in the experiments to investigate their effects. Furthermore, to handle the out of vocabulary problem, words that appear less than 50 times are replaces by the token "UNK".

In our RNN-based approach, we implemented the networks by using tensorflow[8]. The GRU unit is selected as the unit in the RNN model. One hidden layer network is used and a number of nodes in the hidden layer is set at 128. The Adam algorithm [111] is used as the optimization method due to its computational efficiency. The learning rate is set at 0.01. The dataset is fed as the mini-batch, whose size is 50,000 samples. We set the number of the iteration at 5,000, which is enough to reach the convergence.

4.6.2 Experiment 1

Experiment 1 is to investigate the improvement of compound word representations estimated by their individual words using the RNN-based approach. In this experiment, the distributed representations of compound words estimated by the RNN-based approach are compared with the average representations. The average representation is computed by averaging the distributed representations of all individual words in the compound words. To evaluate the improvement, the location and the direction of estimated distributed representations of compound words are considered. The idea to measure the location is that the closer the location to the actual compound word representation, the better it is, while the idea to evaluate the direction is that the more similar the angle to the actual compound word representation, the better it is. Note that the actual compound word representation is the distributed representation of the compound words as a single token, e.g. San_Francisco, learned during the word embedding process. Based on these intuitions, two evaluation metrics: 1) the proportion of the improvement of the

[7]https://radimrehurek.com/gensim/models/word2vec.html
[8]https://www.tensorflow.org/

Euclidean distance, which measures the closeness of the location, and 2) the proportion of the improvement of the cosine similarity, which measures the similarity of the direction, are defined as follows.

$$Location = \frac{\sum_{i=1}^{N} d_{Loc}(v_{w_{cp_i}}, \bar{v}_{w_{cp_i}}, \bar{\bar{v}}_{w_{cp_i}})}{N} \times 100\% \tag{4.17}$$

$$d_{Loc}(v_{w_{cp}}, \bar{v}_{w_{cp}}, \bar{\bar{v}}_{w_{cp}}) = \begin{cases} 1, & \text{if } \|v_{w_{cp}} - \bar{v}_{w_{cp}}\| < \|v_{w_{cp}} - \bar{\bar{v}}_{w_{cp}}\| \\ 0, & \text{otherwise} \end{cases} \tag{4.18}$$

$$Direction = \frac{\sum_{i=1}^{N} d_{Dir}(v_{w_{cp_i}}, \bar{v}_{w_{cp_i}}, \bar{\bar{v}}_{w_{cp_i}})}{N} \times 100\% \tag{4.19}$$

$$d_{Dir}(v_{w_{cp}}, \bar{v}_{w_{cp}}, \bar{\bar{v}}_{w_{cp}}) = \begin{cases} 1, & \text{if } \frac{v_{w_{cp}} \cdot \bar{v}_{w_{cp}}}{\|v_{w_{cp}}\| \|\bar{v}_{w_{cp}}\|} > \frac{v_{w_{cp}} \cdot \bar{\bar{v}}_{w_{cp}}}{\|v_{w_{cp}}\| \|\bar{\bar{v}}_{w_{cp}}\|} \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

where $v_{w_{cp}}$ is the actual representation of the compound word w_{cp}, $\bar{v}_{w_{cp}}$ is the estimated representation of the compound word w_{cp} by the RNN-based approach, $\bar{\bar{v}}_{w_{cp}}$ is the average representation of the compound word w_{cp} and N is a number of compound words.

In this experiment, the dimension of vector representations and the length of window size are varied to investigate their effects because both are the major factors that directly influence the representations of compound words. For observing the effect of the dimension of vector representations, we set the dimension size at 100, 200, 300, 400 and 500, while the length of the window size is fixed at 5. For inspecting the effect of the length of the window size, we set the window size at 5,10, 15 and 20, while the dimension of vector representations is fixed at 200. Also, we conduct the experiment with CBOW and Skip-Gram in order to investigate any influence caused by the models. In the experiment, we conduct the experiment by using the 10-fold cross validation technique.

The results of this experiment are listed in Table 4.5. The results show that most of compound words can be better estimated by using the RNN-based approach in both location and direction aspects. Also, the results indicate that the parameters, the dimension of vector representation and the length of the window size, slightly affect the results. However, for the CBOW setting with larger length of window size, the changes are clearly observed. In this case, only around 86%-88% of compound words can better be estimated by using the RNN-based approach in the location aspect, while around 58%-59% of compound words can better be estimated by using the RNN-based approach in the direction aspect. Nevertheless, the improvement of the estimated representations by the

TABLE 4.5: The results of the improvement in aspects of location and direction of estimation in Experiment 1

# Dimension	# Window Size	CBOW		Skip-Gram	
		Location	Direction	Location	Direction
100	5	97.98%	96.70%	99.01%	98.61%
200	5	98.58%	97.14%	99.18%	98.76%
300	5	98.79%	97.16%	99.10%	98.60%
400	5	98.82%	96.92%	98.79%	98.08%
500	5	98.13%	94.84%	98.12%	97.00%
200	10	98.67%	96.50%	99.19%	98.77%
200	15	88.01%	59.74%	99.06%	98.60%
200	20	86.26%	58.10%	98.88%	98.30%

RNN-based approach over average representations still can be observed. Therefore, the RNN-based approach can do better to estimate the representations of compound words than the average representation approach.

Although the RNN-based approach shows the improvement over the average representation method, to learn the RNN-based approach, the training pairs are still required. Therefore, we analyzed the number of training pairs used in the experiment to achieve the result in Table 4.5 by plotting the learning curve. We used 10% of the data for testing the improvement in the location and direction aspects, while the rest of the data are used as the training data. The number of training example pairs are gradually increased. To plot the learning curve, the standard parameter setting[4] is set. Figure 4.4 shows the learning curve of the RNN-based approach. The learning curve shows after the training size is larger than 10,000 pairs, more than 90% of compound words can be estimated better the aspect of location and direction by the RNN-based approach than the average method in both CBOW and Skip-Gram models.

4.6.3 Experiment 2

Experiment 2 is to evaluate the quality of the estimated representations of compound words by the RNN-based approach. This experiment is to further understand how much improvement of the estimated representations of the compound words we achieved. In this experiment, the experiment setting is the same as Experiment 1 and the average representation method is used as **baseline**. However, we change the evaluation method in order to observe the quality of the estimated distributed representations of compound

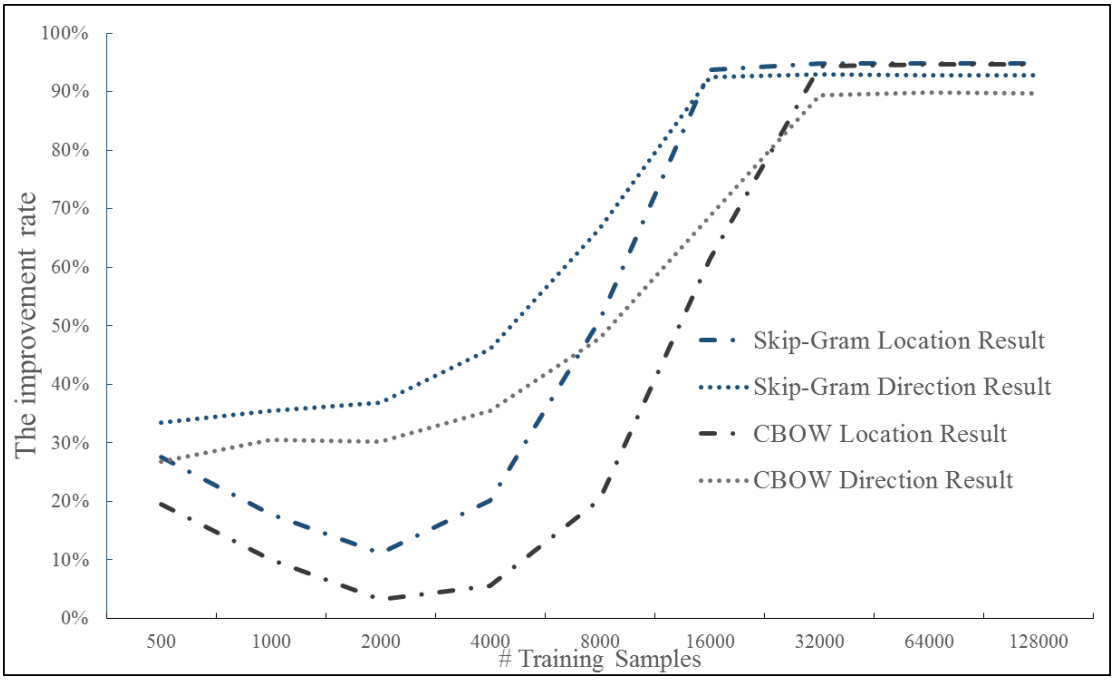


FIGURE 4.4: Learning curve of the RNN approach by using CBOW and Skip-Gram setting with 200-dimensional vector and window size 5 in aspect of location and direction

words. We compute the cosine similarity between the estimated distributed representations of compound words and other word representations in the whole vocabulary. The results are ranked by their similarity score. The higher the similarity score acquire, the lower the rank is. Then, we determine the rank result of the estimated distributed representation by its pair rank. For example given the pair ("Tokyo_Station", "Tokyo" "Station"), the rank result is corresponded to the rank of "Tokyo_Station" in the rank list. In the experiment, we conduct only the ranking by the cosine similarity because many applications [23] used the cosine similarity for evaluating the similarity between representations. Since we need to compare all representations ($\approx$ 968,009 words), only 1,000 pairs are sampled for testing the ranking result, while the rest are used to train the model. To measure the results, we report the mean reciprocal rank (MR), which averages the inverse of the rank and Hit@10 is the proportion of times that the rank of its pair is less than 10.

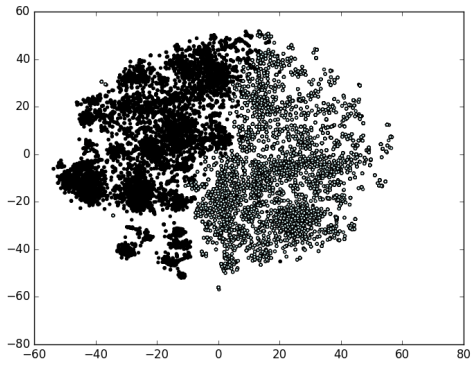
The results of this experiment are listed in Table 5.2. The results show that the RNN-based approach provides significant improvement of the quality of the representations than the baseline in all settings. This result confirms the hypothesis in the study [64], where representations of compound words are not simple average representations of individual words. The experimental results also indicate that the quality of the representations is robust to the length of the dimensional vector representation, while the length of the window size has significant influence. The estimated representations of

TABLE 4.6: The results of quality of the estimated representations of compound words comparing with the representations of compound words

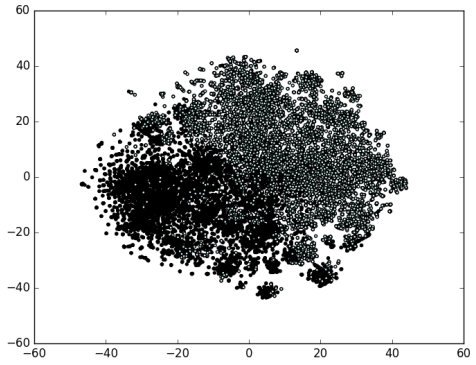
# Dim	# Window Size	Baseline				RNN-based Approach			
		CBOW		Skip-Gram		CBOW		Skip-Gram	
		MR	$Hit@10$	MR	$Hit@10$	MR	$Hit@10$	MR	$Hit@10$
100	5	0.003	0.70%	0.001	0.10%	**0.078**	**22.42%**	0.056	15.92%
200	5	0.006	1.90%	0.002	0.10%	**0.119**	**33.63%**	0.073	21.42%
300	5	0.006	2.20%	0.002	0.40%	**0.133**	**37.54%**	0.076	22.52%
400	5	0.007	2.50%	0.002	0.40%	**0.148**	**42.24%**	0.082	24.42%
500	5	0.007	2.30%	0.002	0.20%	**0.112**	**31.23%**	0.064	17.71%
200	10	0.439	50.55%	0.785	81.48%	0.927	94.90%	**0.988**	**99.10%**
200	15	0.455	52.45%	0.767	79.78%	0.941	95.80%	**0.987**	**99.20%**
200	20	0.466	53.05%	0.784	81.58%	0.939	96.30%	**0.986**	**98.90%**

compound words can be estimated better when the length of the window size becomes larger. One intuition is that the length of the window size is used to capture the long dependency of the context. Since the average length of the compound words in the **dataset** is around 3 tokens, the length of window size at 5 could not capture the whole dependency. Consequently, the quality of the estimated representations is degraded.

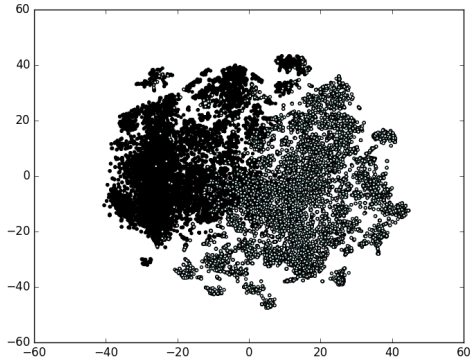
To further understand the representations of compound words and their estimation, the representations of compound words are projected to 2-d space by using the t-distributed stochastic neighbor embedding (t-SNE)[112]. To visualize the representations in the 2-d space, the standard parameter setting[4] is set. Moreover, most of the compound words are person names or location names. When plotting in the 2-d space, black color represents the compound words, which are person, while the grey color denotes the compound words, which is location as shown in Figure 4.5. Figure 4.5 shows that the estimated representations of compound words by using the RNN-based approach are closely similar to the actual representations than using the average representation approach. Furthermore, the average representations show some ambiguity between person and location. This indicates that the average representation could falsely interpret the meaning of the compound words in the vector space.



Actual Representation of Compound Words



Estimated Representation of Compound Words
Using the Average Representation



Estimated Representation of Compound Words
Using the RNN-based approach

FIGURE 4.5: Scatter plot of representations of compound words by their types using CBOW with the 200-dimensional vector and the window size 5

4.7 Summary

Knowledge graphs (KGs) play a crucial role in many modern applications. Consequently, many open information extraction approaches propose the extraction of triples from natural language text in order to populate a KG. Nonetheless, most approaches do not consider forming links between the extracted triples and the KG triples, especially for predicates. Predicate linking is used to identify the predicate in a KG that exactly corresponds to an extracted predicate; this allows the avoidance of the heterogeneity problem when populating a KG. Although there have been a few studies that considered linking predicates, most of them have relied on statistical knowledge patterns, which are not able to generate the possible patterns when dealing with small data. In this paper, we propose a novel distributed representation of the elements in triples and show how this can be used to compute the similarity between predicates in order to find links that would not appear in statistical patterns. There are two aspects to the experimental study of distributed representations of triple elements: one is to evaluate the similarity rank of these representations and compare the results with various similarity metrics; the other is to use these representations to evaluate the links between predicates. We present experimental results that show that the distributed representation of triple elements outperforms other similarity metrics and the linking of predicates is notably improved.

Chapter 5

Knowledge Graph Population

In this Chapter (Chapter 5), we presented **T2KG**: an automatic knowledge graph creation from natural language text. Section 5.1 presented the motivation of the KG construction and the motivation. Then, Section 5.2 described the architecture of T2KG framework for creating a KG from natural language text. Next, we conducted the experiment on KG construction to evaluate the T2KG framework in Section 5.3. Finally, Section 5.4 concluded this Chapter.

5.1 Overview

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[10]. 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 [20]. 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 [20, 43, 44, 48, 54–56]. 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 [44, 98, 113] 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 [56] 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 Chapter, 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.

5.2 Methodology - T2KG

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 5.1, 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

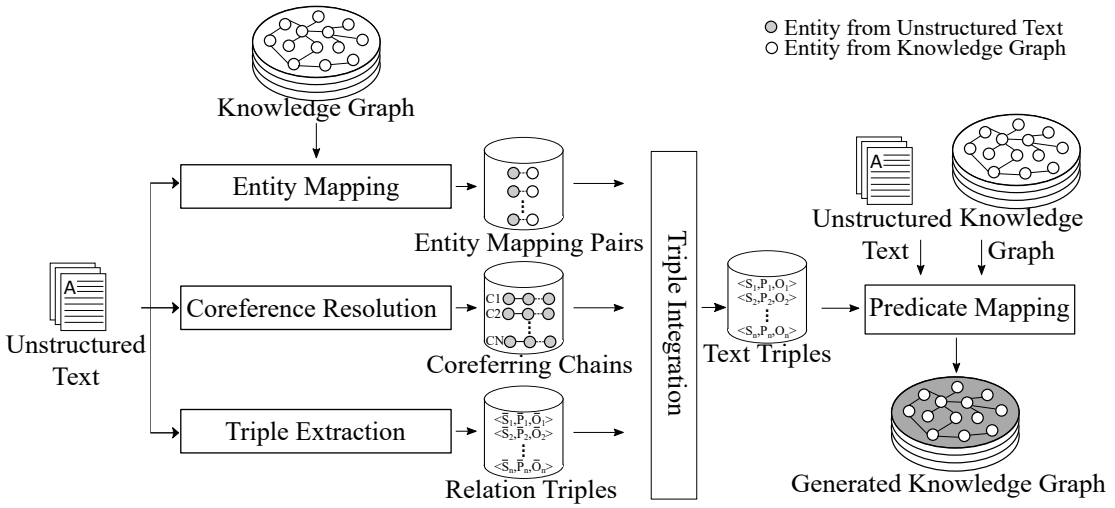


FIGURE 5.1: Architecture of the T2KG system

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.

5.2.1 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 5.2. In Figure 5.2, 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., $Barack\ Obama = \{dbpedia\colon Barack\_Obama\}$.

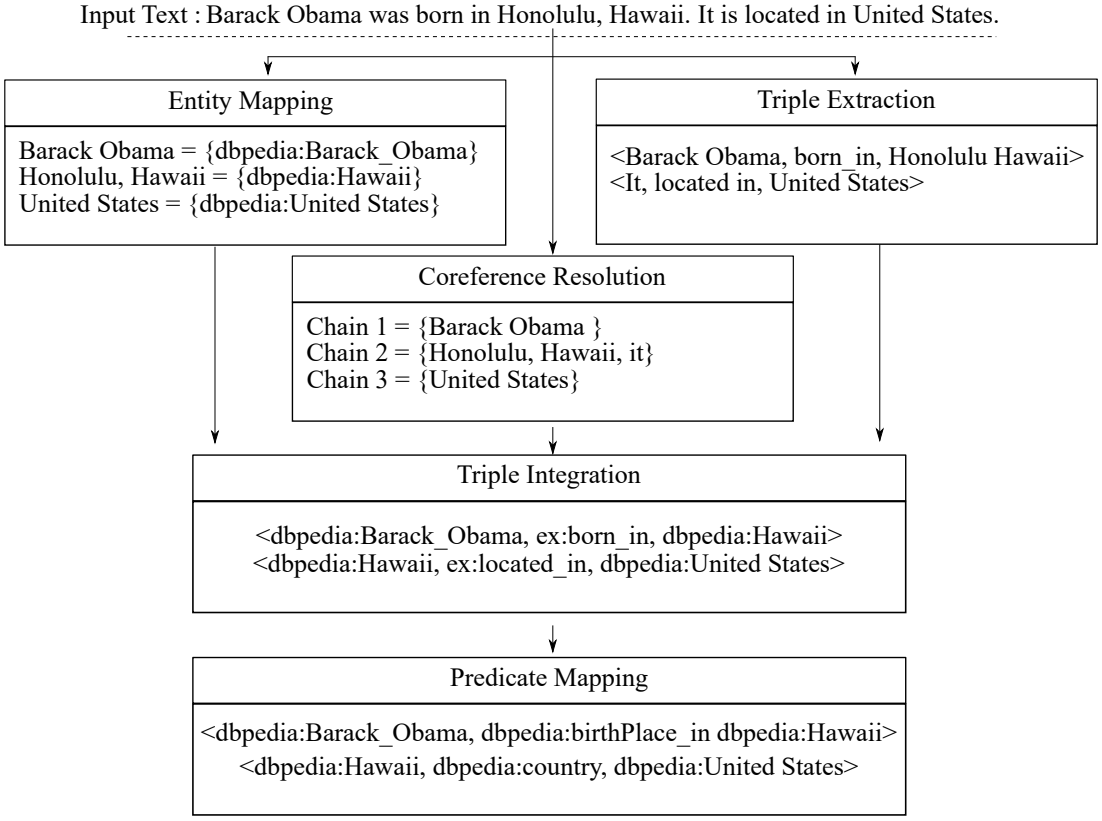


FIGURE 5.2: Example of the data flow in the T2KG system

5.2.2 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 Resolution 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 5.2. The expected results of the Coreference Resolution component are coreferring chains of entities. Based on the input in the example, the coreferring chain is $C2 = \{Honolulu\ Hawaii,\ it\}$.

5.2.3 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 [114], 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 5.2, 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.

5.2.4 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 5.2 shows our 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 still not mapped to any predicate in the KG.

5.2.5 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 [56], 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 [56], 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 5.3. 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 5.2. As shown in the figure, $ex: born\_in$ is mapped to $dbpedia: birthPlace$, and the triple $< dbpedia: Barack\_Obama, dbpedia: birthPlace, dbpedia: Hawaii >$ is created as a triple for the generated KG. The details of each module are as follows.

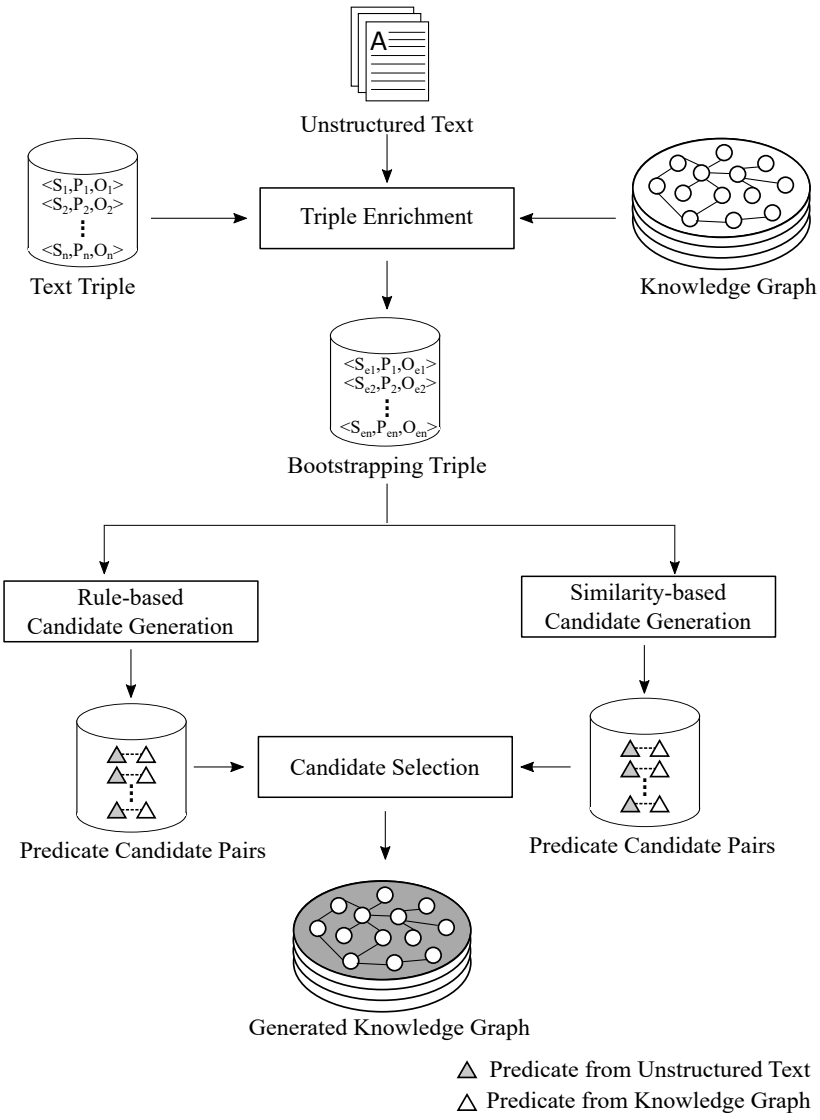


FIGURE 5.3: Diagram of the Predicate Mapping component

5.2.5.1 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 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, $dbpedia: Person$.

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 $< dbpedia: Barack\_Obama, ex: born\_in, dbpedia: Hawaii >$, the result is $< URI,\ ex: born\_in,\ URI >$. All generated triples, called bootstrapping triples, are used as the output of this module.

5.2.5.2 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 [56] 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 $< Person,\ ex: born\_in,\ Location >$ 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.

5.2.5.3 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 [21, 87]. 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 [64]. Inspired by vector representations of words [64], we present elements of triples in the vector space by using other elements in the same triple. The objective function is formulated as follows.

$$L(\theta) = \arg\max_{\theta} \sum_{(e,c)\in T} \log \sigma(v_e \cdot v_c) + \sum_{(e,c)\in T'} \log \sigma(v_e \cdot v_c) \tag{5.1}$$

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.

$$Sim(P_{\hat{T}}, P_{KG}) = \delta(\frac{\bar{P}_{\hat{T}} \cdot \bar{P}_{KG}}{|\bar{PT}||\bar{P_{KG}}|}) + (1-\delta)(\frac{context(P_{KG}) \cdot (\bar{ST} - \bar{OT})}{|context(P_{KG})||\bar{ST} - \bar{OT}|}) \tag{5.2}$$

$$context(P_{KG}) = \frac{\sum_{n=1}^{N}(\bar{S}_{P_{KG_n}} - \bar{O}_{P_{KG_n}})}{N} \tag{5.3}$$

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. 5.2. However, the predicate might be varied by its context. Consequently, in the second term of Eq. 5.2, 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. 5.2 are different, the weight parameter is introduced for adjusting

the salient aspect between the predicate similarity and the context similarity. Eq. 5.3 is proposed to compute an average vector representation of the context of P_{KG}.

5.2.5.4 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, 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.

5.3 Experiment

5.3.1 Experimental Setup

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

In T2KG, each component is implemented and its parameters are configured as follows. In the Entity Mapping component, DBpedia Spotlight [98] 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 [94, 115] is used as a coreference resolver. In the Triple Extraction component, OLLIE [55], which is one of the state-of-the-art 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[1] is used for the training vector representations of the elements of triples. The parameters of word2vec are set by default. To collect the corpus for creating vector representations of the elements of triples, text triples and KG triples are prepared. To gather text triples, 120,000 Wikipedia articles are randomly

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

selected and then the pre-processing step is applied. In the pre-processing step, HTML markups, wiki marks and any hyperlink annotations are removed. Duplicated sentences are reduced to one sentence. After that the pre-processed sentences are passed to the T2KG framework to create text triples. For KG triples, the whole DBpedia [10] is used.

In the Similarity-based Candidate Generation module, there are two hyperparameters: the weight δ and the threshold θ. To fine-tune these two parameters, we automatically create the identical predicate pairs as the training data by the matching strategy between predicates of text triples and predicates of KG. The matching strategy for constructing the training data is done 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 4.3 demonstrates an example for matching between a predicate of a text triple and a predicate of a KG triple for creating the training data. 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 training data construction, the targets of the mapping predicate are 2,800 DBpedia ontology properties. Although this method can help to establish a lot of matching pairs, it is possible that matching pairs 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 text triples when a predicate has multiple matches. According to the matching strategy, the number of remaining mapped predicate pairs is approximately 43,800. After that, we use the grid search algorithm to find the suitable δ and θ parameters on the training dataset. The interval of the parameter searching was [0.00, 1.00], and the step interval was 0.01. The hyperparameters that performed best in the training data were used as the setting.

5.3.2 Experiment 1

The aim of this experiment is to evaluate the performance of our hybrid approach for the predicate mapping task. To investigate the contribution of our approach, the rule-based approach in the study [56] is used as the baseline for comparison.

In the experiment, we manually create the benchmark dataset. To construct the benchmark dataset, we randomly selected 300 text triples and then asked an expert to create

TABLE 5.1: The results of our approach in the predicate mapping task on the benchmark dataset comparing with the baseline

Approach	Macro			Micro		
	Precision	Recall	F1	Precision	Recall	F1
Baseline	**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**

the links between the text predicate and the DBpedia predicate. The dataset is available for download [2].

In this experiment, given a predicate text, an approach returns the DBpedia predicate having the highest similarity. The micro/macro precision, recall and F1 score then use to measure whether the DBpedia predicate and the predicate text are correctly match 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 5.1 shows the results of the rule-based approach compared with our hybrid approach. The experimental results indicate that the hybrid approach can improve recall by 10.60%, 9.67% and F-measure by 4.72%, 4.41% in the macro/micro evaluation respectively. Because the recall 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.

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

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

TABLE 5.2: The performance of T2KG framework for constructing KG comparing with the baseline

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

upon the above example, the text predicate "occur in" can be mapped to dbpedia:place[3]. The hybrid approach therefore can alleviate the problems caused by the limitations of pattern-matching in the rule-based approach.

5.3.3 Experiment 2

The aim of this experiment is to evaluate performance of the T2KG framework in the KG creation from text 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 manual establishing a small set of the gold standard of triple extracted from text. To create the gold standard, we randomly select 100 sentences from Wikipedia articles and then manually extract and map triples to DBpedia triples. Note that to fairly evaluate the performance of T2KG, 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 in order to show the mappingability of the framework. In this experiment, T2KG, which does not install the Similarity-based Candidate Generation module, are used as the baseline. This baseline is mostly the same as the study [56].

The results are listed in Table 5.2. As shown in the result, a number of discovered knowledge, which can integrate to the existing KG, by T2KG is surpassed the baseline. Furthermore, the precision, recall and F1 score of T2KG also outperform the baseline. Note that, the results of generated triples in this experiment are listed in Table A.1 for the reference.

[3]http://dbpedia.org/ontology/place

To 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 as shown in Table 5.3. 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 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, we deeply analyze the types of the error caused by each component. There are two types of errors: 1) Error Type I and 2) Error Type II. Error Type I occurs when the results are mismatch with the standard, while Error Type II is caused by the missing result when comparing with the standard. We also break down the analysis of these errors for each component as listed in Table 5.3. The results show that the majority of the error is Error Type I. To further discuss about these results, we recap the knowledge construction workflow, which consists of two procedures: the knowledge extraction and knowledge integration. The big picture of the KG construction is depicted in Figure 1.3. In the work flow, knowledge is extracted from natural language text and then integrates to existing KGs. In T2KG, the knowledge extraction consists of two components: Coreference Resolution and Triple Extraction, while the knowledge integration also composes two components: Entity Mapping and Predicate Mapping. As shown in Table 5.3,The knowledge extraction tends to produce Error Type II. The knowledge extract encountered the missing knowledge problem, where the existing knowledge in natural language text could not be extracted due to the complex structure of language. For example, given the sentence "KSTP-TV is a television station located in Saint Paul", the relation ⟨ KSTP-TV, located in, Saint Paul ⟩ could not be discovered. Nevertheless, unlike the knowledge extraction, it turns out that the knowledge integration provides Error Type II. Error Type II, also known as mismatch error, is caused by the heterogeneous problem. As a result, it is difficult to select the correct entity or relation. For example, given the sentence "Ann Noreen has been a novelist since 2000", the entity linking fails to map the entity to `dbr:Ann_Widdecombe`. This analysis confirms our assumption on KG construction that the difficulty in knowledge construction is the language complexity while the difficulty in knowledge integration is the heterogeneous problem.

In addition, errors in the elements of generated triples are inspected. We find that the largest number of errors is 38.18% caused by predicates, 36.97% by objects and 24.85% 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

Table 5.3: The analysis of errors on constructing KG

Component	All Error proportion	Error Type I	Error Type II
Coreference Resolution	21.60%	45.90%	54.10%
Triple Extraction	35.21%	48.45%	51.55%
Entity Mapping	20.19%	61.02%	38.98%
Predicate Mapping	23.00%	65.75%	34.25%
All	100.00%	59.86%	40.14%

perfectly extract predicates from natural language 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.

5.3.4 Experiment 3

The aim of this experiment is to empirically investigate the performance of the T2KG framework for populating new knowledge from text to the existing KG. In the experiment, two dataset, gold standard and online article, are used. The gold standard dataset is 100 sentences, which randomly selected from Wikipedia. This dataset is the same as the dataset in Experiment 2. The online article dataset is the dataset,in which articles on the Internet are gathered. To create this dataset, we randomly crawl articles on the website in various domains including, news, movie, book and travel. Both datasets then are passed to T2KG to create the KG. Since this experiment investigate the knowledge population from text to the existing KG, the existing KG is used as scope for populating knowledge to the KG. DBpedia is set as the existing KG. Consequently, only triples, whose subject and predicate can be mapped to DBpedia, (mapped triples) are considered. After acquiring mapped triples, the correctness of such triples is manually checked to study the quality of the populated triples and the number of new knowledge is measured to represent the quantity of the new knowledge. The new knowledge is that the knowledge does not contain in DBpedia and the T2KG framework could populate it from text. To evaluate the results, we therefore report the statistic of the dataset, the number of extracted triples from text, the number of extracted triple, which could mapped to DBpedia, the number of the correct mapped triples and the number of the new knowledge that we could populate to DBpedia. Note that, DBpedia using in this experiment is DBpedia 3.9 (2016-04).

Table 5.4: 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

The experimental results are presented in the Table 5.4. The result shows that T2KG can populate the new knowledge to DBpedia. Although some knowledge is existing in the DBpedia, there are still some knowledge, which is still not discovered yet and could be discover in the text. Furthermore, based on our observation, the result on the online dataset contains more unmapped triples than the gold standard dataset. This characteristic occurs because many entities on the online dataset could not be mapped to DBpedia. For example, some entities from the movie, which is published after the releases of the DBpedia version using in this experiment. As a result, many triples and new knowledge are discarded.

Based on our observation, we found that the knowledge that we can populate by T2KG is the knowledge that explicitly stated on the text. For example, given a sentence "Melania gave birth to their son Barron", we could directly extract the knowledge ⟨ dbr:Melania_Trump, dbo:child, dbr:Barron_Trump ⟩; however, we could not extract knowledge ⟨ dbr:Donald_Trump, dbo:child, dbr:Barron_Trump ⟩ from this example.

To finer analyze the populated knowledge we investigate what kind of acquired knowledge by considering the category of the populated knowledge. In this experiment, the target of the KG to populate the knowledge is DBpedia. Based on our observation on DBpedia, we found that the majority of the knowledge is related to people, places and organizations. Therefore, we defined the four categories: 1) Person, 2) Place, 3) Organization and 4) Misc. These categories are used to evaluate the proportion of discovered knowledge. To evaluate the result, we manually measure the knowledge in each category. We found that for the gold standard dataset 47.06%, 35.29%, 14.71% and 2.94% of discovered knowledge are categorized into the person, place, organization and misc respectively. In the online dataset, 33.78%, 47.30%, 9.46% and 9.46% of new knowledge are person, place, organization and misc respectively. In here, we found that most entities in DBpedia are person and place. This is the reason most of the knowledge populated from DBpedia is related to person and place. Note that, the results of discovered knowledge in this experiment are listed in Table B.1 and Table B.2 as triples for the reference.

5.4 Summary

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, an automatic knowledge graph creation from natural language text. In T2KG, 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 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 achieves approximately 50% in both quality and quantity of triples generated for creating the KG. In addition, the empirical study on the knowledge population from text is investigated. The experimental results indicate that T2KG can successfully populate new knowledge from text to DBpedia. Based on error analyses, the main pitfall of the framework is the Triple Extraction component causing by an open information extraction system. Fortunately, the field of the open information extraction is still active. In the future, we aim to improve the implementation of T2KG by using the later state of the open information extraction system.

Chapter 6

Discussion

In this Chapter (Chapter 6), the achievements, discussion and some limitations of the entity linking task, the predicate linking task and the Knowledge Graph population task were discussed respectively. Firstly, HMiLDs, which uses to tackle mapping entities toward multiple KG resources is discussed in Section 6.1. Secondly, the discussion of HRSim for the predicate linking task was presented in Section 6.2. Finally, T2KG for the Knowledge Graph population was discussed in Section 6.3.

6.1 Entity Linking - HMiLDs

HMILDs is the framework for linking an entity to multiple Knowledge Graph resources. The basic idea of HMILDs is to directly map entities to one particular Knowledge Graph and then gradually expand a search space for discovering identical entities to other Knowledge Graph resources in order to find other identical entities. Due to a large amount of entities in Knowledge Graph resources, an expansion strategy and a heuristic function for limiting the expanding search space are designed into the framework. In experiments, the framework could successfully map entities to the Knowledge Graph by increasing the coverage up to 90%. Experimental results also indicate that the heuristic function in the framework could efficiently limit the expansion space to a reasonable space. Based upon the limited expansion space, the framework could effectively reduce the number of candidate pairs without affecting any performances.

There are many approaches [68–72, 91] proposed for mapping entities to a data set. In Silk [65], three steps are introduced in order to discover and manipulate the matching between different data sets. For the first step, a discovery engine computes identical links between different data sets. Then, the second step fine-tunes the correctness of such links. The third step manipulates the links when changing of data sets is applied. AgreementMaker[66] is a resolution system for matching both ontologies and entities. In AgreementMaker, three phases are performed in order to match between entities. Firstly, candidate pairs of entities are selected by similarity between labels of entities in the candidate generation phase. Secondly, similarity between entity pairs are extracted during the disambiguation phase. Finally, entity pairs are verified, whether they are correct match or not, in the matching phase. In Zhishi.Links[67], which is an enhanced version of Silk, some weighting schema is applied to improve the matching results between entity pairs. ObjectCoref [68] is a self-learning system, which detects identical objects by iteratively learning discriminative property. SERIMI [69] selectes high entropy predicates, which usually possess abilities to discriminate identical objects, in order to select entity pairs and then build a binary classifier to classify whether such entity pairs are correctly matched or not. SLINT [70] and SLINT+ [71] select useful predicates for generating candidate pairs of entities and then such candidate pairs are verified whether they are identical or not. Rong et al. later introduce an entity matching approach using similarity metrics [72]. Several types of the similarity metric are proposed to extract similarity features between candidate entities and then a binary classifier is employed to justify whether candidate pairs are matched. Although those approaches success to identify identical entities, such approaches could not perform mapping entities to multiple data sets.

Owing to a large amount of LOD data sets, we could not know which data set contains an identical entity. As a result, an entity matching system, which map entities to a data set, cannot effectively map entities to the LOD cloud. Kertkeidkachorn et al. therefore introduce an automatic entity expansion framework for mapping entities to the LOD [21]. The framework discovers and maps entities to the LOD cloud by gradually expanding the data set from one data set to another data set. Although their work successfully increases coverage of mapping entities to the LOD cloud, the search space during the expansion process from one data set to another data set is still high. In this article, the major contribution differentiating from the work [21] is a heuristic function for the automatic entity expansion framework in order to limit the expanded search space in the LOD cloud to a reasonable range. Moreover,we also give rigid evidences why mapping entities to multiple data sets are necessary for the LOD cloud and also provide further empirical study of similarity metrics in the entity matching component.

Although we solve the problem on the entity linking to the LOD cloud, there still remaining problems on this approach. Since we utilize the links over KGs, the SPARQL query engine plays a key role to get our candidates. As a result, the most of time-consuming on HMiLDs is SPARQL endpoint. However, this problem is still active. There are some studies to propose the distributed query over KGs such as the studies [116]. In future, we could use a new method to reduce the cost during generate candidates

6.2 Predicate Linking - HRSim

HRSim is the framework for mapping predicate to Knowledge Graph resource. The idea of HRSim is to combine a rule-based approach and a similarity-based approach as the hybrid system for mapping predicate. The rule-based approach is a highly accurate approach but required massive of training data to construct rules. In contrast, the similarity-based does not required much data but the performance of the similarity-based might not be good enough when textual string is significantly different. We therefore propose the hybrid combination between a rule-based approach and a similarity-based approach to gain advantages of each approach. Also, the similarity-based approach greatly relies on similarity measure. In the similarity-based approach, we therefore propose a novel vector-based similarity measure. The experimental result show that our novel vector-based similarity measure outperformance other standard similarity measures. Furthermore, HRSim could perform better than the rule-based approach in the predicate mapping task.

Predicate linking, also known as ontology integration and synonym identification, is to map a predicate to its identical predicate. Most of studies [73–75] focus on predicate

linking between KG triples. Abedjan et al. [73] proposed the association rule mining to learn associated patterns in KGs and applied the patterns to discover identical predicate pairs. Zhao et al. [74] introduced the statistical graph patterns to group candidate predicates and used the string-based similarity approach to verify whether such candidates are identical. Then, identical predicates are used to build the mid-ontology between KGs. Zhang et al. [75] proposed using statistical knowledge patterns to identify identical predicates in the KG. Based upon the results, their approaches [73–75] could identify identical predicates between KGs. Nevertheless, it cannot be applied in a straightforward manner for our task, since we wish to form a link between a predicate in the text triple and its matching counterpart in a KG triple, and there is insufficient information to determine the statistical patterns due to the ambiguity and the sparsity of elements in text triples. The ambiguity of an element in a text triple is the major problem, in which each element of a text triple could convey many meanings because it does not represented by a uniform resource identifier (URI). The sparsity of elements in text triples is caused by the nature of natural language text because some predicates in text triples might appear rarely and the variation of predicates in text triples is higher than predicate in KG triples. Based on these problems, properties for building the statistical knowledge pattern becomes insufficient.

The most applicable study for the predicate linking task in our study is Exner' s study [56]. Exner et al. uses the state-of-the-art natural language processing (NLP) tool to link entities of text triples to KG entities and determines the statistical pattern of the text predicate and the KG predicate based on each subject-object pair, and then forms a link between identical predicates. Although this approach [56] could avoid the ambiguity of an element in a text triple, it has the following severe limitation: due to the sparsity of text, the training data for generating statistical patterns might not cover all possible patterns; consequently, some statistical patterns are missing.

To overcome this limitation, we aim to leverage the similarity-based approach for linking between text predicates and KG predicates. Generally, the similarity-based approaches, the string-based similarity and the wordnet-based similarity, are used in the ontology matching [117]. Although such approach give reasonable results, they fail to identify identical predicates when surface form of string is sufficiently different due to the use of different vocabularies. For example, considering the triple (Lionel Messi, play for, FC Barcelona), "play for" should be linked to dbo: team. However, both the string-based similarity and the wordnet-based similarity fail to identify the predicates as identical.

Since the surface form does not adequately represent the vocabulary, we need to learn and represent the vocabulary in the deeper level than just its surface form. Based upon a review, we found that Mikolov et al. [64] proposed a distributed representation of words

that can capture both the syntactic and the semantic patterns. *Distributed Representation* is to embed the target word into the dense vector in the low-dimensional vector space as its representation. Inspired by distributed representations of words [64], we introduced a distribution representation of an element in a triple in the continuous vector space. The distributed representation of each of the elements in a triple is learned by the other elements in the same triple. The similarity between the distributions of the elements in a triple can then be leveraged to identify identical predicates.

Based on the experimental results, HRSim can linking predicate to the homogenous ontology efficiently; however, the drawback of the *Distributed Representation* is that we need certain amount of samples to embed the elements of triples. Generally, the major workaround for deal with the new elements, which have never been encountered before, is to re-train the whole elements again. Nevertheless, it requires amount of time to train representations. We therefore proposed the way to estimate the distributed representation from the compound components. The experimental results show that our estimation method outperformed the traditional method for estimating the distributed representations of compound words. In this dissertation, we did not directly conduct the experiment to integrate our estimation method in HRSim because our main goal is to evaluate HRSim without the limitation of absent representations. Also, the absent representations enable gaps between the actual representations and the estimated representations. As a result, we could not accurately measure the performance of HRSim because it could accumulate errors across methods. Nevertheless, our estimation method had been adopted to the study [118] in order to identify the cluster of identical predicates. Their results show that estimating representation of compound words can successfully help to cluster identical predicates into the same group [118]. This means that our estimation method is still applicable to the predicate linking task.

6.3 Knowledge Graph Population - T2KG

T2KG: an end-to-end system for constructing/populating a KG from unstructured text is introduced. Based on the reviews, KG construction/population can be separated into three categories, manual, semi- automatic and automatic. We present the comparison among systems proposed so far as well as T2KG in Table 6.1. Also, the compatibility of the approach for the Linked Data standard is listed in the same table. As shown in Table 6.1, Linked Data compatibility identify whether an approach construct or populate knowledge by using the Linked Data standard.

TABLE 6.1: Summary of Category of Knowledge Graph Construction/Population

Approach	KG Constriction/Population			Linked Data Standard
	Manual	Semi-Automatic	Automatic	
Cyc/OpenCyc [28]	✓			
WordNet [30]	✓			
UMLS [32]	✓			
Freebase [11]	✓	✓		✓
Wikidata [36]		✓		
DBpedia [10]		✓		✓
YAGO [12]		✓		✓
NELL [43]			✓	
LODifier [44]			✓	✓
KnowledgeStore [48]			✓	✓
Kriz et al. [20]			✓	✓
Knowledge Vault [50]			✓	✓
TextRunner [53]			✓	
Reverb [54]			✓	
OLLIE [55]			✓	
Exner et al. [56]			✓	✓
T2KG			✓	✓

As shown in Figure 1.3, KG construction/population generally considers the following two tasks: 1) knowledge extraction and 2) knowledge integration. In T2KG, we therefore discuss two main tasks, including the knowledge extraction and knowledge integration.

Based on our observation in the previous studies so far, there are two primary approaches in the knowledge extraction task: 1) schema-based approach and 2) schemaless-based approach, while there are also two approaches for dealing with the knowledge integration task: 1) pattern-based approach and 2) similarity-based approach. In the knowledge extraction of T2KG, the open information, which relied on the schemaless-based approach, has been used to extracted the knowledge. However, since T2KG also considered the knowledge integration task, it is also applicable as the schema-based approach. In the knowledge integration of T2KG, the hybrid approach that combines a rule-based approach and a similarity-based approach for integrating knowledge across existing KGs is proposed. 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. To compare T2KG characteristic with other systems, the methodology in the knowledge extraction task and the knowledge integration task are listed as shown in 6.2. The knowledge extraction consider that the extracted knowledge required schema or does not require schema, while the knowledge integration concerns which methodologies, including rule-based method and a similarity-based method has been used.

TABLE 6.2: Summary of Methodology for Knowledge Graph Construction/Population

Approach	Knwoledge Extraction		Knowledge Integration	
	Schema	Schemaless	Pattern-based	Similarity-based
Cyc/OpenCyc [28]	✓			
WordNet [30]	✓			
UMLS [32]	✓			
Freebase [11]	✓			
Wikidata [36]	✓			
DBpedia [10]	✓			
YAGO [12]	✓			
NELL [43]	✓			
LODifier [44]	✓			
KnowledgeStore [48]	✓			
Kriz et al. [20]	✓			
Knowledge Vault [50]	✓			
TextRunner [53]		✓		
Reverb [54]		✓		
OLLIE [55]		✓		
Exner et al. [56]	✓	✓	✓	
T2KG	✓	✓	✓	✓

In addition, we investigate the error and future direction for T2KG. In T2KG, the new knowledge can be achieved from the natural language text as shown in the experimental results. 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.

Chapter 7

Conclusion

In this Chapter, we concluded our research and presented what we have accomplished so far in Section 7.1. Finally, future directions of the research were discussed in the last section 7.2.

7.1 Conclusion

Knowledge Graph plays an important role in many modern applications, e.g. question answering, browsing knowledge, structured search, and data visualization. Nevertheless, most of the publishing data is natural language text, which is not feasible for constructing Knowledge Graph. Constructing Knowledge Graph therefore becomes the prominent problem for the semantic web community. Recently, there are many studies proposed approaches to transform natural language text to Knowledge Graph in order to construct Knowledge Graph resources. Still, two major issues, entity mapping and predicate mapping, are not solved yet. Mapping entity and predicate to Knowledge Graph provide many advantages since they could reduce the heterogeneous problem and could increase search ability over Knowledge Graph. We therefore aim to propose T2KG: the framework for automatic generating Knowledge Graph from natural language text, where entity mapping and predicate mapping are taken into account. In the framework, two sub-frameworks, namely HMILDs and HRSim, are proposed for dealing with entity mapping and predicate mapping respectively. A Heuristic expansion framework for Mapping Instances to Knowledge Graph data sets (HMILDs) is the framework for mapping entity to multiple Knowledge Graph resources. A Hybrid combination of Rule-based approach and Similarity-based approach (HRSim) is the framework for mapping predicate to predicates in existing Knowledge Graph. The finding of these frameworks are as follows:

- **HMiLDs**

  The key idea of the framework is to gradually expand the search space from one data set to another data set in order to discover identical entities. In experiments, the framework could successfully map entities to the LOD data sets by increasing the coverage to 90.36%. Experimental results also indicate that the heuristic function in the framework could efficiently limit the expansion space to a reasonable space. Based upon the limited expansion space, the framework could effectively reduce the number of candidate pairs to 9.73% of the baseline without affecting any performances.

- **HRSim**

  The experimental results show that our distributed representation-based similarity metric outperforms other traditional similarity metrics. Also, leveraging distributed representation-based similarity metric could help to discover and identify identical KG predicates for text predicates. As a result, our approach could alleviate the problem caused by the limitation of statistical knowledge patterns due to the sparsity of text and improve the discoverability for the predicate linking task.

- **T2KG**

  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 achieves approximately 50% in both quality and quantity of triples generated for creating the KG. In addition, the empirical study on the knowledge population from text is investigated. The experimental results indicate that T2KG can successfully populate new knowledge from text to DBpedia.

7.2 Future Work

In this section, we reported the lessons learned and the future directions for Knowledge Graph integration, including entity linking and predicate linking, as well as the outlook for the Knowledge Graph population respectively.

- **Knowledge Graph Integration**

  In Knowledge Graph integration, we propose HMiLDs and HRSim for solving the specific issues, Entity Linking and Predicate Linking, in Knowledge Graph Resolution. Here, we present the possible direction for these issues respectively.

  - **Entity Linking**

    HMiLDs can find an identical entity in KGs, in particular the LOD cloud, which consists of multiple KGs. Although HMiLDs could discover candidate entities across the LOD cloud, some of such candidate entities could not be successfully paired with source entities. Due to the heterogeneous problem in LOD cloud, the entity matching component in the framework might not be able to correctly verify all candidate pairs generated from the expanded search space whether they are correct match or not. Thanks to the ontology matching community, the entity matching task is still active and has been achieved better and better results every year. Consequently, the future possible direction of HMiLDs is to integrate other powerful entity matching techniques into the entity matching component of the framework to efficiently and effectively deal with the heterogeneous problem.

  - **Predicate Linking**

    HRSim can resolve the problem on the predicate linking task, in which a schema-less predicate are mapped to a predefine predicate. However, HRSim

relies on the distributed representation method to compute similarity between predicate pairs. Due to the drawback of the distributed representation method, an unobserved predicate cannot apply this method to embed the predicate into the continuous vector space. To avoid such problem in this dissertation, we evaluate the predicate, which have been observed over the training data. Nonetheless, we also propose the RNN-based approach to estimate the distributed representation of words when they are not observed on the training data. In the future, we aim to apply the estimated representations to compute the similarity between predicates as the workaround for predicates, which have not been observed in the training phase.

- **Knowledge Graph Population**

  T2KG can successfully populate new knowledge from natural language text to a Knowledge Graph, specifically DBpedia. Nevertheless, in order to cope with the growth of knowledge in the big data era, we have to challenge the scalability problem as well as the evaluation standard. Currently,we investigated T2KG with the small data set so that the quality of knowledge can be accurately evaluated. In practical, new knowledge emerges as stream of the data. Gradually creating the gold standard is not feasible. Therefore, the main direction in the future of T2KG is to improve the scalability of the T2KG framework together with the evolution method.

  Furthermore, based on error analyses of T2KG, the main pitfall of the frame- work is the Triple Extraction component causing by an open information extraction system. Fortunately, the field of the open information extraction is still active. Another possible solution to improve the performance of T2KG is to use the later state of the art of the open information extraction system to avoid this pitfall.

Appendix A

Knowledge Graph Construction with T2KG

Table A.1: Knowledge Construction from Natural Language Text by T2KG

Subject	Predicate	Object
dbr[1]:Poland	dbo[2]:type	dbr:Square
dbr:Poland	ex[3]:be_the_71st_largest_country_in	dbr:World
dbr:Poland	ex:be_the_9th_largest_country_in	dbr:Europe
dbr:Poland	dbo:areaTotal	312,679 square kilometre
dbr:Mao_Zedong	dbo:birthDate	December 26 , 1893
dbr:Mao_Zedong	dbo:birthPlace	dbr:Shaoshan
dbr:Aix-les-Bains	dbo:country	dbr:France
dbr:William_Carew_Hazlitt	dbo:education	dbr:Home
ex:Joseph_Kilroy	dbo:ethnicity	dbr:Irish_people
ex:Joseph_Kilroy	dbo:occupation	dbr:Kinship
dbr:George_J._Mitchell	dbo:athletics	dbr:Basketball
dbr:George_J._Mitchell	dbo:sport	dbr:Team.
dbr:France	dbo:type	dbr:Sovereign_state

[1]dbr : http://dbpedia.org/resource/
[2]dbo : http://dbpedia.org/ontology/
[3]ex : http://ex.org/T2KGResource/

Subject	Predicate	Object
dbr:Francia	dbo:stylisticOrigin	dbr:Latin
dbr:France	dbo:type	dbr:Western_Europe
dbr:France	dbo:type	dbr:Overseas_region
dbr:France	dbo:type	dbr:Territories_of_the_United_States
dbr:Aladdin_(1992_Disney_film)	dbo:stylisticOrigin	dbr:Media_franchise
dbr:Oil_lamp	dbo:canonizedPlace	dbr:Cave
dbr:Magic_in_fiction	dbo:placeOfBurial	dbr:Cave
dbr:Democratic_Republic_of_the_Congo	dbo:leftTributary	dbr:Full_moon
dbr:Republic_of_the_Congo	dbo:location	dbr:Central_Africa
dbr:Democratic_Republic_of_the_Congo	dbo:brand	dbr:Papal_conclave,_August_1978
dbr:West_Africa	dbo:sourceConfluence	dbr:Central_Africa
dbr:Democratic_Republic_of_the_Congo	dbo:twinCountry	dbr:France
dbr:Vichy_France	dbo:governmentType	dbr:Free_French_Forces
dbr:Vichy_France	dbo:officialLanguage	dbr:Free_French_Forces
dbr:Gospel_of_John	dbo:child	dbr:East_Elmhurst,_Queens
dbr:Sandbach	dbo:type	dbr:Civil_parishes_in_England
dbr:Sandbach	dbo:type	dbr:Market_town
dbr:Sandbach	dbo:district	dbr:Unitary_authority
dbr:Cheshire_East	dbo:country	dbr:England
dbr:Royal_Scots_Army	dbo:capital	dbr:Worcester
dbr:Cheshire	dbo:memberOfParliament	dbr:England
dbr:Sandbach	dbo:country	dbr:Cheshire
dbr:Royal_Scots_Army	dbo:countryOrigin	dbr:Battle
dbr:Obsidian_Entertainment	dbo:keyPerson	dbr:Feargus_Urquhart
dbr:Obsidian_Entertainment	dbo:keyPerson	dbr:Chris_Avellone
dbr:Obsidian_Entertainment	dbo:keyPerson	dbr:Christopher_Parker

Subject	Predicate	Object
dbr:Obsidian_Entertainment	dbo:keyPerson	dbr:Darren_Monahan
dbr:Obsidian_Entertainment	dbo:keyPerson	dbr:Chris_Jones_(filmmaker)
dbr:Grande_Arme	dbo:stylisticOrigin	dbr:Lebanese_Armed_Forces
dbr:Film_score	dbo:usedInWar	dbr:Victory
dbr:Release_(music)	dbo:thirdDriverCountry	1976
dbr:Release_(music)	dbo:recordLabel	dbr:Mercury_Records
dbr:Benazir_Bhutto	dbo:largestSettlement	dbr:Ford_Pinto
dbr:Benazir_Bhutto	dbo:birthPlace	dbr:Karachi
dbr:Benazir_Bhutto	dbo:birthDate	21 June 1953
dbr:Benazir_Bhutto	dbo:parent	dbr:Zulfikar_Ali_Bhutto
dbr:Benazir_Bhutto	dbo:parent	dbr:Ali
dbr:Allies_of_World_War_II	dbo:garrison	dbr:Germany
dbr:Helmet	dbo:twinCountry	dbr:Germany
dbr:Victory	dbo:garrison	dbr:Germany
dbr:Yankton,_South_Dakota	dbo:type	dbr:City
dbr:Yankton,_South_Dakota	dbo:government	dbr:Defense_independent_pitching_statistics
dbr:Yankton,_South_Dakota	dbo:type	dbr:County_seat
dbr:Yankton,_South_Dakota	dbo:government	dbr:Source_code
dbr:South_Miami	dbo:largestSettlement	dbr:Miami_metropolitan_area
dbr:South_Miami_(Metrorail_station)	dbo:architectualBureau	dbr:Sunset_Drive
dbr:Miami	dbo:type	dbr:City
dbr:Miami	dbo:largestSettlement	dbr:Sunset_Drive
dbr:Miami	dbo:largestSettlement	dbr:Miami_metropolitan_area
dbr:U.S._state	dbo:leaderName	dbr:Cameron_ministry
dbr:U.S._state	dbo:campus	dbr:Local_government
dbr:U.S._state	dbo:governmentType	dbr:Conservatism
dbr:Norwich_Castle	dbo:foundingYear	the Norman Conquest

Subject	Predicate	Object
dbr:Norwich_Castle	dbo:usedInWar	dbr:Norman_conquest _of_England
dbr:U.S._state	dbo:lowestPosition	dbr:Department_for _Communities_and_Local _Government
dbr:Atchison,_Kansas	dbo:dissolutionYear	1996
dbr:Kansas	dbo:country	dbr:United_States
dbr:Marion_County,_Florida	dbo:type	dbr:Location_(geography)
dbr:Atchison,_Kansas	dbo:governmentPlace	dbr:BNSF_Railway
dbr:Topeka,_Kansas	dbo:governmentPlace	dbr:BNSF_Railway
dbr:Atchison,_Topeka _and_Santa_Fe_Railway	dbo:governmentPlace	dbr:BNSF_Railway
dbr:Topeka,_Kansas	dbo:riverMouth	dbr:Burlington_Northern _Railroad
dbr:Atchison,_Topeka _and_Santa_Fe_Railway	dbo:location	the current BNSF rail- way
dbr:Marion_County, _Florida	dbo:country	dbr:United_States
dbr:Atchison,_Topeka _and_Santa_Fe_Railway	dbo:foundingYear	1996
dbr:Topeka,_Kansas	dbo:dissolutionYear	1996
dbr:Local_government	dbo:riverBranchOf	dbr:Rail_transport
dbr:Atchison,_Topeka _and_Santa_Fe_Railway	dbo:orogeny	dbr:Burlington_Northern _Railroad
dbr:Local_government	dbo:musicFusionGenre	dbr:Atchison,_Topeka _and_Santa_Fe_Railway
dbr:Wales	dbo:country	dbr:Great_Britain
dbr:Wales	dbo:type	dbr:East
dbr:Wales	dbo:type	dbr:Irish_Sea
dbr:Wales	dbo:type	dbr:North_Wales
dbr:United_Kingdom	dbo:district	dbr:Irish_Sea
dbr:Wales	dbo:type	dbr:Longitude
dbr:Wales	dbo:type	dbr:Bristol_Channel
dbr:Great_Britain	dbo:district	dbr:Irish_Sea
dbr:Head_coach	dbo:torchBearer	dbr:Steve_Clifford
dbr:New_Orleans_Hornets	dbo:generalManager	dbr:Steve_Clifford

Subject	Predicate	Object
dbr:Richard_Stone_(composer)	dbo:occupation	dbr:Animator
dbr:Richard_Stone_(composer)	dbo:occupation	dbr:Actor
dbr:Richard_Stone_(composer)	dbo:occupation	dbr:Singing
dbr:Richard_Stone_(composer)	dbo:birthPlace	dbr:United_States
dbr:Richard_Stone_(composer)	dbo:occupation	dbr:Songwriter
dbr:Richard_Stone_(composer)	dbo:occupation	dbr:Film_producer
dbr:Richard_Stone_(composer)	dbo:occupation	dbr:Television_director
dbr:Richard_Stone_(composer)	dbo:occupation	dbr:Screenwriter
dbr:Donald_Smith,_1st_Baron_Strathcona_and_Mount_Royal	dbo:nationality	dbr:Canadians
dbr:Autechre	ex:consist_of	Rob Brown and Sean Booth
dbr:Cavity_Job	dbo:releaseDate	1991
dbr:Autechre	ex:be	a english electronic music duo
dbr:Joseph_Chamberlain	dbo:occupation	dbr:Politician
dbr:Joseph_Chamberlain	dbo:birthPlace	dbr:London
dbr:Joseph_Chamberlain	dbo:occupation	dbr:Imperialism
dbr:Joseph_Chamberlain	dbo:occupation	dbr:Radicalism_(historical)
dbr:Joseph_Chamberlain	dbo:birthPlace	dbr:United_Kingdom
dbr:Orange_County,_New_York	dbo:governmentType	dbr:Independent_(politician)
dbr:Orange_County,_New_York	dbo:foundingDate	November 1, 1683
dbr:Orange_County,_New_York	dbo:governmentType	dbr:Province_of_New_York
dbr:New_Orleans	dbo:type	dbr:City
dbr:New_Orleans	dbo:type	dbr:Port

Subject	Predicate	Object
dbr:Southeastern _Louisiana_University	dbo:place	straddle
dbr:New_Orleans	dbo:country	dbr:United_States
dbr:New_Orleans	dbo:type	dbr:List_of_metropolitan _areas_of_the_United _States
dbr:New_Orleans	dbo:type	dbr:Major
dbr:Bruce_Nauman	dbo:occupation	dbr:Artist
dbr:Bruce_Nauman	dbo:birthPlace	dbr:United_States
dbr:Nikita_Khrushchev	ex:be	dbr:Cold_War
dbr:Nikita_Khrushchev	dbo:birthPlace	dbr:Soviet_Union
dbr:Nikita_Khrushchev	dbo:occupation	dbr:Politician
dbr:Nikita_Khrushchev	dbo:birthPlace	dbr:Russia
dbr:Nikita_Khrushchev	dbo:birthDate	April 15 , 1894
dbr:Nikita_Khrushchev	dbo:birthPlace	dbr:Kalinovka, _Khomutovsky_District, _Kursk_Oblast
dbr:Woodrow_Wilson	ex:serve_as	the 28th president of the United States
dbr:Woodrow_Wilson	dbo:occupation	dbr:Academia
dbr:Woodrow_Wilson	dbo:occupation	dbr:Politics_of_the _United_States
dbr:Ulysses_S._Grant	dbo:almaMater	dbr:United_States _Military_Academy
dbr:Russia	ex:be	the largest country
dbr:Charles_Doolittle _Walcott	dbo:birthPlace	dbr:Saint_Lucia
dbr:Ann,_Burma	dbo:instrument	dbr:Novel
dbr:Ann,_Burma	dbo:subsequentInfras- tructure	dbr:Political_party
dbr:Ann,_Burma	dbo:instrument	dbr:Conservative_Party _(UK)
dbr:Ann,_Burma	dbo:winsInEurope	dbr:Conservative_Party _(UK)
dbr:Ann,_Burma	dbo:instrument	dbr:Political_party
dbr:Ann,_Burma	dbo:winsAtAus	dbr:Novel
dbr:Ann,_Burma	dbo:instrument	dbr:Politician

Subject	Predicate	Object
dbr:Ann,_Burma	dbo:jointCommunity	dbr:Politician
dbr:Rohtak_district	dbo:stylisticOrigin	dbr:Types_of_inhabited _localities_in_Russi
dbr:Florida	ex:be_run_from	the Sarasota-Bradenton-Venice
dbr:Akon	dbo:successor	dbr:Louis_the_Pious
dbr:Akon	dbo:birthPlace	dbr:Saint
dbr:Mike_Rann	dbo:birthPlace	dbr:Sidcup
dbr:David	dbo:birthPlace	dbr:Sidcup
dbr:Little_River_Band	dbo:city	dbr:Bloomington, _Minnesota
ex:The_mall_of_America	dbo:place	dbr:Bloomington, _Minnesota
dbr:Rumson,_New_Jersey	dbo:country	dbr:Borough_(New_Jersey)
dbr:Peekskill,_New_York	dbo:country	dbr:New_York _metropolitan_area
dbr:Population	ex:grow_from	3,304
dbr:Population	ex:grow_in	1871
dbr:Population	ex:grow_to	5,501
dbr:Stanley,_Falkland _Islands	dbo:type	dbr:Foremost_Formation
dbr:L._Ron_Hubbard	ex:be	one of broadcasting 's foremost pioneer
dbr:L._Ron_Hubbard	dbo:occupation	dbr:Foremost_Formation
dbr:L._Ron_Hubbard	ex:be_founder_of	dbr:KSTP_(AM)
dbr:KSTP-TV	ex:be	dbr:Television_station

Appendix B

Knowledge Discovery with T2KG

TABLE B.1: New Knowledge Discovery from Natural Language Text by T2KG (The Gold Standard Dataset)

Subject	Predicate	Object
dbr[1]:William_Carew_Hazlitt	dbo[2]:education	dbr:Home
dbr:George_J._Mitchell	dbo:athletics	dbr:Basketball
dbr:France	dbo:type	dbr:Sovereign_state
dbr:Republic_of_the_Congo	dbo:location	dbr:Central_Africa
dbr:Vichy_France	dbo:governmentType	dbr:Free_French_Forces
dbr:Sandbach	dbo:type	dbr:Civil_parishes_in_England
dbr:Sandbach	dbo:type	dbr:Market_town
dbr:Cheshire_East	dbo:country	dbr:England
dbr:Obsidian_Entertainment	dbo:keyPerson	dbr:Feargus_Urquhart
dbr:Obsidian_Entertainment	dbo:keyPerson	dbr:Chris_Avellone
dbr:Obsidian_Entertainment	dbo:keyPerson	dbr:Christopher_Parker

[1]dbr : http://dbpedia.org/resource/
[2]dbo : http://dbpedia.org/ontology/

Subject	Predicate	Object
dbr:Obsidian _Entertainment	dbo:keyPerson	dbr:Darren_Monahan
dbr:Obsidian _Entertainment	dbo:keyPerson	dbr:Chris_Jones _(filmmaker)
dbr:Benazir_Bhutto	dbo:parent	dbr:Zulfikar_Ali_Bhutto
dbr:Norwich_Castle	dbo:usedInWar	dbr:Norman_conquest_of _England
dbr:Atchison,_Kansas	dbo:dissolutionYear	1996
dbr:Atchison,_Topeka _and_Santa_Fe_Railway	dbo:foundingYear	1996
dbr:Topeka,_Kansas	dbo:dissolutionYear	1996
dbr:New_Orleans_Hornets	dbo:generalManager	dbr:Steve_Clifford
dbr:Richard_Stone _(composer)	dbo:occupation	dbr:Animator
dbr:Richard_Stone _(composer)	dbo:occupation	dbr:Actor
dbr:Richard_Stone _(composer)	dbo:occupation	dbr:Singing
dbr:Richard_Stone _(composer)	dbo:occupation	dbr:Songwriter
dbr:Richard_Stone _(composer)	dbo:occupation	dbr:Film_producer
dbr:Richard_Stone _(composer)	dbo:occupation	dbr:Television_director
dbr:Richard_Stone _(composer)	dbo:occupation	dbr:Screenwriter
dbr:Donald_Smith, _1st_Baron_Strathcona_and _Mount_Royal	dbo:nationality	dbr:Canadians
dbr:Cavity_Job	dbo:releaseDate	1991
dbr:Joseph_Chamberlain	dbo:occupation	dbr:Politician
dbr:Orange_County,_New _York	dbo:foundingDate	November 1, 1683
dbr:Bruce_Nauman	dbo:occupation	dbr:Artist
dbr:Nikita_Khrushchev	dbo:occupation	dbr:Politician
dbr:Woodrow_Wilson	dbo:occupation	dbr:Academia
dbr:Woodrow_Wilson	dbo:occupation	dbr:Politics_of_the _United_States

Table B.2: New Knowledge Discovery from Natural Language Text by T2KG (The Online Article Dataset)

Subject	Predicate	Object
dbr[3]:Julian_Assange	dbo[4]:creator	dbr:WikiLeaks
dbr:Jimmy_Carter	dbo:nationality	dbr:Americans
dbr:Harry_Potter _(character)	dbo:education	dbr:Hogwarts
dbr:Moana_Carcasses _Kalosil	dbo:profession	dbr:Member_of_parliament
dbr:Moana_Hotel	dbo:region	dbr:Pacific_Ocean
dbr:Lin-Manuel_Miranda	dbo:creator	Genius Hamilton
dbr:C._S._Lewis	dbo:influenced	dbr:Jerry_Lewis _(California_politician)
dbr:Lewis_Carroll	dbo:influenced	dbr:Jerry_Lewis _(California_politician)
dbr:Bill_Elliott	dbo:successor	dbr:Eliot_Spitzer
dbr:Samantha_Ryan	dbo:employer	dbr:Feds
dbr:Vonnie_Von_Helmolt	dbo:author	dbr:Great_Depression
dbr:Doctor_Strange	dbo:starring	dbr:Benedict_Cumberbatch
dbr:Ben_Schnetzer	dbo:profession	dbr:Politician
dbr:Doug_Sweetland	dbo:employer	dbr:Pixar
dbr:Bridget_Jones	dbo:activeYearsStartDate	2001
dbr:R._J._Reynolds _Tobacco_Company	dbo:formationYear	a Hollywood darling of the 1950s
dbr:Debbie_Harry	dbo:nationality	dbr:Americans
dbr:Debbie_Reynolds	dbo:nationality	dbr:Americans
dbr:Castellnou_de_Bages	dbo:county	dbr:Valencian_Community
dbr:Castellnou_de_Seana	dbo:county	dbr:Valencian
dbr:Hwang_Kyo-ahn	dbo:occupation	dbr:Prime_minister
dbr:Florida	dbo:region	dbr:United_States
dbr:Viktor_Belenko	dbo:birthPlace	the foothills of the Caucasus mountains
dbr:Artificial _intelligence	dbo:formationYear	the buzzword of 2016
dbr:Artificial _intelligence	dbo:service	dbr:Data

[3]dbr : http://dbpedia.org/resource/
[4]dbo : http://dbpedia.org/ontology/

Subject	Predicate	Object
dbr:Andy_Powell	dbo:coachedTeam	dbr:Cybersecurity_strategy
dbr:Om_Puri	dbo:starring	Veteran Indian
dbr:Barack_Obama	dbo:activeYearsEndDate	2017
dbr:Donald_Trump	dbo:occupation	dbr:President_of_the_United_States
dbr:Julian_Assange	dbo:activeYearsStartYear	2006
dbr:WikiLeaks	dbo:instrument	dbr:Email
dbr:Brisbane_Lions	dbo:formationYear	1996
dbr:Zeus	dbo:child	dbr:Hercules
dbr:Purnululu_National_Park	dbo:location	dbr:Western_Australia
dbr:Purnululu_National_Park__Purnululu__1	dbo:location	dbr:Western_Australia
dbr:Kakadu_National_Park__akadu_National_Park__1	dbo:type	dbr:Kakadu_National_Park
dbr:Sultan_Ahmed_Mosque	dbo:architect	dbr:Syed_Sultan_Ahmed
dbr:Sultan_Ahmed_Mosque	dbo:openingDate	the early 17th century
dbr:Constantinople	dbo:foundingYear	537
dbr:Square_Enix	dbo:formationYear	2003
dbr:Wat_Phra_Kaew	dbo:foundingYear	1782
dbr:Phra_Pathommachedi	dbo:foundingYear	1870
dbr:UEFA_Champions_League	dbo:category	dbr:Association_football
dbr:UEFA_Europa_League	dbo:category	dbr:UEFA_Champions_League
dbr:Wat_Chaiwatthanaram	dbo:name	Wat Chaiwatthanaram
dbr:Taboga_Island	dbo:foundingYear	1515
dbr:Mount_Kilimanjaro	dbo:country	dbr:Tanzania
dbr:Kilimanjaro_International_Airport	dbo:country	dbr:Tanzania
dbr:Kilimanjaro_National_Park	dbo:country	dbr:Tanzania
dbr:Unguja_Park	dbo:country	dbr:Unguja
dbr:Obelisco_(Guatemala_City)	dbo:state	dbr:Buenos_Aires_Central_Business_District

Subject	Predicate	Object
dbr:Ishigaki_Airport	dbo:foundingDate	March 7, 2013
dbr:Ishikawa_Prefecture	dbo:dissolutionDate	the mid-nineteenth century
dbr:Kamakura	dbo:location	dbr:Pacific_Ocean
dbr:Kamakura	dbo:country	dbr:Japan
dbr:Kyoto	dbo:dateCompleted	than 1,000 years ago
dbr:Kyoto	dbo:officialLanguage	dbr:Japanese_language
dbr:Kyoto	dbo:ethnicGroup	dbr:Japanese_people
dbr:Kyoto	dbo:capital	dbr:Japan
dbr:Dale_Earnhardt,_Jr.	dbo:birthPlace	dbr:United_States
dbr:Yangon	dbo:officialLanguage	dbr:Burmese_language
dbr:Yangon_Region	dbo:officialLanguage	dbr:Burmese_language
dbr:Yangon_United_F.C.	dbo:country	dbr:Myanmar
dbr:Yangon_International_Airport	dbo:country	dbr:Myanmar
dbr:Yangon_International_Airport	dbo:officialLanguage	dbr:Burmese_language
dbr:Pilgrimage	dbo:stylisticOrigin	dbr:Stairs
dbr:Singapore_Flyer	dbo:country	dbr:Singapore
dbr:Clarke_Quay_MRT_Station	dbo:country	dbr:Singapore
dbr:Maasai_language	dbo:regionalLanguage	dbr:Mount_Kilimanjaro_International_Convention_Centre
dbr:Baoshan,_Yunnan	dbo:country	dbr:China
dbr:Emperor_of_Japan	dbo:owner	dbr:Ise_Grand_Shrine
dbr:Ise_Grand_Shrine	dbo:location	dbr:Ise,_Mie
dbr:Aschaffenburg	dbo:language	dbr:German_language
dbr:Aschaffenburg_(district)	dbo:language	dbr:German_language
dbr:Johnny_Hart_(footballer)	dbo:ccupation	dbr:Midfielder
dbr:Atsuta_Shrine	dbo:location	dbr:Atsuta-ku,_Nagoya

Bibliography

[1] Tim Berners-Lee. Information management: A proposal, 1989. URL https://www.w3.org/History/1989/proposal.html. [Online; accessed 9-September-2017].

[2] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In *Proceedings of the annual of SEMANTiCS conference (Posters and Demos)*, 2016.

[3] Amit Singhal. Introducing the knowledge graph: things, not strings, 2012. URL https://googleblog.blogspot.jp/2012/05/introducing-knowledge-graph-things-not.html. [Online; accessed 4-April-2017].

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Web and Information Systems Journal*, pages 205–227, 2009.

[5] Tim Berners-Lee. Linked data-design issues (2006), 2011. URL http://www.w3.org/DesignIssues/LinkedData.html. [Online; accessed 4-April-2017].

[6] Graham Cormode and Balachander Krishnamurthy. Key differences between web 1.0 and web 2.0. *First Monday*, 13(6), 2008.

[7] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006. ISSN 1541-1672. URL http://dx.doi.org/10.1109/MIS.2006.62.

[8] Abele Andrejs, McCrae John, P., Buitelaar Paul, Jentzsch Anja, and Cyganiak Richard. Linking open data cloud diagram (2017), 2017. URL http://lod-cloud.net/. [Online; accessed 4-April-2017].

[9] Ora Lassila, Ralph R Swick, et al. Resource Description Framework (RDF) model and syntax specification, 1998. URL https://www.w3.org/TR/1998/WD-rdf-syntax-19980720/. [Online; accessed 4-April-2017].

[10] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the*

6th International the Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-76297-3, 978-3-540-76297-3. URL http://dl.acm.org/citation.cfm?id=1785162.1785216.

[11] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, New York, NY, USA, 2008. ACM. URL http://doi.acm.org/10.1145/1376616.1376746.

[12] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence Journal*, 194:28–61, 2013.

[13] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, Jan 2016. doi: 10.1109/JPROC.2015.2483592.

[14] Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 851–861, 2015.

[15] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3): 59–79, 2010.

[16] Dominic Seyler, Mohamed Yahya, and Klaus Berberich. Generating quiz questions from knowledge graphs. In *Proceedings of the 24th International Conference on World Wide Web*, pages 113–114, New York, NY, USA, 2015. ACM. URL http://doi.acm.org/10.1145/2740908.2742722.

[17] Richard Qian. Understand your world with bing, 2013. URL https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/. [Online; accessed 4-April-2017].

[18] François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5):706–716, October 2008. ISSN 1532-0464. URL http://dx.doi.org/10.1016/j.jbi.2008.03.004.

[19] Alan Ruttenberg, Jonathan A Rees, Matthias Samwald, and M Scott Marshall. Life sciences on the semantic web: the neurocommons and beyond. *Briefings in bioinformatics*, page 4, 2009.

[20] Vincent Kríž, Barbora Hladká, Martin Nečaskỳ, and Tomáš Knap. Data Extraction Using NLP Techniques and Its Transformation to Linked Data. In *Proceedings of 13th Mexican International Conference on Artificial Intelligence*, pages 113–124. Springer, 2014.

[21] Natthawut Kertkeidkachorn, Ryutaro Ichise, Atiwong Suchato, and Proadpran Punyabukkana. An automatic instance expansion framework for mapping instances to linked data resources. In *Proceedings of Joint International Semantic Technology Conference*, pages 380–395. Springer, 2013.

[22] Natthawut Kertkeidkachorn and Ryutaro Ichise. A heuristic expansion framework for mapping instances to linked open data. *IEICE TRANSACTIONS on Information and Systems*, 99(7):1786–1795, 2016.

[23] Natthawut Kertkeidkachorn and Ryutaro Ichise. T2KG: An end-to-end system for creating knowledge graph from unstructured text. In *Proceedings of AAAI Workshop on Knowledge-based Techniques for Problem Solving and Reasoning*, pages 743–750. AAAI, 2017.

[24] Natthawut Kertkeidkachorn and Ryutaro Ichise. Leveraging distributed representations of elements in triples for predicate linking. In *Proceedings of International Conference on Hybrid Artificial Intelligence Systems*, pages 75–87, 2017.

[25] Natthawut Kertkeidkachorn and Ryutaro Ichise. Estimating distributed representations of compound words using recurrent neural networks. In *Proceedings of International Conference on Natural Language & Information Systems*, pages 235–246, 2017.

[26] Natthawut Kertkeidkachorn and Ryutaro Ichise. An automatic knowledge graph creation framework from natural language text. *IEICE TRANSACTIONS on Information and Systems*, 2018. [To appear].

[27] World Wide Web Consortium et al. Rdf 1.1 concepts and abstract syntax, 2014. URL https://www.w3.org/TR/rdf11-concepts/. [Online; accessed 4-April-2017].

[28] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, November 1995. ISSN 0001-0782. URL http://doi.acm.org/10.1145/219717.219745.

[29] Wikipedia. Cyc — wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/w/index.php?title=Cyc&oldid=773238151. [Online; accessed 4-April-2017].

[30] George Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[31] Jorge Morato, Miguel Angel Marzal, Juan Lloréns, and José Moreiro. Wordnet applications. In *Proceedings of Global WordNet Conference*, pages 20–23, 2004.

[32] Olivier Bodenreider. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research*, 32:D267–D270, 2004.

[33] Douglas Lenat. *Understanding Computers: Artificial Intelligence*. Amsterdam: Time-Life Books, 1986.

[34] Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. Towards an integrated crowdsourcing definition. *Journal of Information Science*, 38(2):189–200, April 2012. URL http://dx.doi.org/10.1177/0165551512437638.

[35] Freebase. Freebase api (deprecated), 2017. URL https://developers.google.com/freebase/. [Online; accessed 4-April-2017].

[36] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[37] Wikipedia. Barack obama — wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/w/index.php?title=Barack_Obama&oldid=773703612. [Online; accessed 5-April-2017].

[38] DBpedia. About: Barack obama, 2017. URL http://dbpedia.org/page/Barack_Obama. [Online; accessed 5-April-2017].

[39] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195, 2015.

[40] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, September 2009. ISSN 1570-8268. URL http://dx.doi.org/10.1016/j.websem.2009.07.002.

[41] GeoNames. Geonames, 2017. URL http://www.geonames.org/. [Online; accessed 5-April-2017].

[42] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems*, pages 2–9. ACM, 2001.

[43] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the 24 th AAAI Conference on Artificial Intelligence*, pages 1306–1313. AAAI Press, 2010. URL http://dl.acm.org/citation.cfm?id=2898607.2898816.

[44] Isabelle Augenstein, Sebastian Padó, and Sebastian Rudolph. Lodifier: Generating linked data from unstructured text. In *Proceedings of the 9th International Conference on the Semantic Web: Research and Applications*, pages 210–224, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-30283-1. URL http://dx.doi.org/10.1007/978-3-642-30284-8_21.

[45] Hans Kamp and Uwe Reyle. *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*, volume 42. Springer Science & Business Media, 2013.

[46] James R. Curran, Stephen Clark, and Johan Bos. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 33–36, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1557769.1557781.

[47] Princeton University. Wordnet 3.0 in rdf, 2017. URL http://semanticweb.cs.vu.nl/lod/wn30/. [Online; accessed 5-April-2017].

[48] Francesco Corcoglioniti, Marco Rospocher, Roldano Cattoni, Bernardo Magnini, and Luciano Serafini. The knowledgestore: a storage framework for interlinking unstructured and structured knowledge. *International Journal on Semantic Web and Information Systems*, 11(2):1–35, April 2015. ISSN 1552-6283. URL http://dx.doi.org/10.4018/IJSWIS.2015040101.

[49] Martin Nečaskỳ, Tomáš Knap, Jakub Klímek, Irena Holubová, and Barbora Vidova-Hladka. Linked open data for legislative domain-ontology and experimental data. In *Proceedings of International Conference on Business Information Systems*, pages 172–183. Springer, 2013.

[50] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 601–610, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. URL http://doi.acm.org/10.1145/2623330.2623623.

[51] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. Evaluating entity linking with wikipedia. *Artificial Intelligence Journal*, 194: 130–150, 2013. ISSN 0004-3702. URL http://dx.doi.org/10.1016/j.artint.2012.04.005.

[52] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. Association for Computational Linguistics, 2009.

[53] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 7, pages 2670–2676, 2007.

[54] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. ACL, 2011.

[55] Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. Open language learning for information extraction. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. ACL, 2012.

[56] Peter Exner and Pierre Nugues. Entity extraction: From unstructured text to dbpedia rdf triples. In *Proceedings of the Web of Linked Entities Workshop*, pages 58–69. CEUR-WS, 2012.

[57] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 413–422, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2035-1. URL http://doi.acm.org/10.1145/2488388.2488425.

[58] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6): 707–730, December 2015. ISSN 1066-8888. URL http://dx.doi.org/10.1007/s00778-015-0394-1.

[59] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of Advances in neural information processing systems*, pages 2787–2795, 2013.

[60] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119. AAAI Press, 2014. URL http://dl.acm.org/citation.cfm?id=2893873.2894046.

[61] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2181–2187. AAAI Press, 2015. URL http://dl.acm.org/citation.cfm?id=2886521.2886624.

[62] Han Xiao, Minlie Huang, and Xiaoyan Zhu. Transfig: A generative model for knowledge graph embedding. In *Proceedings of the 29th international conference on computational linguistics. Association for Computational Linguistics*, 2016.

[63] Alfred Horn. On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, 16(01):14–21, 1951.

[64] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in neural information processing systems*, pages 3111–3119, 2013.

[65] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the web of data. In *Proceedings of the 8th International Semantic Web Conference*, pages 650–665, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04929-3. URL http://dx.doi.org/10.1007/978-3-642-04930-9_41.

[66] Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, et al. Results of the ontology alignment evaluation initiative 2009. In *Proceedings of the 4th International Conference on Ontology Matching*, pages 73–126. CEUR-WS, 2009.

[67] Xing Niu, Shu Rong, Yunlong Zhang, and Haofen Wang. Zhishi. links results for oaei 2011. In *Proceedings of the 6th International Conference on Ontology Matching*, pages 220–227. CEUR-WS, 2011.

[68] Wei Hu, Jianfeng Chen, and Yuzhong Qu. A self-training approach for resolving object coreference on the semantic web. In *Proceedings of the 20th International Conference on World Wide Web*, pages 87–96, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0632-4. URL http://doi.acm.org/10.1145/1963405.1963421.

[69] Samur Araujo, Duc Tran, Arjen DeVries, Jan Hidders, and Daniel Schwabe. Serimi: Class-based disambiguation for effective instance matching over heterogeneous web data. In *Proceedings of International Workshop on the Web and Databases*, pages 25–30, 2012.

[70] Khai Nguyen, Ryutaro Ichise, and Bac Le. Slint: a schema-independent linked data interlinking system. In *Proceedings of the 7th International Conference on Ontology Matching*, pages 1–12. CEUR-WS, 2012.

[71] Khai Nguyen and Ryutaro Ichise. Slint+ results for oaei 2013 instance matching. In *Proceedings of the 8th International Conference on Ontology Matching*, pages 177–183. CEUR-WS, 2013.

[72] Shu Rong, Xing Niu, Evan Wei Xiang, Haofen Wang, Qiang Yang, and Yong Yu. A machine learning approach for instance matching based on similarity metrics. In *Proceedings of the 11th International Conference on The Semantic Web*, pages 460–475, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-35175-4. URL http://dx.doi.org/10.1007/978-3-642-35176-1_29.

[73] Ziawasch Abedjan and Felix Naumann. Synonym analysis for predicate expansion. In *Proceedings of Extended Semantic Web Conference*, pages 140–154. Springer, 2013.

[74] Lihua Zhao and Ryutaro Ichise. Ontology integration for linked data. *Journal on Data Semantics*, 3(4):237–254, 2014.

[75] Ziqi Zhang, Anna Lisa Gentile, Eva Blomqvist, Isabelle Augenstein, and Fabio Ciravegna. Statistical knowledge patterns: Identifying synonymous relations in large linked datasets. In *Proceedings of International Semantic Web Conference*, pages 703–719. Springer, 2013.

[76] Ryutaro Ichise. Machine learning approach for ontology mapping using multiple concept similarity measures. In *Seventh IEEE/ACIS International Conference on Computer and Information Science*, pages 340–346, May 2008. doi: 10.1109/ICIS.2008.51.

[77] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady Journal*, pages 707–710, 1966.

[78] William E Winkler. The state of record linkage and current research problems. *Statistical Research Division, US Census Bureau*, 1999.

[79] Matthew A Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[80] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics Journal*, 18(4):467–479, December 1992. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=176313.176316.

[81] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. URL http://dx.doi.org/10.3115/981732.981751.

[82] William W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 201–212, New York, NY, USA, 1998. ACM. ISBN 0-89791-995-5. URL http://doi.acm.org/10.1145/276304.276323.

[83] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155, 2003.

[84] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations*, 2013.

[85] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of International Conference on Machine Learning*, volume 14, pages 1188–1196, 2014.

[86] Corina Dima and Erhard Hinrichs. Automatic noun compound interpretation using deep neural networks and word embeddings. In *Proceedings of International Conference on Computational Semantics*, page 173, 2015.

[87] Daniel Gerber, Sebastian Hellmann, Lorenz Bühmann, Tommaso Soru, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. Real-time rdf extraction from unstructured data streams. In *Proceedings of the 12th International Semantic Web Conference*, pages 135–150, New York, NY, USA, 2013. Springer-Verlag

New York, Inc. ISBN 978-3-642-41334-6. URL http://dx.doi.org/10.1007/978-3-642-41335-3_9.

[88] Mark Vickers. *Ontology-based free-form query processing for the semantic web.* PhD thesis, Brigham Young University, 2006.

[89] Basil Ell, Denny Vrandečic, and Elena Simperl. Labels in the web of data. In *Proceedings of the 10th International Conference on the Semantic Web - Volume Part I*, pages 162–176, Berlin, Heidelberg, 2011. Springer-Verlag. URL http://dl.acm.org/citation.cfm?id=2063016.2063028.

[90] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[91] Juanzi Li, Jie Tang, Yi Li, and Qiong Luo. Rimom: A dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering*, 21(8):1218–1232, August 2009. ISSN 1041-4347. URL http://dx.doi.org/10.1109/TKDE.2008.202.

[92] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. ISBN 0321321367.

[93] José Luis Aguirre, Kai Eckert, Jérôme Euzenat, Alfio Ferrara, Willem Robert Van Hage, Laura Hollink, Christian Meilicke, Andriy Nikolov, Dominique Ritze, François Scharffe, et al. Results of the ontology alignment evaluation initiative 2012. In *Proceedings of the 7th International Conference on Ontology Matching-Volume 946*, pages 73–115. CEUR-WS, 2012.

[94] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL https://doi.org/10.3115/1219840.1219885.

[95] Mark Watson. Practical semantic web and linked data applications. *Raleight, NC USA: Mark Watson*, 2011.

[96] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Proceedings of KDD workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.

[97] Federico Caimi. *Ontology and instance matching for the linked open data cloud.* PhD thesis, Politecnico di Milano, 2012.

[98] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM, 2011.

[99] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[100] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *Proceedings of Association for Computational Linguistics*, pages 455–465, 2013.

[101] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 13, pages 746–751, 2013.

[102] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods on Natural Language Processing*, volume 14, pages 1532–1543, 2014.

[103] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of Empirical Methods on Natural Language Processing*, volume 1631, page 1642, 2013.

[104] Justin Garten, Kenji Sagae, Volkan Ustun, and Morteza Dehghani. Combining distributed vector representations for words. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 95–101, 2015.

[105] Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of Automated Knowledge Base Construction*, 2016.

[106] Jeffrey L Elman. Finding structure in time. *Cognitive science Journal*, pages 179–211, 1990.

[107] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, pages 157–166, 1994.

[108] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation Journal*, pages 1735–1780, 1997.

[109] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[110] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 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*, pages 28–34. ACL, 2011.

[111] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.

[112] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, pages 2579–2605, 2008.

[113] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 1375–1384, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2002472.2002642.

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

[115] Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 492–501. ACL, 2010.

[116] D Graux, L Jachiet, P Genevès, and N Layaïda. Sparqlgx: A distributed rdf store mapping sparql to spark. In *Proceedings of the 15th International Conference on Semantic Systems*, 2016.

[117] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[118] Lihua Zhao, Natthawut Kertkeidkachorn, and Ryutaro Ichise. Knowledge discovery from linked data. In *Proceedings of The 31st Annual Conference of the Japanese Society for Artificial Intelligence*, pages 1–2, 2017.

Index