Building a Knowledge Graph of Vietnam Tourism from Text

Phuc Do and Hung Le

Abstract Most data in the world is in form of text. Therefore, we can say text stores large amount of the knowledge of human beings. Extracting useful knowledge from text, however, is not a simple task. In this paper, we present a complete pipeline to extract knowledge from paragraph. This pipeline combines state-of-the-art systems in order to yield optimal results. There are some other Knowledge Graphs such as Google Knowledge Graph, YAGO, or DBpedia. Most of the data in these Knowledge Graphs is in English. On the other hand, the results from our system is used to build a new Knowledge Graph in Vietnamese of Vietnam Tourism. We use the rich resources language like English to process a low resources language like Vietnamese. We utilize the NLP tools of English such as Google translate, Stanford parser, Co-referencing, ClausIE, MinIE. We develop Google Search to find the text describing the entities in the Internet. This text is in Vietnamese. Then, we translate the Vietnamese text into English text and use English NLP tools to extract triples. Finally, we translate the triples back into Vietnamese and build the knowledge graph of Vietnam tourism. We conduct experiment and discover the advantages and disadvantages of our method.

Keywords Knowledge graph · Google search · Triples extraction · Co-reference resolution · Natural language processing

1 Introduction

The information that we have nowadays is larger than it has ever been before. Most of the time, this enormous amount of data is text and come mostly in form of unstructured data. Text provides a quick and simple way to transform ideas from one person to

P. Do (✉) · H. Le
University of Information Technology,
Vietnam National University, Ho Chi Minh City, Vietnam
e-mail: phucdo@uit.edu.vn

H. Le
e-mail: hungle1abc@gmail.com

R. Alfred et al. (eds.), *Computational Science and Technology*, Lecture Notes in Electrical Engineering 724, https://doi.org/10.1007/978-981-33-4069-5_1

another. In this big dataset, there are knowledge hidden everywhere. We have yet to find the best way to transform this data into useful knowledge.

It is impossible for any person to read all the text in the world. This is the problem we want to address because we need to find the information which we want in this ocean of words without wasting time reading about unrelated subjects.

Text are written in natural language, which is complex due to its ambiguity. The same sentence can have two different meanings in two different paragraphs. Natural language is so flexible that we can use it in different contexts with ease, but this feature made it extremely hard for computer to understand.

The flexibility of natural language makes it impossible to define a set of rules that can cover all use cases of it. Instead, we have to use algorithms that can extract meaning of each sentence and collect the core information from it. The core information is called "facts", or "triples". Any fact can be expressed as a triple of form [Subject, predicate, Object], where Subject and Object are names for real world entities, and predicate is the relationship between these entities. An example triple is [Sesame, is_a, food]. These triples are the basic components of a Knowledge Graph [1].

In this paper, we present a system to build knowledge graph of Vietnamese tourism in Vietnamese. Unluckily, Vietnamese is a low resources language which only has a few NLP resources (software and large data set).

The method we used in this study is that we leverage the powerful resources of English language such as Google translate, Stanford parser, Co-referencing, ClausIE, MinIE. We use Google Search to find the text of specified entity, then we process and translate this bare Vietnamese Text to English. After that, we use English NLP tools to process English Text. Finally, we convert text back to Vietnamese to build the knowledge graph. The knowledge graph will contain a lot of facts about tourism in Vietnam.

In our study, we used NeuralCoref to solve co-reference resolution in the paragraphs [2]. Then we used MinIE [3] to extract triples from sentences. Finally, a graph database called Neo4j [4] is used to store the extracted triples.

We had chosen to solve this problem because it has tremendous applications. One of the applications is currently used by Google search engine. When a user searches for some keywords, Google Search display a box that shows summarized information from articles contains user's keywords.

In this study, we have following contributions:

- We develop a system to utilize the NLP tools of rich resource language like English to extract the triples from text in low resource language like Vietnamese Text. Our research can be a typical application for all low resource languages.
- We design a pipeline containing sequential steps of NLP tools to build a knowledge graph of Vietnamese Tourism in Vietnamese.
- We conduct the experiment and discover the advantages and disadvantages of our proposed method. This result will be a guideline for users who want to build a non-English knowledge graph.

The rest of paper is organized as follows: Section 2 presents the Related Work of our research. Section 3 presents our methodology to solve the problem. Section 4

presents the implementation of our proposed system. Section 5 presents the experiment and the advantages and dis-advantages of our proposed methods. Finally, we conclude what we did and discuss about the future work.

2 Related Work

Google Knowledge Graph, YAGO, and DBpedia are the most well-known knowledge graphs.

Google Knowledge Graph was built in 2012 by Google. It provides direct information quickly by using the relationships between words and concepts from the query. It makes use of user behaviors, related entities and relationships.

YAGO is an open source knowledge base which was developed at the Max Planck Institute for Computer Science. This knowledge base contains more than 10 million entities and more than 120 million facts about these entities. The information was extracted from Wikipedia, WordNet, and GeoNames [5]. This knowledge base is the favorite data source of researchers who interested in testing new ideas on graph database.

DBpedia is a project which was created to extract structured content from Wikipedia articles [6]. The English version of DBpedia knowledge base describes 4.58 million things, ranging from many different topics like persons, places, species, disease, etc. DBpedia extracts information based on Wikipedia article structures and link them together with an extraction manage. Similar to YAGO, DBpedia structured contents can be query using SPARQL and it is free to use.

3 Methodology

3.1 Definition

There is no formal definition of knowledge graph. We consider knowledge graph as a graph where nodes are real world entities and edges are relationships between them. Moreover, knowledge graph also contain rules to enable reasoning to infer new knowledge from existent triples of knowledge graph.

3.2 Extract Triples of Knowledge Graph from Text

To conduct the research, we first looked for other systems that are trying to solve somewhat similar problem. Out of the systems that we saw, a few of them was really stood out. Those are ClausIE, MinIE, NeuralCoref in English, and Neo4j graph database.

We proceeded to combine these systems to build an end-to-end pipeline that can take a paragraph as input and return the knowledge graph as output. This knowledge graph is store in a graph database for later processing. Finally, we ran some experiences, discussed the results of our pipeline and presented some ideas we have moving forward.

3.3 Introduction to ClausIE and MinIE

ClausIE is one of the systems that were built to solve the task of Open Information Extraction (OIE) [7] in English. It consists of two separated steps. The first step is to detect "useful" information from the given sentence. This means ClausIE decides what information is expressed in the sentence, how to identify it, and how much of it worth keeping. The second step is to identify the sentence's representation. This is the part where ClausIE decides what is the form of the relation, should it use triples or n-ary proposition to generate representation of the information in text.

In order to detect useful information, ClausIE makes use of dependency parsing. It uses dependency parsing to detect the set of "clauses" of each sentence. No training data is needed for ClausIE to work properly. After the sets of clauses have been found, ClausIE's second step is to generate propositions for each clause based on the type of the clause. In the Table 1, the first clause pattern, "Tom" is the subject and "laughed" is the intransitive verb. Similarly, in the 7th clause pattern, "Tom" is the subject, "put" is the complex-transitive verb, "his computer" is the direct object, and "down" is the complement.

Though ClausIE achieves high precision and recall, it tends to produces overly-specific extractions. Therefore, MinIE was built on top of ClausIE to generate more useful and semantically richer extractions. For its extractions to be more compact, MinIE uses annotations for capturing the context of an extraction. These annotations represent information about polarity, modality, attribution, and quantities. MinIE also identifies and removes parts that are considered over-specific.

Table 1 Basic clause patterns and their examples

Clause patterns	Example sentences
SV_i	Tom laughed
SV_eA	Tom studied information systems
SV_cC	Tom is a student
$SV_{mt}O$	Tom likes books
$SV_{dt}OA$	Alice gave tom a cup of coffee
$SV_{ct}OA$	Alice taught Tom system design
$SV_{ct}OC$	Tom put his computer down

*S: Subject, V: Verb, C: Complement, O: Direct object, A: Adverbial, O_i: Indirect object, V_i: Intransitive verb, V_c: Copular verb, V_c: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

3.4 Co-reference Resolution and NeuralCoref

Co-reference resolution meaning finding all the words that refer to the same entity in a given piece of text. Let take a look at the following paragraph from Wikipedia: "VNUHCM-University of Information Technology is a public university located in Ho Chi Minh City, Vietnam. Although its name is about information technology, this university teaches many computer studies."

When human read the above paragraph, we can easily know that "its" and "this university" are referring to "VNUHCM-University of Information Technology". An effective co-reference resolution system should be able to do the same thing. Figure 1 shows that NeuralCoref can correctly determine the entities and their antecedences in our example paragraph.

3.5 Store Knowledge Graph in Neo4j Graph Database

Neo4j is a Graph Database management system. It was built to efficiently store, handle, and query highly-connected data. It has a powerful and flexible data model, so it is good choice to store semantic triples. A node in the graph could be a subject or an object in the triple, and the relationship between two nodes is the predicate between the subject and the object.

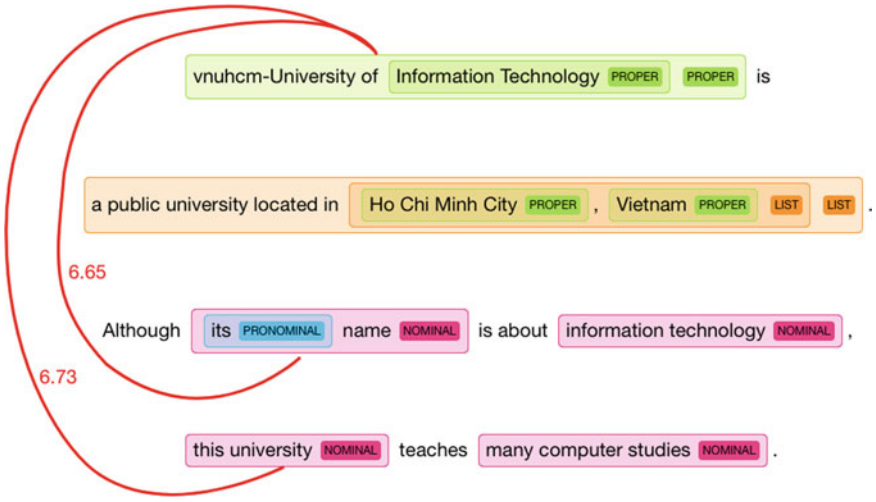

Fig. 1 Results from NeuralCoref with our example

3.6 The Pipeline of Proposed System

The ultimate goal of our research is to build a system which can receive a paragraph written in natural language and create a Vietnamese Knowledge Graph. Figure 2 shows the architecture of our pipeline.

The first step in this pipeline is the collection of English paragraphs related to our topic. We tried to find paragraphs related to tourism in Vietnam because that is our focus. We choose English Text because of two main reasons. The first reason is the amount of text written in English is much larger than the amount of text written in Vietnamese. The second reason is processing text written in Vietnamese has not become as good as processing text written in English. By using English text as input, we can take advantage of the tools that have already been developed for years to process text. As we can see later on in this pipeline, with Google Translate, all the text will be translated to Vietnamese. This method allows us to take advantage of existing tools while being able to have the results in Vietnamese for later use.

The next component is the co-reference resolution followed by triple extraction component. Combine the two components give us more accurate results than using each of them individually. After the triples are extracted, we use Google Translate API to translate them into Vietnamese. Since the triples contain only phrases, not whole sentences, translation systems will do a generally good job. In cases where the phrases of Vietnamese Text are not translated correctly, an expert can step in and edit the translations directly.

The next component is an entity mapping suggestion component. This component uses Jaccard's similarity algorithm and a dictionary in order to give suggestions about the types of entities and relationships.

Finally, we stored the result in Neo4j.

3.7 The Structure of the Knowledge Graph

In this research, we tried to collect data that fall into one of the schemas in Table 2. We intended to build a knowledge graph of Vietnam tourism [8]. In Table 2, we use English and Vietnamese to describe the head, the predicate and the tail of triples of our knowledge graph.

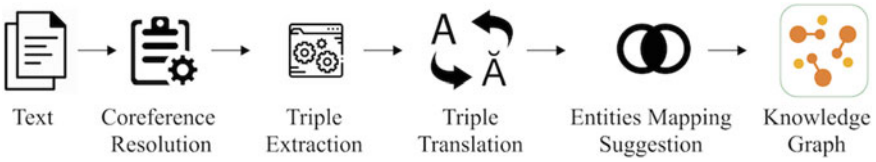

Fig. 2 Knowledge Graph construction pipeline

Table 2 Types of entities and relationships

Head type	Predictate type	Tail type
Landscape (Thắng cảnh)	IN (TOẠ LẠC TẠI)	Place (Địa danh)
Festival (Lễ hội)	FESTIVAL IN (LỄ HỘI TẠI)	Place (Địa danh)
Hero (Anh hùng dân tộc)	WAS BORN IN (SINH RA TẠI)	Place (Địa danh)
Dish (Món ăn)	SPECIAL DISH AT (MÓN ĂN ĐẶC SẢN CỦA)	Place (Địa danh)
Folk song (Dân ca)	TRADITIONAL SONG OF (DÂN CA CỦA)	Ethnic Group (Dân tộc)
Musical instrument (Nhạc cụ)	INSTRUMENT OF (NHẠC CỤ CỦA)	Ethnic Group (Dân tộc)
Ethnic Group (Dân tộc)	LIVING IN (SINH SỐNG TẠI)	Place (Địa danh)

Figure 3 clearly shows the structure of our Knowledge Graph. The nodes represent the types of entities in our Knowledge Graph, and the arrows represent the types of the relationships between them.

4 System Implementation

The implementation of the experimental system is as follow: (i) On the Client: just install a browser like Chrome or Firefox; (ii) On the Server: install Docker, set up environment variables and run the command "docker-compose up –build"; (iii) Go to http://127.0.0.1:5000 to test the system.

In order to generate the knowledge graph from paragraphs, our system has the following services as described in Table 3.

Detail descriptions and algorithms of the services of our system are described below.

Table 3 The systems' services

No.	Services	Description
1	Co-reference resolution	Resolved mentions of same entities in the paragraph
2	Triples extraction and translation	Extract triples from the paragraph
3	Types recommendation	Translate the triples into Vietnamese and recommend the type of entities/relationships

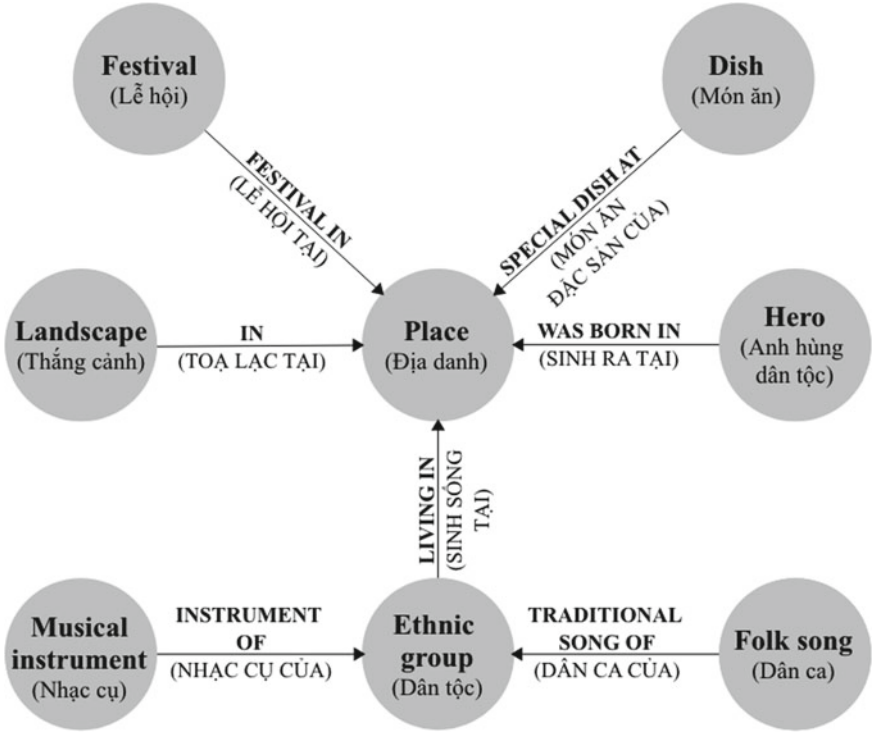

Fig. 3 Structure of the Knowledge Graph

(a) The "Co-reference Resolution" service:

The *Co-reference Resolution* service is the service that take the responsibility to resolve entity co-occurrences. Within this service, we used NeuralCoref library to extract mentions and their duplications.

The input of this service is a normal paragraph, and the output of this service the resolved paragraph. The service did this by replacing the mentions of the same entity with the first occurrence of that entity in the paragraph. The details algorithm of the *Co-reference Resolution* service is described in Algorithm 1.

Algorithm 1 The Co-reference Resolution algorithm

Input: Normal paragraph.
Output: Resolved paragraph.

1: Run NeuralCoref with the input paragraph to find mentions of the same entities in the paragraph.
2: Replace all the mentions of the same entities with the first occurences of those entities in all sentences of the paragraph.
3: Return the resolved paragraph.

(b) The Triples Extraction and Translation service

After we have the resolved paragraph from the *Co-reference Resolution* service, the next step is to extract useful triples to create our Knowledge Graph. This service takes each sentence of the resolved paragraph as its input and produces triples that the sentence contains.

The service detects triples by analyzing the English clause types. This analysis was done within ClausIE. MinIE then eliminates triples that are consider too specific and adds annotation to the triples. Our service then go a bit further and translate these triples into Vietnamese with human's verification. The result is a list of triples in English and Vietnamese along with their polarity and modality. Algorithm 2 describes how *Triples Extraction and Translation* service works.

Algorithm 2 The Co-reference Resolution algorithm

Input: A sentence from the resolved paragraph.
Output: All the triples (in English and Vietnamese) that the sentence contains.

1: The sentence is processed with MinIE to produces triples that that sentence contains along with their polarity and modality.
2: An expert can change the translations at this step directly from the browser.
3: Return the list of triples that the sentence contains.

(c) The Types Recommendation service

The *Types Recommendation* service will recommend the type that an entity or a relationship is supposed to have. The list of possible type in this research is as described in Table 2.

We recommend the type by using VNCoreNLP [8] to segmentate Vietnamese words from the triples first, then compare these words with our dictionary using the Jaccard similarity algorithm to determine the type of the entity or relationship. Algorithm 3 is the explanation of this service.

Algorithm 3 The Types Recommendation algorithm

Input: A Vietnamese triple.
Output: Recommended types of entities and the relationship of the triple.

1: The phrases of the triple is seperated into words with VNCoreNLP Word Segmentation [9, 10].
2: The words are compared with our dictionary using Jaccard similarity algorithm with the coefficient is 80
3: If their is no recommended type found, the returned type will be "UNKNOWN".
4: Return the recommended types for the triples.

5 Experiment and Discussion

In our system, the paragraphs are obtained through two sources: Google Search and Wikipedia. First, we collected the list of entities (written in Vietnamese) that we are interested in. Next, we either run these entities with our questions using Google Search to get the desired paragraphs about the entities; or we get the summary paragraphs of our entities by Python's Wikipedia API. Finally, these paragraphs are translated into English and feed to our system.

In this section, we will show the results of our system when we process the following paragraph which was taken from Wikipedia:

"$Dà$ Lạt city is the capital of Lâm $Dồng$ Province in Vietnam. The city is located 1,500 m above sea level on the Langbian Plateau in the southern parts of the Central Highlands region. Da Lat is the most popular tourist destination in Vietnam."

For clarity, we use a Text in English. Normally, Text is in Vietnamese and is translated to English by Google Translator and verified by man.

5.1 Results from the Co-Reference Resolution

For this paragraph, the system understood the paragraph correctly. The mention "The city" in the second sentence of the paragraph is replaced with the entity "Da Lat". Fig 4 shows the resolved paragraph in our system.

5.2 Results from Triples Extraction and Translation

For our example paragraph, the system extracted and translated a total of seven triples. They are listed in Table 4.

As the results shows that, some of the triples are wrong (No. 2), some of them are useless (No. 5), but some are pretty useful (No. 1, No.4, No.6) for our Knowledge Graph.

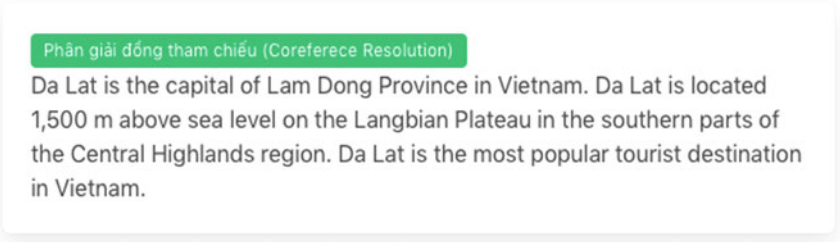

Fig. 4 Resolved paragraph

Table 4 Extracted triples from the example paragraph

No.	Subject	Predicate	Object
1	Da Lat	Is capital of	Lam Dong Province
2	Da Lat	Is capital in	Vietnam
3	Da Lat	Is	Capital
4	Da Lat	Is located	1,500 m above sea level on Langbian Plateau in southern parts of Central Highlands region
5	Da Lat	Is	Located
6	Da Lat	Is most popular tourist destination in	Vietnam
7	Da Lat	Is	Most popular tourist destination

The triple No. 7 in Table 4 is translated into the Vietnamese triple: (“Đà Lạt”, “là hầu hết các du lịch nổi tiếng đích trong”, “Vietnam”). This is not the most accurate translation but as Google Translate get smarter, we can expect to get a better translation.

5.3 *Results from the “Type Recommendation” Service*

Fig 5 shows the recommended types for the triple No. 7 in Table 4.

The system can identify the entity ““Đà Lạt”” as a Place (“(“Địa danh”)”). For the other entity and the relationship, the system could not guess the type. However, the expert can step in and set the types of the entity/relationship directly, therefore, the system will be able to provide more suggestions over time.

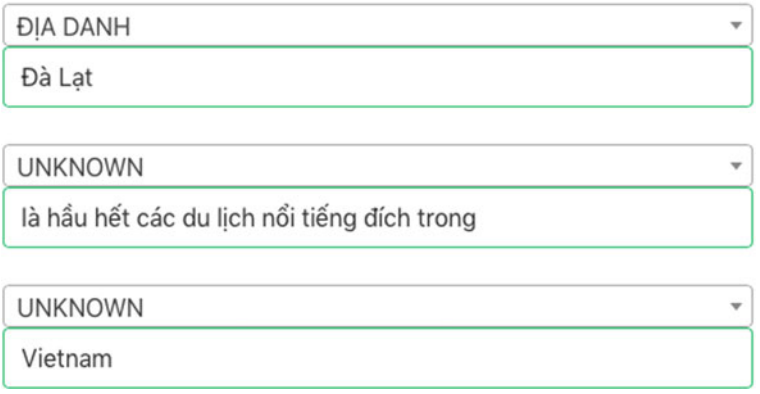

Fig. 5 A triple with its recommended type

5.4 Discussions

Through our experiments, we had found some advantages as well as some dis-advantages in our proposal system.
Advantages Our approach presents the following advantages:

- We can stay away from the complexity of the sub-problem which other tools have been trying to solve. Instead, we focused on the demands of our specific system such as the entities' types or the structure of our knowledge graph.
- By utilizing existing NLP tools, we were able to build our system quickly.

Dis-advantages Our approach, however, still has some cons:

- Although we had design our system so that we can easily replace any component with a better version, our system still depends on other NLP tools.
- Translations from Google Translate are not always accurate, and we lost some of the native features and characteristics that only exist in Vietnamese.

6 Conclusion and Future Work

In this paper, we proposed a system that can build a knowledge graph in Vietnamese of Vietnam Tourism. Our system is assembled by taking advantages of state of the art components of English like co-reference resolution, open information extraction, word segmentation, etc. Each of these components can also be further optimized independently. We hope that this system could be the baseline system that future systems in this domain can compare to.

In the future, we would like to use Deep Learning to exploit information from text.

Acknowledgements This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCMC) under the grant number DS2020-26-01.

References

1. Ehrlinger L, Woss W (2016) Towards a definition of knowledge graphs
2. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Cistac P, Rault T, Louf R, Funtowicz M, Brew J (2019) Transformers: state-of-the-art natural language processing. ArXiv
3. Gashteovski K, Gemulla R, Del Corro L (2017) MinIE: minimizing facts in open information extraction. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp 2630–2640
4. Webber J (2012) A programmatic introduction to Neo4j. In: Proceedings of the 3rd annual conference on systems, programming, and applications: software for humanity, pp 217–218
5. Suchanek F, Kasneci G, Weikum G (2007) YAGO: a core of semantic knowledge. In: 16th international world wide web conference, WWW2007, pp 697–706

6. Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes P, Hellmann S, Morsey M, Van Kleef P, Auer S, Bizer C (2014) DBpedia—a large-scale. Multilingual knowledge base extracted from Wikipedia, semantic web journal
7. Corro L, Gemulla R (2013) ClausIE: clause-based open information extraction. In: WWW 2013—proceedings of the 22nd international conference on world wide web, pp 355–366
8. Do P (2019) SparkHINlog: extension of sparkDatalog for heterogeneous information network. J Intell Fuzzy Syst 37(6):7555–7566
9. Vu T, Nguyen DQ, Nguyen D, Dras M, Johnson M (2018) VnCoreNLP: a vietnamese natural language processing toolkit. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: demonstrations, pp 56–60
10. Vu T, Nguyen DQ, Nguyen D, Dras M, Johnson M (2017) From word segmentation to POS tagging for Vietnamese. In: Proceedings of the 15th annual workshop of the Australasian Language technology association, pp 108–113