BUILD KNOWLEDGE GRAPH FROM HETEROGENEOUS DOCUMENTS

HIEU CHI NGUYEN
Industrial University of Ho Chi Minh City,
nchieu@iuh.edu.vn

Abstract. Knowledge Graphs are applied in many fields such as search engines, semantic analysis, and question answering in recent years. However, there are many obstacles for building knowledge graphs as methodologies, data and tools. This paper introduces a novel methodology to build knowledge graph from heterogeneous documents. We use the methodologies of Natural Language Processing and deep learning to build this graph. The knowledge graph can use in Question answering systems and Information retrieval especially in Computing domain.

Keywords. Knowledge graph, Question answering, Graph databases.

1 INTRODUCTION

Most of human knowledge can be formalized in entities, abstract concepts, categories and the relation between them. A knowledge graph (KG) is a natural candidate for representing this. NELL [1], Freebase [2], and YAGO [3] are examples of large knowledge graphs that include millions of entities and facts. Facts are represented as triples, each consisting of two entities connected by a binary relation, e.g., (concept: city: Hanoi, relation: country capital, concept: country: Vietnam). The entities such as Hanoi and Vietnam are represented as nodes and the relation country capital is represented as binary link which connect these nodes. In recent years, knowledge graph embedding (KGE) has been applied to many fields. In KGE, entities and relations are embedded in vector space, and operations in this space are used for defining a confidence score function Θ_{ijk} that approximates the truth value of a given triple (e_i, e_j, r_k).

Although the knowledge graph such as Freebase has the millions of entities and the billions of relations, but it seems the incomplete knowledge graph because there are not many relations among the entities in this graph. Therefore, one of the big problems in knowledge graph embedding is that the knowledge graph is completed.

Our key contributions are as follows: (i) We have crawled a large-scale dataset from the ACM Digital Library and Wikipedia focus on computing domain for knowledge graph embedding; (ii) We propose new structure of knowledge graph;

The rest of this paper is organized as follows: section 2 - related works; section 3 - automatic subject labeling of text document; section 4 - experimental results and discussion; section 5 - conclusions and future works.

2 RELATED WORKS

In recent years, Knowledge graph are interested in the researchers for representation the big data. As outline from Xin Lv et al. [4], they proposed a novel knowledge graph embedding model named TransC by differentiating concepts and instances. Specifically, TransC encodes each concept in knowledge graph as a sphere and each instance as a vector in the same semantic space. Besides, their knowledge graph is shown the relations between concepts and instances and the relations between concepts and sub-concepts. G. Zhu et al. [5] proposed a knowledge graph for exploiting semantic similarity for named entity disambiguation. They also proposed a Category2Vec embedding model based on joint learning of word and category embedding, in order to compute word-category similarity for entity disambiguation. B. Kotnis and V. Nastase [6] proposed Knowledge graphs including only positive relation instances, leaving the door open for a variety of methods for selecting negative examples. They also present an empirical study on the impact of negative sampling on the learned embeddings, assessed through the task of link prediction. They used state-of-the-art knowledge graph embedding methods including Rescal, TransE, DistMult and ComplEX. S.S. Dasgupta et al [7] proposed HyTE, a temporally aware knowledge graph embedding method which explicitly incorporates time in the entity-relation space by associating each timestamp with a corresponding hyperplane. HyTE not only performs knowledge graph inference using temporal guidance, but also predicts temporal scopes for relational facts with missing time annotations. X. Huang et al. [8] proposed a Question answering system over knowledge graph to use facts in the knowledge graph to answer natural language questions. It helps end users more efficiently and more easily access the substantial and valuable knowledge in the knowledge graph, without knowing its data structures. The Question answering over knowledge graph is a nontrivial problem since capturing the semantic meaning of natural language is difficult for a machine. Meanwhile, many knowledge graphs embedding methods have been proposed. The key idea is to represent each predicate/entity as a low-dimensional vector, such that the relation information in the knowledge graph could be preserved.

Generally, there is a lot of methods to have knowledge graph for applied to many different fields. The researches can apply approaches related to NLP, Machine Learning, Deep learning or hybrid approaches. In this paper, we use NLP and deep learning for data training to build knowledge graph focusing computing domain.

3 HETEROGENEOUS DOCUMENTS BASED KNOWLEDGE GRAPH EMBEDDING

Definition 1. Heterogeneous document is mean that they include text documents from ACM digital library, XML documents from Wikipedia and data stream form WordNet database.
Definition 2. A knowledge graph G includes vertex representing entities, class, subclass and edges representing relationship among vertex.

3.1 Knowledge Graph Embedding (KGE)

Knowledge graph KG = (V, E) contain knowledge in the form of relation triple (s, r, o) where s, o ϵ V are entities and r ϵ E are relationship between entities. S denotes Subject; O denotes Object and r denotes Relation.

According to Quan Wang [9], KGE represents entities as low- dimensional vector, such that the original structures and relations in the KG are preserved in these learned vectors. The core idea of most of the existing KG embedding methods could be summarized as follows. For each fact (s, r, t) in G, we denote its embedding representations as (e_s, p_r, e_t). The embedding algorithm initializes the values of e_s, p_r, and e_t randomly or based on the trained word embedding models. Then, a function f(·) that measures the relation of a fact (s, r, t) in the embedding spaces is defined, i.e., $e_t \approx f(e_s, p_r)$.

TransE in knowledge graph interprets relations as a translation operation from the source to the target mediated by the relation. Such KGs contain rich structured knowledge and are useful for many NLP tasks.

3.2 Building KGE from text documents of ACM Digital Library

The process for training text documents of ACM Digital Library includes 2 phrases:

- The first phrase is data preprocessing
- The second phrase is using Keras framework including word embedding model on text data.

In the first phrase, we merge all of text file into a single text file based on their category. These files are as input and it is sent to Tokenizer. The Tokenizer split the sentences into words based on whitespace character. The tokenized words are taken to extractor for converting to lowercase, removing punctuation from each token and filtering out remaining tokens that are not alphabetic as well as filtering out tokens that are stop words. After removing stop words from the text files, these text files are taken to extractor again for stemming process. Stemming refers to the process of reducing each word to its root or base. For example, fishing, fished, fisher all reduces to the stem fish. Some applications, like document classification, may benefit from stemming in order to both reduce the vocabulary and to focus on the sense or sentiment of a document rather than deeper meaning. There are many stemming algorithms, although a popular and long-standing method is the Porter Stemming algorithm. In addition, we use Natural Language Tool Kit (NLTK) [10] for data preprocessing.

In the second phrase, we use Keras [11] framework including word embedding for training data. The model is shown in Fig 1.

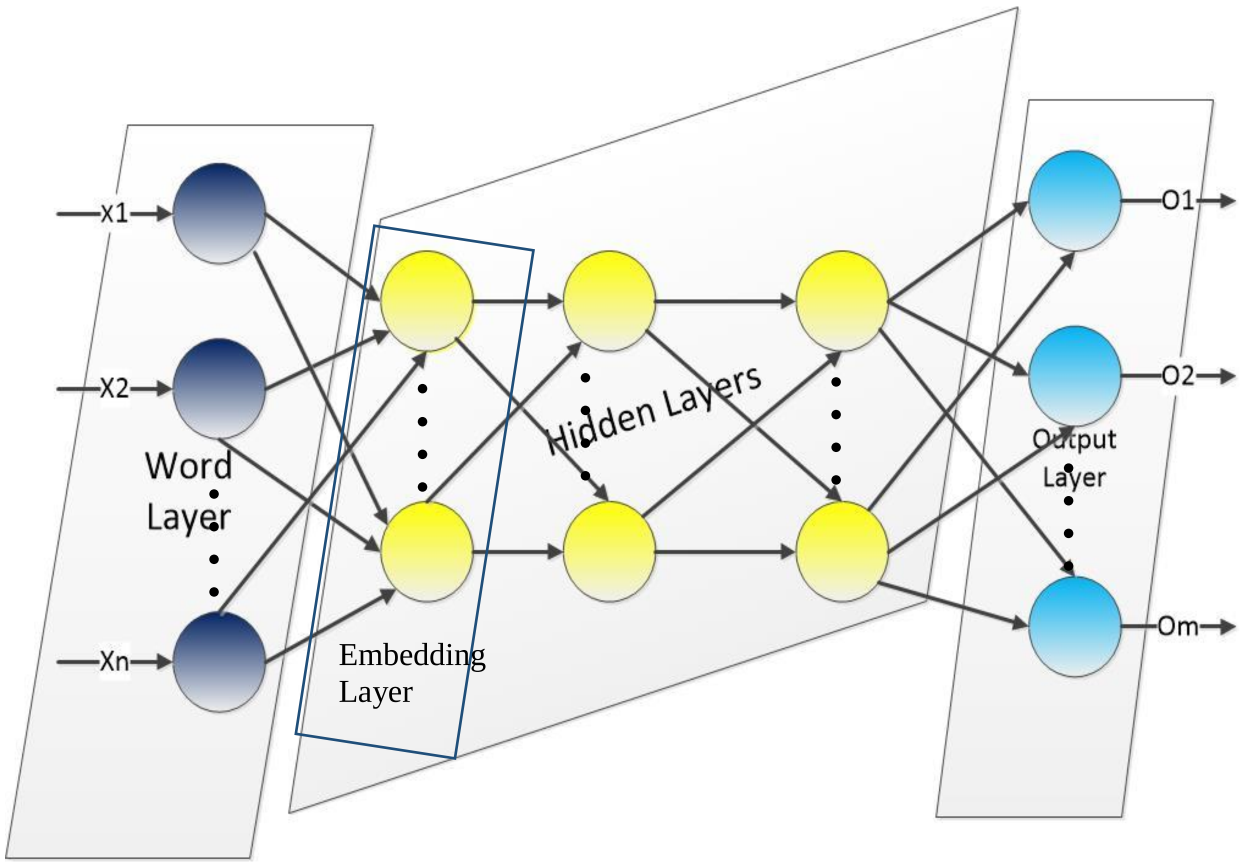

Figure 1: The model using Keras framework including word embedding (word2Vec)

After finishing data training, the set of word vectors are used to build KG. The structure of KGE is separated into two layers and the root of KGE is Computing domain.

The first layer is known as the Subject layer. This layer including categories from ACM Classification Categories [12]. We obtain over 30 different categories from this site.

Next layer is known as the Object layer. In this layer, there are many different objects which are output from Word2Vec word embedding model, e.g., “Computer”, “Memory”, “Programming”, “Processor”, “Model and Principle”, etc.

Initially, relationship between the root and the Subject layer is called “Belong to” and the relationship between the Subject and Object layer also is called “Belong to”. However, our model also supports extra relationships such as Part-Of and Has-Part, Is-Member-Off and Has-Member, Hypernymy and Hyponymy (defined in WordNet), or Relevant-Of, using for pair of subjects, or pair of objects, to build hierarchical networks of linked and levelized items in each layer. This extension can provide more powerful ability for our KGE to structurizing a large amount of items in different semantics levels. The relationships are detected through by the Object extraction from by ACM Digital Library and Wikipedia. The KGE representing for Computing domain is shown as Fig 2. They can also be recognized base on the synsets of WordNet with many predefined semantics networks of data objects.

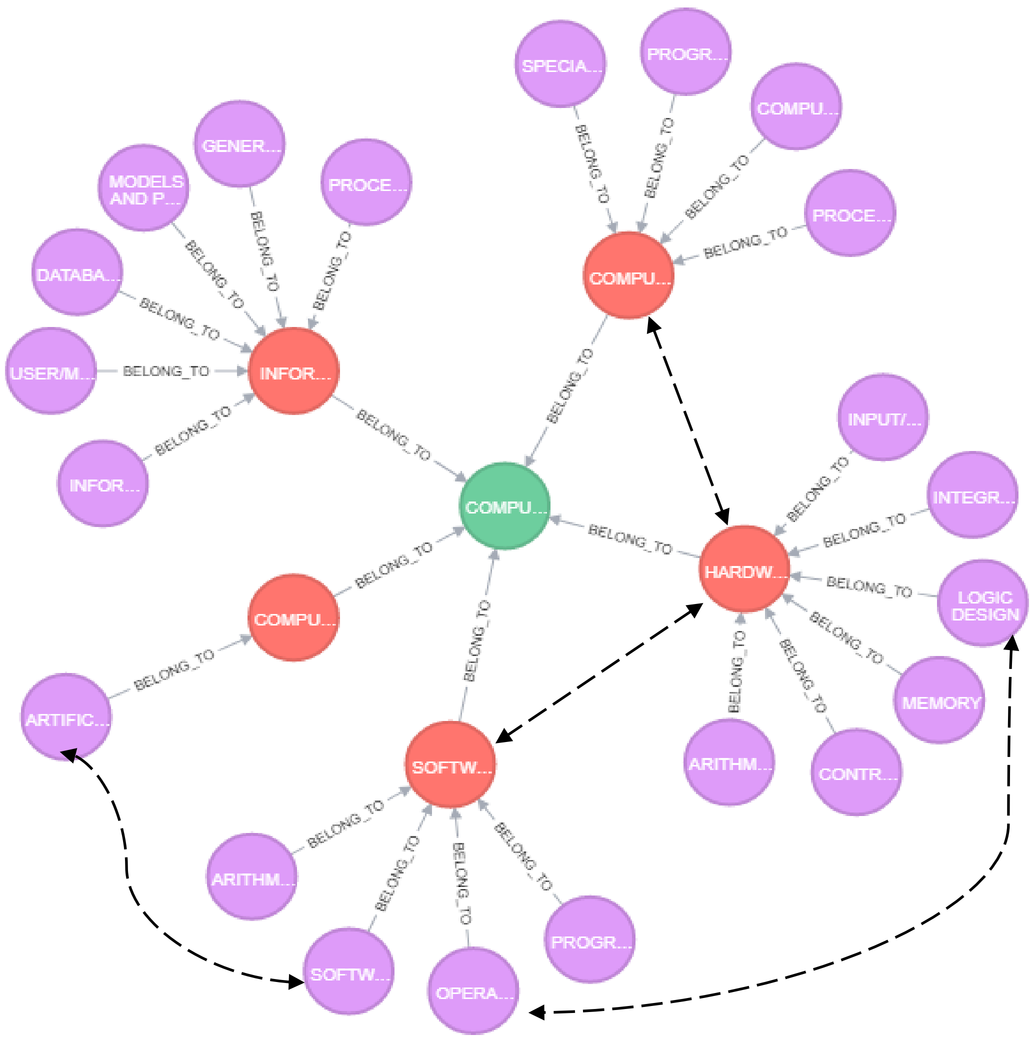

Figure 2: The hierarchy of KGE

3.3 Updating KGE from XML documents of Wikipedia

The process to update KGE by objects extracted from Wikipedia [13] includes three phrases:

- The first phrase is to prepare XML files including objects belong to categories of ACM Digital Libraries
- The second phrase and the third phrase are like building KGE from text documents of ACM Digital Libraries (3.2)

In order to access and extract data belong to a category from Wikipedia, we use Wikipedia API as like "https://en.wikipedia.org/w/api.php?action=query&list=computing&cmtitle=Computing:Wikipedia&cmtype=Database". After accessing data, we can save to xml files. We remove the HTML tags in these files before processing the second and third phrase. The object which are extracted from Wikipedia (XML files) will be updated into Object layer of KGE by category and by relationships predefined in KGE in the similar processing approach of the section 3.2

4 EXPERIMENTAL RESULT AND DISCUSSION

4.1 Evaluation based on three measures

We implement numerous experiments for studying the efficiency of the proposed approach. We select papers which have only abstract part belong to five categories from ACM Digital Library for testing as following. These groups cover multiple subgroup with items leveled in hierarchical structure.

– 100 abstracts in Software category;
– 100 abstracts in Process Management category;

– 100 abstracts in Artificial Intelligent category
– 100 abstracts in Operating system category
– 100 abstracts in Logic Design category

We use three measures: Precision (P), Recall (R) and F-measure for experimental evaluation.

$$P(C_i) = \frac{Correct(C_i)}{Correct(C_i) + Wrong(C_i)} \quad (1)$$

$$R(C_i) = \frac{Correct(C_i)}{Correct(C_i) + Missing(C_i)} \quad (2)$$

$$F - measure(C_i) = 2\frac{Precision(C_i) * Recall(C_i)}{Precision(C_i) + Recall(C_i)} \quad (3)$$

Where:

C_i denotes a category in CDO; Correct (C_i) denotes a number of the sentences which are found in CDO and they accurately belong to the category C_i; Wrong (C_i) denotes a number of the sentences which are found in CDO, but they do not belong to category C_i; Missing (C_i) denotes a number of the sentences which are not found in CDO. The experimental evaluation is shown as Table 1.

Table 1. The experimental evaluation for building KGE relating to Computing domain

Categories	Precision	Recall	F-Measure
Artificial Intelligent	94.12%	88.41%	91.18%
Logic Design	92.78%	56.81%	70.47%
Operating System	85.70%	82.14%	83.88%
Process Management	95.45%	75.10%	84.06%
Software	96.12%	91.63%	93.82%

4.2 Comparative approach

We use TF/IDF approach [14] for comparative approach. TF/IDF is short for Term Frequency–Inverse Document Frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The TF/IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general [13].

We use the same corpora for comparative approach. The corpora like as 4.1.

The results are shown in Table 2.

Table 2. Data comparison between TF/IDF and Deep Learning Approaches

Categories	Precision	Precision (DL)	Recall	Recall (DL)
Artificial Intelligent	93.03%	94.12%	88.62%	88.41%
Logic Design	91.41%	92.78%	54.72%	56.81%
Operating System	83.71%	85.70%	81.37%	82.14%
Process Management	94.72%	95.45%	76.02%	75.10%
Software	94.52%	96.12%	92.19%	91.63%

The Figure 3, 4, 5, 6 show the different data for each category in detail.

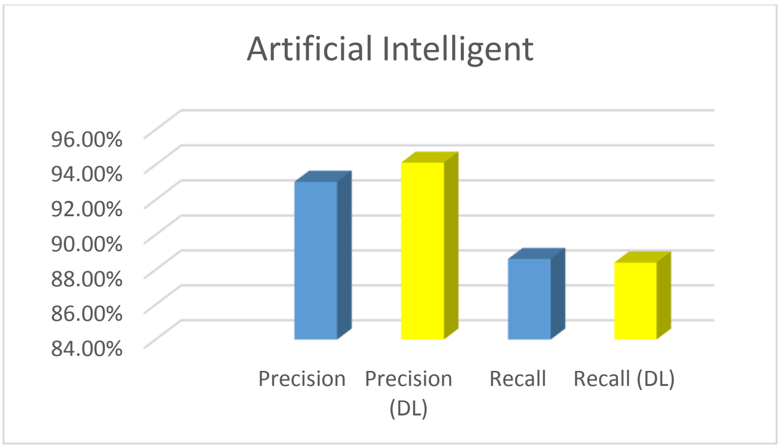

Figure 3: Data comparison between TF/IDF and Deep Learning by Artificial Intelligent category

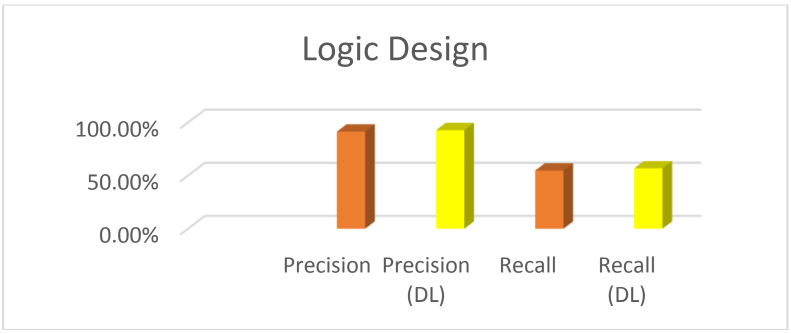

Figure 4: Data comparison between TF/IDF and Deep Learning by Logic Design category

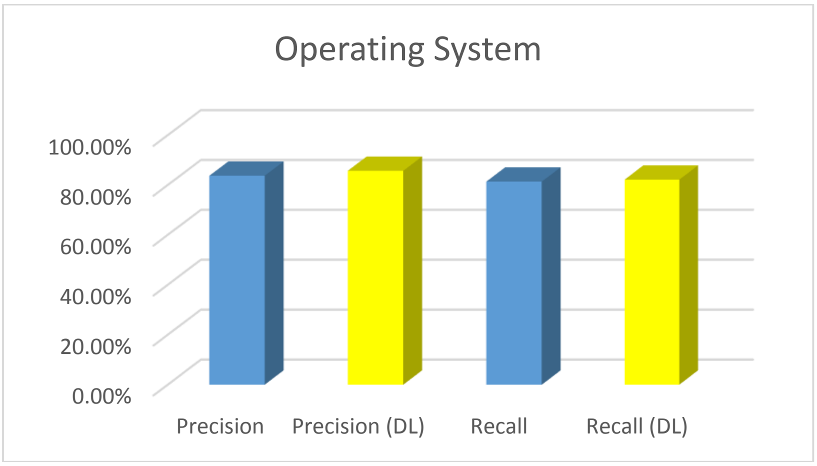

Figure 5: Data comparison between TF/IDF and Deep Learning by Operating System category

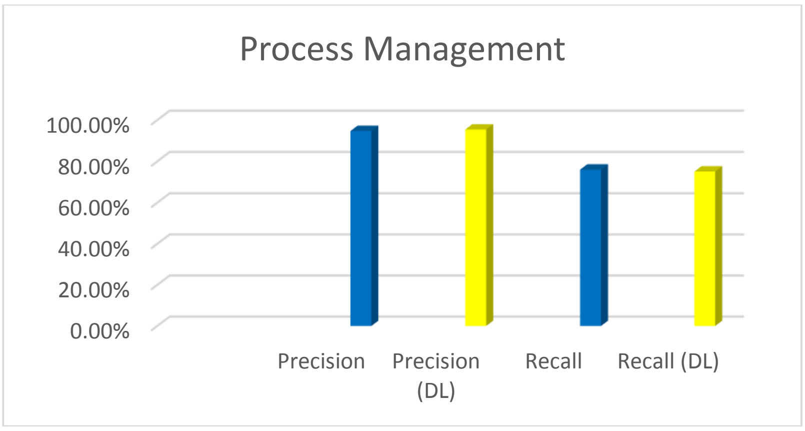

Figure 6: Data comparison between TF/IDF and Deep Learning by Process Management category

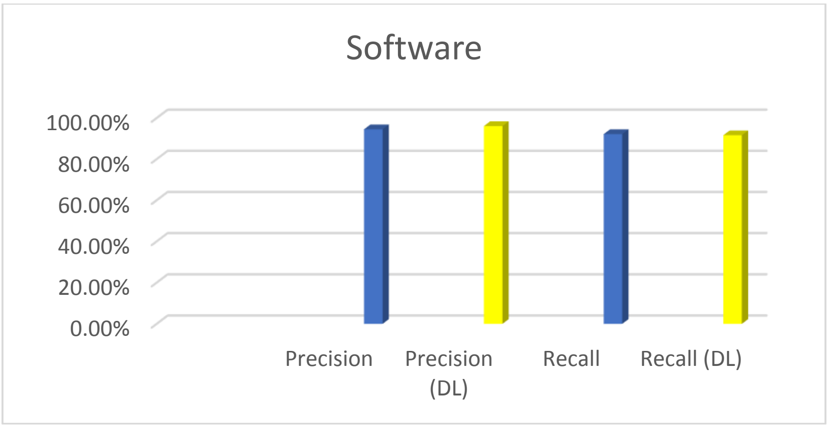

Figure 7: Data comparison between TF/IDF and Deep Learning by Software category

In general, the scores reported in Fig. 3, 4, 5, 6, 7 reveals that using Deep Learning for data training improves the Precision measure, but the Recall measure can be improved or not depend on the category.

5 CONCLUSIONS AND FUTURE WORKS

Our experiment tried to build KGE from documents of ACM Digital Library and XML files which are extracted from Wikipedia focus on Computing domain. We proposed an approach has two phases: the first phase is data preprocessing including tokenized words, converting to lowercase, removing punctuation and stemming. In the second phase, we use Keras based on Theno with embedding and some hidden layers for data training. The data after training will be filtered base on the predefined relationships and synsets in WordNet then be used for building KGE in multiple hierarchical networks and semantics layers. This KGE can be applied for many applications relating to Information Retrieval. We apply three measures as Precision, Recall and F-Measure for evaluation our approach. Besides, we also use the TF/IDF on the same corpora for comparative approach. In the future, we use some available special ontologies only focusing Computing domain for enriching this KGE also enhancing the relationships with statistical factors to optimize the networks of each layers either improve the trained data quality.

REFERENCES

[1] Never-Ending Language Learning - NELL. Online: http://rtw.ml.cmu.edu/rtw/
[2] Online: https://developers.google.com/freebase
[3] Online: https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

[4] X. Ly et al (2018), *Differentiating Concepts and Instances for Knowledge Graph Embedding*, in the Proceedings of the Conference on Empirical Methods in Natural Language Processing.
[5] G. Zhu (2018), *Exploiting semantic similarity for named entity disambiguation in knowledge graphs*, International Journal of Expert Systems with Applications, Vol 101.
[6] Kotnis, V. Nastase (2017), *Analysis of the impact of negative sampling on link prediction in knowledge graphs*, CoRR, 2017
[7] S.S. Dasgupta et al. (2018), *HyTE: Hyperplane-based Temporally Aware Knowledge Graph Embedding*, in the Proceedings of the Conference on Empirical Methods in Natural Language Processing, Belgium, November 2018, pages 2001–2011.
[8] X. Huang et al (2019), Knowledge Graph Embedding Based Question Answering, in the Proceedings of the Conference on Web Search and Data Mining (WSDM 2019).
[9] Q. Wang et al (2017), *Knowledge Graph Embedding: A Survey of Approaches and Applications*, in the Proceedings of the Conference on IEEE Transactions on Knowledge and Data Engineering (TKDE 29), Dec 2017, page 2724–2743.
[10] NLTK Project. Online https://www.nltk.org/news.html
[11] Keras Project. Online https://keras.io/
[12] The ACM Computing Classification System. Online: https://www.acm.org/publications/computing-classification-system/1998/ccs98
[13] Wikipedia. Online: https://en.wikipedia.org/wiki/Tf%E2%80%93idf
[14] Chien. Ta Duy Cong, Tuoi. Phan Thi (2015), *Building Ontology Based-on Heterogeneous Data*, Journal of Computer Science and Cybernetics, vol. 31, no.2, 2015, ISSN: 1813-9663.

XÂY DỰNG ĐỒ THỊ TRI THỨC TỪ TÀI LIỆU KHÔNG ĐỒNG NHẤT

NGUYỄN CHÍ HIẾU
[1] *Khoa CNTT, Trường Đại học Công nghiệp Tp.HCM;*
Email: nchieu@iuh.edu.vn

Tóm tắt. Trong những năm gần đây, đồ thị tri thức được áp dụng trong nhiều lĩnh vực của khoa học máy tính như công cụ tìm kiếm, phân tích ngữ nghĩa và trả lời câu hỏi... Tuy nhiên, có nhiều trở ngại cho việc xây dựng đồ thị tri thức (phương pháp, dữ liệu và công cụ). Bài viết này giới thiệu một phương pháp mới để xây dựng đồ thị tri thức từ các tài liệu không đồng nhất. Chúng tôi sử dụng các phương pháp xử lý ngôn ngữ tự nhiên và học sâu (deep learning) để xây dựng đồ thị này. Đồ thị tri thức có thể sử dụng trong các hệ thống trả lời câu hỏi và truy xuất thông tin, đặc biệt là trong ngôn ngữ học tính toán.
Từ khóa. Knowledge graph, Question answering, Graph databases

Ngày nhận bài: 06/04/2020
Ngày chấp nhận đăng: 15/06/2020