The Thirty-Eighth Annual Conference on Neural Information Processing Systems

Vancouver Convention Center
Tuesday Dec 10 through Sunday Dec 15

NeurIPS 2024 Main Conference

Text2NKG: Fine-Grained N-ary Relation Extraction for N-ary relational Knowledge Graph Construction

Haoran Luo[1], Haihong E[1*], Yuhao Yang[2], Tianyu Yao[1], Yikai Guo[3], Zichen Tang[1], Wentai Zhang[1], Shiyao Peng[1], Kaiyang Wan[1], Meina Song[1], Wei Lin[4], Yifan Zhu[1], Luu Anh Tuan[5]

[1] School of Computer Science, Beijing University of Posts and Telecommunications, China
[2] School of Automation Science and Electrical Engineering, Beihang University, China
[3] Beijing Institute of Computer Technology and Application [4] Inspur Group Co., Ltd., China
[5] College of Computing and Data Science, Nanyang Technological University, Singapore
{luohaoran, ehaihong, yifan_zhu}@bupt.edu.cn, anhtuan.luu@ntu.edu.sg

CONTENTS

Text2NKG: Fine-Grained N-ary Relation Extraction for N-ary relational Knowledge Graph Construction

NEURAL INFORMATION PROCESSING SYSTEMS

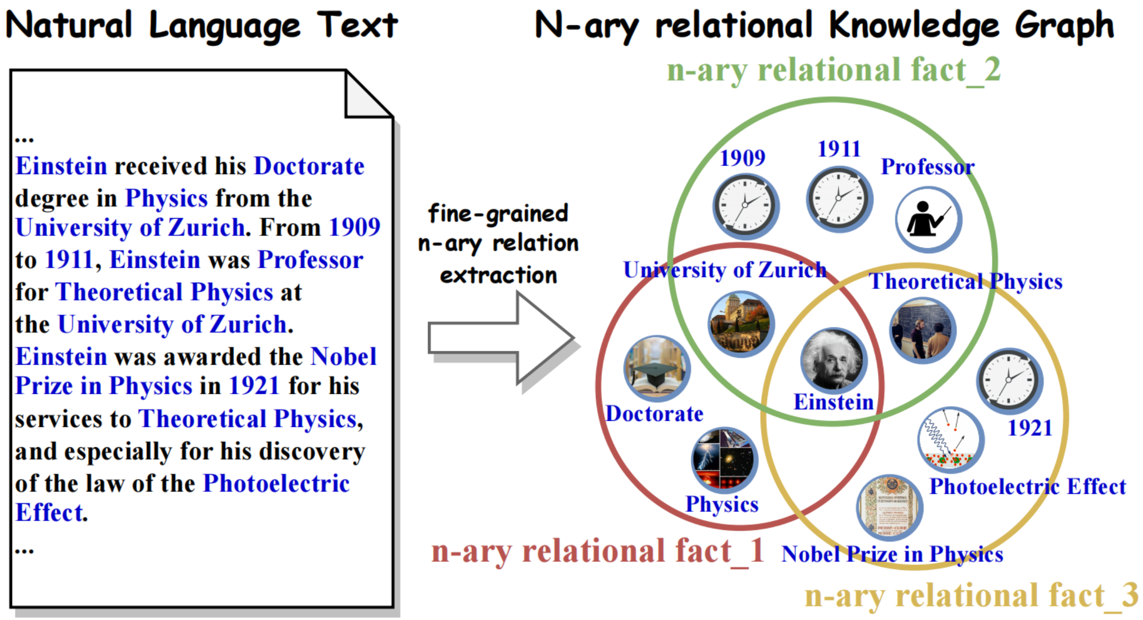
1

P A R T

Background of Study

N-ary Knowledge Graph Construction

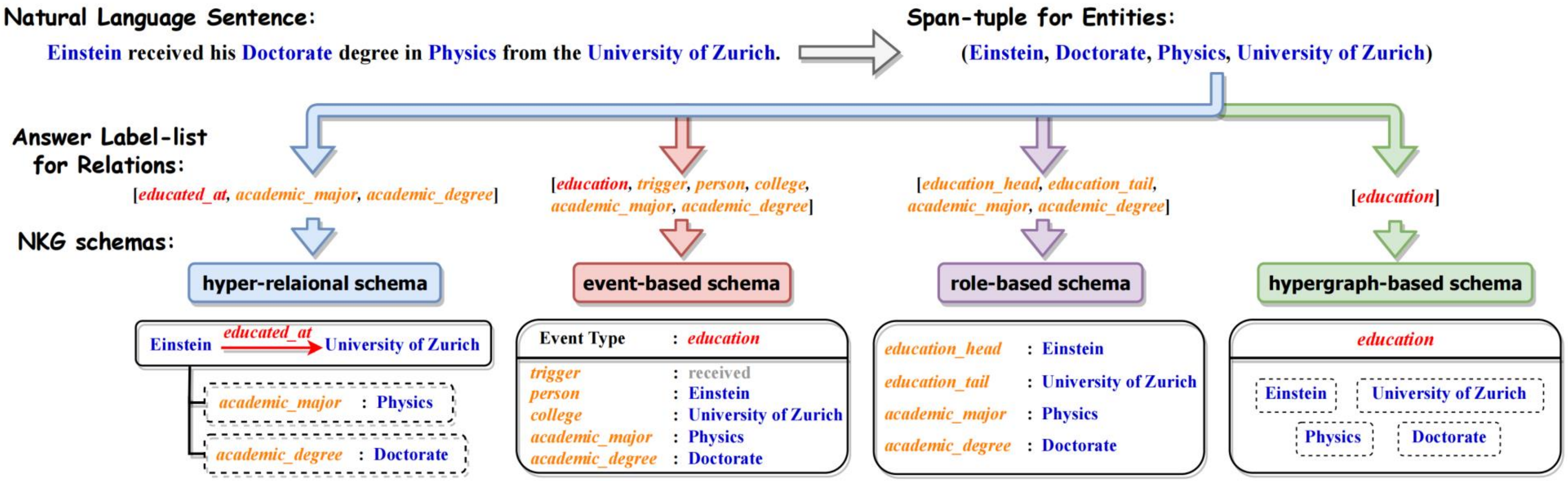
Traditional KGs are mostly composed of binary relational facts (*subject*, *relation*, *object*), which represent the relationship between two entities. However, it has been observed that over 30% of real-world facts involve n-ary relation facts with more than two entities ($n \geq 2$).



As shown in Figure, an n-ary relational knowledge graph (NKG) is composed of many n-ary relation facts, offering richer knowledge expression and wider application capabilities.

Fine-Grained N-ary Relation Extraction

As a key step of constructing NKGs, n-ary relation extraction (n-ary RE) is a task of identifying n-ary relations among entities in natural language texts. Compared to binary relational facts, n-ary relational facts in NKGs have more diverse schemas for different scenarios.



For example, Wikidata utilizes n-ary relational facts in a *hyper-relational schema,* which adds (n − 2) key-value pairs to the main triple to represent auxiliary information. In addition to the *hyper-relational schema*, the existing NKG schemas also include *event-based schema*, *role-based schema*, and *hypergraph-based schema*, as shown in Figure.

Task Definition

Formulation of NKG. An NKG $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{F}\}$ consists of an entity set $\mathcal{E}$, a relation set $\mathcal{R}$, and an n-ary fact (n≥2) set $\mathcal{F}$. Each n-ary fact $f^n \in \mathcal{F}$ consists of entities $\in \mathcal{E}$ and relations $\in \mathcal{R}$. For hyper-relational schema [20]: $f^n_{hr} = (e_1, r_1, e_2, \{r_{i-1}, e_i\}^n_{i=3})$ where $\{e_i\}^n_{i=1} \in \mathcal{E}$, $\{r_i\}^{n-1}_{i=1} \in \mathcal{R}$. For event-based schema [16]: $f^n_{ev} = (r_1, \{r_{i+1}, e_i\}^n_{i=1})$, where $\{e_i\}^n_{i=1} \in \mathcal{E}$, $\{r_i\}^{n+1}_{i=1} \in \mathcal{R}$. For role-based schema [12]: $f^n_{ro} = (\{r_i, e_i\}^n_{i=1})$, where $\{e_i\}^n_{i=1} \in \mathcal{E}$, $\{r_i\}^n_{i=1} \in \mathcal{R}$. For hypergraph-based schema [26]: $f^n_{hg} = (r_1, \{e_i\}^n_{i=1})$, where $\{e_i\}^n_{i=1} \in \mathcal{E}$, $r_1 \in \mathcal{R}$.

Problem Definition. Given an input sentence with l words $s = \{w_1, w_2, ..., w_l\}$, an entity e is a consecutive span of words: $e = \{w_p, w_{p+1}, ..., w_q\} \in \mathcal{E}_s$, where $p, q \in \{1, ..., l\}$, and $\mathcal{E}_s = \{e_j\}^m_{j=1}$ is the entity set of all m entities in the sentence. The output of n-ary relation extraction, $R()$, is a set of n-ary relational facts $\mathcal{F}_s$ in given NKG schema in $\{f^n_{hr}, f^n_{ev}, f^n_{ro}, f^n_{hg}\}$. Specifically, each n-ary relational fact $f^n \in \mathcal{F}_s$ is extracted by multi-label classification of one of the ordered span-tuple for n entities $[e_i]^n_{i=1} \in \mathcal{E}_s$, forming an answer label-list for n_r relations $[r_i]^{n_r}_{i=1} \in \mathcal{R}$, where n is the arity of the extracted n-ary relational fact, and n_r is the number of answer relations in the fact, which is determined by the given NKG schema: $R([e_i]^n_{i=1}) = [r_i]^{n-1}_{i=1}$, when $f^n = f^n_{hr}$, $R([e_i]^n_{i=1}) = [r_i]^{n+1}_{i=1}$ when $f^n = f^n_{ev}$, $R([e_i]^n_{i=1}) = [r_i]^n_{i=1}$ when $f^n = f^n_{ro}$, and $R([e_i]^n_{i=1}) = [r_1]$ when $f^n = f^n_{hg}$.

Three Main Challenges

However, there are still three main challenges in automated n-ary RE for NKG construction, which remains at a coarse-grained level:

(1) Diversity of NKG schemas.
Previous methods could only perform N-ary RE based on a specific schema, but currently, there is no flexible method that can perform n-ary RE for arbitrary schema with different number of relations.

(2) Determination of the order of entities.
N-ary RE involves more possible entity orders than binary RE, for example, as shown in Figure 2, in a *hyper-relational schema*, there is an order issue regarding which entity is the head entity, tail entity, or auxiliary entity. Previous methods often ignored the joint impact of different entity orders, leading to inaccurate extraction.

(3) Variability of the arity of n-ary RE.
Previous methods usually output a fixed number of entities and are not adept at determining the variable number of entities forming an n-ary relational fact.

2 PART

Contributions

NEURAL INFORMATION PROCESSING SYSTEMS

Contributions

To tackle these challenges, we introduce **Text2NKG**, a novel fine-grained n-ary RE framework designed to automate the generation of n-ary relational facts from natural language text for NKG construction.

- Text2NKG employs a **span-tuple multi-label classification** method, which transforms n-ary RE into a multi-label classification task for span-tuples, including all combinations of entities in the text. Because the number of predicted relation labels corresponds to the chosen NKG schema, Text2NKG is adaptable to all NKG schemas, offering examples with *hyper-relational schema*, *event-based schema*, *role-based schema*, and *hypergraph-based schema*, all of which have broad applications.

- Moreover, Text2NKG introduces a **hetero-ordered merging** method, considering the probabilities of predicted labels for different entity orders to determine the final entity order.

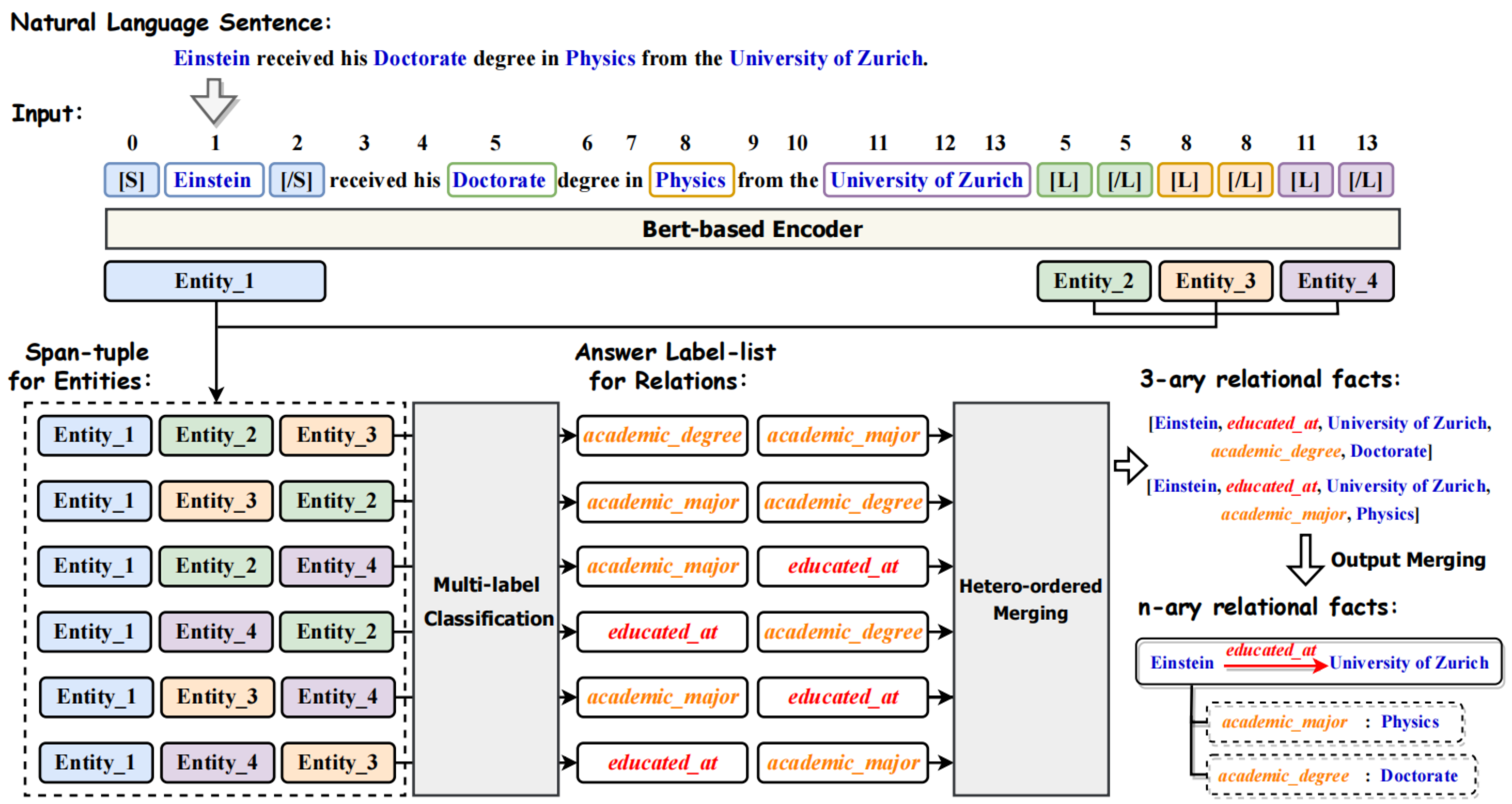
- Finally, Text2NKG proposes an **output merging** method, which is used to unsupervisedly derive n-ary relational facts of any number of entities for NKG construction.

3

P A R T

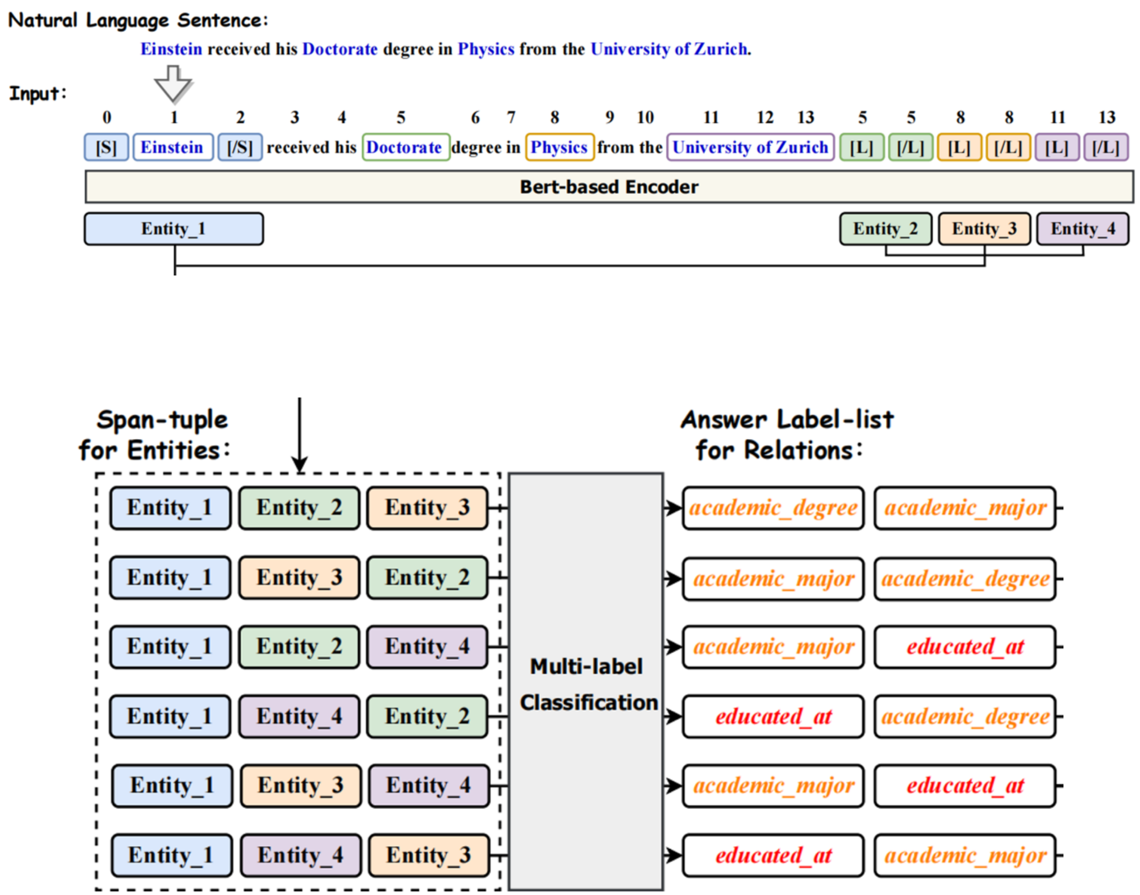
Methodology

Overview of Text2NKG



- First, Text2NKG encodes the entities using BERT-based Encoder with a packaged levitated marker for embedding. Then each arrangement of ordered span-tuple with three entity embeddings will be classified with multiple labels, and the framework will be learned by the weighted cross-entropy with a null-label bias.
- In the decoding stage, in order to filter the n-ary relational facts whose entity compositions have isomorphic hetero-ordered characteristics, Text2NKG proposes a hetero-ordered merging strategy to merge the label probabilities of 3! = 6 arrangement cases of span-tuples composed of the same entities and filter out the output 3-ary relational facts existing non-conforming relations.
- Finally, Text2NKG combines the output 3-ary relational facts to form the final n-ary relational facts with output merging.

Span-tuple Multi-label Classificaition



For the given sentence token $s = \{w_1, w_2, ..., w_l\}$ and the set of entities $\mathcal{E}_s$, in order to perform fine-grained n-ary RE, we need first to encode a span-tuple (e_1, e_2, e_3) consisting of every arrangement of three ordered entities, where $e_1, e_2, e_3 \in \mathcal{E}_s$. Due to the high time complexity of training every span-tuple as one training item, inspired by [27], we achieve the reduction of training items by using packed levitated markers that pack one training item with each entity in $\mathcal{E}_s$ separately. Specifically, in each packed training item, a pair of solid tokens, [S] and [/S], are added before and after the packed entity $e_S = \{w_{p_S}, ..., w_{q_S}\}$, and $(|\mathcal{E}_s| - 1)$ pairs of levitated markers, [L] and [/L], according to other entities in $\mathcal{E}_s$, are added with the same position embeddings as the beginning and end of their corresponding entities span $e_{L_i} = \{w_{p_{L_i}}, ..., w_{q_{L_i}}\}$ to form the input token $\mathbf{X}$:

$$\begin{aligned}\mathbf{X} =& \{w_1, ..., [S], w_{p_S}, ..., w_{q_S}, [/S], ..., \\ & w_{p_{L_i}} \cup [L], ..., w_{q_{L_i}} \cup [/L], ..., w_l\}.\end{aligned} \tag{1}$$

We encode such token by the BERT-based pre-trained model encoder [6]:

$$\{h_1, h_2, ..., h_t\} = \text{BERT}(\mathbf{X}), \tag{2}$$

where $t = |\mathbf{X}|$ is the input token length, $\{h_i\}_{i=1}^{t} \in \mathbb{R}^d$, and d is embedding size.

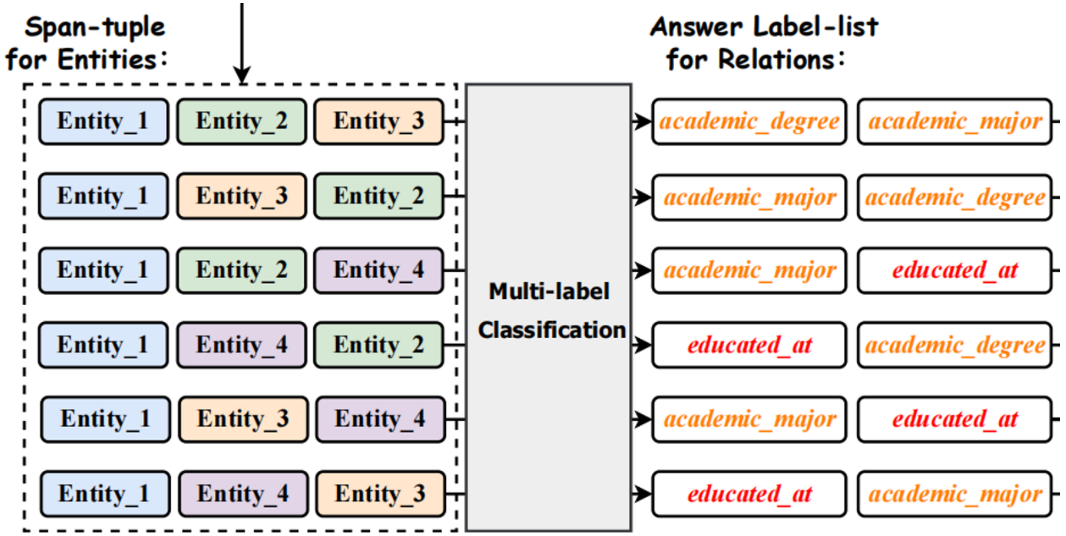
There are several span-tuples (A, B, C) in a training item. The embedding of first entity $h_A \in \mathbb{R}^{2d}$ in the span-tuple is obtained by concat embedding of the solid markers, [S] and [/S], and the embeddings of second and third entities $h_B, h_C \in \mathbb{R}^{2d}$ are obtained by concat embeddings of levitated markers, [L] and [/L] with all A_{m-1}^2 arrangement of any other two entities in $\mathcal{E}_s$. Thus, we obtain the embedding representation of the three entities to form A_{m-1}^2 span-tuples in one training item. Therefore, every input sentence contains m training items with $mA_{m-1}^2 = A_m^3$ span-tuples for any ordered arrangement of three entities.

We then define n_r linear classifiers, each of which consists of 3 feedforward neural networks $\{\text{FNN}_i^k\}_{i=1}^{n_r}, k = 1, 2, 3$, to classify the span-tuples for multiple-label classification. Each classifier targets the prediction of one relation r_i, thus obtaining a probability lists $(\mathbf{P}_i)_{i=1}^{n_r}$ with all relations in given relation set $\mathcal{R}$ plus a null-label:

$$\mathbf{P}_i = \text{FNN}_i^1(h_A) + \text{FNN}_i^2(h_B) + \text{FNN}_i^3(h_C), \tag{3}$$

where $\text{FNN}_i^k \in \mathbb{R}^{2d \times (|\mathcal{R}|+1)}$, and $\mathbf{P}_i \in \mathbb{R}^{(|\mathcal{R}|+1)}$.

Training Strategy



To train the n_r classifiers for each relation prediction more accurately, we propose a data augmentation strategy for span-tuples. Taking the *hyper-relational schema* as an example, given a hyper-relational fact (A, r_1, B, r_2, C), we consider swapping the head and tail entities, and changing the main relation to its inverse (B, r_1^{-1}, A, r_2, C), as well as swapping the tail entities with auxiliary values, and the main relation with the auxiliary key (A, r_2, C, r_1, B), also as labeled training span-tuple cases. Thus $R_{hr}(A, B, C) = (r_1, r_2)$ can be augmented with $3! = 6$ orders of span-tuples:

$$\begin{cases} R_{hr}(A,B,C) = (r_1, r_2), \\ R_{hr}(B,A,C) = (r_1^{-1}, r_2), \\ R_{hr}(A,C,B) = (r_2, r_1), \\ R_{hr}(B,C,A) = (r_2, r_1^{-1}), \\ R_{hr}(C,A,B) = (r_2^{-1}, r_1), \\ R_{hr}(C,B,A) = (r_1, r_2^{-1}). \end{cases} \tag{4}$$

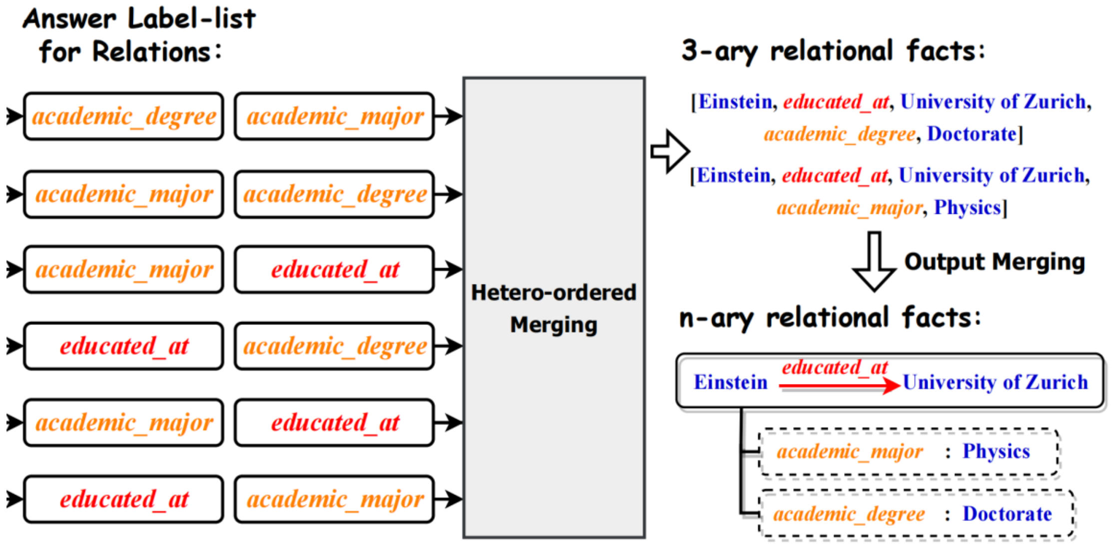
For other schemas, we can also obtain 6 fully-arranged cases of labeled span-tuples in a similar way, as described in Appendix A. If no n-ary relational fact exists between the three entities of span-tuples, then relation labels are set as null-label.

Since most cases of span-tuple are null-label, we set a weight hyperparameter $\alpha \in (0, 1]$ between the null-label and other labels to balance the learning of the null-label. We jointly trained the n_r classifiers for each relations by cross-entropy loss $\mathcal{L}$ with a null-label weight bias $\mathbf{W}_\alpha$:

$$\mathcal{L} = -\sum_{i=1}^{n_r} \mathbf{W}_\alpha \log \left(\frac{\exp(\mathbf{P}_i[r_i])}{\sum_{j=1}^{|\mathcal{R}|+1} \exp(\mathbf{P}_{ij})} \right), \tag{5}$$

where $\mathbf{W}_\alpha = [\alpha, 1.0, 1.0, ...1.0] \in \mathbb{R}^{(|\mathcal{R}|+1)}$.

Hetero-ordered Merging

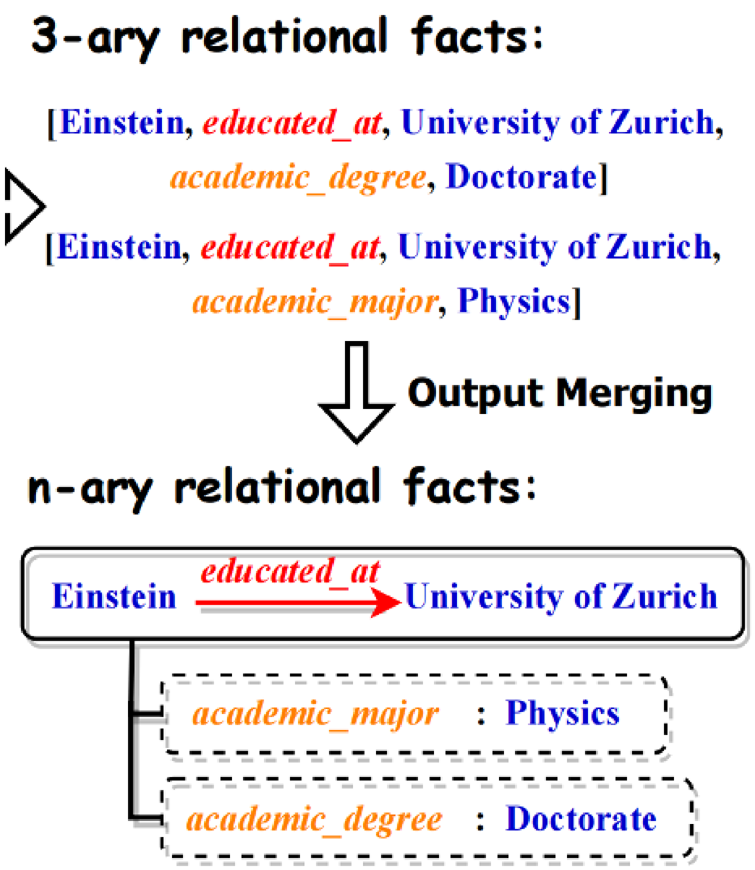


In the decoding stage, since Text2NKG labels all 6 different arrangement of the same entity composition, we design a hetero-ordered merging strategy to merge the corresponding labels of these 6 hetero-ordered span-tuples into one to generate non-repetitive n-ary relational facts unsupervisedly. For *hyper-relational schema* ($n_r = 2$), we combine the predicted probabilities of two labels $\mathbf{P}_1, \mathbf{P}_2$ in 6 orders to (A, B, C) order as follows:

$$\begin{cases} \mathbf{P}_1 = \mathbf{P}_1^{(ABC)} + I(\mathbf{P}_1^{(BAC)}) + \mathbf{P}_2^{(ACB)} \\ \qquad + I(\mathbf{P}_2^{(BCA)}) + \mathbf{P}_2^{(CAB)} + \mathbf{P}_1^{(CBA)}, \\ \mathbf{P}_2 = \mathbf{P}_2^{(ABC)} + \mathbf{P}_2^{(BAC)} + \mathbf{P}_1^{(ACB)} \\ \qquad + \mathbf{P}_1^{(BCA)} + I(\mathbf{P}_1^{(CAB)}) + I(\mathbf{P}_2^{(CBA)}), \end{cases} \tag{6}$$

where $I()$ is a function for swapping the predicted probability of relations and the corresponding inverse relations. Then, we take the maximum probability to obtain labels r_1, r_2, forming a 3-ary relational fact (A, r_1, B, r_2, C) and filter it out if there are null-label in (r_1, r_2). If there are inverse relation labels in (r_1, r_2), we can also transform the order of entities and relations as equation 4. For *event-based schema*, *role-based schema*, and *hypergraph-based schema*, all can be generated by hetero-ordered merging according to this idea, as shown in Appendix B.

Output Merging



After hetero-ordered merging, we merge the output 3-ary relational facts to form higher-arity facts, with *hyper-relational schema* based on the same main triple, *event-based schema* based on the same main relation (event type), *role-based schema* based on the same key-value pairs, and *hypergraph based schema* based on the same hyperedge relation. This way, we can unsupervisedly obtain n-ary relational facts with dynamic number of arity numbers for NKG construction.

4

P A R T

Experimental Results

NEURAL INFORMATION PROCESSING SYSTEMS

Main Results

Model	PLM	*hyper-relational schema* / Dev			*hyper-relational schema* / Test		
		Precision	Recall	F_1	Precision	Recall	F_1
Unsupervised Method							
ChatGPT	gpt-3.5-turbo	12.0583	11.2764	11.6542	11.4021	10.9134	11.1524
GPT-4	gpt-4	15.7324	15.2377	15.4811	15.8187	15.4824	15.6487
Supervised Method							
Generative Baseline	BERT-base (110M)	63.79 ± 0.27	59.94 ± 0.68	61.80 ± 0.37	64.60 ± 0.47	59.67 ± 0.35	62.03 ± 0.21
Pipelinge Baseline	BERT-base (110M)	69.23 ± 0.30	58.21 ± 0.57	63.24 ± 0.44	69.00 ± 0.48	57.55 ± 0.19	62.75 ± 0.29
CubeRE	BERT-base (110M)	66.14 ± 0.88	64.39 ± 1.23	65.23 ± 0.82	65.82 ± 0.84	64.28 ± 0.25	65.04 ± 0.29
Text2NKG w/o DA	BERT-base (110M)	76.02 ± 0.50	72.28 ± 0.68	74.10 ± 0.55	73.55 ± 0.81	70.63 ± 1.40	72.06 ± 0.34
Text2NKG w/o α	BERT-base (110M)	88.77 ± 0.85	78.39 ± 0.47	83.26 ± 0.70	88.09 ± 0.69	76.64 ± 0.45	81.97 ± 0.58
Text2NKG w/o HM	BERT-base (110M)	61.74 ± 0.34	76.97 ± 0.44	68.52 ± 0.69	61.07 ± 0.73	76.16 ± 0.59	67.72 ± 0.48
Text2NKG (ours)	BERT-base (110M)	**91.26 ± 0.69**	**79.36 ± 0.51**	**84.89 ± 0.44**	**90.77 ± 0.60**	**77.53 ± 0.32**	**83.63 ± 0.63**
Generative Baseline	BERT-large (340M)	67.08 ± 0.49	65.73 ± 0.78	66.40 ± 0.47	67.17 ± 0.40	64.56 ± 0.58	65.84 ± 0.25
Pipelinge Baseline	BERT-large (340M)	70.58 ± 0.78	66.58 ± 0.66	68.52 ± 0.32	69.21 ± 0.55	64.27 ± 0.24	66.65 ± 0.28
CubeRE	BERT-large (340M)	68.75 ± 0.82	68.88 ± 1.03	68.81 ± 0.46	66.39 ± 0.96	67.12 ± 0.69	66.75 ± 0.28
Text2NKG (ours)	BERT-large (340M)	**91.90 ± 0.79**	**79.43 ± 0.42**	**85.21 ± 0.69**	**91.06 ± 0.81**	**77.64 ± 0.46**	**83.81 ± 0.54**

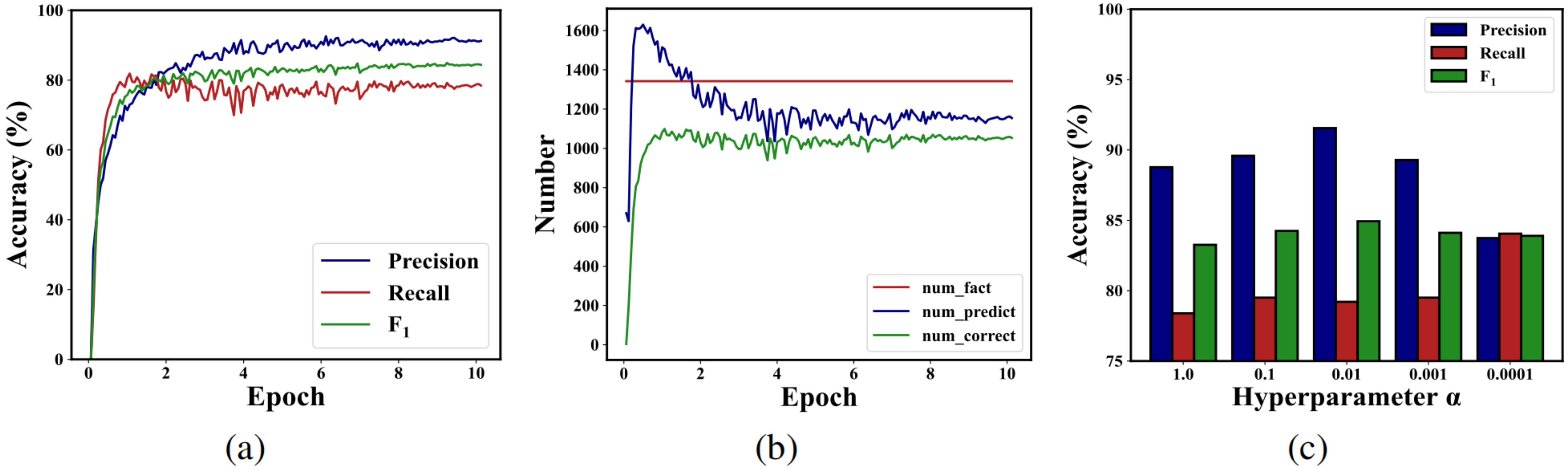
Comparison of Text2NKG with other baselines in the hyper-relational extraction on HyperRED. Results of the supervised baseline models are mainly taken from the original paper. The best results in each metric are in **bold**.

Results on Various NKG Schemas

Model	**PLM**	*event-based schema*			*role-based schema*			*hypergraph-based schema*		
		Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
Unsupervised Method										
ChatGPT	gpt-3.5-turbo	10.4678	11.1628	10.8041	11.4387	10.4203	10.9058	11.2998	11.7852	11.5373
GPT-4	gpt-4	13.3681	14.6701	13.9888	13.6397	12.5355	13.0643	13.0907	13.6701	13.3741
Supervised Method										
Text2Event	T5-base (220M)	73.94 ± 0.76	70.56 ± 0.58	72.21 ± 1.25	72.73 ± 0.79	68.45 ± 1.34	70.52 ± 0.62	73.68 ± 0.88	70.37 ± 0.51	71.98 ± 0.92
UIE	T5-base (220M)	76.51 ± 0.28	73.02 ± 0.66	74.72 ± 0.18	72.17 ± 0.29	69.84 ± 0.11	70.98 ± 0.31	72.03 ± 0.41	68.74 ± 0.13	70.34 ± 1.07
LasUIE	T5-base (220M)	79.62 ± 0.27	78.04 ± 0.75	78.82 ± 0.26	77.01 ± 0.20	74.26 ± 0.25	75.61 ± 0.24	76.21 ± 0.07	73.75 ± 0.17	74.96 ± 0.42
Text2NKG	BERT-base (110M)	**86.20 ± 0.57**	**79.25 ± 0.33**	**82.58 ± 0.20**	**86.72 ± 0.80**	**78.94 ± 0.59**	**82.64 ± 0.38**	**83.53 ± 1.18**	**86.59 ± 0.38**	**85.03 ± 0.86**
Text2Event	T5-large (770M)	75.58 ± 0.53	72.39 ± 0.82	73.97 ± 1.19	73.21 ± 0.45	70.85 ± 0.67	72.01 ± 0.31	75.28 ± 0.93	72.73 ± 1.07	73.98 ± 0.49
UIE	T5-large (770M)	79.38 ± 0.28	74.69 ± 0.61	76.96 ± 0.95	74.47 ± 1.42	71.84 ± 0.77	73.14 ± 0.38	74.57 ± 0.64	71.93 ± 0.86	73.22 ± 0.19
LasUIE	T5-large (770M)	81.29 ± 0.83	79.54 ± 0.26	80.40 ± 0.65	79.37 ± 0.92	76.63 ± 0.44	77.97 ± 0.76	77.49 ± 0.35	74.96 ± 0.60	76.20 ± 0.87
Text2NKG	BERT-large (340M)	**88.47 ± 0.95**	**80.30 ± 0.75**	**84.19 ± 1.29**	**86.87 ± 0.87**	**80.86 ± 0.29**	**83.76 ± 1.17**	**85.06 ± 0.33**	**86.72 ± 0.36**	**85.89 ± 0.69**

Comparison of Text2NKG with other baselines in the n-ary RE in event-based, role-based, and *hypergraph-based schemas* on HyperRED. The best results in each metric are in **bold**.

Ablation Study



Ablation results.

(a) Precision, Recall, and $F1$ changes in the dev set during the training of Text2NKG.

(b) The changes of the number of true facts, the number of predicted facts, and the number of predicted accurate facts during the training of Text2NKG.

(c) Precision, Recall, and $F1$ results on different null-label hyperparameter (α) settings.

Analysis of N-ary Relation Extraction in Different Arity

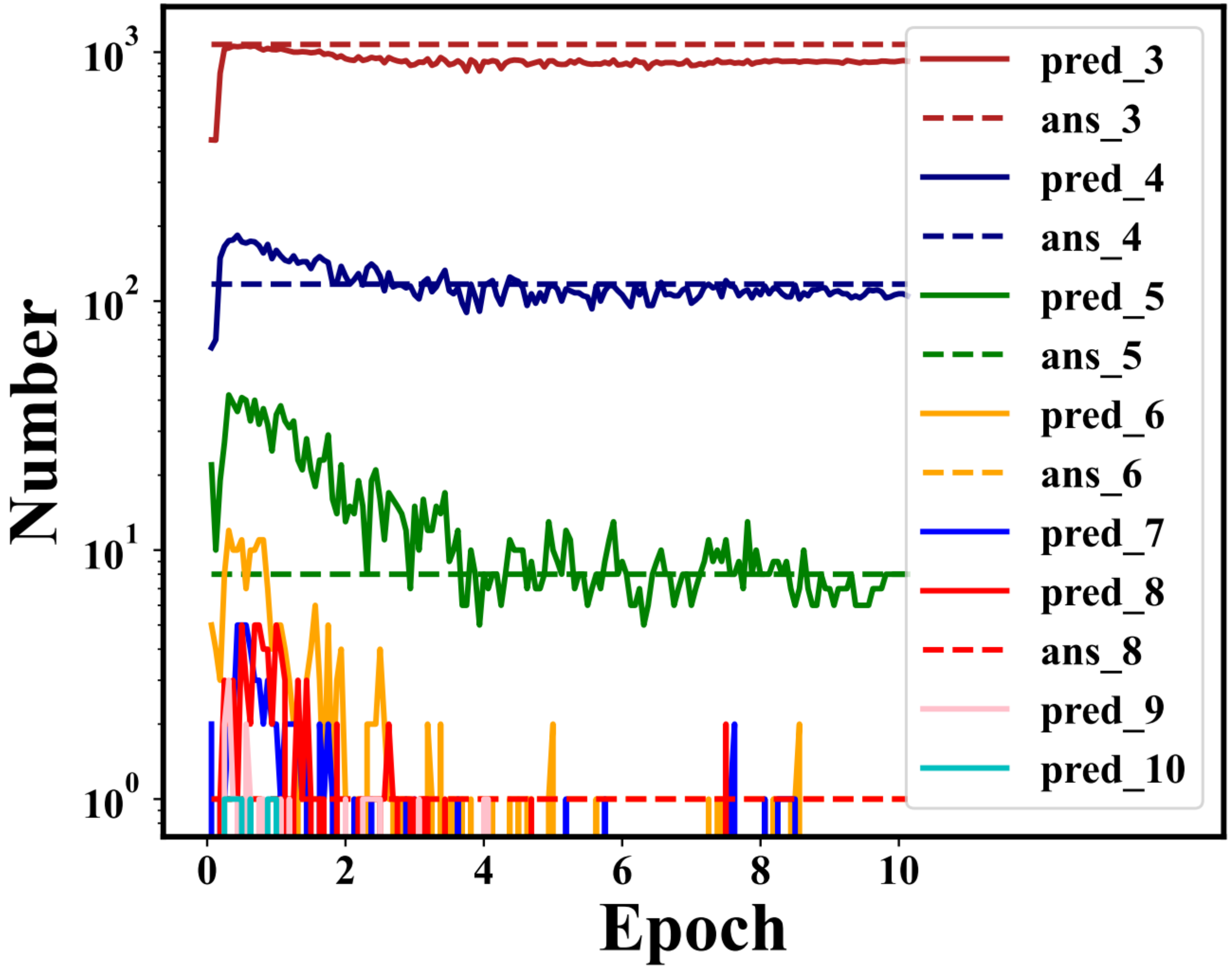


Figure shows the number of n-ary relational facts extracted after output merging and the number of the answer facts in different arity during training of Text2NKG on the dev set. We find that, as the training proceeds, the final output of Text2NKG converges to the correct answer in terms of the number of complete n-ary relational facts in each arity, achieving implementation of n-ary RE in indefinite arity unsupervised, with good scalability.

Case Study

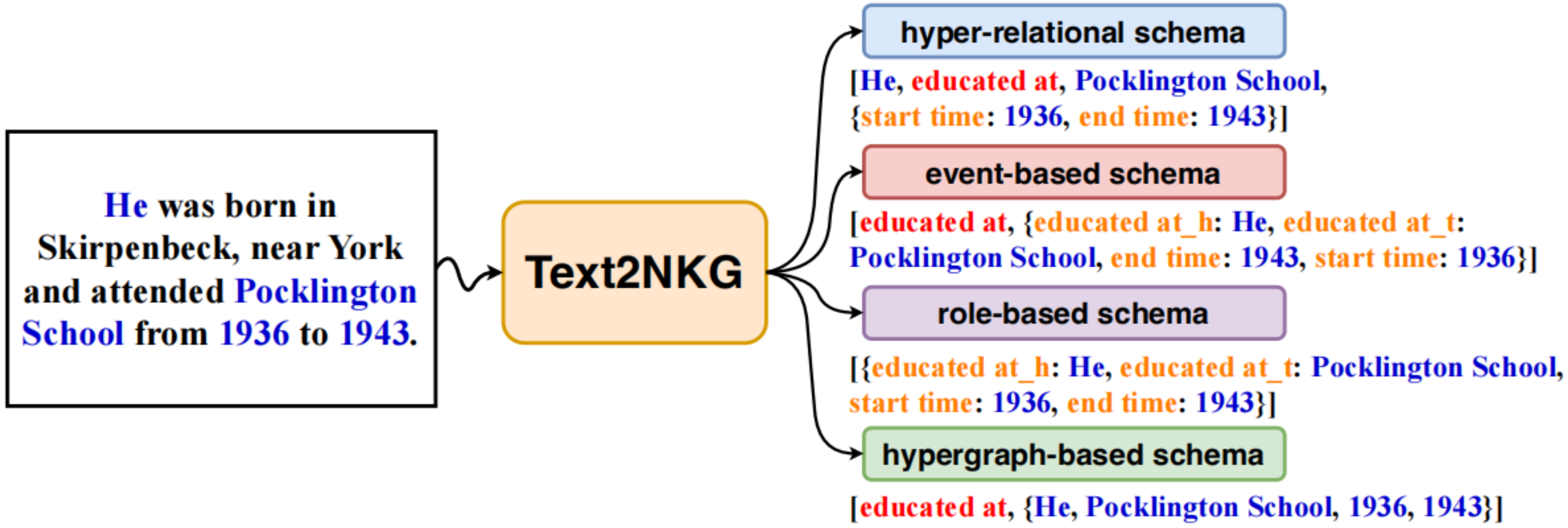


Figure shows a case study of n-ary RE by a trained Text2NKG. For a sentence, "He was born in Skirpenbeck, near York and attended Pocklin.", four structured n-ary RE can be obtained by Text2NKG according to the requirements. Taking the *hyper-relational schema* for an example, Text2NKG can successfully extract one n-ary relational fact consisting of a main triple [He, educated at, Pocklington], and two auxiliary key-value pairs {start time:1936}, {end time:1943}. This intuitively validates the practical performance of Text2NKG on fine-grained n-ary RE to better contribute to NKG construction.

The Thirty-Eighth Annual Conference on Neural Information Processing Systems

Vancouver Convention Center
Tuesday Dec 10 through Sunday Dec 15

NeurIPS 2024 Main Conference

Thanks for Listening

Code & Data https://github.com/LHRLAB/Text2NKG

Contact luohaoran@bupt.edu.cn