World Wide Web (2022) 25:1403–1423
https://doi.org/10.1007/s11280-021-00932-0

Knowledge based natural answer generation via masked-graph transformer

Xiangyu Li[1] · Sen Hu[1,2] · Lei Zou[1,3]

Received: 16 March 2021 / Revised: 25 June 2021 / Accepted: 20 July 2021 /
Published online: 26 October 2021

Abstract

Natural Answer Generation on Knowledge Base (NAG-KB), which generates natural answer sentences for the given question, has received much attention in recent years. Compared with traditional QA systems, NAG could offer specific entities fluently and naturally, which is more user-friendly in the real world. However, existing NAG systems usually utilize simple retrieval and embedding mechanism, which is hard to tackle complex questions. They suffer issues containing knowledge insufficiency, entity ambiguity, and especially poor expressiveness during generation. To address these challenges, we propose an improved knowledge extractor containing post disambiguation and simplifying strategy to retrieve supporting graphs from KB, an masked-graph transformer to encode the supporting graph, which introduce special vertex setting, communication path calculation and mask mechanism. Moreover we design a multi-task training combining classification and sequence decoding jointly. In summary, we propose a framework called G-NAG in this paper, including a knowledge extractor, an incorporating encoder, and an multi-task generator. Experimental results on two complex QA datasets demonstrate the efficiency of G-NAG compared with state-of-the-art NAG systems and transformer baselines.

Keywords Question answering · Natural answer generation · Graph attention network · Mask mechanism

This article belongs to the Topical Collection: *Special Issue on Graph Data Management, Mining, and Applications*
Guest Editors: Xin Wang, Rui Zhang, and Young-Koo Lee

✉ Lei Zou
zoulei@pku.edu.cn

Xiangyu Li
xiangyu_li@pku.edu.cn

Sen Hu
husen@pku.edu.cn

1 Peking University, Beijing, China

2 Ant Group, Beijing, China

3 ChongQing Research Institute of Big Data, Peking University, Beijing, China

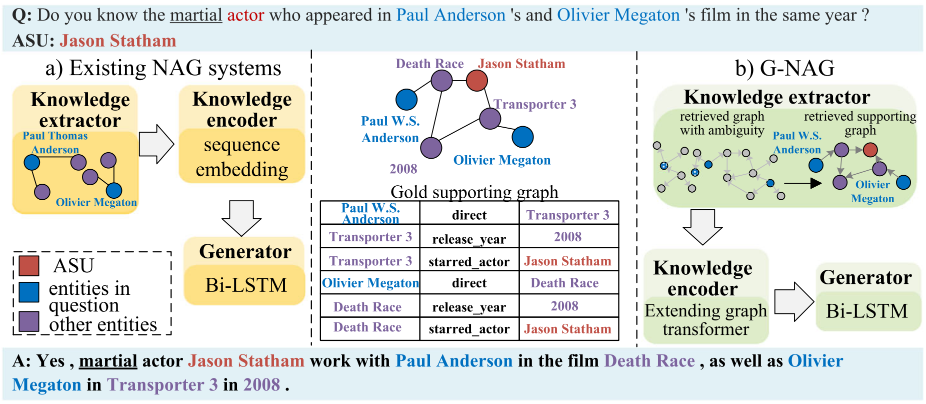
1 Introduction

Natural Answer Generation on Knowledge Base (NAG-KB), which devotes to providing fluent answers in the form of natural language sentences based on KB, has received much attention in recent years. Compared with traditional question answering (QA) systems that merely offer accurate Answer Semantic Units (ASU) [11], NAG could satisfy users in real-world scenarios where fluency is of strong demand.

Generally, the popular NAG framework consists of three modules, as shown in Figure 1-a. Knowledge extractor recognizes the topic entity and retrieves its related triples from the underlying KB. After Knowledge encoder representing these candidate triples and the question as two sequences, Generator could generate the natural answer with an attention mechanism. Existing NAG systems [6, 11, 32] have achieved some success focused on simple question (only one entity or only one relation).

However, there are still many non-trivial issues due to linguistic complexity that the above systems do not perform well. (1) In Knowledge extractor. On the one hand, existing NAG systems recognize one topic entity and retrieve its one-hop neighbors related to the question. When a question contains more entities and multi-hop relations, they may leave out some critical entities. Take Q in Figure 1 as an example, the ASU `Jason Statham` should be retrieved through 2-hops from the mentioned entities in question so that it may be left by previous knowledge extractor with one-hop retrieval mechanism. On the other hand, without considering the global structure in KB, the above systems do disambiguation before retrieving triples. Thus, they may choose irrelevant entities far apart from others, such as `Paul Thomas Anderson` which may be confused with the correct entity `Paul W.S. Anderson` but unrelated to the question in Figure 1-a. (2) In Knowledge encoder. Previous NAG systems encode triples as a sequence, such as a list by LSTM [11, 32] or key-value structure by Memory Network [6], which is too simple to express complicated semantic information. For the same example, triple-list or key-value could not represent the topological structure of the supporting graph clearly, which is the key to generate answers logically.

We focus on these challenges above and propose some novel solutions. (1) In Knowledge extractor, we design a two-channel retrieval scheme to deal with simple and complex QA conditions. Then we propose a novel extract mechanism containing multi-hop retrieval,


Q: Do you know the <u>martial</u> actor who appeared in Paul Anderson 's and Olivier Megaton 's film in the same year ?

ASU: Jason Statham

Paul W.S. Anderson	direct	Transporter 3
Transporter 3	release_year	2008
Transporter 3	starred_actor	Jason Statham
Olivier Megaton	direct	Death Race
Death Race	release_year	2008
Death Race	starred_actor	Jason Statham

A: Yes , <u>martial</u> actor Jason Statham work with Paul Anderson in the film Death Race , as well as Olivier Megaton in Transporter 3 in 2008 .

Figure 1 Natural Answer Generation process of an example question

post disambiguation based on global structure and three-stage simplifying method based on semantic similarity, as shown in Figure 1-b. The proposed mechanism solves the triple missing problem, ambiguity and information redundancy (2) In Knowledge encoder. Since graph neural network is proposed in Graph2Seq task like KB based summarization [14] and achieves excellent performance, we propose a masked-graph transformer as encoder, which has more capacity to encode complicated pair-wise relationships than the sequence structure. To fit the NAG problem, we introduce the communication path and special vertices to capture global or variable information, respectively (to be discussed in Section 4.2). (3) In Generator. To improve the accuracy of our system, we train question type classification task with answer decoding jointly. We define two kinds of approach to distinguish KB questions which will be discussed in Section 5.

In this paper, we propose a framework called G-NAG (Graph-based NAG) to implement the generation process, which also consists of three modules, as shown in Figure 1-b. Compared with previous work, we enlarge the retrieval range before disambiguation and then propose a simplifying strategy based on semantic similarity in Knowledge extractor. Moreover, we replace the sequence encoder with a graph transformer considering communication path as well as global and variate vertices. Based on `Wikimovie` dataset [20] and `DBpedia`, we reconstruct a QA dataset in movie domain aimed at multi-hop question answering. Experimental results on the original `Wikimovie` and new dataset demonstrate the efficiency of our model compared with state-of-the-art NAG systems and transformer baselines.

We summarize our main contributions of this paper as follow:

- We design a generation framework G-NAG, which generates natural and logical answer based on KB. To our knowledge, it is the first framework that aims at addressing complex NAG problem.
- We present a novel knowledge extractor which distinguish different question channels, enlarges retrieval range before disambiguation and simplifies triples based on global information and semantic similarity to gain the supporting graph.
- We propose an masked-graph transformer to represent the supporting graph in QA condition, which considers the communication path and captures global or variable information by extra vertices.
- We design a multi-task training combining question type classification and answer sentence decoding jointly, and propose two approach to distinguish question type in KBQA.
- We implement experiments on two KBQA datasets in the movie domain. The results demonstrate that G-NAG performs better compared with three groups of baselines, especially in complex natural answer generation problems.

2 Question definition and methodology framework

2.1 Question definition

In this section, we firstly introduce the notations employed in this paper and provide some important definition.

Definition 1 NAG(Natural Answer Generation)

Given a natural language question Q, its accurate Answer Semantic Units as ASU and specific RDF knowledge base D. The knowledge triples in D are in the form $\langle s, p, o \rangle$, where s, o are entity vertices (*ent*) and p is relation edge (*rel*). NAG aims at generating target natural language answer which is denoted as A.

Specifically, we focus more on the question that only has one type of ASU, i.e., for a question based on a movie domain knowledge base, the ASU maybe two actors but not an actor and a writer.

Definition 2 Supporting graph
Given a specific RDF knowledge base D and a natural language question Q. If there exist a sub-graph G having the same semantic meaning of Q, that is the ASU can in inferred by G with the same reasoning process, we refer to G as supporting graph of G.

In G-NAG, the supporting graph is the output of Knowledge Extractor. Besides, there is some pivotal intermediate results in different stage, we provides their definition and notation following.

Initial retrieved graph In Knowledge Extractor, the initial graph retrieved by multi-hop search is defined as a inter-connected graph set $\mathbb{G} = [G_i = (V_q, V_o, E_i)]$, where vertex $v \in V_q$ is mentioned in question, $v \in V_o$ denotes other retrieved vertex, and E_i is a set of relation edges that link vertices. After disambiguation and graph simplifying, the final supporting graph is denoted as G.

Unlabeled graph In encoding section, G is converted to an unlabeled graph $G' = (V', P')$, where V' is a set of all vertices and P' is a matrix describing communication path among vertices.

2.2 Framework overview

2.3 Framework

Our G-NAG framework [15] consists of three modules: Knowledge Extractor, Incorporating Encoder, and Generator. We depict an overview with a concrete example in Figure 2.

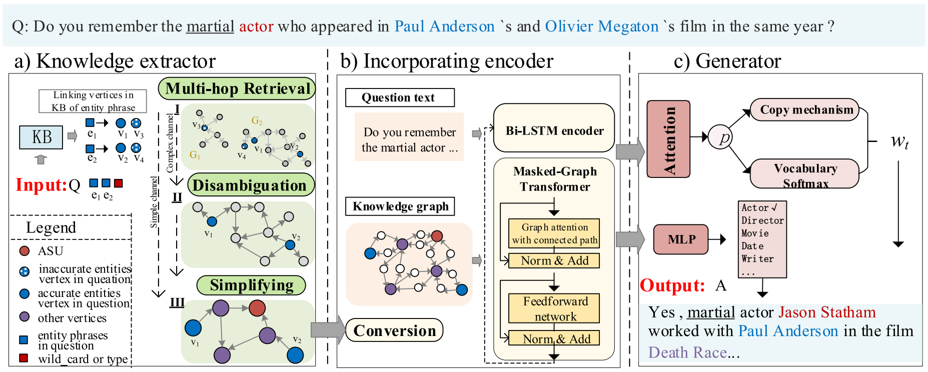


Figure 2 Natural Answer Generation process of an example question

Knowledge Extractor: Given the question Q and underlying KB D, G-NAG distinguishes between simple and complex conditions and maps each entity phrase to its candidate linking vertices $v \in V_q$. For complex channel, employ k-hop retrieval and global-based ambiguity as as illustrated in Figure 2 a-I and a-II. For another channel, only pass one-hop retrieval and similarity-based ambiguity described in Section 3.1. Further, G-NAG removes redundant vertices and edges by semantic similarity to acquire a simplified supporting graph G as shown in Figure 2 a-III.

Incorporating Encoder: G-NAG propose Masked-Graph Transformer (to be discussed in Section 4.2) to encode supporting graph obtained from Knowledge Extractor and fuse the graph network outputs with question embedding (to be discussed in Section 4.3) to express Q, so we regard it as an Incorporating Encoder, as shown in Figure 2-b. Besides, we define a graph conversion operation to set special vertex and communication path.

Generator: G-NAG designs a multi-task training approach which combines sequence prediction and question type classification. On the one hand, the Generator distinguishes question type based on graph embedding. On the other hand, it predicts output word w_t at each time step t by generating from vocabulary or copying from supporting graph G and question Q via a soft switch p. As illustrated in Figure 2-c, token with underline is copied from question text, and the colored token is from the graph, while other ordinary words are generated from the vocabulary.

3 knowledge extractor

We propose an improved knowledge extractor in this section to provide a more accurate supporting graph. Specifically, we enlarge the extraction range by multi-hop retrieval before entity disambiguation, then solve the ambiguity based on the global graph structure, and simplify the graph by semantic similarity eventually.

3.1 Multi-hop retrieval

Given question Q and RDF KB D, we employ entity identification to acquire entity phrase list $E = [ent_i]$. Then we map each entity phrase ent_i to its candidate linking vertices $v \in V_q$ in KB, while a entity phrase $ent1$ may be matched more than one vertex as a set, such as $\mathbb{V}_{ent1} = [v_i^1] \subseteq V_q$.

To avoid unnecessary retrieving, a two-channel retrieval scheme is set up in G-NAG system. Channels that solve simple problems have two opening conditions:

– List E has length 1 which means only one core entity existing in question.
– List E has length 2 and there is a triple in D that perfectly matches entities, namely $< s, p, o >$ corresponds to $< ent_1, rel, ent_2 >$ while rel also appears in Q.

Simple questions assigned to this channel require only one hop retrieval. Then do ambiguity and simplifying in one step based on sequence semantic similarity calculation as Eq. 2 to obtain supporting graph G, where T denotes text label in each vertex that faces filtering.

For another channel which belongs to complex questions, allowing for the ambiguity of phrase linking temporarily, we retrieve k-hop neighbors of each linking vertex to construct a large graph. As some linking vertices are far apart in KB, the large graph could be divided into a graph set $\mathbb{G} = [G_i = (V_q, V_o, E_i)]$ where G_i are unconnected to each other, V_q denotes linking vertices for entity phrase in question and V_o denotes the other vertices retrieved. Factual questions usually have only a group of core multi-hop relationships, that is, the distances between exact entities are all within a fixed number of hops, so target entities are rarely distributed on two completely different graphs G_i.

3.2 Entity disambiguation

In this stage, G-NAG deals with the ambiguous vertices in V_q. To ensure the integrity of the supporting graph, we remain at most M graphs in $\mathbb{G}$ with more linking entity phrases (Assume n graphs cover all entity phrases and denote m as a parameter). Then considering one of the remaining graphs G_i, we compute its cost motivated by [30] formulated as follow:

$$M = max(m, n), \quad Cost_{G_i} = \sum_{(s,p,o)\in G_i} \|s + p - o\|_2^2 \tag{1}$$

Because of the cumulative effect of error, the candidate G with the minimum cost could be selected with the strongest internal relevance.

Definition 3 Distance on graph for vertex
Given a supporting graph $G = [\mathbb{V}_{ent_1}, \mathbb{V}_{ent_2}, ..., \mathbb{V}_{ent_n}]$ with ambiguous vertices and a vertex v on G, where $\mathbb{V}_{ent_1} = [v_i, v_{i+1}, ..., v_j]$ only containing one correct entity vertex. The minimum-distance from v to $\mathbb{V}_{ent_1}$ is the shortest path between v and any vertex in $\mathbb{V}_{ent_1}$. And the distance on graph G for v is the minimum-distance sum within 1 to n.

Moreover, we propose a minimum-distance method to delete redundant linking vertices in G for each entity phrase. Take $\mathbb{V}_{ent_1} = [v_1, v_2]$ in Figure 3-a as an example, we define the shortest path (number of edges) between v_1 and each vertex of $\mathbb{V}_{ent_2}$ as the minimum-distance between v_1 and $\mathbb{V}_{ent_2}$. Then we rank vertices v in the same $\mathbb{V}_{ent_j}$ according to the minimum-distance sum of v and other $\mathbb{V}_{ent_J}$. Further, we only keep the vertex with the minimum sum in each $\mathbb{V}$.

3.3 Graph simplifying

In this stage, G-NAG deletes redundant vertices in V_o. For each vertex $v \in V_o$ in graph, we keep it if there exists a communication path between two linking vertices $v_i, v_j \in V_q$ containing it. In other words, we remove $v \in V_o$ only related to one entity phrase, which means a weak correlation with Q. Here, we regard two vertices as isomorphic if they share the same neighborhood and connect every common neighbor vertex with edges of the same relation. Then we merge isomorphic vertices and concatenate their text attributions as one vertex.

G-NAG further deletes redundant vertices in V_o using aggregated semantic similarities based on word embedding [10]. For this step, we only consider the alternative vertex $v \in V_o$ that could be removed without affecting the connectivity of the graph. Specifically, for each vertex, we concatenate triples containing it as a word sequence T, then use Word2Vec [19] to compute string similarities between T and question Q following [25]. where, w represents a word of the string, and the average is used as the aggregation functions. Finally, we keep the top-k alternative vertices in V_o with a higher score.

$$Similarity(Q, T) = Agg\ cos(w_Q, w_T) \tag{2}$$

Different from existing NAG systems, which match triples with the question directly, G-NAG performs multi-hop retrieval in entity-level without considering relation phrases, then simplifies the graph based on semantic similarity. This strategy allows G-NAG to handle implicit relations, where predicates are missed in question, more effectively.

4 Incorporating encoder

The encoder receives discrete inputs (question text and supporting graph defined before) and encodes them into numerical representations jointly [3], to accomplish neural network feeding. For the supporting graph input, we define the graph conversion operation containing extra vertices adding. Then we employ a novel Masked-Graph Transformer to update the vertex representations by aggregating information from their neighbors. Then we learn the question embedding based on the output of Masked-Graph Transformer by bi-LSTM.

4.1 Graph conversion

Inspired by [2, 14], we convert the extracted graph to an unlabeled bipartite graph. Specifically, we replace each relation edge with two vertices, where one represents the forward direction of the relation and the other represents the reverse. The key difference with the above work is G-NAG introduces two extra vertices v_v, v_g to capture effective information.

Definition 4 Global Vertex and Variate Vertex

Given a Knowledge graph G related to a QA pair, while some vertices have been mentioned in question denoted as $v \in V_q$ and the other have been not denoted as $v \in V_o$. Define a extra vertex which is connected to $v \in V_o$ as variate vertex. Then define another extra vertex having edges to variate vertex and $v \in V_q$ as global vertex.

Specifically, global vertex is set to update the global state following the reasoning perspective as humans, while variate vertex concerns more on retrieved vertices $v \in V_o$ especially ASU, which is of vital importance for the generation. If the question type is known or can be extracted, the global vertex's text label is the type recognized by knowledge extractor, i.e., actor. Otherwise, the global vertex would has a wild-card as text label, i.e., who, what, where, etc.

As shown in Figure 3-b, global vertex v_g, which concentrates information via two mentioned vertices and the variate vertex by yellow edges, could reach all vertices in G', while variate vertex connects $v \in V_o$ by blue edges. The conversion result is $G' = (V', P')$, where V' is a vertex set and P' is a matrix storing communication path between every vertices-pair.

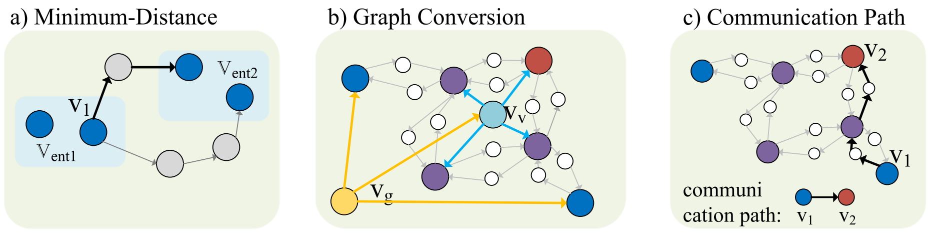


Figure 3 Example of minimum distance, graph conversion and communication path

Take vertices pair v_1, v_2 as an example in Figure 3-c, a sequence of vertex text attribution along the path from v_1 to v_2 expresses the communication path. Note we choose the shortest path between vertices-pair (numbers of edges) and adopt embedding average when two or more equal length paths exist.

During vertex embedding updating, the global vertex vector represents the whole subgraph embedding. When we take out it in ablation studies, we apply average function on all vertexs to represent the sub-graph.

4.2 Masked-graph transformer

Motivation Our G-NAG deals with a Graph2Seq task that converts a knowledge graph to a sentence, which is similar to the knowledge-based scientific summarization proposed by [14]. Since graph encoder pay an important role in Graph2Seq task and GAT(Graph Attention Network) successfully presents knowledge graph in many studies, i.e., the graph Transformer called Graph Writer in [14], we design the Masked-Graph Transformer based on GraphWriter. Here, we analyze the differences between our task and GraphWriter, which is also challenges for knowledge encoding.

– In NAG, vertices of supporting graph could be divided into mentioned vertices $v \in V_q$ and other retrieved vertices $v \in V_o$, while the former linked by entity phrases extracted directly from question but the latter may be ASU or supplementary information that could be omitted.

– In NAG, there may be redundant vertices, although the knowledge extractor tries to filter them out. While it is more like a translation problem in GraphWriter, so all vertices are necessary for sentence decoding.

– There are closer relationships between entities than that of a summarization since the supporting graph is for a specific question in NAG, not for a broad scientific topic.

In traditional Transformer, every two vertices are visible to each other, so it employ self-attention calculation between every vertex pairs. While in GAT or Graph Transformer, only one-hop neighbors are in the scope of calculation. Since we introduce communication path into our graph encoder, we attempts to characterize finer-grained relationships between vertices not limited to one-hop neighbors. However, consider every two vertices and their communication means high time complexity and some unnecessary calculations. So we propose masked mechanism motivated by GNNExplainer [33], that is for each vertex, the other vertices in graph can be divided into visible vertices and masked vertices

Masked-graph transformer Therefore, we propose masked-graph transformer as following, which introduces communication path and masked-mechanism. Specifically, we design three methods based on the following definitions. One designs mask mechanism according to the position of virtual vertexs, while the visibility between vertexs decides different computational formula during embedding updating. The second is optimized on the basis of the former, in which we set a learned parameter to control the masking degree. The third approach designs another mask mechanism based on semantics of sub-graph, in other words, semantically related vertexs are visible to each other. Following we give the definition of communication path and mask mechanism in the first two methods.

Definition 5 Communication Path

Given a knowledge graph G extracted from a RDF knowledge base D, every two vertex v_i and v_j have at least one connected path. For one of them, arrange the text labels of the vertex on the path as sequence $p = [p_i]$, that is communication path between v_i and v_j.

Definition 6 Masked Vertex

Given a knowledge graph G extracted from a RDF knowledge base D, For global vertex as well as variate vertex, only one-hop neighbors are visible since these edge,having no labels, are special. For vertices $v \in V_q$, the vertex over the whole supporting graph are visible, while for vertices $v \in V_o$, other retrieved vertices $v \in V_o$ are masked and other question mentioned vertices $v \in V_q$ are visible. Noticed one-hop neighbors are visible in any case (Figure 4).

Based on Communication path and Masked vertex mechanism, we design the Masked-Graph Transformer to encode supporting graph. We employ the common Transformer architecture expressed in Figure 2b, which is composed of a stack of D = 6 identical layers. each layer consists of a self-attention mechanism and feed-forward network, both around by a residual connection.

Initially, the text attribution of vertices in G is embedded as $V = [v_i]$, $v_i \in R^d$ in a dense continuous space using bi-LSTM, which would be the input of graph encoder. Same as the typical transformer, each vertex has 3 vector representations **q**, **k**, **v**, which means query, key, value respectively.

Then we acquire the vertex embedding by masked graph transformer algorithm proposed following.

Finally, we adopt the conventional transform architecture, the final representation of vertices is denoted as $V^D = [v^D]$. To present the support graph, we employ two methods of average pooling and global vertex representation. The former adds a pooling layer to calculate all vertices by average function, while the latter regard the global vertex's embedding as graph embedding.

4.2.1 Algorithm 1

Firstly, for visible vertex pair v_i, v_j in graph that has multi-hop relations, we encode their communication path representation described following into d-size dimension space and add it to vertex v_j's **k**(*key*) vector for calculation. Then, we represent v_i as the weighted sum of all visible vertices' **v**(*value*) vectors with the consideration of communication path, formulated as follow:

$$\hat{v}_i = \Big\Vert_{n=1}^{N} \sum_{j \in V_{visible}} \alpha_{ij}^n W_V^n v_j, \ \ where\ a_{ij}^n = \frac{exp((\mathbf{W_k k_j} + \mathbf{W_R r_{ij}})^\top W_Q q_i)}{\sum_{z \in V} exp((\mathbf{W_k k_z} + \mathbf{W_R r_{iz}})^\top W_Q q_i)} \tag{3}$$

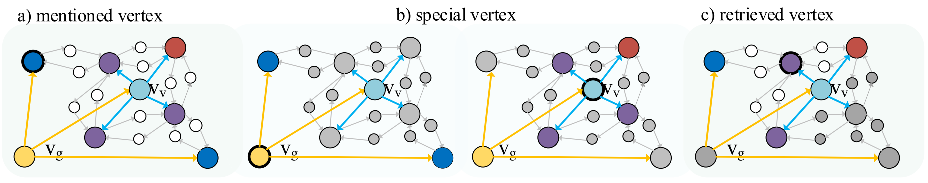


Figure 4 Masked mechanism defined in supporting graph

where $\|$ represents concatenation, α_{ij}^{n} is normalized attention coefficient computed by self-attention mechanism per head, and W_V^n is transformation's weight matrix of $\mathbf{v}$(*value*). For each attention function α, W_K, W_Q are transformation's weight matrix of $\mathbf{k}$ (*key*) and $\mathbf{q}$ (*query*), where r_{ij} denotes the embedded communication path between v_i, v_j and $W_R \in R^{d*d}$ is a parameter matrix of $\mathbf{r}$.

Secondly, we compute extra vertices' representation over neighbor vertices without path encoding respectively. As discussed before, we capture retrieved information by variate vertex and obtain global state by global vertex, which allows graph transformer to better articulate global patterns and ASU location. Since the edges around each extra vertex do not represent real relation in KB, we only contextualize global and variate vertices' representation by attending over their neighborhoods. As a result, these two vertices' representations are calculated attending over their neighborhoods in G' formulated as follows. Here, $\mathcal{N}_g$ denotes the neighborhoods of v_g and the representation calculation of v_v is the same as v_g.

$$\hat{v_g} = \overset{N}{\underset{n=1}{\|}} \sum_{j \in \mathcal{N}_g} \alpha_j^n W_V^n v_j, \quad where\ a_j^n = \frac{exp((\mathbf{W_k k_g})^\top W_Q q_i)}{\sum_{z \in \mathcal{N}_g} exp((\mathbf{W_k k_z})^\top W_Q q_g)} \tag{4}$$

4.2.2 Algorithm 2

Furthermore, we propose a soft switch information propagation control scheme based on the same masking mechanism. When the mask mechanism is in effect, m_1 is set to 0 so as to block information spreading. Otherwise, m_1 is learned during training. In this approach, the transmission of information between vertexs is not simply divided into two categories, but is controlled by a parameter that can be learned.

$$m_1 = \sigma(W_q q_a + b_{ptr}) \tag{5}$$

$$a_{ij}^n = \frac{exp((\mathbf{m_1 W_k k_j} + \mathbf{W_R r_{ij}})^\top W_q q_i)}{\sum_{z \in V} exp((\mathbf{m_1 W_k k_z} + \mathbf{W_R r_{ij}})^\top W_q q_i)} \tag{6}$$

4.2.3 Algorithm 3

In this method, we do not propose a new calculation formula, but adopt a different mask mechanism. We use mask mechanism to distinguish different semantics, while only vertexs semantically related are visible to each other. Take the answer sentence in figure 1 as an example, vertices *Death_Race*, *Jason_Statham*, *Paul_w.s._Anderson*, *2008* are visible to each other, since these vertices together express a simple meaning 'Jason Statham and Paul W.S. Anderson cooperate Death Race in 2008'.

Since the semantic relevancy is difficult to automatically annotate, we currently use semi-automated manual labeling method to carry out experiments. Semi-automated manual labeling means annotating typical cases by humans and applying to corresponding cases which have the same structure. Noted in this method, we do not add extra vertex.

Communication path representation Given a communication path $p = [p_i]$ between two vertices, we acquire d-sized corresponding embedding sequence $s = [s_i]$ inspired by the label sequence embedding procedure in [34]. Considering continuous or discrete representations separately, we employ the LSTM method and self-attention method to calculate representation vector r_{ij}.

– LSTM method: encode the sequence s into vector r by LSTM units, that is using the hidden state in last time step as r
– Self-attention method: use attention function as presented in Eq. 4 to acquire the representation of s as $h^s = [h_i^s]$, then define a weight γ to calculate weighted sum of h^s as r:

$$\gamma_i = \frac{exp(e_i)}{\sum_{k=0}^{L} exp(e_k)}, e_i = v^\top tanh(W_{h^s} h_i^s + b) \tag{7}$$

where L denotes the length of communication path.

4.3 Question encoder

The question encoder transforms the question text into a vector representation and fuse it with graph representations $V^D = [v^D]$ to supplement semantic information in KG.

$$Hq = F(hq, V^D) \tag{8}$$

Firstly, token q_i are fed into a single-layer bidirectional LSTM [12] one by one, producing a sequence of concatenated encoder hidden states hq_i. While hq_i is expressed by $[\overrightarrow{hq_i}, \overleftarrow{hq}_{L-i+1}]$, which are encoded by a forward and a backward LSTM independently.

Then we employ a fusion function F based on matching rules to recognize related vertex in supporting graph for each question token. For each token q_i, we match it to the smallest granularity vertex v_i^D that contains q_i and concatenate the vertex representation v_i^D with hq_i. If there is no related vertex v_i^D, concatenate q_i with vector $\overrightarrow{0}$ to maintain dimension.

So the encoder output list $Hq_i = [\overrightarrow{hq_i}, \overleftarrow{hq}_{L-i+1}, v_i^D]$ after concatenation operation represents question tokens, which is stored for attention distribution calculation later. Beside, the encoder state $hq_L = [\overrightarrow{hq_L}, \overleftarrow{hq_1}]$ represents the whole question.

5 Generator

We propose an multi-task Generator combine a classification task and a sequence decoding task, so as to improve the accuracy of our answer sentence. There is no doubt that question type plays a decisive role in answer decoding since it influence the core topic generated.

5.1 Question type classification

We design two approach to distinguish question type based on common KBQA datasets.

– Domain-based: For some KB covered one specific domain, the topic of question are usually clearly, such as actor, movie, writer, etc,. in movie dataset, or prices, material, cost, etc,. in business domain.
– Speech-based: For some KB especially in open domain, there are a wide variety of question type. So we propose to classify question into what, when, who, where, and so on.

Corresponding to the two kinds of question type definition, we design recognizing strategy as following.

- Topic-based: This approach is aimed at specific domain KBQA datasets with fixed topic, such as `wikimovie` which can be divided into 10 topics. We determine question type by the ASU's topic, so this annotation method applies to topic-specific datasets where the ASU is known.
- Cluster-based: This approach deals with specific domain KBQA datasets, where there is no one-to-one correspondence between ASU and question type or ASU is unknown. So we propose to classify question according to clustering algorithm.
- Keyword-based: This approach applies to open-domain datasets by recognize sentence beginning word, such as what, when, who, and special keywords like the name of a person or place.

In training stage, G-NAG classify the question type by a MLP layer based on the graph embedding from Masked-Graph Transformer as shown in Figure 2. Usually, there are two ways to acquire graph embedding from V^D : vertex-based and pooling based, the former set a global vertex connected to other vertices and updated with others, the latter employ a average pooling function on all vertices.

5.2 Text decoder

G-NAG employ a LSTM to predict answer words y_t in each time step. During training, it accepts the previous outputs $y_{<t} = y_1, y_2, ..., y_{t-1}$, context vector c_t and decoder hidden state of the last step s_{t-1} to update hidden state: $s_t = f(y_{t-1}, s_{t-1}, c_t)$. Meanwhile, we apply the copy mechanism [7], to deal with some unknown or special words directly. Then we describe the generation process in decoder at each time step as follow.

Firstly, initialize the decoder state s_0 as global vertex representation. Then we acquire the graph context vector c_g by N-headed attention as follows.

$$c_g = s_t + \mathop{\Big\Vert}_{n=1}^{N} \sum_{i \in V} \alpha_i^n W_G^n v_i^D, \qquad where \alpha_i = \frac{exp((\mathbf{W_k k_i})^\top W_Q s_t)}{\sum_{z \in V} exp((\mathbf{W_k k_z})^\top W_Q s_t)} \tag{9}$$

Similarly, the question context vector c_q is computed attending over the question [1]. Then concatenate c_g and c_q as final context vector c_t. Below, parameters W_h, W_s, b^* are learned during training, and L indicates the length of question sequence.

$$c_q = \sum_{j=1}^{L} \beta_j^n h_j, \; where \beta_j = \frac{exp(e_j)}{\sum_{k=0}^{L} exp(e_k)}, e_j = v^\top tanh(W_h h_j + W_s s_t + b^*) \tag{10}$$

G-NAG model generates answer words both from vocabulary based on attention and copying words via pointing. Therefore, we define a soft switch g within 0 to 1, which chooses between predicting a vocabulary word by distribution P_v or copying a word via attention distribution $[\alpha_i, \beta_j]$. Eventually, we acquire a final probability distribution over the extend vocabulary as follows.

$$P(w) = g P_{copy}(w) + (1-g) P_v(w), \; where \; g = sigmoid(W_h^\top h_t + W_s^\top s_t + b_g) \tag{11}$$

$$P_{copy} = \sum_{j: w_j = w} (\alpha_j + \beta_j) \qquad P_v = softmax(W_{v1}(W_{v2}[s_t, c_t] + b_1) + b_2) \tag{12}$$

Besides, we minimize negative log-likelihood of the target word w_t^* for each time step, and the overall loss is defined as their sum.

$$\mathcal{L} = \frac{1}{T} \sum_{t=0}^{T} (-log P(w_t^*)) \tag{13}$$

Table 1 Data statistics of dataset

Dataset	Total movie num	QA-pairs	Avg length of question	Avg length of answer	Avg triples per QA-pair
wikimovie*	6429	12037	17	14	4.7
multihop	13066	34472	15	15	5.5

6 Experiment

6.1 Datasets

Our model attempts to generate natural answers, especially for complex questions that contain various relations between entities. To our knowledge, there is not an existing dataset naturally fitted to this problem. Thus, we tailor the `Wikimovie`[1] dataset [20] according to our requirements as `wikimovie*`. Moreover, we reconstruct a multi-hop dataset `wikimovie-multihop` from the `Wikimovie` and `DBpedia` by manual annotation. The original `Wikimovie` dataset consists of simple question-ASU pairs, external KB and natural sentences from Wikipedia about the movie, which covers 10 topics. To expand knowledge, we search cast members' related triples in DBpedia by DBpedia Lookup Service . Statistics of the two datasets are available in Table 1.

wikimovie* Take each natural sentence in `Wikimovie` as an ideal answer, we search the related triples in underlying KB and choose one o (*object*) among the triples as ASU. Then let annotators generate the corresponding question, which contains the triple information mentioned in the answer without variate and movie name as Example 1. We remove the QA-pair if its ASU is not unique. Since each natural sentence in `Wikimovie` is around one movie, the related graph is star-like and within 2-hops.

Example 1 Given the natural answer "*Resident Evil is a 2013* ***English*** *movie directed by Paul Anderson and starring Li_Bingbing*". One possible question is "*What is the language of the 2013 film by Paul Anderson and Li_Bingbing*".

wikimovie-multihop We extract sub-graph randomly in underlying KB with limited size, while the sub-graph should contain more than 2-hop relations between entities, but no more than 4-hop for the longest path. Then for each sub-graph, we mask one vertex to be ASU that is not in the border. Based on the sub-graph, let annotators generate QA pairs in natural language sentences, while the question must be answerable and the answer should contain all information without missing. Note that entities not essential for reasoning ASU could be omitted or replaced, i.e., in 2008 replaced by in the same year in Section 1. After annotators providing 460 QA pairs, we extend the dataset by replacing the sub-graph in underlying KB with the same graph structure.

6.2 Evaluation metrics

Automatic evaluation Refer to existing NAG systems [11, 32], we use ASU-acc to measure the accuracy of ASU. Following [5], we adopt some word-overlap based metrics

[1] http://fb.ai/babi

(WBMs)[2] including BLEU-4 [24], and METEOR [4] to measure the co-occurrences of references and generated answers.

Manual evaluation The evaluation result of automatic metric a little one-sided, so we design manual evaluation such as Fluency, Semantic and Syntax [21] employed on 100 generated samples. Annotator measure these index respectively by a score among 0-5, where the higher the score, the better the evaluation. The Kappa coefficient for inter-annotator is 0.744, and the p-value for scores is less than 0.01.

Evaluation in knowledge extractor Precision ratio represents the proportion of accurate vertices in all vertices of the sub-graph extracted. Recall ratio represents the proportion of extracted vertices of the accurate sub-graph.

6.3 Comparison models

We set up three groups of comparative experiments. Firstly, throughout existing researches on the natural language answer generation problem, we compare our model (G-NAG) with state-of-the-art NAG models from different perspectives.

- GenQA [32], a standard seq2seq model with attention using encoder-decoder structure. It retrieves the best-matched triple by MLP and encodes it with the question encoded by LSTM to generate a natural answer.
- COREQA [11], a similar structure to GenQA. Moreover, it retrieves more one-hop triples and introduces the copy mechanism.
- HM-NAG [6], an improvement of COREQA. It encodes all related triples in key-value structure without matching with the question and selects proper triples completely by attention during generation.

Secondly, we compare several baselines containing GAT(Graph Attention Network) or transformer. Since these models have no knowledge extractor module, we feed the same simplified graph after converting as input.

- GraphWriter [14], a graph2seq model for summarization containing graph transformer without variate vertex and communication path during the self-attention calculation.
- Tranformer [28], a sequence transformer proposed originally without graph structure.

Thirdly, we perform ablation studies to assess the impact of different components on the G-NAG model performance.

- $-w/o$ Question: Remove question encoder and maintain other module unchanged.
- $-w/o$ Multi-task: Do not consider question type classification, that is only decode answer sentence in Generator.

6.4 Implementation details

In knowledge extractor, we recognize entity phrases by StanfordCoreNLP tools, use Word2Vec [19] with 300 dimension vectors trained on the EN-wiki dataset to initialize text attribution for semantic similarity computation in Graph Simplifying stage, and employ TransE to encode underlying KB. Besides, we retain the top-2 alternative vertices in V_o with

[2]WBMs are implemented in https://github.com/Maluuba/nlgeval.

Table 2 Performances on dataset $Wikimovie*$

Model	ASU-acc	BLEU-4	Meteor	Fluency	Syntax	Semantics
GenQA	0.6506	0.3421	0.3722	2.5	2.6	2.0
COREQA	0.6680	0.3792	0.3990	2.7	2.7	2.5
HM-NAG	0.6818	0.3879	0.4113	2.7	2.9	2.7
GraphWriter	0.8171	0.4282	0.4527	3.4	3.5	3.4
Transformer	0.7913	0.4014	0.4371	3.1	3.2	3.1
G-NAG	**0.8601**	**0.4421**	**0.4828**	**3.4**	**3.6**	**3.7**
$-w/o$ Q	0.8201	0.4242	0.4490	3.1	3.3	3.3
$-w/o$ M	0.8310	0.4417	0.4811	**3.4**	3.5	3.6

a higher score, and set $k = 2$, $m = 3$ in extractor module. We use topic-based method to classify question type and employ global vertex's embedding as graph embedding for classifier. The choice of parameters k will be explained in subsequent experiments

In experiments, G-NAG and baseline models are trained for about 40 epochs with the learning rate as 0.03, where gradients are updated by Adam [13] learning rule. In both datasets, we add word occurring more than 5 times into vocabulary and the state size of word embedding and batch size are both set to 256. For the transformer, we set layer D as 6, attention heads as 4, following the setting in [14], and use a self-attention based method to encode the communication path described in Section 4.2.

For question type, this dataset has 10 topic around movie and each question only has one kind of answer type. Thus we use the original question type already reported before [20].

6.5 Result

6.5.1 NAG task experiment result

Table 2 shows the answer generation performance on the `wikimovie*` dataset, while G-NAG performs better than NAG baselines[3] both in the automatic or manual evaluation due to the improved knowledge extractor. Meanwhile, G-NAG outperforms GraphWriter and Transformer in ASU-acc, BLEU-4, and METEOR with stronger information express-ability of masked-graph transformer. Compared with $-w/o$ M, multi-task mechanism helps to improve ASU-acc as the classification task limits answer's topic (Table 3).

As for manual evaluation, both graph-based encoder systems generate more fluent natural answers with the same score in Fluency. Moreover, our G-NAG obtains a higher score in Semantics as Masked-Graph Transformer express graph more fine-grained.

Table 4 shows the results when we apply different algorithm in graph encoder. Since semi-automated manual annotation still requires high cost, and the mask mechanism based on semantics is more applicable for multi-hop QA pairs, we do not apply algorithm 3 in this dataset. Thus we can see that graph encoder using extra parameter performers better.

Next, we prove the best effectiveness of our model in `wikimovie-multihop` dataset in Table 4, in which we apply the algorithm 2 for graph encoder. In comparison to G-NAG, ASU-acc metrics of NAG baselines are unsatisfactory as they use one-hop triple

[3]Since different tailoring for the dataset, the result of HM-NAG is not the same as it reported

Table 3 G-NAG Performances on dataset *Wikimovie**

Model	ASU-acc	BLEU-4	Meteor	Fluency	Syntax	Semantics
$G-NAG^1$	0.8411	0.4274	0.4701	3.4	3.6	3.5
$G-NAG^2$	**0.8601**	**0.4421**	**0.4828**	**3.4**	**3.6**	**3.7**

retrieval method, which solves multi-hop relations hardly in a complex situation. Based on the same knowledge extractor, we see that G-NAG achieves higher ASU-acc than the latter two groups of comparison models since both multi-task mechanism and variate vertex setting can capture the ASU more correctly.

A comparison in manual evaluation between sequence-based knowledge representation, such as NAG baselines or Transformer, and graph transformer-based framework proves the express-ability of graph transformer. We analyze that sequence-based systems may miss information during retrieving or generating stage, therefore the generated answers get a low score. Further, as for graph transformer, G-NAG could generate more logical and perfect answers than GraphWriter, which is reflected in BLEU-4 and Semantics metrics.

Table 5 shows the results by different algorithm applied in graph encoder. We can see when the question require multi-hop relation to answer, the masked mechanism based on semantics performers better.

As mentioned before, G-NAG can handle implicit relations in questions, which is a challenge to NAG but the common situation in daily life, i.e., Q in Figure 2. Thus, we select the QA pairs in `Wikimovie*` where the questions have no obvious attribute or relational predicates. As shown in Table 6, G-NAG performs better than NAG baselines as it extracts triples depending more on the entity, not the relation, which is reflected in the decline value of ASU-acc and BLEU-4 compared with Table 4. Moreover, G-NAG keeps retrieved vertices as well as relation edges with higher scores in graph simplifying so as to identify these implicit relations. Furthermore, we can see that although the automatic metrics have fallen, the Fluency and Semantics of G-NAG stay essentially flat because of the ability of generator module. However, G-NAG may generate redundant information in this situation, which will be discussed in the case study (Table 7).

Table 4 Performances on dataset *Wikimovie-multihop*

Model	ASU-acc	BLEU-4	Meteor	Fluency	Syntax	Semantics
GenQA	0.3071	0.1608	0.2034	2.1	2.8	1.7
COREQA	0.4129	0.2011	0.2351	2.3	2.8	2.1
HM-NAG	0.4513	0.2106	0.2509	2.4	3.0	2.2
GraphWriter	0.7544	0.3322	0.3777	3.1	3.3	3.2
Transformer	0.7403	0.3078	0.3541	3.0	3.1	2.9
$G-NAG^3$	**0.7977**	0.3424	**0.4081**	**3.3**	**3.4**	**3.5**
$-w/o$ Q	0.7311	0.3121	0.3677	2.9	3.1	3.1
$-w/o$ M	0.7826	**0.3473**	0.3919	**3.2**	**3.4**	3.3

Table 5 G-NAG Performances on dataset $Wikimovie - multihop$

Model	ASU-acc	BLEU-4	Meteor	Fluency	Syntax	Semantics
$G-NAG^1$	0.7889	0.3470	0.3922	3.2	3.4	3.4
$G-NAG^2$	0.7719	0.3498	0.3812	3.3	3.4	3.4
$G-NAG^3$	**0.7977**	0.3424	**0.4081**	**3.3**	**3.4**	**3.5**

6.5.2 Additional experiments

To validate the effect of each strategy in Knowledge Extractor, we randomly choose 100 samples carried out extraction on the data step by step. As shown in Table 5, multi-hop retrieval and post disambiguation increases the probability of finding the right answer ASU. And graph simplifying reduce the vertex redundant obviously. In Table 5, simplifying-1 means the pruning operation based on connected path and simplifying-2 means the semantic similarity-based strategy. Here, we measure the result directly in supporting graph not containing the encoder and decoder structure.

Although disambiguation stage may reduce the precision due to the recognizing error, it significantly reduces the recall metric, as well as the graph simplifying stage. To strike a balance between accuracy and brevity, the combined extracting strategy is necessary.

Then we discuss the selection of neighbor hops. Consider that the size of the sub-graph and the cost of calculation will increase significantly with the increasing of k, we focus on k at 2 and 3. As shown in Table 5, the recall rate was maintained at a similar index after the extractor steps. As for precision, the large number of redundant nodes makes simplification difficult, so the larger k performers bad in this metric. In conclusion, the experiment results reported before are carried out when $k = 2$ (Table 8).

6.6 Case study

Table 6 gives some outputs from our model, GraphWriter, and HM-NAG which performs better than the other two NAG baselines. ASU and other entities in this table are marked as **bold** and *italics* separately, while copy words marked as underline and superscript denotes the dataset QA pairs from. Besides, we use (movie-1,movie-2) to denote mentioned movies in the order that they appear in gold answers.

Table 6 Performances on implicit relation dataset $Wikimovie$

Model	GenQA	COREQA	HM-NAG	GraphWriter	Transformer	G-NAG
ASU-acc	0.5217 (-0.129)	0.5513 (-0.117)	0.5904 (-0.091)	0.7744 (-0.0430)	0.7502 (-0.041)	**0.7911 (-0.038)**
BLEU-4	0.2904 (-0.052)	0.3122 (-0.067)	0.3212 (-0.067)	0.3884 (-0.040)	0.3571 (-0.044)	**0.4037 (-0.038)**
Meteor	0.3317	0.3520	0.3688	0.4243	0.3914	**0.4441**
Fluency	2.3	2.6	2.6	**3.4**	3.0	**3.4**
Syntax	2.6	2.8	3.0	3.4	3.2	**3.5**
Semantics	1.8	2.2	2.3	3.2	2.7	**3.5**

Table 7 Performances of Knowledge Extractor on 100 samples

Extract step	one-hop	multi-hop	disambiguation	simplifying-1	simplifying-2
recall k=2	27%	94%	90%	90%	88%
precision k=2	54%	22%	40%	53%	73%
recall k=3	27%	98%	92%	92%	89%
precision k=3	54%	14%	20%	37%	51%

In Case 1, though HM-NAG recognizes entities correctly, it fails to generate ASU and accurate movie names because of triple missing. Meanwhile, because GraphWriter does not consider path information in the attention calculation, it has not a comprehensive grasp of graph structure to generate *year* in the right position. In Case 2, when the given question contains implicit relations, it is hard for HM-NAG to recognize all accurate relations and ASU. Moreover, even fed with a more accurate graph, GraphWriter misses the relation reflected in gold answer by *co-writer*. As implicit relation affects the simplifying stage, G-NAG obtains a supporting graph with more redundant entities while it generates extra information as *horror*.

7 Related work

Our work belongs to the NAG task and draws inspiration from the related research fields.

In the early days, [18] create a pattern based system to solve NAG problem, which retrieves triples related to the topic entity from KB so as to fills the commonly used message-response patterns. To improve diversity, [11, 32] propose an end-to-end model, which encodes question and related knowledge as a sequence. Further, [6] put these triples into Key-Value memory proved effective by [20]. The above work provides a feasible framework consists of retrieving and generating that is followed by G-NAG. However, limited by the simple retrieval and sequence representation structure, these systems do not perform well in complex questions, which stimulates us to make improvements.

To find alternative representation structure for NAG, we notice that converting graph to sequence is wildly studied from different aspects. The above work proves that the graph is an effective structure to encode complex information [16], which fits our requirements. As for graph representation, the key idea is to learn a mapping to embed vertexs as points in a low-dimensional vector space. Motivated by [14, 29, 34], we employ graph transformer considering communication path to encode the supporting graph.

Since unknown or special words in source text may impede predicting, Copying based on Attention has been proven extremely useful for a broad range of text generation tasks. To judge where to copy from, Copynet [7] utilizes the soft attention distribution to produce an output sequence containing elements from the input. This solution is applied to dialogue system [7], NMT [8], summarization [22, 26], QA system [11], etc.

Multi-task Learning [23] leverage the information contained in multiple tasks simultaneously so it usually has better results than the single-task model. There are two ways to combine tasks, one is setting different neural network layers correspond to different sub-task [9, 27]. Another one is public-private mechanism in which all sub-tasks own independent model and share a public model meanwhile [17]. And the third way is Gated-mechanism, different modules share information by gate [31].

Table 8 Example outputs of various systems versus Gold

Question*	Do you remember the *César-winner* actress who appeared in director Rupert Sanders' and Drew Goddard's film in the same year?
Knowledge	(movie-1, release_year, 2012), (movie-1, directed_by, Rupert Sanders), (movie-1, starred_actor, ASU),(movie-2, release_year, 2012), (movie-2, directed_by, Drew Goddard),(movie-2, starred_actor,ASU)
HM-NAG	*César-winner* Rupert Sanders worked with Rupert Sanders in the film Snow White and the Huntsman, and Drew Goddard in the film Bad Times at the El Royale in the same year .
GraphWriter	*César-winner* **Kristen Stewart** worked with Rupert Sanders in the film Snow White and the Huntsman in 2012, and Drew Goddard in the film The Cabin in the Woods .
G-NAG	*César-winner* **Kristen Stewart** worked with Rupert Sanders in the film Snow White and the Huntsman, and Drew Goddard in the film The Cabin in the Woods in 2012 .
Gold	Yes, *César-winner* **Kristen Stewart** worked with Rupert Sanders in the film Snow White and the Huntsman, as well as Drew Goddard in The Cabin in the Woods in 2012 .
Question^{multi}	What is the release date of the *animated* movie by Kurt Frey and Ben Stassen?
Knowledge	(movie-1,directed_by,Ben Stassen),(movie-1,written_by,Ben Stassen) , (movie-1, written_by, Kurt Frey),(movie-1, release_year, ASU)
HM-NAG	Haunted Castle is a *animated* film written by writer Kurt Frey and Ben Stassen .
GraphWriter	Haunted Castle is a **2001** *animated* horror film written by writer Kurt Frey and directed by Ben Stassen .
G-NAG	Haunted Castle is a **2001** *animated* horror film written by writer Kurt Frey and directed by co-writer Ben Stassen .
Gold	Written by Kurt Frey and directed by co-writer Ben Stassen, Haunted Castle is a **2001** *animated* film .

8 Conclusion and future work

In this paper, we propose a G-NAG framework based on graph neural network to address the natural answer generation(NAG) problem. G-NAG pays more attention on complex QA conditions and proposes a novel knowledge extractor and an incorporating encoder. Specifically, it employs multi-hop retrieval before disambiguation and simplifying strategy during knowledge extraction, mainly increases express-ability by designing Masked-Graph Transformer to encode knowledge graph, and propose multi-task mechanism containing question type classification and sentence decoding to improve generation ability. Experimental results on two closed-domain datasets demonstrate that our model significantly outperforms existing NAG models, and prove the effectiveness of Masked Graph Transformer and multi-task mechanism meanwhile. In the future, we expect G-NAG to find the balance between enlarging retrieval range and controlling graph size. Moreover, we try to solve the repetition problems by coverage model or other approaches.

Acknowledgements This paper is an extended version of a published one which can be accessed at https://doi.org/10.1007/978-3-030-60259-8 and https://www.springerprofessional.de/de/natural-answer-generation-via-graph-transformer/18488332. Thus, part strategy and experiment results have been reported before.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 (2014)
2. Beck, D., Haffari, G., Cohn, T.: Graph-to-sequence learning using gated graph neural networks, pp. 273–283. Association for Computational Linguistics, Melbourne (2020)
3. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv:1406.1078 (2014)
4. Denkowski, M., Lavie, A.: Meteor universal: Language specific translation evaluation for any target language. pp. 376–380. ACL
5. Elsahar, H., Gravier, C., Laforest, F.: Zero-shot question generation from knowledge graphs for unseen predicates and entity types. ACL (2018)
6. Fu, Y., Feng, Y.: Natural answer generation with heterogeneous memory. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2018)
7. Gu, J., Lu, Z., Li, H.ang., Ok Li, V.: Incorporating copying mechanism in sequence-to-sequence learning. arXiv:1603.06393 (2016)
8. Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., Bengio, Y.: Pointing the unknown words. arXiv:1603.08148 (2016)
9. Hashimoto, K., Xiong, C., Tsuruoka, Y., Socher, R.: A joint many-task model: Growing a neural network for multiple nlp tasks. arXiv:1611.01587 (2016)
10. Hasibi, F., Balog, K., Erik Bratsberg, S.: Dynamic factual summaries for entity cards. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 773–782. ACM (2017)
11. He, S., Liu, C., Liu, K., Zhao, J.: Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (2020)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput **9**(8), 1735–1780 (1997)
13. Kingma, D.P., Adam, J.B.: A method for stochastic optimization. arXiv:1412.6980 (2014)
14. Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., hajishirzi, H.: Text generation from knowledge graphs with graph transformers. arXiv:1904.02342 (2019)
15. Li, X., Hu, S., Zou, L.: Natural Answer Generation via Graph Transformer. In: Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data, pp. 302–318. Springer (2020)
16. Lin, P., Song, Q., Wu, Y.: Fact checking in knowledge graphs with ontological subgraph patterns. Data Sci. Eng. **3**(4), 341–358 (2018)
17. Liu, P., Qiu, X., Huang, X.: Adversarial multi-task learning for text classification. arXiv:1704.05742 (2017)
18. McTear, M., Callejas, Z., Griol, D.: The Conversational Interface: Talking to smart devices. Springer international publishing, Berlin (2016)
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
20. Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., Weston, J.: Key-value memory networks for directly reading documents. arXiv:1606.03126
21. Mohammed, S., Shi, P., Lin, J.: Strong baselines for simple question answering over knowledge graphs with and without neural networks. pp. 291–296. ACL
22. Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al.: Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv:1602.06023
23. Pan, L., Xie, Y., Feng, Y., Chua, T.-S., Kan, M.-Y.: Semantic graphs for generating deep questions. arXiv:2004.12704 (2020)

24. Papineni, K., Roukos, S., Ward, T., Zhu, W.ei.-J.ing.: Bleu: a method for automatic evaluation of machine translation. Association for Computational Linguistics
25. Reinanda, R., Meij, E., de Rijke, M.: Mining, ranking and recommending entity aspects. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 263–272. ACM (2015)
26. See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. arXiv:1704.04368 (2017)
27. Søgaard, A., Goldberg, Y.: Deep multi-task learning with low level tasks supervised at lower layers. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 231–235 (2016)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł.ukasz., Polosukhin, I.: Attention is All You Need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
29. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv:1710.10903 (2017)
30. Wang, R., Wang, M., Liu, J., Chen, W., Cochez, M., Decker, S.: Leveraging knowledge graph embeddings for natural language question answering. In: International Conference on Database Systems for Advanced Applications (2020)
31. Xiao, L., Zhang, H., Chen, W.: Gated multi-task network for text classification. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pp. 726–731 (2018)
32. Yin, J., Jiang, X., Lu, Z., Shang, L., Li, H., Li, X.: Neural generative question answering. arXiv:1512.01337 (2015)
33. Ying, R., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. Adv Neural Inform Process Syst **32**, 9240 (2019)
34. Zhu, J., Li, J., Zhu, M., Qian, L., Zhang, M., Zhou, G.: Modeling graph structure in transformer for better amr-to-text generation (2019)