A New Question Answering Approach with Conceptual Graphs

Kyung-Min Kim[1], Jinhong Jung[2], Jihee Ryu[1], Ha-Myung Park[1], Joseph P. Joohee[1], Seokwoo Jeong[1], U Kang[2], Sung-Hyon Myaeng[1]

291 Daehak-ro
34141 Daejeon, Republic of Korea
[1] Korea Advanced Institute of Science and Technology
1 Gwanak-ro
08826 Seoul, Republic of Korea
[2] Seoul National University
[1]{kimdarwin, jiheeryu, blackmochi, wjdtjrdn1201, myaeng}@kaist.ac.kr
[2]{jinhongjung, ukang}@snu.ac.kr

RÉSUMÉ. *Avec la disponibilité croissante des bases de connaissances structurées à grandes échelles et des techniques de traitement automatique du langage naturel (TALN) aidées par des techniques avancées de recherche d'information (RI), les systèmes de questions-réponses (QR) sont entrés dans une ère de commercialisation. Cependant, les types de questions auxquelles on peut répondre sont un peu limités aux connaissances encyclopédiques qui sont souvent bien structurées sous forme de triplets ou, par ailleurs, localisées dans un segment de texte. Dans cette étude, nous proposons un graphe conceptuel basé sur les structures de questions-réponses (GCQR), qui permet l'inférence non-programmée et la représentation des connaissances à base de contexte. Cette approche a été mise en œuvre avec les techniques TALN pour générer non seulement des graphes conceptuels à partir du texte en question, mais aussi des algorithmes efficaces de correspondance graphique servant comme un mécanisme d'inférence, qui est pensé pour répondre à des questions, non seulement classiques mais aussi 'difficiles'.*

ABSTRACT. *With the increasing availability of large-scale structured knowledge bases and natural language processing (NLP) techniques aided by advanced information retrieval (IR) techniques, question answering (QA) systems have entered into a commercialization era. However, the types of questions that can be answered are somewhat limited to encyclopedic knowledge that are often either well structured like triplets or localized in a text segment. In this work, we propose a conceptual graph based question answering (CGQA) framework that enables informal inference and context-driven knowledge representation. This approach has been implemented with NLP techniques for generating conceptual graphs from text and efficient graph matching algorithms as an inference mechanism, which is geared toward answering not only conventional but also 'hard' questions.*

MOTS-CLÉS: *graphique conceptuel, cadre de questions réponses, appariement graphique*

KEYWORDS: *conceptual graph, question answering framework, graph matching*

1. Introduction

With the increasing availability of large-scale structured knowledge bases (KB) and natural language processing (NLP) techniques aided by advanced information retrieval (IR) techniques, question answering (QA) systems have entered into a commercialization era. A well-known example is IBM Watson that won a Jeopardy quiz contest against human champions (Ferrucci *et al.*, 2010). QA technology has been considered for different applications including medical diagnosis assistance and law service assistance.

However, the types of questions that can be answered are somewhat limited to encyclopedic knowledge that are often either well structured like triplets or localized in a text segment. In knowledge base based QA (KBQA) systems (Lopez *et al.*, 2007; Unger *et al.*, 2012; Berant *et al.*, 2014; Fader *et al.*, 2014), a knowledge base is composed of set of triples. Each triple is meant to be a universal fact, and context information such as time, location, related objects, and topics from the original information source are rarely linked in KB. Therefore, KBQA usually has an advantage for relatively simple factoid questions.

On the other hand, information retrieval based QA (IRQA) systems (Manandhar *et al.*, 2008) lean heavily on statistical approaches and ad-hoc processing. They first retrieve documents that are similar to the question and then extract and rank answer candidates. IRQA has a natural weakness that it is hard to collect dispersed pieces of evidence among documents in a principled way.

The existing QA techniques cannot handle some 'hard' questions because of two major reasons. They either rely on formal inference of fragmented and context-independent knowledge units like triples or are based on ad-hoc question answering algorithms together with an identification of linguistic constructs like named entities, co-references, and semantic types of entities. The first problem is especially critical for questions that require evidences coming from more than one context, making it important to explicitly handle context in question answering.

In this work, we propose a conceptual graph based question answering (CGQA) framework that enables context-driven knowledge representation and informal inference. This approach has been implemented with heavy NLP techniques for generating conceptual graphs from text, and efficient graph matching algorithms as an inference mechanism, which is geared toward answering not only conventional but also 'hard' questions. We describe how it applies to questions in a quiz contest similar to Jeopardy and demonstrate feasibility of the CGQA framework to help answering ‘hard’ questions that require a processing of contexts in a novel way.

2. Related Work

Building automatic QA system has been an active research area with a large number of researchers during the last decade. The KB-based question answering (KBQA) approach utilizes a massive KB such as Yago (Suchanek *et al.*, 2007), DBpedia (Lehmann *et al.*, 2013), and Freebase (Bollacker *et al.*, 2008). Several QA systems use an RDF form of data as in a template-based SPARQL learner (TBSL) (Unger *et al.*, 2012) and Parsempre (Berant *et al.*, 2014). Since a conventional KB is curated and verified by humans, KBQA can provide relatively accurate answers. A recent approach is to automatically curate a KB with information extraction (IE) techniques. P. Yin (Yin *et al.*, 2015) proposed to use *n*-tuple assertions and n-tuple open KB (nOKB) to answer questions with complex prepositional or adverbial constraints.

KBQA has a fundamental limitation with the lack of an ability to discriminate against different contexts. Since knowledge in KB is represented in the form of fragmented triples, it tends to contain universal facts. Another limitation is the difficulty in approximate reasoning. Since KBQA usually makes inference by connecting triples, it is not straightforward to mimic probabilistic inference and answer ranking, which incur exponentially increasing computational cost.

IRQA systems search raw text and thus can handle contextually sensitive questions. Since it heavily relies on statistical methods, however, some structural and semantic aspects in the original question and the text in the knowledge source cannot be enforced. IRQA also has a weakness as it usually relies on local search in extracting an answer, not being able to collect pieces of evidence dispersed among different documents. To overcome the problem of lost semantics, Manandhar *et al.* (2008) proposed to use different semantic features for an answer extraction process by using question types (factoid vs. non-factoid).

Recently new QA systems based on a deep learning approach (deepQA) have been introduced (e.g. Yu *et al.*, 2017). Although they are at an infant stage, they have a great potential with their high-dimensional information representation ability and flexibility. M. Tan (Tan *et al.*, 2016) proposed two hybrid models which combine the strength of both recurrent and convolutional neural networks to deal with answer passage selection task. However, deepQA has a fundamental drawback as it is impossible to interpret an intermediate result of inference in most of cases. Therefore, several attempts have been proposed to link a symbolic approach like KBQA or IRQA and a deep learning based approach as a promising research area (Socher *et al.*, 2013).

3. Simplified Conceptual Graphs for QA

Conceptual graphs (CG) are a knowledge representation language for a semantic network. It is a highly expressive system of logic with a direct mapping to and from natural languages (Sowa, 1992). With the definitional mechanisms, conceptual graphs can be used as an intermediate stage between natural languages and the rules and frames of expert systems and QA systems as well as an important tool for knowledge acquisition (Sowa, 1992). In this work, we adopt and simplify the CG representation as a basis for representing the content of the text from which questions need to be answered.

While the original conceptual graph framework has the knowledge representation power equivalent to first order logic and even has the potential for the second order logic with its flexibility, we take the liberty of relaxing some of the theories such as quantification so that we mainly follow its graph structure and the notion of type hierarchy so that the concepts and relations are considered at a semantic level. The resulting representation is much less formal but amenable for more flexible 'inference' based on graph matching.

3.1. Key Elements in CGQA

We first define a few key elements of conceptual graphs to be used, such as concepts and relations in such a way that they are suitable for QA tasks.

Definition 1. **(Concept)** A *concept* $c \in C$ is an abstract notion (e.g. 'economy') or an entity of a real-world (e.g. 'tiger' and 'Obama') mentioned in a text. A concept manifests itself in natural language text usually in the form of a noun or noun complex. C is a set of concepts, each of which is encoded with a unique identifier in the QA system. For the current implementation, we only consider concepts extracted from Wikipedia as it is the only resource from which an answer is obtained for questions.

Definition 2. (Relation) A *relation* $r \in R$ is a property of a concept or an action or a state between two concepts (e.g. 'PARENT_OF' and 'LIVES_IN'). Like *C*, *R* is a set of relations, each of which is encoded with a unique identifier in the QA system. A relation is usually expressed in text with an adjective phrase, verb phrase, or prepositional phrase but can be revealed by a sequence of words between and around two concepts in a text.

Definition 3. (Knowledge Triple) A *knowledge triple* $t =< c_i, r, c_j >$, where $c_i, c_j \in C$ and $c_i \neq c_j$, is a basic unit of knowledge that constitutes a conceptual graph. Note $t \in \boldsymbol{T}$ where $\boldsymbol{T}$ is the entire set of triples in a KB.

For example, a sentence "A robot appears in a play" can be represented as: <*robot, appear, play*>. While we use word forms in expressing a concept and a relation, they are encoded with unique identifiers in the QA system. It is one of the main differences between the proposed approach and the conventional information retrieval (IR) based QA approach or an open-domain relation extraction (Open-IE) approach, where strings composed of words or phrases appearing in the source text are used for a concept or relation, and answers are obtained based on string matching rather than semantic matching.

Using triples as building blocks, a conceptual graph can be constructed by merging the common concept nodes of triples. Clauses in natural language text are basic units for constructing conceptual graphs.

Definition 4. (Bare Conceptual Graph) A set of triples constitute a bare conceptual graph (BCG) $\hat{g} = \{t_1, t_2, \dots,\}$ which is a connected graph representing the meaning of a natural language clause. A collection of bare conceptual graphs is denoted as $\hat{G}$

That is, a concept node is reachable from any other concept node in a BCG. This constraint is important for the graph matching algorithm.

In a problem solving domain, context plays a critical role as it constrains the problem solving without intervening in it explicitly (Brézillon *et al.*, 1999). Likewise, context is important in answering questions. First of all, a question may not be answered correctly without a specific reference to a context in which an answer would satisfy the conditions. Let us consider the following:

Question: *What is the type of our galaxy (Milky Way) based on Hubble's categorization?*

Answer: *Barred spiral galaxy*

The question includes two contexts: 'our galaxy' and 'Hubble's categorization'. Without referring to the specific contexts, the system may find a wrong answer like 'spiral galaxy' (from the 'Andromeda galaxy' context) or 'black hole based galaxy' (from other categorization context).

Second, when BCGs are combined to represent the meaning of a connected text (e.g. in a document) or remote texts (e.g. two remote sentences referring to an event), it would not make sense to merge two concepts without considering their contexts. For instance, a triple <*robot, appear, play*> extracted from a sentence "The word 'robot' firstly appeared in a play." should not be combined with a triple extracted from an entirely different context like <*Hubo, is_a, robot*>.

To address this issue, we associate a context for each BCG.

Definition 5 (Context) A context $x \in X$, where X is a set of possible contexts in a QA system, is associated with BCGs or a higher level knowledge. A function $h: X \rightarrow \hat{G}$ identifies a collection of BCGs that are associated with a single context. Note that h^{-1} returns one or more contexts for a bare CG. A set of B CGs for a context *i* is denoted as $\hat{G}_i$.

A QA system must have its own definition of possible contexts or *X*, which can cover various types such as time, space, group of people, dialogue, etc. In our current research, we simply define *X* to be different Wikipedia articles. That is, we make an assumption that the concepts with the same surface form appearing in a single Wikipedia article refer to the same entity or abstract notion because they are in the same context.

Two BCGs in the same context can be merged when there is common concept node on each to form a larger graph. This is based on the assumption that two concepts of the surface form in the same context refer to the same concept. Although there can be an exception like the same surface forms even in the same sentence that may refer to two different entities, differentiation of such concepts is deferred to future research. On the other hand, the same surface forms in different contexts (i.e. different articles in the current work) are treated as different concepts with unique IDs. The only exception is that the concepts corresponding to a named entity appearing in different articles are treated as the same concept and hence merged at the time of graph matching. Now we can define a CG with its proper context.

Definition 6 (Context-Anchored Conceptual Graph) A collection of context-anchored CG (or XCG) $G_X = \{(\hat{G}, x)\}$ is defined to be a set of BCG collections associated with their contexts. An XCG instance $g_{x_i} = (C, R, X)$ is an ordered triples of concepts, relations, and contexts.

To illustrate all the definitions, we show an example:

Input sentence: *"The word 'robot' firstly written in a play"* (from wikipedia document titled 'robot')

XCG: {<robot, is_a, word> : Wikipedia:robot), (<robot, appear, play> : Wikipedia:robot)}

Note that relations are a member of predefined set of relations. For a notational convenience, the above XCG instance is expressed as

XCG: Wikipedia:robot :: {<robot, is_a, word>, <robot, appear, play>}

Fig. 1 shows a visualized representation of two XCG instances constructed from two different contexts (source documents). Rectangles and ovals represent concepts and relations, respectively. The rounded boxes represent context labels (in this case, source documents' titles) that are referred to by the BCGs.

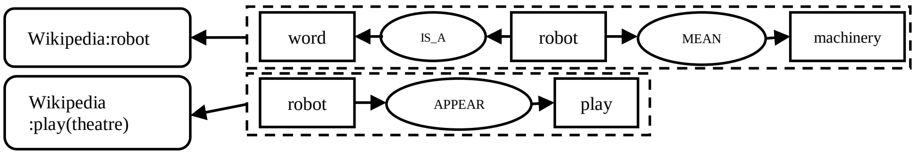


Figure 1. A visualized representation of two XCG instances

Note that BCGs in the same context may be merged with a common concept node to form a larger graph. This context-sensitive merging process simulates the usual inference with the triples in a conventional KB. Since this type of informal inference is conducted at the time of CG construction only within the same context, the way contexts are defined in a QA system determines the flexibility of its 'inference' and hence its characteristics in answering a question.

3.2. Converting Text to CG

This process involves many NLP tasks, which are challenging by themselves. In this paper, we only describe the gist of the key steps with key issues, and overall approaches taken in the current implementation.

Concept Extraction. Since concepts are the most essential building blocks of CGs and the core elements in question answering, concept extraction is the first task that determines the quality of a CG. From an NLP point view, we consider concepts come from nouns or noun phrases to cover abstract notions and entities as well as named entities. So the concept extraction task boils down to that of identifying a text boundary corresponding to a concept in such a way that its type can be determined or it can be mapped to a type hierarchy (aka ontology).

The main thrust of the current implementation is that we primarily rely on the headwords in multiple dictionaries and the titles of Wikipedia articles (referred to as reference sources) in defining the space of concepts and determining whether any words or phrases should be treated as concepts. We also apply some heuristic rules for identifying the concepts not found in the reference sources, such as newly coined words, unknown named entities, or phrasal entities. We deliberately avoid well-known technique like named entity recognition (e.g. [Cohen and Sarawagi, 2004]) because they are computationally expensive and labor-intensive for building training data. Based on a preliminary experiment with a small test set containing 200 concept instances, we obtained 88.5% accuracy in the concept detection task.

After a concept is extracted, we determine its type especially when it is ambiguous (e.g. human name vs. film's title with the same name). The type classifier is trained by word2vec (Mikolov *et al.*, 2013) features, which is a widely used deep-learning method based on text representation features. The training instances are sentences from the Korean Wikipedia corpus, selected by a string matching method with each category's concept. The concept categories which grant unique identifier to each type come from both Korean standard dictionaries and Wikidata.

Relation Extraction. Once all the concepts have been extracted from a text, all the relations between pairs of the concepts need to be found. Given a catalogue of pre-defined relations adopted from Wikidata (e.g. 'spouse'), the relation extraction task is treated as a classification problem because it is a matter of selecting one or more of the available relations based on the target sentence. Wikidata has thousands of unique relations defined by humans, and most of the triple instances are automatically generated from the Wikipedia corpus including the semi-structured 'info-box' information. By counting the concept pairs with a particular relation in the dataset, we selected 47 most frequent relations that would be useful in QA.

Formally the relation extraction function ρ is defined as follows:

$$\rho\colon \boldsymbol{C} \times \boldsymbol{C} \times \boldsymbol{X} \rightarrow \boldsymbol{R}$$

where $\boldsymbol{X}$ is broadly defined to be a context as introduced earlier but is limited to a narrower scope in the context of the two concepts extracted from the text. As a result of the mapping function, we end up generating triples as building blocks for a CG.

While this function can be defined in a number of different ways, our current approach is to integrate both pattern-based and supervised learning based classifiers. The former has an advantage of high precision but a weakness in recall. The pattern-based classifier determines a relation based on the POS patterns in the context between two concepts in the sentence. To handle the remaining cases, we train a classifier based on labeled sentences. We employ a distant supervision approach in (Mintz *et al.*, 2009). The precision of extracted relation scored 79.0% in the experimental set consisting of 300 randomly selected concept pairs.

4. CGQA Process

Question answering is defined to be a function $m: G_Q \rightarrow G_{KB}$ that maps a question CG (also referred to as a *query graph*) to a KB CG so that a relevant subgraph (typically a concept node) is returned. Since a question analysis often leads to an arbitrarily complex result including a semantic answer type (SAT), a lexical answer type, and a question focus for a natural language query, a QA process can take advantage of them by developing an ad-hoc answer selection algorithm. We define a core of CGQA as the graph matching kernel that returns a set of subgraphs that are likely to contain the answer.

Definition 7 (CGQA System Kernel) A CG-based QA system includes a kernel that receives a question graph g_Q which is in the form of G_X with or without explicit value for X and searches a KB $G_{KB} = \cup \mathrm{G}_{X_j}$ covering all the contexts.

The entire CGQA for a natural language query consists of four steps: query CG construction, context search, graph matching, and answer ranking.

The q*uery graph construction* process determines the question type such as 'fill-in-the-blank' and 'association inference' and then converts question sentences into a query graph. A query graph may or may not include a wildcard (*) concept node, depending on the question type. While a 'fill-in-the-blank' question asks for a concept that is missing in a query graph, expressed with a wildcard, an 'association inference' question asks for a concept that can be inferred from a set of concepts mentioned in the query.

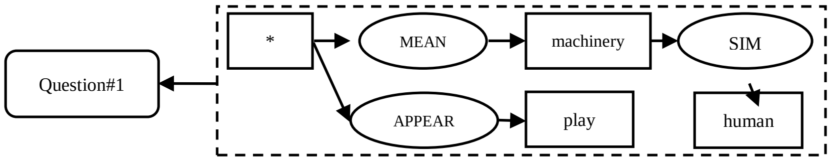
Given the sheer size of the KB, it would be impractical to match a query graph against the entire document CG in the KB. The *context search* process alleviates the problem by limiting the search only to those graphs in a specific context so that we can significantly reduce the search space for graph matching. Since the current implementation defines context with the boundaries of Wikipedia articles, the result of the context search process is a set of Wikipedia articles. Note that the search function can be implemented with more refined contexts such as time interval or location boundaries of any entity mentioned in the question.

Graph matching generates candidate subgraphs by invoking the core process in Definition 7. Details of the current algorithm are explained in Section 4.3. Finally, *answer ranking* process aggregates the graph matching result and ranks answer candidates extracted from the matched document graphs.

4.1. Query Graph Construction

Given a natural language question, the first step is to determine its type. While several types exist for the quiz contest, we pay attention to two: 'fill-in-the-blank' and 'association inference' types. The former is named as such because the question contains an explanation about something with a wildcard ('*') that should be filled in to make the question sentences correct. During the graph matching process, wildcard concepts should be matched with any concepts that are returned as potent answer candidates. The missing concept node is marked with a wildcard, and the QA process needs to find a concept to replace it. In this work, a query graph for the 'fill-in-the-blank' type has only one wildcard node. The latter type asks for a concept that should be inferred by a list of explanations in the question. For CGQA, we need to find a subgraph or node that is semantically related to the list of explanations. The question type is automatically determined by a pre-built classifier built by Heo *et al.*, (2015). Fig. 3 shows an example for the two types of questions and their CG versions.

1) **'Fill-in-the-blank' Question Type**: "*This word firstly appeared in a play. The modern meaning of it is 'a machinery similar to human*'. *What is this?*" [Answer: 'robot']



2) **'Association Inference' Question Type:** *"Apollon, Inka empire, and Louis XIV… What is related to all the above?"* [Answer: 'sun']

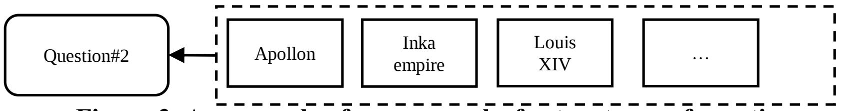


Figure 3. An example of query graphs for two types of questions

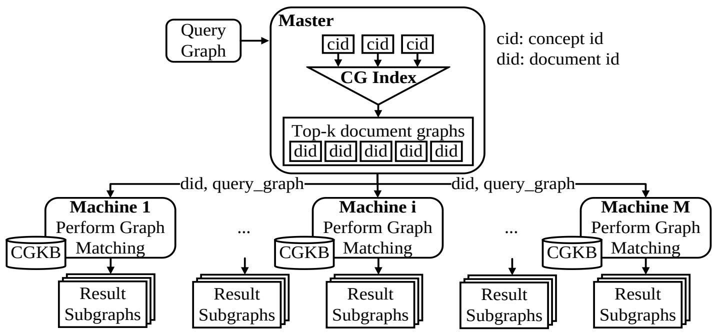


Figure 4. Overall process of context search for graph matching

4.2. Context Search

The answer for a query CG is likely to be in the same contexts as those of the concepts of the query. To find top-*k* context-anchored CGs (or document graphs in this case), we currently employ a simple *TF-IDF* term weighting scheme heavily used in information retrieval.

$$tf(c,d) = log(deg(c,d) + 1)$$
$$idf(c,D) = log(|D| \;/\; |\{d' \in D\colon c \in d'\}|)$$
where $deg(c,\ d)$ is the number of relations directly linked to the concept c in the context-anchored graph d. The score for the context graph associated with the query CG (qCG) is the sum of the *TF-IDF* scores of all concepts in the query graph. Since calculating all the scores of the context graphs requires very high computational cost, we create two indices to reduce the online calculation cost: 1) the graph-theoretic degree of each concept node in a CG, and 2) the number of context graphs containing each concept. As a result of context search, we can retrieve top-k document graphs to be matched with the query graph. The overall process is described in Fig. 4.

4.3. *Graph Matching*

We apply a graph matching algorithm to find appropriate answer candidates in document graphs. The graph matching approach finds or extracts subgraphs which are likely to contain answer candidates for a query graph from document graphs. The detailed policies for searching answer candidates in result subgraphs are different for the question types. Definition 8 describes the task of graph matching.

Definition 8 (Graph Matching) The inputs are a query graph $\boldsymbol{H}_q$, a set of document graphs $\boldsymbol{G} = \{g_1,\ ...,\ g_t\}$ where g_t is a document graph, and a top-k parameter k. According to the question types, the algorithm performs one of the following tasks: 1) partial subgraph matching for ‘fill-in-the-blank’ questions, and 2) center-piece subgraph extraction for ‘association inference’ questions. The outputs are top-k matched subgraphs $\boldsymbol{H}_t$ from each document graph in $\boldsymbol{G}$. The algorithm is applied to multiple document graphs in parallel as shown in Fig. 4.

Partial Subgraph Matching. The answer node of a ‘fill-in-the-blank’ query graph is likely to be close to subgraphs in a document graph when the subgraphs have the similar structure to the query graph. Our approach first matches a query graph except for a wildcard node using a subgraph matching algorithm based on G-Ray (Tong *et al.*, 2007); then we match the wildcard based on the best matched subgraph $\boldsymbol{H}_t$. The nodes matched to the wildcard are considered as answer candidates. We design a wildcard score function in order to match a document node to the wildcard. Let $\boldsymbol{S}(w)$ and $\boldsymbol{T}(w)$ denote in- and out-neighbors of the wildcard w in $\boldsymbol{H}_q$, respectively. Suppose $m\colon q \rightarrow v$ is a matching function from a query node q to a document node v. The wildcard score function of a document node v w.r.t. the wildcard w is computed as follows:

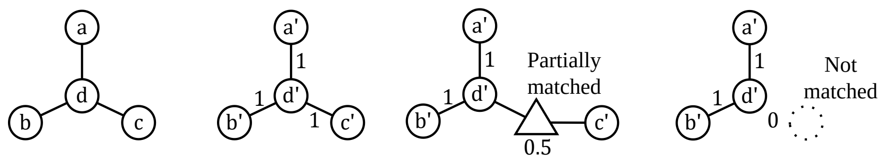
$$h_w(v) = \prod_{s \in \boldsymbol{S}(\boldsymbol{w})} r_{m(s)\rightarrow v} \times \prod_{t \in \boldsymbol{T}(\boldsymbol{w})} r_{v \rightarrow m(t)}$$

where $r_{u\rightarrow v}$ is the node relevance such as Random Walk with Restart (Jung *et al.*, 2016) from node u to node v. This indicates that node v’s wildcard score will be high if node v is close to other nodes matched to neighbors of w. We choose k document nodes $\{a_1, a_2, \dots, a_k\}$ for matching the wildcard w in the order of wildcard scores. Let $\boldsymbol{P}(w, a_i) = \{path(a_i, m(q)) | q \in \boldsymbol{S}(w) \cup \boldsymbol{T}(w)\}$ be the set of paths between a_i and other document nodes matched to neighbors of the wildcard w where the paths are discovered by a path finding algorithm suggested in G-Ray. Then, we compute the union of $\boldsymbol{H}_t$ and $\boldsymbol{P}(w, a_i)$ to form i-th matched subgraph.

Subgraph Score for Partial Subgraph Matching. We design a subgraph score to compare subgraphs obtained from multiple document graphs (Fig. 4). Suppose $\boldsymbol{H}_q(\boldsymbol{E})$ is the set of edges in $\boldsymbol{H}_q$, and $\boldsymbol{H}_t$ is a result subgraph. Consider an edge in $\boldsymbol{H}_q(\boldsymbol{E})$ is partially matched to a path p in a document graph. Let $\boldsymbol{P}$ be the set of paths in $\boldsymbol{H}_t$ for all edges in $\boldsymbol{H}_q(\boldsymbol{E})$. The subgraph score $\sigma(\boldsymbol{H}_t)$ is defined as follows:

$$\sigma(\boldsymbol{H}_t) = \frac{\sum_{p \in \boldsymbol{P}} |p|^{-1}}{|\boldsymbol{H}_q(\boldsymbol{E})|}$$

where $|p|^{-1}$ is the inverse of the length of the path p (i.e., a long path is penalized), and $|\boldsymbol{H}_q(\boldsymbol{E})|$ is the number of edges in $\boldsymbol{H}_q$. The score indicates the structure similarity between $\boldsymbol{H}_q$ and $\boldsymbol{H}_t$. Fig. 5. illustrates the examples of subgraph scores.



(a) Query graph (b) Result subgraph 1 (c) Result subgraph 2 (d) Result subgraph 3

$$\sigma(\boldsymbol{H}_t) = \frac{1+1+1}{3} = 1 \quad \sigma(\boldsymbol{H}_t) = \frac{1+1+0.5}{3} = 0.83 \quad \sigma(\boldsymbol{H}_t) = \frac{1+1}{3} = 0.67$$

Figure 5. Examples of subgraph scores when a query concept a is matched to a document node $\mathbf{a'}$. The subgraph score in (b) is 1 since the structure of the result subgraph is the same as that of the query graph. In (c), the length-2 path $\mathbf{d' - \Delta - c'}$ is partially matched to the edge $\mathbf{d - c}$ in the query graph; the score is penalized by the long path. Also, the score is penalized by non-matched nodes as in (d).

Center-piece Subgraph Extraction. Our goal is to find concepts that are most relevant to query concepts for an 'association inference' question. For the purpose, we extract subgraphs containing nodes relevant to query nodes using a subgraph extraction algorithm based on CePS (Tong *et al.*, 2006). CePS first matches query nodes, and extracts subgraphs based on nodes relevant to the query concepts. To measure the relevance of node v for query nodes, a center score is defined as follows:

$$h_{\boldsymbol{Q}}(v) = \prod_{q \in \boldsymbol{Q}} r_{m(q) \to v}$$

where $\boldsymbol{Q}$ is the set of query nodes, $m(q)$ is the node matched to query node q, and $r_{u \to v}$ is the node relevance score from node u to node v. We choose k document nodes $\{a_1, a_2, \ldots, a_k\}$ as answer candidates in the order of center scores. Let $\boldsymbol{P}(\boldsymbol{Q}, a_i) = \{path(a_i, m(q)) | q \in \boldsymbol{Q}\}$ be the set of paths between node a_i and other document nodes matched to the query nodes where the paths are obtained by a path finding algorithm in CePS. Then, we build i-th matched center-piece subgraph by computing the union of the paths in $\boldsymbol{P}(\boldsymbol{Q}, a_i)$.

Each answer candidate extracted from each document graph contains a local score such as a center score. To increase the performance of overall QA process, the local score has to be integrated into a global score. However, in our current implementation, we take an approach of choosing top-k answer candidates over all matched subgraphs in the order of local scores for simplicity.

5. Experiment

5.1. Task and Dataset

Our CGQA system was applied to a real-world QA task as a way of proving its feasibility and identifying key research issues for further improvements. In particular, our aim is to test the performance of the current implementation of the graph matching and answer ranking algorithms for the questions of the two types so that we can identify the areas for further refinement and extension.

The QA task came from a quiz contest on Korean broadcasting, called *Jang-Hak Quiz*. The task in our experiment is to answer a subset of the questions by finding a ranked list of answer candidates from document CGs constructed from the Wikipedia corpus. As the evaluation measure, we employ success@n which represents the ratio of questions which succeed in finding correct answers within top n answer candidates. The comparison with state-of-art QA systems is not conducted because current language we focus is Korean.

We have collected the questions and answers broadcasted from 2009 to 2014. From 2,355 questions in total, we selected 11.8% that are classified as 'hard' questions belonging to the 'association inference' type. We also tested our system for questions belonging to the 'fill-in-the-blank' type, which constitutes 60.6% of the entire set. Other minor question types like 'comparison' and 'calculation' are not used for our experiment.

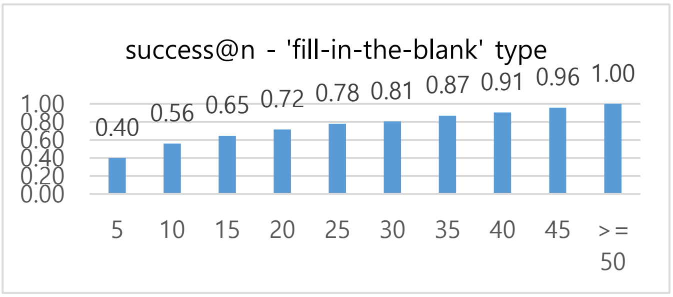
To construct a CG-based KB, we used the Korean Wikipedia corpus containing over 500,000 documents. After removing redirection, reference and category documents, we used 301,070 documents for actual experiment. A total of 1,618,458 concepts were extracted and the number of distinct IDs was 350,902. The number of triples formed after relation extraction was 20,676,272. We adopted 47 relations from Wikidata based on frequency.

5.2. Results

5.2.1. 'Fill-in-the-blank' questions

Out of 766 questions belonging to this type, we selected 170 for which the query graphs were correctly constructed and context search was conducted successfully with a reasonable number of candidate graphs. By isolating the errors incurred in the process of CG generation and context search, the experimental result would show the efficacy of the matching algorithms. The correctness of each query graph was determined by three judges.

Fig. 6 shows the result where the X axis represents the number of returned answer candidates (*n)* and the Y axis represents the ratio of correctly answered questions to the answer candidates, using success@n criteria. For example, for 40% of the questions, the correct answer was found within five answer candidates. The correct answers were found in over 96% of the questions within the top 45 answer



candidates. It means we can find the correct answer with a very high probability when we search only 45 answer candidates. Note that the result is based on the graph matching algorithm only, and the performance can be further improved if we use other features available in the question CG such as semantic answer type (SAT).

Figure 6. Success@n of 'fill-in-the-blank' type questions

5.2.2. 'Association inference' question type

This question type is atypical compared to usaual factoid questions because the clues included in a question are not likely to be found together with the answer in triples. It turns out that both IRQA and KBQA experience diffuculty in answering questions of this type. Out of 128 questions of this type in the dataset, we selected 30 questions for which their query graphs were constructed correctly and context search was also done successfully like the other query type case mentioned above.

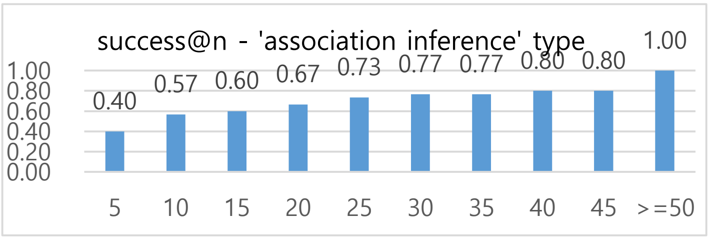


Figure 7. Success@n of 'association inference' type questions

Fig. 7 shows the result. As expected, the scores for this type are relatievly lower than the other question on average but not much. This result is very encouraging because this question type is considered very difficult to answer by traditional QA systems. Although finding the correct answer from the candidates may be more difficult for this type of questions, the use of semantics avaiable in query graphs can also be a big plus in finding the correct answers.

5.2.3 Failure Analysis

We show a failure case where the correct answer was ranked very low among the answer candidates, so that we can identify the limitations of the current implementation. The question we analyzed was of 'association inference' type. For the question *"A method or a tool to imply events that would occur in the future in a narrative of a novel or a play"*, the answer is '*foreshadow'*. The titles of the top five document graphs retrieved by the context search were: *Dramatization, Humanities, New_Crobuzon* (nation in a novel), *SF* (genre), and *Richard_Wagner* (composer). The first four are considered appropriate but the last one is completely irrelevant.

When the candidate graphs were matched against the query graph, the correct answer 'foreshadow' was ranked at 84. The highly ranked incorrect answers like 'train', 'setup', 'sex', and 'death' can be seen as instances of foreshadow in actual artwork but not appropriate as a general answer.

5.2.4. Discussion

One of the key contributions of this work is the explicit use of context for QA. The context search is valuable not only in reducing the search space but also enhancing the effectiveness of CGQA. The performance in the experiment would have been much lower without the context search as shown in a preliminary experiment where the performance without context search was much worse.

Existing QA approaches based on a context-insensitive KB rarely have knowledge structure to handle context, making it difficult to deal with 'hard' questions that refer to more than one context. Since the clue concepts in the questions are far apart from the correct answer, they cannot contribute to limiting the search in IRQA and making inference in KBQA. Fig. 8 shows a part of document graphs from three contexts to answer a question (from Fig. 3-1)). Although the clue concepts are dispersed widely, the correct answer is found successfully. For example, the concepts 'Human', 'Play', and 'Writer' found from several different contexts ('Wikipidia:Atheism' and 'Wikipidia:Humanities') contributed to find the correct answer 'Robot' by CGQA.

In our graph matching algorithm, the subgraph scores indicate how much a result subgraph is topologically similar to the query graph without considering the importance of each concept in the query graph. The proposed method would result in poor performance when the resulting subgraphs cover only a small portion of the concepts in the query graph and when they are similar to each other. Therefore we need to devise an enhanced scoring function by considering various semantics like concept-concept similarity.

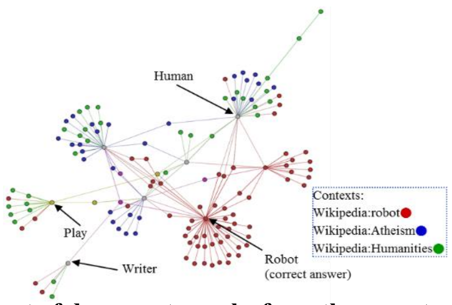


Figure 8. A subset of document graphs from three contexts to answer a question

6. Conclusion

We proposed a conceptual graph based question answering (CGQA) framework that enables informal inference and context-driven knowledge representation. This approach has been implemented with NLP techniques for generating conceptual graphs from text and efficient graph matching algorithms as an inference mechanism, which is geared toward answering not only conventional but also 'hard' questions.

We described how the new QA method applies to questions in a quiz contest dataset and the feasibility of CGQA framework that helps answering 'hard' questions that require processing of contexts. We reassured that the availability of context-based knowledge structure seems crucial for the given QA task.

There are a number of avenues to explore for future research. Using the framework, we can define a variety of context types so that the graph matching can be limited to more coherent candidate graphs, thereby improving efficiency and effectiveness. Another important area is to consider approximate matching in computing concept-concept or relation-relation similarity so that we can relax the rigidity of graph matching for higher recall. On the other hand, the structural constraints would compensate for a potential loss of precision.

7. Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. R0101-15-0176, Development of Core Technology for Context-aware Deep-Symbolic Hybrid Learning and Construction of Language Resources).

8. Bibliographie

Berant J., Liang P., « Semantic parsing via paraphrasing », *In Proceedings of ACL*. Vol. 7, No. 1. 2014.

Bollacker K., Evans C., Paritosh P., *et al.*, « Freebase: a collaboratively created graph database for structuring human knowledge », *In Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008. p. 1247-1250.

Brézillon P. « Context in problem solving: a survey », *The Knowledge Engineering Review*, Vol. 14, No. 1, 1999. p. 47-80.

Chen G., Kotz D., « A Survey of Context-Aware Mobile Computing Research », *Technical Report of Dartmouth College Hanover*, NH, USA. 2000.

Cohen W. W., Sarawagi S., « Exploiting Dictionaries in Named Entity Extraction », *SIGKDD,* 2004. p. 89-98.

Ferrucci D., Brown E., Chu-Carroll J., *et al.*, « Building Watson: An overview of the DeepQA project », AI magazine, Vol. 31, No. 3, 2010.

Heo J., Ryu P. M., Kim H. K., Ock C. Y., « Artificial Intelligence: Recognition of Answer Type for WiseQA », *KIPS Transactions*, Vol. 4, Issue 7, 2015. p. 283-290.

High R., « The Era of Cognitive Systems: An Inside Look at IBM Watson and How it Works », *Redbooks published by IBM corp*, 2012.

Jung, J., Shin, K., Sael, L., Kang, U., « Random Walk with Restart on Large Graphs using Block Elimination », ACM Transactions on Database Systems, 2016. 41(2), 12.

Lehmann J., Isele R., Jakob M., *et al.*, « DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia », *Semantic Web Journal*, 2013.

Mikolov T., Chen K., Corrado G., Dean J., « Efficient Estimation of Word Representations in Vector Space », *arXiv*:1301.3781 [cs.CL], 2013.

Mintz M., Bills S., Snow R., Jurafsky D., « Distant supervision for relation extraction without labeled data », *In Proceedings of the IJCNLP of the AFNLP*, 2009. p. 1003-1011.

Prediger S., « Nested Concept Graphs and Triadic Power Context Families A Situation-Based Contextual Approach », *ICCS 2000*, LNAI 1867, 2000. p. 249–262.

Quarteroni S., Manandhar S., « Designing an Interactive Open-Domain Question Answering System », Natural Language Engineering Vol. 1, No. 1, 2008. p. 1–23.

Schmidt A., Beigl M., Gellersen HW., « There is more to context than location », *Computers & Graphics,* Volume 23, Issue 6, 1999. p. 893–901.

Socher R., Chen D., Manning C. D., « Reasoning with neural tensor networks for knowledge base completion », *Neural Information Processing Systems*, 2013.

Sowa J. F., « Conceptual graphs as a universal knowledge representation », *Computers Math. Applic*. Vol. 23, No. 2-5, 1992. p. 75-93.

Suchanek F. M., Kasneci G., Weikum G., « Yago: a core of semantic knowledge », *In Proceedings of the 16th international conference on World Wide Web,* 2007, p. 697-706.

Tan M., Santos C., Xiang B., Zhou B., « Improved Representation Learning for Question Answer Matching », *In Proceedings of the ACL.* 2016. p. 464-273.

Tong, H., Faloutsos, C., « Center-piece subgraphs: problem definition and fast solutions », In Proceedings of the 12th ACM SIGKDD, 2006. p. 404-413.

Tong, H., Faloutsos, C., Gallagher, B., Eliassi-Rad, T., « Fast best-effort pattern matching in large attributed graphs », In Proceedings of the 13th ACM SIGKDD, 2007. p. 737-746.

Unger C., Bühmann L., Lehmann J., *et al.*, « Template-based question answering over RDF data », *In Proceedings of the WWW*. 2012. p. 639-648.

Yin P., Duan N., Kao B., *et al.*, « Answering Questions with Complex Semantic Constraints on Open Knowledge Bases», *In Proceedings of the IKM*, 2015. p.1301-1310.

Yu L., Hermann K. M., Blunsom P., Pulman S., « Deep Learning for Answer Sentence Selection », *NIPS deep learning workshop*, 2017. (to appear)