824

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 30, NO. 5, MAY 2018

Answering Natural Language Questions by  
Subgraph Matching over Knowledge Graphs

Sen Hu , Lei Zou , Jeffrey Xu Yu , Haixun Wang, and Dongyan Zhao

Abstract—RDF question/answering (Q/A) allows users to ask questions in natural languages over a knowledge base represented by  
RDF. To answer a natural language question, the existing work takes a two-stage approach: question understanding and query  
evaluation. Their focus is on question understanding to deal with the disambiguation of the natural language phrases. The most  
common technique is the joint disambiguation, which has the exponential search space. In this paper, we propose a systematic  
framework to answer natural language questions over RDF repository (RDF Q/A) from a graph data-driven perspective. We propose a  
semantic query graph to model the query intention in the natural language question in a structural way, based on which, RDF Q/A is  
reduced to subgraph matching problem. More importantly, we resolve the ambiguity of natural language questions at the time when  
matches of query are found. The cost of disambiguation is saved if there are no matching found. More speciﬁcally, we propose two  
different frameworks to build the semantic query graph, one is relation (edge)-ﬁrst and the other one is node-ﬁrst. We compare our  
method with some state-of-the-art RDF Q/A systems in the benchmark dataset. Extensive experiments conﬁrm that our method not  
only improves the precision but also speeds up query performance greatly.

Index Terms—RDF, graph database, question answering

Ç

1 INTRODUCTION

AS more and more structured data become available on

the web, the question of how end users can access this  
body of knowledge becomes of crucial importance. As a de  
facto standard of a knowledge base, Resource Description  
Framework(RDF) repository is a collection of triples, denoted  
as hsubject, predicate, objecti, and can be represented as a  
graph, where subjects and objects are vertices and predicates  
are edge labels. Although SPARQL is a standard way to  
access RDF data, it remains tedious and difﬁcult for end users  
because of the complexity of the SPARQL syntax and the RDF  
schema. An ideal system should allow end users to proﬁt  
from the expressive power of Semantic Web standards (such  
as RDF and SPARQLs) while at the same time hiding their  
complexity behind an intuitive and easy-to-use interface [1].  
Therefore, RDF question/answering (Q/A) systems have  
received wide attention in both natural language processing  
(NLP) [2], [3] and database areas [4].

Generally, there are two stages in RDF Q/A systems:  
question understanding and query evaluation. Existing systems  
in the ﬁrst stage translate a natural language question N  
into SPARQLs [1], and in the second stage evaluate all  
SPARQLs translated in the ﬁrst stage. The focus of the

(cid:2) S. Hu and D. Zhao are with Peking University, Beijing 100080, China.

E-mail: {husen, zhaody}@pku.edu.cn.

(cid:2) L. Zou is with Peking University, Beijing 100080, China, and the Beijing  
Institute of Big Data Research., Beijing, China. E-mail: zoulei@pku.edu.cn.  
J.X. Yu is with The Chinese University of Hong Kong, China.  
E-mail: yu@se.cuhk.edu.hk.

(cid:2)

(cid:2) H. Wang is with Facebook, Menlo Park, CA 94025.

E-mail: haixun@google.com.

Manuscript received 19 Feb. 2017; revised 11 Oct. 2017; accepted 13 Oct.  
2017. Date of publication 26 Oct. 2017; date of current version 30 Mar. 2018.  
(Corresponding author: Lei Zou.)  
Recommended for acceptance by A. Singh.  
For information on obtaining reprints of this article, please send e-mail to:  
reprints@ieee.org, and reference the Digital Object Identiﬁer below.  
Digital Object Identiﬁer no. 10.1109/TKDE.2017.2766634

existing solutions is on question understanding. Let us con-  
sider a running example in Fig. 1. The RDF dataset is given  
in Fig. 1a. Given a natural language question N1 ¼ “What is  
the budget of the ﬁlm directed by Paul Anderson?”, it is ﬁrst  
interpreted as a SPARQL query that is evaluated to get the  
answers (as shown in Fig. 1b).

1.1 Motivation  
The inherent hardness of RDF Q/A lies in the ambiguity of  
un-structured natural language question sentences. Gener-  
ally, there are two main challenges.

Phrase Linking. A natural language phrase wsi may have  
several meanings, i.e., wsi correspond to several semantic  
items in RDF graph G. As shown in Fig. 1b, the entity phrase  
“Paul Anderson” can map to three persons hPaul\_Anderson\_  
(actor)i, hPaul\_S.\_Andersoni and hPaul\_W.\_S.\_Andersoni.  
For a relation phrase, “directed by” also refers to two possible  
predicates hdirectori and hwriteri. Sometimes a phrase needs  
to be mapped to a non-atomic structure in knowledge graph.  
For example, “uncle of” refers to a predicate path (see  
Table 4). In RDF Q/A systems, we should eliminate “the  
ambiguity of phrase linking”.

Composition. The task of composition is to construct cor-  
responding query or query graph by assembling the identi-  
ﬁed phrases.  
In the running example, we know the  
predicate hdirectori is to connect subject hﬁlmi and object  
hPaul\_W.\_S.\_Andersoni; consequently, we generate a triple  
h?ﬁlm, director, Paul\_W.\_S.\_Andersoni. However, in some  
cases, it is difﬁcult to determine the correct subject and  
object for a given predicate, or there may exist several possi-  
ble query graph structures for a given question sentence.  
We call it “the ambiguity of query graph structure”.

In this paper, we focus on how to address the two chal-  
lenges. Different from existing solutions that try to solve  
ambiguity in the question understanding stage, we propose  
to combine disambiguation (for both phrase linking and  
query graph construction) and query evaluation together.

1041-4347 (cid:1) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.  
See ht\_tp://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

HU ET AL.: ANSWERING NATURAL LANGUAGE QUESTIONS BY SUBGRAPH MATCHING OVER KNOWLEDGE GRAPHS

825

Fig. 1. Question answering over RDF dataset.

(cid:1)(cid:1)! and u2u3

Speciﬁcally, we resolve the ambiguity of natural language  
questions at the time when matches of query are found. The  
cost of disambiguation is saved if there is no match found.  
We call this as the graph data-driven approach for RDF Q/A.  
We illustrate the intuition of our method by an example.  
Example 1. Consider a subgraph of graph G in Fig. 1a (the  
(cid:1)(cid:1)!  
subgraph induced by vertices u1, u2, u3 and c1). Edge u2c1  
(cid:1)(cid:1)!  
says that “Resident Evil: Retribution is a ﬁlm”. Edge u2u1  
says that “The budget of Resident Evil: Retribution is $ 65  
(cid:1)(cid:1)! says that “Paul W. S. Anderson directed  
million”. Edge u2u3  
the ﬁlm Resident Evil: Retribution”. The natural language  
question N1 is “What is the budget of the ﬁlm directed by  
Paul Anderson”. Obviously, the subgraph formed by edges  
(cid:1)(cid:1)! is a match of N1. “6.5E7” is a correct  
(cid:1)(cid:1)!, u2u1  
u2c1  
answer. On the other hand, we cannot ﬁnd a match (of N1)  
containing h Paul\_Anderson\_(actor)i inc G, i.e., the phrase  
“Paul Anderson” (in N1) cannot map to hPaul\_Anderson\_  
(actor)i. Therefore, we address the ambiguity issue of  
phrase linking when the matches are found. We can also  
resolve the ambiguity of query graph structure following  
the same idea. More details will be discussed in Section 5.  
The above example illustrates the intuition of our graph  
data-driven approach. A fundamental issue in our method  
is how to deﬁne a “match” between a subgraph of G and a  
natural language question N. Because N is unstructured  
data and G is graph structure data, we should ﬁll the gap  
between them. Therefore, we propose a semantic query  
graph QS (deﬁned in Deﬁnition 1) to represent the question  
semantics of N. An example of QS is given in Fig. 1c, which  
represents the semantic of the question N. Answering natu-  
ral language question equals to ﬁnding matches of QS over  
the underlying RDF graph G. To build QS, we propose two  
different frameworks: relation (edge)-ﬁrst and node-ﬁrst.

1.2 Our Approach  
Although there are still  
two stages “question under-  
standing” and “query evaluation” in our method, we do not

generate SPARQL at the question understanding step as  
existing solutions do. As we know, a SPARQL query can  
also be represented as a query graph, which does not  
include any ambiguity. Instead, our method builds a query  
graph that represents users’ query intention, but it allows  
for the ambiguity at the question understanding stage, such  
as the ambiguity of phrase linking and query graph struc-  
ture. We resolve the ambiguity when the matches are found  
at the query evaluation.

In the ﬁrst framework, we ﬁrst extract semantic relations  
based on the dependency tree structure of question senten-  
ces to build a semantic query graph QS. A semantic relation  
is a triple hrel; arg1; arg2i, where rel is a relation phrase, and  
arg1 and arg2 are its associated node phrases. For instance,  
h“directed by”,“ﬁlm”,“Paul Anderson”i is a semantic rela-  
tion. In QS, two edges share one common endpoint if the  
two corresponding relations share one common node  
phrase. Each node (entity/class mention) and edge (relation  
mention) in QS may have multiple candidates. The ﬁrst  
framework addresses the ambiguity of phrase linking when  
the matches (see Deﬁnition 2) of QS are found. Note that the  
ﬁrst framework does not address the ambiguity of query  
graph’s structure and assumes that the query graph can be  
uniquely ﬁxed at the question understanding step.

The second framework takes another perspective. When  
there exist some implicit or uncertain relations in N, the  
relation-ﬁrst framework often fails to extract such relations.  
Therefore, the second framework starts with extracting  
nodes from the question sentence N and connects these  
nodes to form a query graph. Furthermore, different from  
the relation-ﬁrst  
framework  
allows for the ambiguity of query graph structure at the  
beginning. It does not intend to build QS in the question  
understanding step. Instead, it builds a super graph QU of  
QS that includes uncertain edges. To match QU over the  
underlying RDF graph G, we allow for mismatching some  
edges in QU ,  
i.e., approximate match (Deﬁnition 5). We

the node-ﬁrst

framework,

826

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 30, NO. 5, MAY 2018

TABLE 1  
Notations

Notation

GðV; EÞ  
N  
Q  
QS  
QU  
Y  
DE=DR  
vi/ui  
Cvi /Cvivj

Deﬁnition and Description

RDF graph and vertex and edge sets  
A natural language question  
A SPARQL query (of N)  
The Semantic Query Graph (of N)  
The Super Semantic Query Graph (of N)  
The dependency tree (of N)  
The entity/relation mention dictionary  
A vertex in query graph / RDF graph  
Candidate mappings of vertex vi / edge vivj

resolve the ambiguity of phrase linking and query graph  
structure together when the approximate matches are  
found. Actually, the approximate matching position (in  
RDF graph G) deﬁnes the semantic query graph QS that we  
aim to build. In other words, we push down resolving the  
ambiguity of QS’s structure to the query evaluation stage.  
In a nutshell, we make the following contribution.

In the ﬁrst

(1) We propose two graph data-driven frameworks for  
RDF Q/A task, different from exiting solutions, in  
which the disambiguation and query evaluation are  
combined together.  
framework, we  
address ambiguity of phrase linking at the query  
evaluation; while in the second framework,  
the  
ambiguity of phrase linking and query graph’s struc-  
ture are both resolved. The graph data-driven frame-  
works not only improve the precision but also speed  
up query processing time greatly.  
In the ofﬂine processing, we propose a graph mining  
algorithm to build a relation mention dictionary, i.e.,  
mapping natural language phrases to possible predi-  
cates, which is used for question understanding in  
RDF Q/A.  
In the online processing, in order to speed up query  
evaluation, we propose efﬁcient top-k (approximate)  
graph matching algorithms of matching QS and QU  
over RDF graph.

(3)

(2)

(4) We conduct extensive experiments over several real  
RDF datasets (including QALD benchmark and  
WebQuestions benchmark) and compare our system  
with some state-of-the-art systems. The performance  
of our approach beat the other systems on QALD  
benchmark while close to the best on the WebQues-  
tions benchmark.

2 OVERVIEW

The problem of this paper is to ﬁnd the answers to a natural  
language question N over a RDF graph G. Table 1 lists the  
notations used throughout this paper.

There are two key issues in RDF Q/A problem. The ﬁrst  
one is how to represent the query intention of the natural lan-  
guage question N in a structural way. The second one is how  
to address the ambiguity of natural language N. In this paper,  
we focus on the ambiguity of phrase linking and query graph  
structure (composition) that are mentioned in Section 1.1.

2.1 Semantic Query Graph  
We deﬁne a semantic query graph (Deﬁnition 1) to repre-  
sent the query intention of the question N in a graph struc-  
tured way.

Deﬁnition 1 (Semantic Query Graph). A semantic query  
graph (denoted as QS) is a graph, in which each vertex vi is  
associated with an entity phrase, class phrase or wild-cards in  
the question sentence N; and each edge vivj is associated with a  
relation phrase in the question sentence N, 1 (cid:3) i; j (cid:3) jV ðQSÞj.

1 is given in Fig. 2b. In QS

Given the question sentences N1, the corresponding  
semantic query graphs QS  
1 , nodes  
v1, v2 and v3 are associated with “what” (wild-card), “ﬁlm”  
(a class phrase) and “Paul Anderson” (an entity phrase),  
respectively. The relation phrase “(be) budget of ” denotes  
the relation between v1 and v2, as well as the relation phrase  
“directed by” between v2 and v3.

As mentioned in the introduction, we want to ﬁnd a  
“match” of the semantic query graph QS over RDF graph G.  
When the matches are found, we resolve the ambiguity of  
natural language question sentence; meanwhile we ﬁnd the  
answers to the question. Generally, a “match” is deﬁned  
based on subgraph isomorphism. Given a node vi  
in a  
semantic query graph QS, if vi is an entity phrase or a class  
phrase, we can use entity linking algorithm [5] to retrieve all  
entity/class (in RDF graph G) that possibly correspond to vi,  
denoted as CðviÞ; if vi is a wild-card (such as wh-word), we  
assume that CðviÞ contains all vertices in RDF graph G. Anal-  
ogously, each edge vivj in QS also maps to a list of candidate  
predicates, denoted as Cvivj . Consider the semantic query  
graph QS in Fig. 2b. We also visualizes the candidates  
for each vertex and edge in QS in Fig. 2c. For example, v3  
(“Paul Anderson”) corresponds to hPaul\_Anderson\_(actor)i,  
hPaul\_S.\_Andersoni and hPaul\_W.\_S.\_Andersoni; and edge  
“v2v3” maps to hdirectori, hwriteri and hproduceri. Formally,  
we deﬁne the match as follows.  
Deﬁnition 2 (Match). Consider a semantic query graph QS  
with n nodes fv1; . . . ; vng. Each node vi has a candidate list  
Cvi , i ¼ 1; . . . ; n. Each edge vivj also has a candidate list Cvivj ,  
where 1 (cid:3) i 6¼ j (cid:3) n. A subgraph M containing n vertices  
fu1; . . . ; ung in RDF graph G is a match of QS if and only if  
the following conditions hold:

(1)

(2)

(3)

If vi is mapping to an entity ui, i ¼ 1; . . . ; n, ui must  
be in list Cvi ; and  
If vi is mapping to a class ci, i ¼ 1; . . . ; n, ui is an  
entity whose type is ci (i.e., there is a triple hui rdf:type  
cii in RDF graph) and ci must be in Cvi ; and  
(cid:1)(cid:1)! 2 M. Furthermore,  
8vivj 2 QS , uiuj  
(cid:1)(cid:1)!) is in  
(cid:1)(cid:1)! (or ujui  
the predicate Pij associated with uiuj  
Cvivj , 1 (cid:3) i; j (cid:3) n.

(cid:1)(cid:1)! 2 M \_ ujui

Let us see Fig. 2. The subgraph included by vertices c1, u1,  
u2 and u3 (in RDF graph G) denotes a match of semantic query  
graph QS in Fig. 2b. When the matches are found, we resolve  
the ambiguity, e.g., “Paul Anderson” should refer to hPaul\_W.  
\_S.\_Andersoni rather than others., meanwhile that we ﬁnd the  
answers to the question, i.e., “6.5E7”1 is the ﬁlm budget.

The core of our graph data-driven solution lies in two  
aspects: one is how to build a semantic query graph QS accu-  
rately and the other one is how to ﬁnd matches efﬁciently. In  
order to address the above issues, we propose two different  
frameworks. The ﬁrst one is called “relation (edge)-ﬁrst”. It  
means that we always extract relations from the natural lan-  
guage question sentence N and represent them as edges.  
Then, we assemble these edges to form a semantic query

1. Sixty-ﬁve million

HU ET AL.: ANSWERING NATURAL LANGUAGE QUESTIONS BY SUBGRAPH MATCHING OVER KNOWLEDGE GRAPHS

827

Fig. 2. Question answering with semantic query graph in relation-ﬁrst framework.

graph. The second framework takes another perspective,  
called “node-ﬁrst”. It starts with ﬁnding nodes (entity/class  
phrases and wild-cards) and try to introduce edges to con-  
nect them to form a semantic query graph QS. Furthermore,  
another major difference between the two frameworks is  
that the node-ﬁrst framework deﬁnes a super graph (called  
QU ) of QS when there exist some implicit or uncertain rela-  
tions in the question sentence. In other words, the node-ﬁrst  
framework is not to ﬁx the QS’s structure before subgraph  
matching evaluation as the relation-ﬁrst framework does.

2.2 Relation-First Framework  
Given a natural language question sentence N, the relation-  
ﬁrst framework begins with extracting semantic relations  
(edge together with two end points) from N.  
Deﬁnition 3 (Semantic Relation). A semantic relation is a  
triple hrel; arg1; arg2i, where rel is a relation mention, arg1  
and arg2 are the two node phrases.

In the running example, h“directed by”, “ﬁlm”,“Paul  
Anderson”i is a semantic relation, in which “directed by” is a  
relation mention (phrase), “who” and “actor” are its associ-  
ated node phrases. We can also ﬁnd another semantic relation  
h“budget of”, “what”,“ﬁlm”i from the question sentence N1.

2.2.1 Question Understanding  
The goal of the question understanding in the ﬁrst frame-  
work is to build a semantic query graph QS for representing  
users’ query intention in N. Speciﬁcally, we ﬁrst extract all  
semantic relations in N, each of which corresponds to an  
edge in QS. The semantic relation extraction is based on the  
dependency tree of users’ question sentence and a relation  
mention dictionary (see more details in Section 4.1). If the  
two semantic relations have one common node, they share  
one endpoint in QS. In the running example, we get two  
semantic relations, i.e., h“directed by”, “ﬁlm”,“Paul Ander-  
son”i and h“budget of”, “what”,“ﬁlm”i, as shown in Fig. 2.  
They can be combined through the common node phrase  
“ﬁlm” as showed in Fig. 2c. In addition, if two node phrases

refer to same thing after “coreference resolution” [6], we  
also combine the corresponding two semantic relations.

2.2.2 Query Executing  
As mentioned earlier, a semantic query graph QS is a  
structural representation of N. In order to answer N, we  
need to ﬁnd subgraphs of RDF graph G that match QS. The  
match is deﬁned according to the subgraph isomorphism  
(see Deﬁnition 2)

Each subgraph match has a score, which is derived from  
the conﬁdences of each edge and vertex mapping. Deﬁnition  
8 deﬁnes the score, which we will discuss later. Our goal is to  
ﬁnd all subgraph matches with the top-k scores. A best-ﬁrst  
algorithm is proposed in Section 4.2 to address this issue.  
Each subgraph match of QS implies an answer to the natural  
language question N, meanwhile, the ambiguity is resolved.

2.3 Node-First Framework  
The relation-ﬁrst framework has two main obstacles. The  
ﬁrst is that some relations are difﬁcult to be extracted. If the  
relation does not explicitly appeared in the question sen-  
tence, it is difﬁcult to extract such semantic relations, since  
our relation extraction relies on the relation mention in the  
relation mention dictionary. Let us consider two examples  
“show me all ﬁlms started by a Chinese actor”, “show me all  
ﬁlms stared by an actor who was born in China”. Obviously,  
the latter question has one explicit relation mention “(be)  
born in”, where the relation in the former one is implicitly  
mentioned. Therefore, it is difﬁcult to extract these implicit  
relations. Second, in the relation-ﬁrst framework, semantic  
relation extraction relies on the syntactic dependency tree of  
users’ question sentence and heuristic linguistic rules. If the  
syntactic dependency tree has some mistakes, it inevitably  
leads to wrong semantic query graph QS’s structure and  
wrong answers.

Considering the above two obstacles, we design a robust  
framework even in the presence of implicit relations and  
mistakes in the dependency parse tree. There are two key  
points in the second framework:

828

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 30, NO. 5, MAY 2018

Fig. 3. Question answering with super semantic query graph in node-ﬁrst framework.

(1)

The ﬁrst step is to extract node phrases (such as  
entity phrase, class phrase and wh-words) from the  
question sentence N, instead of relation extraction in  
the ﬁrst framework.

(2) We do not intend to build a semantic query graph QS  
at the question understanding step. Instead, we build  
a super semantic query graph QU , which possibly has  
some uncertain or implicit relations (i.e., edges). In  
other words, we allows the structure ambiguity of  
query graph in the question understanding step,  
which will be resolved at the query evaluation step.  
A super semantic query graph QU is analogue to QS (see Deﬁ-  
nition 4), but allows for explicit or uncertain relations (edges).  
Deﬁnition 4 (Super Semantic Query Graph). A super  
semantic query graph (denoted as QU ) is a graph, in which  
each vertex vi is associated with an entity phrase, class phrase or  
wild-card in the question sentence N; and each edge vivj is asso-  
ciated with a relation in N, 1 (cid:3) i; j (cid:3) jV ðQU Þj. If the relation is  
explicit, the edge label is the relation mention occurring in N;  
otherwise, the edge label is empty when the relation is implicit.

The following example illustrates the intuition of the sec-

ond framework.  
Example 2. Consider N2 in Fig. 3. “What is the budget of the  
ﬁlm directed by Paul Anderson and starred by a Chinese  
actor?”. The correct SPARQL query of N2 has two addi-  
tional triples than N1, which are t1 ¼ h?ﬁlm,starring, ?actori  
and t2 ¼ h?actor, country,Chinai. The relation-ﬁrst frame-  
work cannot generate t2 because the predicate “country”  
has no explicit relation mention in N2. In the node-ﬁrst  
framework, we introduce an edge between v4 (“actor”) and  
v5 (“Chinese”) in Fig. 3b, whose edge label is empty. For  
detected relation mention “starred by”, it is difﬁcult to  
determine its corresponding two nodes. There are three can-  
didate nodes: “Paul Anderson”, “ﬁlm”, and “actor”. In QU ,  
we introduce two edges between “ﬁlm” and “actor”; and  
“Paul Anderson” and “actor”. In the query evaluation step,  
we perform the approximate match (deﬁned in Deﬁnition 5)  
to match QU with RDF graph G, i.e., ﬁnding the occurrences  
of QU in RDF graph G with (possible) mismatching edges.

In this example, the ﬁnal match is denoted using bold lines  
in Fig. 3, in which the edge between “Paul Anderson” and  
“actor” (in QU ) is not matched.  
It is easy to infer that an approximate match of QU equals to  
an exact match of a connected spanning subgraph2 of QU ,  
where the spanning subgraph is the semantic query graph  
QS that we aim to build. Therefore, in the second framework,  
we ﬁx the semantic query graph QS when the matches are  
found; meanwhile the answers to the question have been  
found. In other words, we resolve the “structure ambiguity”  
of query graph at the time the matches are found. We also  
brieﬂy discuss the two steps of the node-ﬁrst framework as  
follows. More technical details are given in Section 5.

2.3.1 Question Understanding  
Given a natural language question sentence N, we ﬁrst  
extract all constant nodes from N by applying entity extrac-  
tion algorithms, which are referred to entities or classes. We  
also extract all wh-words (such as who, what and which  
et al.) from N as variable nodes. Then, to build QU , we need  
to introduce an edge between two nodes if there is a seman-  
tic relation between them. A naive solution is to introduce  
an edge between any two nodes. Obviously, this method  
introduces more noises and ambiguity for the query graph’s  
structure. On the other hand, the approximate match in the  
node-ﬁrst framework allows mis-matching one or more  
edges in QU . The naive solution leads to Oð2nÞ possible  
matching structures in the ﬁnal evaluation step, where n is  
the number of nodes in QU . This is quite costly.

To eliminate more noises and reduce the search space,

we propose a simple yet effective assumption:  
Assumption 1. Two nodes v1 and v2 has a semantic relation if  
and only if there exists no other node v(cid:4) that occurs in the sim-  
ple path between v1 and v2 of the dependency parse tree of ques-  
tion sentence N.

2. A spanning subgraph for graph Q is a subgraph of Q which con-

tains every vertex of Q.