A Question Answering System based on Conceptual Graph Formalism

Wael Salloum[1]
[1:] Faculty of Applied Sciences, University of Kalamoon, UOK, Damascus, Syria
edu@wsalloum.com

***Abstract*—This paper proposes a new text-based question answering system. It models knowledge in documents and questions with conceptual graph formalism (CGF). To prepare knowledge to be modeled, natural language processing is applied to the text using OpenNLP, and then syntactic and semantic information is realized using VerbNet and WordNet. We handle different types of questions, especially questions constructed with wh- pronouns, and questions constructed with "how", and we propose a model to represent them in CGF so that their *target* is realized and marked. Each question's type has different conceptual graph (CG) representations; thus, for each question, many CGs are generated using *formulas*. Some of these formulas are introduced here. The *projection* operator is used to compare a question's CG to a sentence's CG, and then the exact answer is extracted from the part of the sentence's CG that has been projected under the question target's concept.**

Keywords-*Question Answering; Knowledge Representation; Conceptual Graphs; Natural Language Processing; WordNet*

1. INTRODUCTION

Web search engines provide only directions on the web and are insufficient for people who seek knowledge. Therefore, a search engine will never satisfy the needs of a Knowledge Seeker (KS) no matter how huge is the document base (the Web for example). KSs need only the answer to their questions no more and no less, not 10 millions web pages that they must search in them again manually to find out the answer, and they don't care if the search engine boasts with a response time of 0.02 seconds. KSs need a question answering system that consolidates our slogan "Take your time, and give me nothing but the answer".

In this paper, we propose a text-based question answering system that acquires its knowledge from reliable text materials, and answers question semantically. It returns just the answer not the documents. Our system represents knowledge found in a document in the conceptual graph formalism (CGF). It 1) splits document into sentences; 2) parses sentences via natural language processing (NLP) techniques; 3) maps words in sentences to concepts via WordNet ontology; 4) uses the syntactical relations between words to construct conceptual relations between concepts so that each sentence is represented with a conceptual graph (CG); 5) uses sentence connectors (e.g. "therefore") between sentences, if found, to construct conceptual relations between their CGs; and 6) stores resulting CGs in a database.

Our system answers a question by 1) representing its knowledge as a CG, actually it generates many CGs to the question according to its type; 2) projecting question CGs on documents' CGs via the projection operator; 3) extracting answers from the projected sentences; and 4) comparing, combining, ranking and presenting the answers to the user. In this paper we focus on questions constructed with wh- pronouns and the ones constructed with "how". For each question type, we present some formulas by which the CG representations of the question are generated.

Section 2 describes the knowledge acquisition phase where the documents are parsed and their knowledge is represented. Section 3 explains how different types of questions are constructed and represented as CGs, and then how they are compared to documents' CGs in section 4. Finally, section 5 concludes and proposes further research.

2. KNOWLEDGE ACQUISITION (KA)

2.1. Construe text

This stage parses the text and builds its linguistic structure. Firstly, Natural Language Processing algorithms will be performed on each sentence to build a *syntax tree*. We utilize an open source toolkit which is OpenNLP to do that. Secondly, the *sentence pattern* will be identified.

For example, let's take the sentence: "The human liver secretes the bile." The resulting syntax tree of this sentence from OpenNLP is shown in fig. 1. The pattern of the syntax tree it is NP-VP-NP (Noun Phrase - Verb Phrase - Noun Phrase). The system identifies the main verb of this sentence "secrete", and then gets all VerbNet classes that match it. Then, it will examine all the semantic frames associated with each class to find the best match of the pattern of the syntax tree. In our case it will be "NP:Agent-VP-NP:Patient". That means, the first NP is the subject of the verb, and the next NP is the object of the verb. We call this best match the *sentence pattern*.

To alleviate the negative effect of paraphrasing discussed in [2], [5] and [6], the system preprocesses the text before parsing it with OpenNLP by replacing certain phrases with their synonyms (words or phrases) from a list. For example, replacing "X gave birth to Y" with "X bore Y", because in the first phrase the object will be *birth* not Y. This list is built manually by linguists and domain experts.

The system must post-process the syntax tree resulted from OpenNLP so that the sentence pattern suits the next stage "Construct". The post-processing simplifies compound and complex sentences to become simple ones, and converts passive voice sentences to active voice ones.

Thus the "Construe" steps are to 1) pre-process text; 2) parse text with OpenNLP; 3) post-process syntax tree; and 4) find the best frame that matches the sentence pattern.

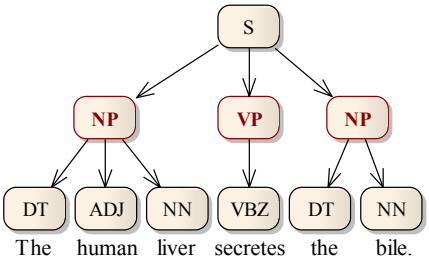

Figure 1. Example of a syntax tree

2.2. Construct Conceptual Graphs

In this KA's stage, the system realizes the meaning of the sentence by building the semantic model of the knowledge in its text based on its syntax recognized in the previous stage.

The first step is to convert the words found in the sentence into concepts using WordNet ontology which maps each word with one or more concepts. To find out which concept is meant, the system disambiguates these words depending on the context of the sentence. One approach to carry out word-sense disambiguation (WSD) is Domain Driven Disambiguation (DDD) which considers each word to be ambiguous when isolated from the text, but when it is found within a context, it will belong to the domain of that context (i.e. the domain of all the words in that context), given that a context will rarely belong to more than one domain with the same degree. We use WordNet Domains, an extension of WordNet knowledge base, which assigns each concept with a certain domain.

The second step is to build the relations between these concepts based on the syntax tree and sentence pattern. We choose the *Conceptual Graph Formalism (CGF)* introduced in [7] to represent knowledge in our system. CG is defined as a directed, bipartite graph; it contains two sets of vertices and a set of edges: $cg = (C, R, E)$. The first set of vertices (C) contains the *concepts* while the second set of vertices (R) contains the *relations* between these concepts. Each edge of E connects a concept from C with a relation form R. Each concept vertex contains a *Type*, which specifies the class of the entity represented by the concept, and a *Referent*, which identifies an instance of the concept's type. The graphical representation in fig. 2 illustrates this formalism.

The system determines the structure of the CG based on the sentence pattern and the syntax tree. Metadata about the represented knowledge must be annotated, which depicts many attributes of the CG such as the tense of the sentence, and its domain (medical, technology, etc.).

For example, we can represent the knowledge in the sentence: "Mark Twain wrote Tom Sawyer" as in the CG in fig. 3. At first, we obtain the concepts of the words/phrases of the sentence from WordNet: "Author" with a referent *Mark Twain*, "Write", and "Novel" with a referent *Tom Sawyer*. Then we will construct the CG according to the sentence pattern "NP:Agent-VP-NP:Patient": the verb is "Write", its subject is "Author" (connected be "Agent" relation), and its object is "Novel" (connected by "Patient" relation). The tense of the sentence is stored in the metadata along with the domain "Literature" extracted from WordNet Domains. We also can represent this CG in a simplified linear form as:

[Write]-
(Agnt) -> [Author: Mark Twain]
(Ptnt) -> [Novel: Tom Sawyer]

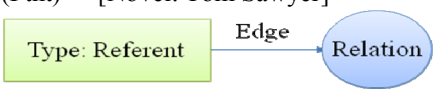

Figure 2. Conceptual Graph Formalism

Figure 3. Example of a Conceptual Graph

The knowledge is represented in a form that suits the question answering purpose of the system. Therefore, many concepts and relations are constructed and manipulated to increase the efficiency of question answering. For such concepts, the system recognizes named entities (NEs) [3] such as *Person*, *Organization*, *Location*, *Date/Time*, and *Number*. If a number (figure or text) is associated with a unit ($, mph, etc.), then the resulting concept *type* will be figured out from the unit and the concept *referent* will be the number.

2.3. Connect sentences

The system uses words and phrases that connect sentences (sentence connectors) to form a relation between the two CGs of these sentences. For example, words and phrases such as *because*, *therefore*, *thus*, *hence*, *consequently*, *as a result*, *for that reason*, *so*, *then* are used to form the "Implicate" relation between two CGs. Many other relations are constructed to facilitate sentence comparing and contrasting in order to summarize answers. In addition, the "Reason" relation, which answers "Why" questions, is constructed when one sentence contains "to <verb>" and the other one is in the past tense (stored in CG's metadata). However, if the other sentence is in the present tense, a "HowTo" relation is constructed instead. For example, the sentence "Sam killed his wife to get the insurance money." is represented as the following CG:

{ [Kill]-
(Agnt) -> [Person: Sam]
(Ptnt) -> [Wife: *] <- (Attr) <- [Person: Sam] }
-> (Reason) ->
{ [Get]-
(Agnt) -> [Person: Sam]
(Ptnt) -> [Money] -> (Attr) -> [Insurance] }

In contrast, the sentence "To get help, press F1." is represented as the following CG:

{ [Get]-
(Agnt) -> [Person: *]
(Ptnt) -> [Help] }
-> (HowTo) ->
{ [Press]-
(Agnt) -> [Person: *]
(Ptnt) -> [Computer-Key: F1] }

3. Question Construction

To represent the question, first, the system construes the question's sentence, and then constructs the knowledge in the question's sentence in the same knowledge representation form of the acquired knowledge: a conceptual graph and its metadata (we denote it as *QSCG*). However, the CG representation of a question differs from the CG representation of a document sentence in that the target of the question must be identified and marked, and in that, for each question, many CG representations will be generated to alleviate the negative effect of paraphrasing problem. CG generation subjects to the *type* of the question where each type is associated with a set of *formulas* to generate question CGs. These formulas are built manually and each one has a *weight*. We present some of them in the following. Finally, the domains of the question are extracted from WordNet Domains and stored in CG's metadata in order to narrow the search range.

3.1. The Whx Model

This model represents knowledge in wh-questions, which are constructed with *who*, *whom*, *what*, *which*, *when*, *where*, *why*, and *whose*. It identifies the question's target according to the wh- pronoun used in the question. The following sample formulas give a good explanation:

Who, Whom ➔ [Person: *]? <-(*rel*)<- [QSCG]
What, Which ➔ [Thing: *]? <-(*rel*)<- [QSCG]
When ➔ [Date: *]? <-(Date)<- [QSCG]
Where ➔ [Place: *]? <-(*rel*)<- [QSCG]
Whose ➔ [Thing: *]? <-(Agnt)<- [Possess] <- [QSCG]
➔ [Thing: *]? <-(Attr)<- [QSCG]
Why ➔ [CG]? <- (Reason)<- [QSCG]
➔ [CG]? ->(Implicate)-> [QSCG]

We mean by *rel* any relation that can be found; sometimes it can be identified from the question's context. Concepts followed by a question mark "?" are the target of the question, and will be a potential answer when found. QSCG is the CG of the question's sentence.

In the following we present some questions and their knowledge representation:

- Who invented the light bulb?
[Invent]-
(Agnt) -> [Person: *] ?
(Ptnt) -> [Light-Bulb]
- Who was assassinated on November 22, 1963, in Dallas?
[Assassinate]-
(Agnt) -> [Person: *]
(Ptnt) -> [Person: *] ?
(Date) -> [Date: 11/22/1963]
(Location) -> [City: Dallas]
- What is the gland that regulates sugar? Or,
Which gland regulates sugar?
[Regulate]-
(Agnt) -> [Gland: *] ?
(Ptnt) -> [Sugar]
- Where was Edison born?
[Bear]-
(Agnt) -> [Person: *]
(Ptnt) -> [Person: Thomas Edison]
(Location) -> [Place: *] ?
- What did Thomas Edison invented and when?
[Invent]-
(Agnt) -> [Person: Thomas Edison]
(Ptnt) -> [Thing: *] ?
(Date) -> [Date: *] ?

Although the named entity recognition has simplified the answering of wh-questions (except *why*) given the previous simple formulas, there are many special cases that cannot be answered without the creation of a reasonably-large set of formulas associated with each question type. Section 5 explains how they can be created automatically.

3.2. Questions formed by "How"

"How" questions are divided into the following types:

1) "How many/much"

A question that starts with "how many/much" is usually followed by a noun *X* where the question's target is the amount of *X*. "How many" is used for countable nouns (days) while "how much" is used for uncountable nouns (money).

The CG representation of the two types is:
[Number: *]?<-(Attr)<-[*X*]<-(*rel*) <- [QSCG]

If the system can't find a match for "Attr", it will counts all the answers for *X*, and then presents the answer as this: "I know *n X*s: X_1, X_2, …, and X_n.", where *n* is the count of *X*s.

Sometimes "how much" questions are not followed with a noun. In this case, CG representation is:
How much ➔ [Number: *]? <-(*rel*)<- [QSCG]

2) "How" followed by an adjective:

A question that starts with "how" followed by an adjective is asking for the amount of an attribute of a named entity (NE) found in the question. We first identify this NE, and then discover its attribute that matches the adjective. We maintain a list that contains adjectives assigned to nouns and units. Table 1 contains a sample.

TABLE I. HOW ADJECTIVES' LIST

Adjectives	Nouns	Units
Tall/short	Height/Length	meter, inch…
Long	Distance/Length	meter, mile…
	Interval	year, hour…
Old	Age	year, hour…
Far (away)	Distance	meter, inch…
Wide	Space/Area	acre, square foot…
Heavy	Weight	gram, pound…
Hot	Temperature	celsius, fahrenheit…
Fast/slow	Speed	mph, kmph…
Often	Frequency	times…

Some formulas of this question's type are:
[QSCG]->(*rel*)->[NE]->(Attr)->[*noun*]->(is)->[Number:*]?
[NE] <-(Agnt)<- [*verb*] ->(Ptnt)-> [Number:*]?
[QSCG] ->(*rel*)-> [Number:*]?

We use the second formula when a verb *verb* can be derived from the Noun. The relation *rel* means any possible relation that can be found. The third formula is the worst case. Some Examples of applying these formulas are:

- How heavy is the World Cup?
[World Cup]->(Attr)->[Weight]->(is)->[Number: *]?
[World Cup]<-(Agnt)<-[weigh]->(Ptnt)->[Number: *]?
- How fast can a Corvette go? Or, How fast is a Corvette?
[Corvette] ->(Attr)-> [Speed] ->(is)-> [Number: *]?
[Corvette] <-(Agnt)<- [go] ->(Ptnt)-> [Number: *]?

The system avoids the silly answers such as "very", "a little bit" and "many". Simply, it excludes adverbs from the answers' list when it contains a number.

3) "How" followed with an auxiliary verb in the past:

This type asks for an event happened in the past. The target is about "in what manner or way"; "by what means or process"; or "for what reason" that event occurred. Some formulas that generate Question's CGs are:

- [QSCG] ->(Instrument)->[Thing/CG: *]?

The "Instrument" relation is created in the KA phase when an instrument is identified by propositions such as *by* and *with* or words and phrases such as *using* and *by means of*. For example, "You can buy it with a credit card", "He created this document using MS-Word".

- [CG]? ->(Reason)->[QSCG]

In this relation, the target CG answers a *why* question with *certainty*, and the source CG answers a *how* question with *uncertainty* (note that in the following example).

E.g.: "To get the insurance money, he killed his wife."
Question: *Why* did he kill his wife?
Answer: To get the insurance money.
Question: *How did* he get the insurance money?
Answer: He killed his wife.

- [CG]? ->(Implicate)-> [QSCG]

In this case, "how" and "why" questions are the same. E.g.: "He got the insurance money because his wife died."

Question: *Why* did he get the insurance money?

Answer: Because his wife died.

Question: *How did* he get the insurance money?

Answer: His wife died.

4) "How" followed with an auxiliary verb in present:

The target of this type is a process or a fact. When the target is a process, the answer can be an article or a section in an article if its title matches the question. So the system must search in titles' and captions' CGs (the metadata of the CG specify that if it is a title or a caption).

On the other hand, when the target is a fact, the CG representations will be similar to the previous form and the system will generate CG representations according to the same formulas. However, "Reason" relation is replaced with "HowTo" relation, for example, for the sentence "To get help, press F1.", questions as "*How to* get help?" and "*How can* I get help?" will be answered with "Press F1.".

3.3. Other Questions

1) True/False Questions

The CG of a question of this type is constructed as for a sentence and projected on all CGs stored in the database.

2) Question formed with Show, Discribe, Name, etc.

This form of questions can be manipulated in a similar way to the previous cases.

4. Question Answering

When a user asks the system a question, the system builds its CGs as discussed previously, and then compares each question's CG to each sentence's CG in the same domains of the question by performing the projection operator. Then it extracts a potential answer (PA) from each successful projection, where PA is the part of the sentence's CG projected under the question's target. PA can be a CG (denoted as CG_{pa}) or a single concept node (denoted as $[C_{pa}]$). In the first case, the exact answer will be the sentence reconstructed from CG_{pa}. In the second case where $[C_{pa}]$ is a *restriction* of the question's target (which is an existential concept denoted as $[C_{qt}: *]$), the exact answer will be the type C_{pa} if it is a subtype of C_{qt} and/or it will be the referent r if C_{pa} is an *individual concept* $[C_{pa}: r]$, otherwise ($[C_{pa}]=[C_{qt}: *]$) it will be excluded from the answers. For example, the question "Who wrote Tom Sawyer?" seeks the *referent* "Mark Twain" not the type "Author". Finally, the system filters, summarizes, ranks and presents the answers to the user.

The projection operator is discussed in many papers as [1, 4, 8] and different algorithms are proposed. We will *not* discuss it here. We embrace the algorithm proposed in [8].

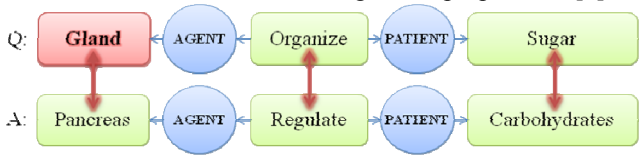

Figure 4. Example of projection of a question and a sentence. The "Gland" is the question's target. Bidirectional arrows represent a projection on concepts computed using WordNet and VerbNet.

For example, given the sentence "Pancreas regulates carbohydrates." and the question "What is the gland that organizes sugar?" the system projects the question's CG in the sentence's CG and extract the exact answer "Pancreas" (a subtype of "Gland") as illustrated in fig. 4. Then it calculates the relevance of the answer based on many factors such as the *weight* of the formula by which the question's CG was generated and the *similarity* between the two CGs. Then the system compares all the answers to eliminate replication and solve contradiction. Finally, it displays the final answer.

5. Conclusions and Further Work

We have introduced a model to represent knowledge in text documents and questions as conceptual graphs, and then find the answers of these questions by projecting the question's CGs on the documents' CGs, finally, extracting the answer from the projection results. For each question type, we provided some sample formulas that generate question's CGs for questions of that type. A considerable list of formulas can be learned by training the system as follows: 1) We can obtain a large set of question/answer pairs (e.g. TREC QA Track); 2) For each pair, the system searches for a string of both question and its answer in a large, well-written corpus (such as Wikipedia); 3) When a question/answer pair is found in a sentence, the CG of this sentence will be constructed; 4) A new formula will be learned by generalizing this CG; 5) The weight of this formula can be calculated as the frequency of its appearance; and 6) Formulas with low weights (below a certain threshold) will be dropped from the model.

Furthermore, research is needed on the summarization of the answers and comparing and contrasting them in order to present one final answer if possible. Summarization can be done by defining a model for *joining* answer's CGs such as reassembling complex and compound sentences from simple sentences and combining sentences of similar meaning. Comparing and contrasting can be achieved by utilizing relations constructed from sentence connectors such as *however* and *in contrast* for contrasting, and *besides* and *furthermore* for comparing.

References

[1] J. F. Sowa, "Knowledge Representation: Logical, Philosophical, and Computational Foundations," Brooks/Cole Publishing Co., Pacific Grove, CA, 2000.

[2] D. Mollá and M. Van Zaanen, "Learning of Graph Rules for Question Answering," Proc. ALTW05, Sydney, December 2005, 2005.

[3] M. Van Zaanen and D. Mollá, "A Named Entity Recogniser for Question Answering," Proceedings PACLING 2007, 8 pages, Melbourne, 2007.

[4] K. Willett and G. Lendaris, "Projection in conceptual graphs using neural networks," 1995, pp. 120+. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=485025

[5] F. R. James, J. Dowdall, K. Kaljur, M. Hess, and D. Mollá, "Exploiting Paraphrases in a Question Answering System," In Proc. Workshop in Paraphrasing at ACL2003, 2003.

[6] F. D. France, F. Yvon and O. Collin, "Learning Paraphrases to Improve a Question-Answering System," In Proceedings of the 10th Conference of EACL Workshop Natural Language Processing for Question-Answering, 2003.

[7] J. F. Sowa, "Conceptual structures: information processing in mind and machine," Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984.

[8] G. Amati and I. Ounis, "Conceptual graphs and first order logic," The Computer Journal, vol. 43, no. 1, pp. 1-12, January 2000.