Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 38 (2011) 4520-4523

ISSN: 2229-712X

Dependency based query answering of the simple english text

Aditya Tiwari[1] and Samta Gajbhiye[2]

[1]Shri Shankaracharya College of Engineering and Technology

[2]Department of Computer Science and Engineering, CSVTU University, Bhilai, India.

ARTICLE INFO

Article history:

Received: 22 August 2011;

Received in revised form:

26 August 2011;

Accepted: 31 August 2011;

Keywords

Question Answering,

Answer Finder,

Semantic Processing.

ABSTRACT

The improvement of current Question Answering (QA) systems relies on finding ways to support the traditional statistic approach to QA with logic reasoning. In this presentation we show one way of supporting an Interactive Question answering system with logic reasoning. The most likely answer to a question by searching a predicate format of topic-arranged question patterns and responses. Answer Finder is a framework for the development of question-answering systems. Answer Finder is currently being used to test the applicability of graph representations for the detection and extraction of answers. In this paper we briefly describe Answer Finder and introduce our method to learn graph patterns that link questions with their corresponding answers in arbitrary sentences. The method is based on the translation of the logical forms of questions and answer sentences into graphs, and the application of operations based on graph overlaps and the construction of paths within graphs. The method is general and can be applied to any graph-based representation of the contents of questions and answers.

Introduction

Web queries can be considered as implicit questions or commands, in that they are performed either to find information on the web or to initiate interaction with web services. Web users, however, rarely express their intent in full language.

For example, to find out "what are the movies of 2010 in which AKSHAY KUMAR stars", a user may simply query "AKSHAY KUMAR m,ovies 2010". Today's search engines, generally speaking, are based on matching such keywords against web documents and ranking relevant results using sophisticated features and algorithms.

As search engine technologies evolve, it is increasingly believed that search will be shifting away from "ten blue links" toward understanding intent and serving objects. This trend has been largely driven by an increasing amount of structured and semi-structured data made available to search engines, such as relational databases and semantically annotated web documents. Searching over such data sources, in many cases, can offer more relevant and essential results compared with merely returning web pages that contain query keywords.

Consider the query "AKSHAY KUMAR movies 2010". It is possible to retrieve a set of movie objects that satisfy the constraints Year = 2010 and Cast 3 AKSHAY KUMAR. This would deliver direct answers to the query rather than having the user sort through list of keyword results.

In no small part, the success of such an approach relies on robust understanding of query intent. Most previous works in this area focus on query intent classification (Shen et al., 2006; Liet al., 2008b; Arguello et al., 2009). Indeed, the intent class information is crucial in determining if a query can be answered by any structured data sources and, if so, by which one. In this work, we go one step further and study the semantic structure of a query, i.e., individual constituents of a query and their semantic roles. In particular, we focus on noun phrase, also trying to focus on whole sentences queries. My project works on sentences, active voice as well as passive voice sentences. A key contribution of this work is that we formally define query semantic structure as comprised of predicate format.

Identifying the semantic structure of queries can be beneficial to information retrieval.

Knowing the semantic role of each query constituent, we can reformulate the query into a structured form or reweight different query constituents for structured data retrieval (Robertson et al., 2004; Kim et al., 2009; Paparizos et al., 2009).

A second contribution of this work is to present methods that automatically extract the semantic structure of noun phrase queries. In particular, we investigate the use of transition, lexical, semantic and syntactic features. The semantic features can be constructed from structured data sources or by mining query logs, while the syntactic features can be obtained by readily-available syntactic analysis tools. We compare the roles of these features in two discriminative models, Markov and semi- Markov conditional random fields. The second model is especially interesting to us since in our task it is beneficial to use features that measure segment-level characteristics. Finally, we evaluate our proposed models and features on manually annotated query sets from three domains, while our techniques are general enough to be applied to many other domains.

Relationships between Phrases:

Semantic Processing

Semantic interpretation indicates dependencies among the concepts identified by mapping noun phrases to concepts in the Metathesaurus. We represent these depen-dencies in a predicate argument structure that we call conceptual structure, which is closely related to logical. The arguments in conceptual structure are labelled with semantic case roles in order to more clearly specify the relationships among the concepts represented. Conceptual structures are built through the application of semantic rules which fall into two major categories. As much as possible we rely on the UMLS Semantic Network.

Tele:

E-mail addresses: adityarise0609@gmail.com, Samta.gajbhiye@gmail.com

Logical Graph

We are developing a graph notation for the expression of the logical contents of questions and answer sentences. Our Logical Graphs are inspired on Conceptual Graphs (Sowa, 1979), though our graphs do not attempt to encode the full semantics of a sentence. Instead, the focus of our Logical Graphs is on robustness and practicability.

Robustness. It should be possible to automatically produce the Logical Graph of any sentence, even of those sentences that are not fully grammatical. The importance of this feature becomes obvious once one looks at the quality of the English used in typical corpora used for QA.

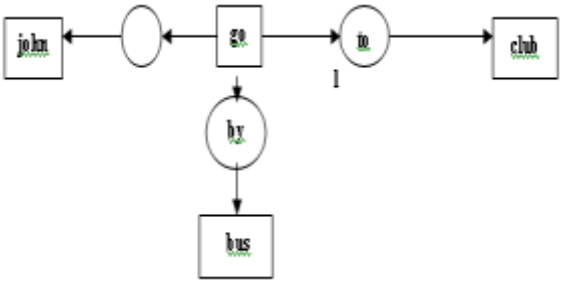
Practicability. The Logical Graphs should be automatically constructed in relatively short run time. The operations with the graphs should be computable within relatively short time. Like Sowa's Conceptual Graphs, our Logical Graphs are directed, bipartite graphs with two types of vertices, concepts and relations:

Concepts. Examples of concepts are objects dog, table, events and states run, love, and properties red, quick. Concepts may be arranged in a network of word relations (such as ontologies), though our method does not yet exploit this possibility in full.

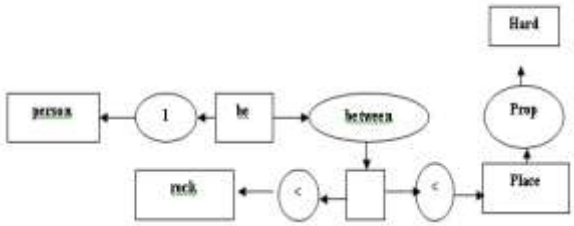
Relations. Relations act as links between concepts. Traditional examples of relations are grammatical roles and prepositions. However, to facilitate the production of the Logical Graphs we have decided to use a labelling of relations which is relatively close to the syntactic level of linguistic information. For example, instead of using the usual thematic roles agent, patient, and so forth, we use syntactic roles subject, object, etc. For convenience, and to avoid entering into a debate about the possible names of the syntactic roles, we have decided to use numbers. Thus, the relation 1 indicates the link to the first argument of a verb (that is, what is usually a subject). The relation 2 indicates the link to the second argument of a verb (usually the direct object), and so forth.

Figure 1 shows various examples of Logical Graphs. The first example shows the use of a relation 1 to express the subject of the go event, and two relations, to and by, that represent two prepositions. The second example shows the use of lattice structures to represent complex entities (such as the ones formed when a conjunction is used). This use of lattices is inspired from the treatment of plurals and complex events (Link, 1983; Moll´a, 1997). Finally, the third example shows the expression of clauses and control verbs. These examples only cover a few of the Figure 2: Graph overlaps of sentences John saw a book and Mary saw a table and John saw a table. The two overlaps are shown in thick lines. The stright lines show the correspondence relation from the graph vertices of each overlap and the projected subgraphs in the original graphs (the correspondence relation from the edges is not shown to improve readability).

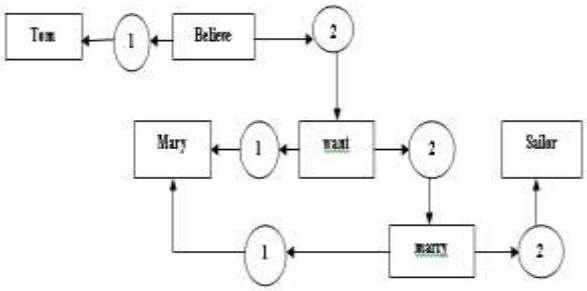
Linguistic features but we hope they will suffice to show the expressive power of our Logical Graphs.



John is going to club by bus



A person is between a rock and a hard place



Tom believes that Mary wants to marry a sailor

Figure 1: Examples of logical graphs

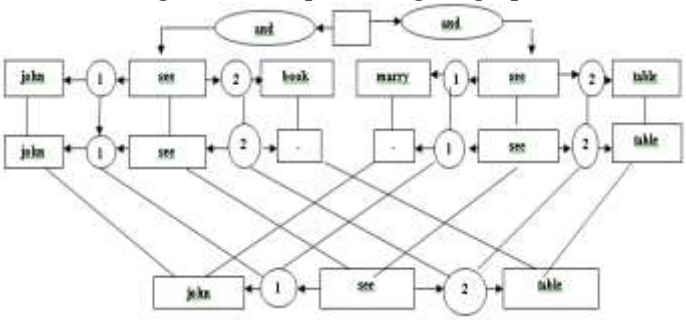


Figure 2: Graph overlaps of sentences John saw a book and Mary saw a table and John saw a table. The two overlaps are shown in thick lines. The stright lines show the correspondence relation from the graph vertices of each overlap and the projected subgraphs in the original graphs (the correspondence relation from the edges is not shown to improve readability)

Learning of logical graph rule

With the help of a training set of questions and sentences containing the answers, a set of Logical Graph rules can be learnt. Figure 3 shows an example of a rule learnt between two sentences. The graph notation has been simplified by replacing the relation vertices with labeled edges.

The algorithm for learning rules is fairly Straight forward and is shown in Figure 4. Rules Learnt with this algorithm are very specific to the question/answer pair. For example, the

1 2 prop

Q:Where was Peter born? Genitive 1 2

A:Peter's birthplace was Paris

The Rule (ro in regular lines, rp in straight lines, ra in thick lines)

Figure 3: A logical graph rule

FOR every question/answer Sentence pair

Gq = the graph of the question

Gs = the graph of the answer sentence

Ga = the graph of the exact answer

FOR every overlap O between Gq and Gs

FOR every path P between O and Ga

Build a rule R of the form

Ro = O

Rp = P
Ra = Ga
Figure 4: Learning of graph rules
The current list of stop concepts is:
and, or, not, nor, if, otherwise, have,
be, become, do, make
The resulting generalised rules may then overgeneralise and therefore they must be weighted according to their ability to detect the correct answer in the training corpus. The weight W(r) of a rule r is computed following the formula:
W(r) = # correct answers found

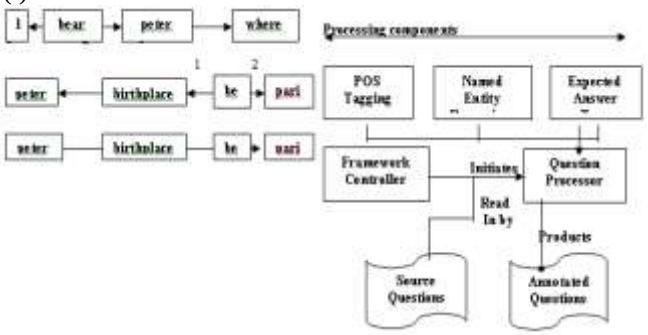


Figure 4: processing input questions

Question Processing

The question processing (QP) stage is responsible for analyzing and Understanding questions posed to the QA system. To accomplish this, the stage takes in provided questions, and subsequently make use of a variety of processing components to generate additional information that can help in interpreting the questions.

As an example, given a question it will be useful to know the expected answer type of the question. A processing component that can provide this information is used to provide this information.

Figure illustrates how the question processing stage is structured. The keen eyed reader will recognize that this architecture is very similar to that for the IBP stage earlier. This uniformity is intentional to make the QANUS framework easier to understand and pick up. Input questions in are fed to the Question Processor which then passes the documents through various processing components as shown in the figure. The output from the various components is called ANNO TATION S, and these annotations and the original questions are stored for use in subsequent stages in the QA pipeline.

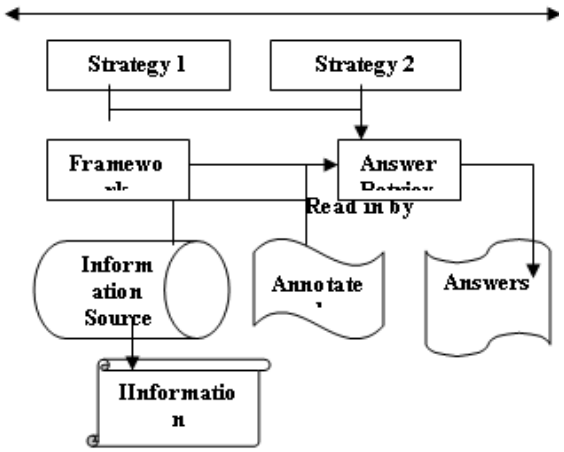


Figure 5. Getting answers from information source and questions

The architecture shown in Figure 4 again exhibits a high similarity to that used in the previous two stages explained above. The Answer Retrievor reads in the information source and annotated questions, and sends these information to various strategy components. Each of these components can make use of the provided information to derive answers to the posed questions. If more than one of these components is used, the Answer Retrievor class chooses one answer amongst the various proposals by each component. The final answers to the questions are then output.

There are two issues that may need to be elaborated on.

1. In instances where the information source is not built in the IBP stage, an implementation that observes the Information Base Querier interface needs to be provided. This interface specifies the interaction between the information source and the Answer Retrievor and needs to be observed before the Answer Retrievor can invoke the information source.
2. The architecture allows for multiple answer retrieval strategies to be used.

However we have not implemented the logic to choose between the answers provided by different strategies.

Conclusions and Further Work

We have introduced a methodology for the learning of graph patterns between questions and answers. Rules are learnt on the basis of two graph concepts: graph overlap, and paths between two subgraphs in a graph.

The techniques presented here use graph representations of the logical contents between questions and answer sentences. These techniques are being tested in AnswerFinder, a framework for the development of question answering techniques that is easily configurable.

We believe that our method can generalise to any graph representation of questions and answer Sentences. Further work will include the use of alternative graph representations, including the output of a dependency-based parser.

Finally, we plan to continue our evaluation of the method by integrating it into the AnswerFinder system and other QA systems to fully assess its potential..

References

[1] Understanding the Semantic Structure of Noun Phrase Queries 2010. Xiao Li Microsoft Research,One Microsoft Way Redmond, WA 98052 USA xiaol@microsoft.com

[2]Learning of Graph Rules for Question Answering Diego MOLLA and Menno VAN ZAANEN Centre for Language Technology, Macquarie University Sydney, Australia, {diego,menno}@ics.mq.edu.au

[3] Semantic Processing in Information Retrieval 2009. Thomas C. Rindflesch and Alan R. Aronson National Library of Medicine Bethesda, MD 20894. BOOK -ELAINE RICH AND KELVIN KNIGHT.

[4] NATURAL LANGUAGE PROCESSING Thomas C. Rindfleschr

[5] A Fundamental Algorithm for Dependency ParsingMichael A. Covington Artificial Intelligence Center The University of Georgia Athens, GA 30602-7415 U.S.A.mc@uga.edu

[6] Dependency Grammar and Dependency Parsing Joakim Nivre

[7] Handling Arabic Morphological and Syntactic Ambiguity within the LFG. Framework with a View to Machine Translation,(2008) Mohammed A. Attia School of Languages, Linguistics and Cultures.

[8]A Method of Cross Language Question-Answering Based on Machine Translation and Transliteration —Yokohama National University at NTCIR-5 CLQA1 — Tatsunori MORI and

Masami KAWAGISHI Graduate School of Environment and Information Sciences, Yokohama National University 79-7 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan mori,kawagisi_@forest.eis.ynu.ac.jp

[9]BOOK -ELAINE RICH AND KELVIN KNIGHT

[10]Complex Question Answering: Unsupervised Learning Approaches and Experiments Yllias Chali chali@cs.uleth.ca University of Lethbridge Lethbridge, AB, Canada, T1K 3M4