Simple Structures in Question Answering

Matthew Purver

Department of Computer Science
King's College London
Strand, London WC2R XXX, UK
matthew.purver@kcl.ac.uk

April 8, 2001

Paper ID: ACL-2001-XXXX

Keywords: question answering, sentence structure, transformation

Contact Author: as above

Under consideration for other conferences (specify)? none

Abstract

This paper describes the design and implementation of an open-domain question answering system based on notions of transformation and matching of simple structural relations extracted from query and answer passages. I describe the techniques used, together with some proposals for further development.

Simple Structures in Question Answering

Paper-ID: ACL-2001-XXXX

Abstract

This paper describes the design and implementation of an open-domain question answering system based on notions of transformation and matching of simple structural relations extracted from query and answer passages. I describe the techniques used, together with some proposals for further development.

1 Introduction

1.1 The Need for Structure

Conventional information retrieval (IR) techniques, which do not take sentence structure into account, are not ideally suited to question answering (QA) - in terms both of determining whether a passage really answers a query, and of delivering a satisfactory answer. The following examples illustrate the two main problems for IR techniques:

Q *Who is the president of the USA?*

A1 *George Bush is the president of the USA.*
A2 *George Bush, despite polling fewer votes nationwide than Al Gore, is now the president of the USA.*
A3 *Laura Bush is the wife of the president of the USA.*

Example A1 does not seem to pose a problem: the sentence answers the question and could even be returned whole as an acceptable answer.

Example A2 shows the first problem: what do we return as the answer? Returning the whole sentence does not seem to be acceptable, and if we apply windowing techniques to reduce the amount of text returned in the manner of e.g. (Allan et al., 1999; Cormack et al., 1999; Lin and Chen, 1999), there is no guarantee that we will capture the essential words *George Bush* due to the distance from the keywords. Techniques that distinguish a required entity type (see e.g. (Abney et al., 2000; Radev et al., 2000; Moldovan et al., 1999; Srihari and Li, 2000)) - in this case, the *Who* query indicates that a person is required - will still have to somehow rule out *Al Gore* as the answer, despite the proximity to the keywords *is the president of the USA*.

Example A3 shows the second problem: although it matches the keywords and contains a suitable entity, it does not contain an answer at all.

In order to reliably distinguish between genuine answers and non-answers which match keywords, while also identifying the entities required for answering, we need some information about how the entities in the sentence are related (i.e. about their roles in the sentence structure).

1.2 System Overview

1.2.1 Description

A system was developed which used sentence structure to extract simple relations between words and phrases, broadly similar to the *semantic triple* approach of Litkowski (1999; 2000), but requiring only a shallow parse. These relations could then be matched between query and answer passages, thus both ensuring that a genuine answer is present, and identifying the essential matching words for use as an answer. In order to deal with a range of structural phenomena, while keeping matching restrictions tight in order to ensure rejection of non-answers, a set of allowable structural transformations was also introduced.

Section 2 describes the text processing and structural relation extraction, and section 3 describes the matching and transformation

process. A detailed description of the system is available in (Purver, 2000).

The system was evaluated over a wide range of query types and answer sentence phenomena. The performance is discussed in section 4, together with some proposals for further work.

1.2.2 Restrictions in Scope

Due to the restricted time available, certain necessary but existing-technology elements of a real-world system were simulated rather than developed. The initial identification of candidate answer passages containing question keywords through standard IR techniques was assumed, by only considering such passages (hereafter referred to as *answers*). The identification and classification of unknown named entities (e.g. proper names) was also assumed, by adding all entities to a lexicon.

For the same reason, it was also decided not to attempt to deal with certain classes of query/answer passage phenomenon: those that deal with tense and thus require knowledge of document creation time; synonyms; and those that require inference and real-world knowledge. Previous TREC participants have shown that treatments of tense (Morton, 1999) and synonyms (Ferret et al., 1999) are possible.

2 Sentence Processing

2.1 Shallow Text Processing

The query and answers are PoS-tagged using `shallowproc` (Knight, 2000), stemmed using a rule-based stemmer based on the Oxford Advanced Learner's Dictionary (OALD - see (**?**)), and then parsed to give a shallow tree structure. Various robust shallow parsers are available (see e.g. (Strzalkowski, 1997)) and could be used within this approach, but as requirements were initially not precisely known, a simple non-deterministic finite-state syntactic processor was developed specifically for this project.

The resulting syntactic trees (see figure 1) consist of noun groups (NGs) and verb groups (VGs) under a single sentence node, with modifiers (prepositional phrases (PPs) and adverb groups) optionally attached to these groups.

Various features are attached to the groups for later use in matching: number (singular/plural) and entity type (e.g. person, organization, location) are used for NGs, and voice (active/passive) and predicate name for VGs. These types and predicate names were listed manually in a lexicon.

Query term NGs (those containing query words such as *who*, *how many*) were treated similarly, with entity type being attached in the same way. Entity type was determined either from the wh-word itself (e.g. *who*, *why*), or the other constituents in the NG (e.g. *what company*, *how rich*).

Entity types for PPs (e.g. location, time, manner, possession) were also determined, based on the analysis in (Sparck Jones and Boguraev, 1987). These were used in matching and in transformation.

2.2 Structural Relation Extraction

The resulting tree is subjected to coreference resolution and sub-sentential unit identification before simple relational structures are extracted. These structures can either convey a general predicate-argument relation (for example, the subject-verb-object relation *"X likes Y"*) or an existential relation (*"X is Y"*).

In order that the structures carry all the information required, we first need to go through a process of coreference resolution: pronouns and definite NGs, together with expressions of number, time and location, are (non-deterministically) resolved as references to other entities within the passage under consideration. The number and entity type features are used to restrict the possible references to suitable entities.

Complex sentences are then syntactically simplified to allow the structures to be easily extracted. Punctuation is used to split sentences into simple units at e.g. colons, semicolons; conjunctions of sentences, VGs and NGs are expanded into multiple simple units; and sentences containing relative and subordinate clauses are similarly simplified (using

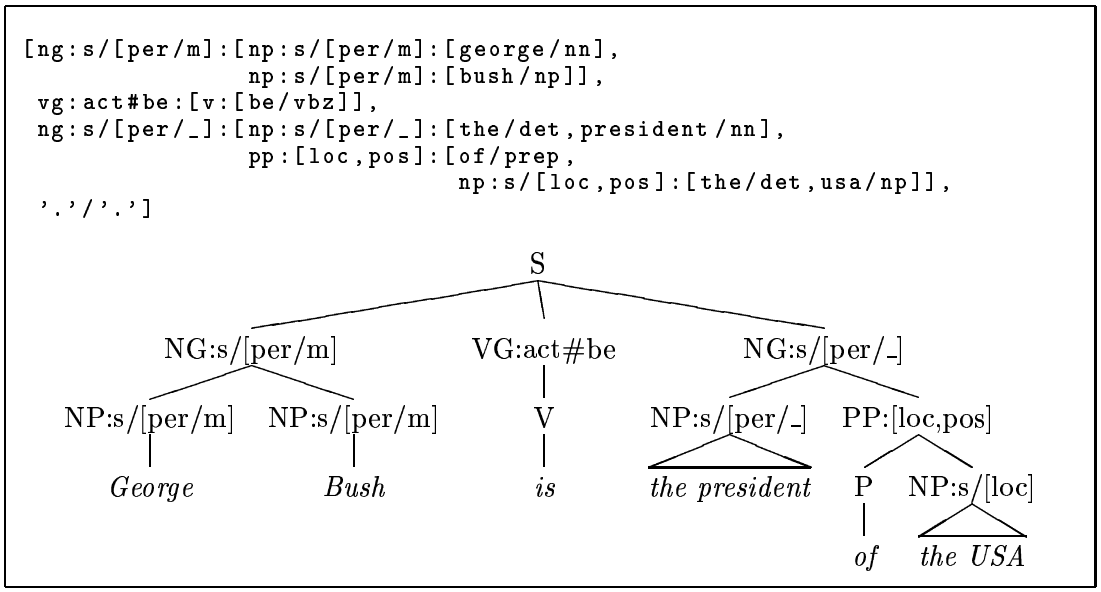

Figure 1: Example A1 after shallow text processing

punctuation to determine clause extent, after Briscoe (1994; 1996)). Care is taken to share all available information between the newly separated simple sentences.

2.2.1 Predicate-Argument Structures

Any relation involving a predicate (a non-existential verb) is then extracted as a *predicate-argument* (PA) structure. A PA-structure takes the form

`s:[Predicate, Argument1, Argument2]`

. This can be considered as a more general version of a verb-subject-object relationship: the number of arguments could be varied for verbs of different subcategories (information available in the OALD) if required - this was not attempted in this implementation.

The extraction of these structures is based entirely on word order. The predicate is the VG of the simple sentence. Any NGs and other words before the predicate are compounded to form the first argument. Similarly, all NGs after the predicate form the second argument.

2.2.2 State Structures

Although it is possible to represent existential verbs (e.g. *be*, but also others such as *be located*, *include*) as predicates in a PA-structure as above, this was found to have disadvantages. Existential relations are not only expressed by verbs, but by forms such as compound NGs or PP attachment: we would like our example Q to be matched by a sentence containing the NG *George Bush, the president of the USA*.

In order to facilitate the matching that this requires, existential verbs and NGs (but not conjunctions) were used to produce structures (*state*-structures), of the form

`s:[Argument1, Argument2]`

Our example A1 will produce the structure

`s:[[george bush],[the president of the usa]]`

as will A2 (along with other less relevant structures). It is usually the case that many possible structures can be extracted from a single sentence, and Prolog backtracking is used to allow each structure to be considered in turn.

3 Structural Transformation and Matching

3.1 Matching Principles

Until this point, query and answers have been treated in the same way. The extracted state- or PA-structures are now compared between query and potential answer, and a match is

attempted. Matching is governed by the following basic structural matching principles:

1. *Structures match if all constituent entities match (predicate, if applicable, and arguments).*

$$s : [Q1, Q2, \ldots] \Leftrightarrow s : [A1, A2, \ldots] \quad (1)$$
$$\text{if} \quad Q1 \Leftrightarrow A1,\ Q2 \Leftrightarrow A2,\ \ldots$$

2. *Structures match if a defined transformation (of one or both) allows them to match.*

$$Q \Leftrightarrow A \quad \text{if} \quad T_Q(Q) \Leftrightarrow T_A(A) \quad (2)$$

and the two following entity matching principles:

3. *A query entity (predicate or argument) can match any answer entity of the same type which contains at least the same information.*

4. *Query term arguments can match any answer argument of the correct entity type.*

3.2 Entity Matching

Principle (3) is applied by a set of word and group matching rules. They are applied simplest first (from individual word matching to complex NG matching), but can be used in any combination via Prolog backtracking.

The requirement that the answer entity contain at least *the same* information as the query is met by:

1. Requiring all groups (NGs, VGs, modifiers) within a query entity to be matched by corresponding groups of the same category in the matching answer entity.

2. Requiring matching predicate VGs to have identical voice and predicate-name attributes, and requiring any modifiers in the query groups to be matched in the answer group.

3. Requiring matching argument NGs/PPs to have identical entity types, and requiring each constituent (word/group) of a query NG to be matched in the answer NG.

4. Requiring matching words to have identical stems.

Note that there is no requirement for every word in a VG to be matched, or for identity of prepositions - only predicate names and PP types must match. For NGs, number is not required to match - a singular query can be matched by a plural answer. Determiners (except universal quantifiers) are also not required to match.

In this way, the example query Q can only be matched by answers where an existential argument role is filled by a *president* with an attached modifier *of the USA*.

The requirement that the answer entity contain *at least* the same information is met by:

1. Allowing answer arguments to contain any extra information (modifiers, adjectives etc.) not required for the matching described above.

2. Allowing query adjectives/adverbs to be matched by equivalents with at least the same comparative degree.

This allows answers about, say, *the 43rd white male president of the USA* to be acceptable.

Principle (4) is applied directly, by allowing query term arguments (groups consisting of query words, e.g. *who*, *how high*, *which city*) to match only entities of the same entity type.

3.3 Structural Transformation

Structural transformations are required to allow general concepts of sentence structure equivalence to be captured. These general equivalences include:

1. Active-passive: a PA-structure with passive voice and an agentive PP argument can be transformed into the equivalent active PA-structure.

2. Existential equivalence: a state-structure can have the order of its arguments reversed.

3. Predicative nouns: a state-structure, or a PA-structure with a generic utility predicate (e.g. *do*, *have*, *make*), one of whose

arguments contains a noun whose lemma corresponds to a predicate, can be transformed into an equivalent PA-structure with this lemmatised predicate.

4. Predicative prepositions: a similar transformation can be applied to structures with a PP argument whose type corresponds to a predicate.

5. Quantity "possession": a PA-structure with a predicate of possession, one of whose arguments is of numerical quantity type (height, cost etc.), can be transformed into the equivalent state-structure.

These transformations allow less direct examples such as *The USA has a new president, George Bush* to match query Q. Testing on a specially created corpus showed that the majority of cases required at least one transformation.

4 Conclusions

The extraction of simple relational structures, together with the principles of matching and transformation, allowed the system to deal with a large number of query types and answer passage phenomena. These range from simple phenomena such as extra unnecessary content in answer passages, to those which could be more problematic for IR-based approaches such as treatment of relative clauses, and predicates expressed by nouns and prepositions. Some example phenomena are shown in table 1 - other phenomena such as passive constructions and verb nominalisation can also be dealt with but cannot be illustrated using the current running example. The matching of query terms also allows suitable portions of the answer text to be identified for use as the system output.

4.1 Performance

The unavailability of the TREC document corpus, together with the restriction to a manually specified lexicon, prevented assessment by methods which could be meaningfully compared with other systems. In order to test the system on a wide range of answer structure phenomena, while keeping the demands on the manually specified lexicon to a minimum, specially created examples were used rather than real-world (e.g. newspaper) text.

Results were encouraging, however. Precision was high, with few non-answers being falsely allowed. Recall was lower: this was partly due to problems with the parser, which could be solved by use of a more fully developed and fully robust shallow parser, and partly due to the fact that the transformation approach is not inherently robust to new structural phenomena which require transformation. A real-world system based on this approach would require a large amount of training data in order to allow the set of transformations to be determined and specified in the most general form - this was not available during the course of this project.

4.2 Further Work

4.2.1 Into the Real World

The restricted scope of this experiment allowed manual specification of all required information in a lexicon, including the predicative properties of nouns and prepositions. Extension of this method to a real-world system is not realistic. The problem can be seen by considering a typical question from TREC-8, which began:

Q1 *Who is the author of the book [...]?*

The answer in this case was of the form *[...], by Hugo Young.* The transformation approach could deal with this, but needs the information that *by* and *author* can both express the same predicate *write.* This kind of knowledge might be extractable from lexical resources such as CIDE+ (**?**) or WordNet (**?**). An alternative approach might be to learn such equivalences from a large set of training data containing questions and corresponding (certified correct) answers.

4.2.2 Answer Ranking

The approach taken so far has been purely one of polar decisions as to whether a pas-

Direct Match	*Bush is the president*
Singular/Plural	*Bill and Hillary were the presidents*
Unnecessary Modifiers	*Bush is a Republican president*
Existential (not "be")	*Bush became president ...* *Bush, the president of the USA, ...*
Existential Ordering	*The president is Bush*
PP Type Matching	*The USA's president is Bush*
PP Predicates	*The USA has a new president, Bush*
Coreference	*Bush knows he is president*
Relative Clauses	*Bush, who is now president, ...* *Bush, who ..., is president*
Subordinate Clauses	*Bush, despite ..., is president*

Table 1: Some Example Phenomena

sage contains an answer or not. When presented with many candidate passages, it is likely to be useful to rank in some way those that contain answers. It is hoped that the use of matching and transformations can give a basis for ranking answers based on closeness of match and number of transformations required.

4.2.3 Negatives

Negative sentences are currently treated exactly as their affirmative counterparts. To a certain extent, this approach seems reasonable, as our aim is to identify a passage that answers a query: *Al Gore is not president* would be a perfectly acceptable and informative answer to the query *Is Al Gore president?*. It is less clear for wh-questions, however: the same example is probably not a useful answer to our original query Q, and could be positively misleading if the system attempted to answer the question directly (so producing the output *Al Gore*). A solution might be to treat negatives as adverbial modifiers, creating a "negative" feature for VGs or structures, although complex cases will require some thought.

4.2.4 Ellipsis, Inference, Reasoning

Elliptical expressions will cause problems for the predicate-argument approach, and will require some resolution or reconstruction procedure to be treated within this framework. Answers that require inference will require, as a minimum: some real-world knowledge; a logical capability such as theorem-proving; calculation capability; pragmatic theory. These areas are not being considered for immediate work.

References

Steven Abney, Michael Collins, and Amit Singhal. 2000. Answer extraction. In *Proceedings of the Sixth Applied Natural Language Processing Conference*, pages 296–301. Association for Computational Linguistics, Morgan Kaufmann, April-May.

James Allan, Jamie Callan, Fang-Fang Feng, and Daniella Malin. 1999. INQUERY and TREC-8. In *The Eighth Text REtrieval Conference (TREC 8)*. NIST Special Publication 500-246.

E. J. Briscoe. 1994. Parsing (with) punctuation etc. Rank Xerox Research Laboratory, Grenoble, MLTT-TR-002.

E. J. Briscoe. 1996. The syntax and semantics of punctuation and its use in interpretation. In B. Jones, editor, *Punctuation in Computational Linguistics*, pages 1–8. Proceedings of ACL SIGPARSE Workshop.

G. V. Cormack, C. L. A. Clarke, C. R. Palmer, and D. I. E. Kisman. 1999. Fast automatic passage ranking (multitext experiments for TREC-8. In *The Eighth Text REtrieval Conference (TREC 8)*. NIST Special Publication 500-246.

Olivier Ferret, Brigitte Grau, Gabriel Illouz, Christian Jacquemin, and Nicolas Masson. 1999. QALC - the question-answering program

of the language and cognition group at LIMSI-CNRS. In *The Eighth Text REtrieval Conference (TREC 8)*. NIST Special Publication 500-246.

Sylvia Knight. 2000. *First Year Report*. Ph.D. thesis, Computer Laboratory, University of Cambridge. In publication.

Chuan-Jie Lin and Hsin-Hsi Chen. 1999. Description of preliminary results to TREC-8 QA task. In *The Eighth Text REtrieval Conference (TREC 8)*. NIST Special Publication 500-246.

Kenneth C. Litkowski. 1999. Question-answering using semantic relation triples. In *The Eighth Text REtrieval Conference (TREC 8)*. NIST Special Publication 500-246.

Kenneth C. Litkowski. 2000. Syntactic clues and lexical resources in question-answering. In *The Ninth Text REtrieval Conference (TREC 9)*. NIST Special Publication 500-XXX.

Dan Moldovan, Sanda Harabagiu, Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju, and Vasile Rus. 1999. LASSO: A tool for surfing the answer net. In *The Eighth Text REtrieval Conference (TREC 8)*. NIST Special Publication 500-246.

Thomas S. Morton. 1999. Using coreference in question answering. In *The Eighth Text REtrieval Conference (TREC 8)*. NIST Special Publication 500-246.

Matthew Purver. 2000. Simplistic question answering. M.Phil. thesis, Computer Laboratory, University of Cambridge.

Dragomir R. Radev, John Prager, and Valerie Samn. 2000. Ranking suspected answers to natural language questions using predictive annotation. In *Proceedings of the Sixth Applied Natural Language Processing Conference*, pages 150–157. Association for Computational Linguistics, Morgan Kaufmann, April-May.

Karen Sparck Jones and Branimir Boguraev. 1987. A note on a study of cases. *Computational Linguistics*, 13(1-2):65–68, January-June.

Rohini Srihari and Wei Li. 2000. A question answering system supported by information extraction. In *Proceedings of the Sixth Applied Natural Language Processing Conference*, pages 166–172. Association for Computational Linguistics, Morgan Kaufmann, April-May.

Tomek Strzalkowski. 1997. Robust text processing in automated information retrieval. In Karen Sparck Jones and Peter Willett, editors, *Readings in information retrieval*, pages 317–322. Morgan Kaufmann.