Building a repository of background knowledge using semantic skeletons

Boris Galitsky

School of Computer Science and Information Systems
Birkbeck College, University of London
Malet Street, London WC1E 7HX, UK
galitsky@dcs.bbk.ac.uk
http://www.dcs.bbk.ac.uk/~galitsky/NL/book

Abstract

We build the knowledge representation machinery for answering complex questions in poorly formalized and logically complex domains. Answers are annotated with deductively linked logical expressions (semantic skeletons), which are to be matched with formal representations for questions. Technique of semantic skeletons is a further development of our semantic header-based approach to question answering.

The question-answering technique has been implemented for the financial and legal domains, which are rather sophisticated on one hand and requires fairly precise answers on the other hand.

Introduction

Domain-specific ontologies are an important component of question-answering (Q/A). While the keyword search and open domain question answering are oriented for a wide (horizontal) domain, handling relatively simple questions containing an entity and its attribute, this is not the case for a legal, financial or business Q/A system. Representation of the above knowledge, oriented to the general audience, is much less structured and requires much richer set of entities than a natural language interface to SQL databases (Maybury 2000, Popescu et al 2003, Galitsky 2003). Furthermore, the structure of links between these entities is significantly more complex in such domains.

Domain-specific ontology for Q/A must be designed in a way to deliver a limited portion if information which:

1. is adjusted to the question;
2. is linked to additional resources if they are necessary;
3. indicates its position in the taxonomy of given Q/A domain;
4. is consistent with other answers and provides a uniform coverage of Q/A domain.

Earlier studies into design of natural language-based and expert systems showed that adequate commonsense reasoning is essential for answering complex questions

(Winograd 1972). A number of recent studies have shown how application of advanced reasoning is helpful to compensate for a lack of linguistic or domain-dependent knowledge while answering complex questions (Ng et al 2001, Moldovan et al 2002, Pasca 2003, Rus and Moldovan 2002, Baral et al 2004, Galitsky 2004).

In our earlier studies we have explored what forms of reasoning can overcome the bottleneck of a limited accuracy of delivered answers. Default logic has been shown to provide a significant assistance disambiguating questions formed by a domain non-expert (Galitsky 2005). An architecture for merging vertical question answering ontologies has been proposed in (Galitsky and Pampapathi 2005). In this paper we continue development of the knowledge representation and reasoning technique for building sharable reusable ontologies for answering complex questions.

The technique of semantic headers (SH, Galitsky 2003) is intended to represent and reason about poorly structured knowledge, manually extracted from text and match it with formalized questions. Having undergone the commercial evaluation, this technique demonstrated the superior performance in the market niche of expensive question answering systems, requiring a substantial domain-representation work of knowledge engineers. However, its accuracy and complexity of delivered advices is much higher than that of open-domain question answering with automatic annotation. SHs are special logical forms oriented to represent partial (most important) information from a textual document.

Semantic skeletons (SSK) extend the functionality of question answering by means of commonsense reasoning machinery. Designed for the above market niche, a semantic skeleton – enabled knowledge domain provides a better coverage of a totality of possible questions. This is due to the fact that an "emergent" question is expected to be deductively linked to one or more of the existing annotated answers by application of commonsense reasoning, inherent to SSK. Moreover, SSK expressions closer follow natural language expressions than knowledge representations which abstract away from natural language means. Hence SSK technique seems to be a good candidate for building a domain-specific repositories for Q/A. A number of encoded domains is available for reuse at www.dcs.bbk.ac.uk/~galitsky/NL/book .

To illustrate the target complexity of questions the proposed repository will provide knowledge for, we present questions from Mortgage domain. NLP system needs to handle up to four entities; neither keyword search–based nor statistical nor syntactic match can provide satisfactory information access in vertical domains.

> How much lower is an adjustable rate mortgage compared to a fixed rate loan?
> Does the "start" rate quoted by lenders on a loan stay in effect for the term of the mortgage?
> How can I avoid negative amortization on an adjustable rate mortgage?
> How risky is a 125 percent loan to value second mortgage?

The desired suite of features we are attempting to achieve by SSK – based knowledge representation machinery is as follows:

1) simplicity and expressive power;
2) capability to reason with incomplete information;
3) existence of a well developed programming methodology;
4) availability of rather efficient reasoning features;
5) encoding defeasible relations, defaults, causal relations, argumentations, and inheritance hierarchies (Galitsky 2005);
6) being elaboration-tolerant knowledge base, i.e., be able to accommodate new knowledge without doing large-scale modification.

Building semantic headers of answers

The problem of question answering in a vertical domain is posed as building a mapping between formal representations of the fixed set of answers and formal representations of possible questions. The technique of semantic headers is intended to be the means of conversion of an abstract textual document into a form, appropriate to be associated to a question and to generate an answer from pre-designed textual components (Galitsky 2003). Only the data, which can be explicitly mentioned in a potential query, occurs in semantic headers. The rest of the information, which would be unlikely to occur in a question, but can potentially form the relevant answer, does not have to be formalized.

Let us consider the *Internet Auction* domain, which includes the description of bidding rules and various types of auctions.

"Restricted-Access Auctions. This separate category makes it easy for you to find or avoid adult-only merchandise. To view and bid on adult-only items, buyers need to have a credit card on file with eBay. Sellers must also have credit card verification. Items listed in the Adult-Only category are not included in the New Items "

What is this paragraph about? It introduces the "Restricted-Access" auction as a specific class of auctions, explains how to search for or avoid selected category of products, presents the credit card rules and describes the relations between this class of auctions and the highlighted sections of the Internet auction site. We do not change the paragraph in order to adjust it to the potential questions answered within it; instead, we consider all the possible questions this paragraph can serve as an answer to:

What is the restricted-access auction? This question is raised when a customer knows the name of the specific class of auction and wants to get more details about it.
What kind of auctions sells adult-only items? How to avoid adult-rated products for my son? Do you sell adult items? These are similar questions, but the class of auctions is specified implicitly, via the key attribute *adult-only.*

When does a buyer need a credit card on file? Why does a seller need credit card verification? These are more specific questions about what kind of auctions requires having credit cards on file, and what is the difference in credit card processing for the auction seller/buyer. The above paragraph serves as answer to these questions as well, and since we are not dividing the paragraph into smaller fragments, the question addressee will get more information than he/she has directly requested; however this additional information is relevant to that request.

Below is the fragment of SH repository which lists of headers for the above answer.

```
auction(restricted_access):-addAnswer(restrAuction).
product(adult):-addAnswer(restrAuction).
seller(credit_card(verification,_),_):-
                    addAnswer(restrAuction).
credit_card(verification,_)):-addAnswer(restrAuction).
buyer(credit_card(reject(_,_),_),_):-
                    addAnswer(restrAuction).
bidder(credit_card(_,_),_):-addAnswer(restrAuction).
seller(credit_card(_,_),_):-addAnswer(restrAuction).
what_is(auction(restricted_access,_),_):-
                    addAnswer(restrAuction).
```

What happens when the system receives a question e.g. *What if buyers' credit card is not approved immediately when I shop at restricted access auction?* Firstly, it is translated into representation (we do not present the details here)

```
buyer(credit_card(_,_),
shop(auction(restricted_access,_),_).
```

Secondly, we search for a proof of this conjunction, given all available SHs, including ones for the above answer. The first conjunctive member will be satisfied by the clause

```
buyer(credit_card(reject(_,_),_),_):-
                    addAnswer(restrAuction).
```

Finally, the predicate *addAnswer(restrAuction)* is called and the above paragraph is added to the current answer, which may consists of multiple pre-prepared ones. The

second conjunctive member might be satisfied with another SH, which would deliver another answer.

Now we will briefly introduce a generic set of SHs for an entity. For an entity e, its attributes $c_1, c_2, ...,$ variables over these attributes $C, C_1,$ as well as other involved entities $e_1,...,$ and the ID of resultant answer, SHs look like the following:

$e(A):-var(A), answer(id)$. This is a very general answer, introducing (defining) the entity e. It is not always appropriate to provide a general answer (e.g. to answer *What is tax?*), so the system may ask a user to be more specific:

$e(A):-var(A), clarify([c_1, c_2, ...])$. If the attribute of e is unknown, a clarification procedure is initiated, suggesting the choice of an attribute from the list $c_1, c_2,...$ to have the specific answer about $e(c_i)$ instead of general one for $e(\_)$ (definition for e).

$e(A):-nonvar(A), A = c_1, answer(id)$. The attribute is determined and the system outputs the answer associated with the entity and its attribute.

$e(e_1(A)):-nonvar(A), A= c_1, e_1(A)$.

$e(e_1(A),e_2):-nonvar(A), A \neq c_1, e_2(\_)$. Depending on the existence and values of attributes, an embedded expression is reduced to its innermost entity that calls another SH.

$e(A,id)$. This (dead-end) semantic header serves as a constraint for the representation of a complex query.

Note that *var/1* and *nonvar/1* are the built-in PROLOG metapredicates that obtain the respective status of variables.

From the perspective of logic, the choice of SHs to be matched against a formal representation of a query corresponds to the search for a proof of this representation, considering SHs as axioms.

Hence, SHs of answer are formal generalized representations of potential questions which contain the essential information from answers and serve to separate them, being matched with formal representations of questions. SHs are built taking into account the set of other semantically close answers, and the totality of relevant questions, semantically similar to generic questions above.

Semantic skeletons for common sense

Evidently, a set of SH represents the associated answer with the loss of information. What kind of information can be saved given the formal language that supports semantic headers?

When we extract the answer identifying information and construct the semantic headers we intentionally lose the commonsense links between the entities and objects used. This happens for the sole purpose of building the most robust and compact expressions for matching with the query representations. Nevertheless, it seems reasonable to conserve the answer information which is not directly connected with potential questions, but useful for completeness of knowledge being queried. A semantic skeleton (SSK) can be considered as a combination of semantic headers with *mutual explanations* of how they are related to each other from the answers perspective. SSKs are domain-specific and coded manually by knowledge engineers
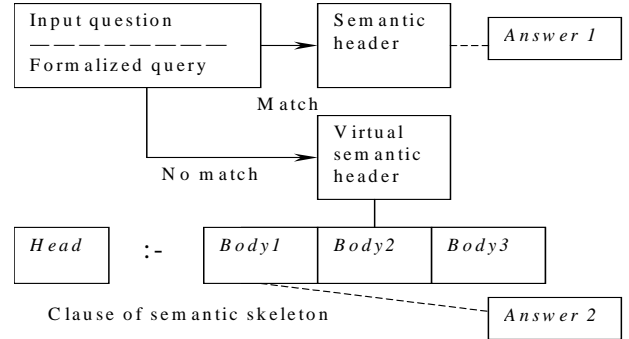

Fig. 1: Illustration for the idea of semantic skeletons

SSKs serve the purpose of handling the queries not directly related to the informational content of the answers, represented by semantic headers. For an answer and a set of semantic headers, an SSK derives an additional set of *virtual* headers to cover those questions which require a deductive step to be linked with this answer. In other words, a semantic skeleton extends a set of questions which is covered by existing semantic headers towards the superset of questions, deductively connected with the former ones. It happens during a question answering session, unlike the creation of regular SHs which are built in the course of domain construction.

Yielding virtual SHs in a domain can be written as $\forall a$ $SSK : \{SH(a)\} \rightarrow \{vSH(a)\}$, where $\{SH(a)\}$ is the set of original semantic headers for answer $a,$ and $\{vSH(a)\}$ is the set of virtual semantic headers derived from SSK for an answer a. A virtual semantic header (vSH) can be yielded by multiple answers (Galitsky 2003). However, a vSH cannot be a regular header for another answer (note that two semantic headers for different answers are allowed to be deductively linked): $\forall a,a'$ $vSH(a) \cap SH(a')=\varnothing$. Hence, a vSH for a query is an expression that enters a clause of the semantic skeleton and can be matched with the translation formula of a query or its conjunctive component. In the latter case, the terms of mentioned clauses must not match with the negations of the (conjunctive) components of that translation formula.

The idea of SSK is depicted in Figure 1. The input query is matched against the vSHs if there is no appropriate regular semantic header to match with. Virtual semantic headers are obtained given the terms of SSK clauses. The SHs are assigned to answers directly. However, vSHs are assigned to answers via clauses. Both *Answer1* and *Answer2* may have other assigned regular and virtual SHs.

For example, imagine a semantic header *tax(income)* that is intended to handle questions about *tax brackets* in the *Tax* domain: how the *tax* amount depends on *income*. Evidently, this answer would be a relevant one to the question *What would my tax be if I lost my job last year?* Since *losing a job* is not directly related to *tax* (the former is deductively linked to the latter via *income*, *job(lost)→ not income(_)*), it would be unreasonable to have a special semantic header to link *tax* and *job-lost*. Therefore, the expression *job(lost)* serves as a virtual semantic header in the TAX domain, being generated dynamically from the clause *job(lost)→ not income(_)*, instead of being a regular one. If we do not use the regular semantic header instead of the virtual one for the entities which are neither deductively nor syntactically linked in a query, it would damage the domain structure and lead to an excess number of semantic headers. Indeed, this used to happen before the concept of the SSK was introduced.

At the same time, in the IRA domain the *loosing job* scenario is under special consideration, and expressions *ira(tax(income))* and *ira(job(lost))* are expected to be the semantic headers for different answers; one for calculating *tax* on *IRA distribution amount* that depends on *income*, and the other for the special case of *tax* on *IRA distribution* under *employment termination.* Thus a pair (triple, etc.) of entities may form a vSH (that requires a SSK-clause that would yield, generally speaking, multiple links between entities) or, form a regular header, depending on whether these entities are directly semantically or syntactically linked in a query. The clauses of the semantic skeleton are not *directly* used to separate answers, so they can be built as complete as possible irrespectively on the knowledge correlation with other answers. Furthermore, semantic skeletons for a pair of answers may overlap, having the common clauses.

SSK handling of complex questions

Semantic skeletons are helpful for formalizing queries which are conjunctions of multiple terms (This happens for complex queries consisting of two or more components, for example *Can I qualify for a 15-year loan if I filed bankruptcy two years ago with my partner? → loan(qualify)& bankruptcy(file, 2years))*. If a term is either matched against none of the SHs or delivers too many of them, then this term can serve as a virtual SH. In the Table 1 below we analyze various cases of the satisfaction (matching) of a translation formula with two terms against regular and virtual SHs.

In a complex question, we distinguish two parts: *leading* and *assisting*. These parts are frequently correlated with the syntactic components of questions. In accordance to our observations (Galitsky 2003), the leading part is usually more general than the assisting

part. One of the canonical examples of a complex query is as follows (note that we accent its semantic structure rather than its syntactic one): *How can I do this* **Action** *with that* **Attribute***, if I am* **AdditionalAttribute1** *of / by / with / from/and* **AdditionalAttribute2***. We enumerate the properties of our informational model above as follows:

1) *Action* and its *Attribute* are more important (and more general) than *AdditionalAttribute1* and *AdditionalAttribute2;* they are more likely to point to a specific topic (group of answers).

2) *AdditionalAttribute1* or *AdditionalAttribute2* are more specific and more likely to point to an exact answer.

Therefore, if we mishandle the leading part, we would probably dissatisfy the assisting one and find ourselves with a totally irrelevant answer. Conversely, if the assisting part is mishandled when the leading part has been matched, we frequently find ourselves in the situation where we have either a marginally relevant answer or too many answers. In general the variety of assisting components is much higher than that of the leading ones. Therefore, it is reasonable to represent assisting expressions as vSHs. Proper domain coding is intended to achieve the coverage of all leading components, so most of them are represented by SHs.

Assuming that the complete (linking all entities) SSK is built, we approach the Table 1. It shows the rules for building SSK ontology to handle complex questions. There are a few additional models for complex questions. When a question does not follow our two-part model, SHs and SSKs can be individually built to handle particular asking schema. However, if no special means have been designed for a (semantically) deviated question, the resultant answers may be irrelevant.

As a final SSK example, we show how to handle the question *What if* **my** *credit card is not approved immediately when I shop at restricted access auction?* To the domain discussed above. If a *buyer* is mention in one way or another, SH technique would deliver the answer, but not otherwise. A simple SSK
buyer(person,_):- shop(person, auction(Any,_))
is required to express knowledge that a shopper is a potential buyer. Obviously, such kind of SSKs assure that a wide class of complex questions is properly understood.

Evaluation of Q/A improvement due to SSK

A series of tax return assisting, investment, mortgage and financial companies have been using the Q/A systems with SSK-based knowledge representation. Q/A can replace human agents, automatically answering tax questions in up to 85% of all cases. Human agents were ready to get involved in the Q/A process in case of a failure of the automatic system.

First term (leading)	Second term (assisting)	Resultant answer and comments
Matches with multiple SHs	Matches with a single SH(a)	The case for a "dead end" semantic header for an assisting term, which reduces the number of matched SHs for the first term, having the common variable (answer Id). Answer a is chosen in this situation which had required special preparation.
Matches with a single SH(a)	Matches with multiple SHs	Answer a from the leading term is taking over the multiple ones delivered by the assisting term. The confidence of that right decision would grow if the assisting term matches with a vSH of a; otherwise, we conclude that the assisting component is unknown.
Matches with a single SH(a) or only vSH(a)	Matches with a single SH(a) or only vSH(a)	The answer is a. Higher confidence in the proper decision would be established if the leading term matches with SH and the assisting one with vSH.
Matches with a set of SH(a), $a \in A$	Matches with a single SH(a)	The answer is a. The assisting term matches against a single semantic header and therefore reduces the answers yielded by the leading term.
Matches with a set of SH(a), $a \in A$	Matches with a vSH(a) only	All answers from A. The fact that the assisting term matches against a virtual semantic header is insufficient evidence to reduce the answers yield by the leading term.
Matches with a set of vSH(a), $a \in A$	Matches with a single SH(a)	The answer is a. The assisting term contributes to that decision, consistent with the match of the leading term.
Matches with a set of vSH(a), $a \in A$	Matches with a vSH(a) only	All answers from A. The resultant confidence level is rather low and there is insufficient evidence to reduce the answers yielded by the set of vSH of the leading term.
Matches with a single SH(a)	Matches with a virtual SH(a') only	The answers are both a and a' except in the case when the first term matches with virtual SH(a') and the answer is just a.
Matches with a virtual SH(a) only	Matches with a virtual SH(a') only	All answers which are yielded by vSH(a). The question is far from being covered by SH, so it is safer to provide all answers, deductively linked to the leading term and ignore the assisting term.
Matches with a set of vSH(a): $a \in A$	Matches with a set of vSH(a'): $a' \in A'$	We find the answer which delivers most of vSH in $\{ vSH(a) \cap vSH(a'): a \in A, a' \in A' \}$

Table 1: Various cases of matching (disagreements) for the leading and assisting components of complex query. First and second columns enumerate matching possibilities for the leading and assisting components.

In particular, the suite of legal (family law) domain has been created, which covers sufficient information for the general audience of using about 1000 answers in the principle and accompanying domains. The domain includes more than 240 entities and more than 2000 of their parameters in these sub-domains. More than 3000 semantic headers and semantic skeletons were designed to provide an access to these answers. During the beta testing, the Family Law adviser was subject to evaluation by a few hundred users starting from the summer of 2000. Customers had the options to provide the feedback to the system concerning a particular answer if they were dissatisfied or not fully satisfied with it (too long, non-relevant, partially relevant, etc.). With the answer size not to exceed 6 paragraphs, the system correctly answers more than 70% of all queries, in accordance to the analysis of the Q/A log by the experts. Even with 82% resultant accuracy (Table 2), which is relatively low for traditional pattern recognition systems, over 95% of customers and quality assurance personnel agreed that the legal advisor is the preferable way of accessing information for non-professional users.

Usually, customers tried to rephrase questions in case of the system's misunderstanding or failure to provide a response. Reiteration (rephrasing the question) was almost always sufficient to obtain the required information. At the beginning of the evaluation period, the number of misunderstood question was significantly exceeded by the number of answers not known by the system. This situation was dramatically reversed later, however the number of misunderstood questions was monotonically decreasing in spite of an increase in overall represented knowledge.

Use of SSK allowed increasing the percentage of correctly answered questions from 60 to 67 (Table 2): about 7 % of questions are indirect and require to apply a commonsense reasoning to link these questions to formalized answer components. In 2% of cases vSHs were built but they derived multiple inconsistent SHs because of a lack of a specific knowledge (which has been added later). As one would expect applying SSK technique, the decrease of cases with a lack of understanding (6%) was higher than (twice as much as) the decrease of cases with misunderstanding (3%). To estimate the results of matching procedure without a SSK, the reader may hypothetically replace matching with a virtual SH by "no match" and track the number of situations with the lack of proper handling where SSKs are not in use.

Development step	Source of questions	Correct answer	No knowledge	No understanding	Misunderstanding
Initial coding	Initially designed (expert) questions for SH	47	0	35	18
Testing & reviewing of initial coding	Initially designed & accompanying questions	52	18	21	9
Adjustment to testers' questions	Additional and reformulated and rephrased testers' questions	60	15	10	15
Adding SSKs without Table1 rules	Domain-specific knowledge	63	17	7	13
Adding SSKs with Table1 rules	**Domain-specific knowledge**	**67**	**17**	**4**	**12**
Adjustment to content providers' questions	More questions, reflecting a different viewpoint	74	8	4	14
Adjustment to users' questions	No additional questions	82	4	4	10

Table 2: The progress of question answering enhancement at consecutive steps of domain development (%). SSK step is shown in bold. *Commonsense domain knowledge* helps to yields questions which were not encoded during initial phase of domain development, but are nevertheless relevant.

Conclusions

Application of the SSK technique to NL Q/A showed the following. There is a superior performance over the knowledge systems based on the syntactic matching of NL queries with the previously prepared NL representation of canonical queries, and the knowledge systems based on fully formalized knowledge. Moreover, the domain coverage of SSK is better than that of SH because a new question can be reduced to existing pre-coded ones by means of commonsense reasoning.

The SSK approach to knowledge representation for Q/A gives a higher precision in answers than the SH and syntactic–matching based ones because it involves the semantic information in higher degree. The SSK technique gives more complete answers, possesses higher consistency to context deviation and is more efficient than the fully formalized ontology-based approach because all information in answers does not have to be obtained via reasoning.

The achieved accuracy of providing an advice in response to a NL question is much higher than an alternative approach to advising in a vertical domain would

provide, including open-domain question answering, an expert system on its own, a keyword search, statistical or syntactic pattern matcher. Indeed, SSK technique approaches the accuracy of a Q/A in a fully-formalized domain, assuming the knowledge representation machinery obeys the features outlined in the Introduction.

In this paper we described the design of a single Q/A domain. To merge multiple vertical domains to form a horizontal one, we suggest a multiagent question answering approach, where each domain is represented by an agent which tries to answer questions taking into account its specific knowledge. The meta–agent controls the cooperation between question answering agents and chooses the most relevant answer(s). In (Galitsky & Pampapathi 2005) we argue that multiagent question answering is optimal in terms of access to business and financial knowledge, flexibility in query phrasing, and efficiency and usability of advice.

Using semantic resources like WordNet, automated statistic-based annotation systems and commonsense ontology resources would decrease the cost of domain development at the expense of lower accuracy. In our future studies we plan to perform the above integration for less narrow domains and less complex questions, but automated annotation.

References

Galitsky, B. 2003. Natural Language Question Answering System: Technique of Semantic Headers. Advance Knowledge International, Australia.

Galitsky, B., Pampapathi, R. 2005. Can many agents answer questions better than one? First Monday v 10 n1. http://firstmonday.org/issues/issue10_1/galitsky/index.html

Galitsky, B. 2005. Disambiguation via Default Reasoning for Answering Complex Questions. Intl J. AI Tools N1-2 157-175.

Pasca, M. 2003. Open-Domain Question Answering from Large Text Collections *CSLI Publication series.*

Baral, C., Gelfond, M. and Scherl, R. 2004. Using answer set programming to answer complex queries. In *Workshop on Pragmatics of Question Answering at HLT-NAAC2004*

Moldovan, D., Pasca, M., Harabagiu, S. and Surdeanu, M. 2002. Performance issues and error analysis in an open-domain question answering system. In ACL-2002.

Rus, V. and Moldovan, D. 2002. High Precision Logic Form Transformation, Intl J. AI Tools, vol. 11 no. 3.

Maybury, M.T. 2000. Adaptive Multimedia Information Access - Ask Questions, Get Answers. *First International Conference on Adaptive Hypertext AH 00.* Trento, Italy.

Ng, H.T., Lai Pheng Kwan, J. and Xia, Y. 2001. Question Answering Using a Large Text Database: A Machine Learning Approach. *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing. EMNLP 2001.* Pittsburgh, PA

Popescu, A.-M., Etzioni, O. and Kautz, H. 2003. Towards a Theory of Natural Language Interfaces to Databases. *Intelligent User Interface.*

Winograd, T. 1972. Understanding natural language. NewYork: Academic Press.