DIMACS Technical Report 2002-10
April 2002

Use of virtual semantic headers to improve the coverage of natural language question answering domains

by

Boris Galitsky[1]

[1] KnowledgeTrail, Inc., 9 Charles Str. Natick MA 01760 USA.
http://dimacs.rutgers.edu/~galitsky
(galitsky@dimacs.rutgers.edu)

The author gratefully acknowledges the grant NSF STC 91-19999 for the support while visiting DIMACS.

ABSTRACT

We report on the knowledge representation mechanism designed for natural language question-answering system to function in such poorly-formalized and heterogeneous domains as the financial, legal, pharmaceutical and psychological. The system is oriented to provide a customized expert advice, given the pre-designed set of textual templates and the database that contains customers' profiles and preferences, parameters of products and services, etc. Question answering is based on matching a formal representation of a query against the formalized representations of answers' essential ideas (semantic headers of these answers). Semantic headers are designed to be independent on the query phrasing and are the means of approximate reasoning while generating the most relevant advice. A semantic skeleton of an answer includes semantic headers and deductive links between them and their entities, based on the common-sense domain knowledge. Semantic skeleton includes the virtual semantic headers, which do not have to be explicitly programmed but are generated on the fly, using the clauses of semantic skeleton, to be matched with a question.

We present the evaluation of the released question-answering system, advising the customers of H&R Block and CBS Market Watch on various taxes and associated legal issues starting from 1999. Domain development and maintenance implications of semantic header technique, answer accuracy, meaning deviations and overall customer impressions are analyzed.

Introduction

In recent years, question-answering (Q/A) systems are becoming important means for finding relevant information in a variety of domains. Attracting new customers by the valuable content and keeping them surfing a portal is essential to establish the competitive advantage. Proper knowledge representation for natural language (NL) Q/A allows addressing the particular topic and obtaining an answer in an exact, convenient and immediate way.

The role of Q/A systems to introduce customers to new knowledge domains is rather high relatively to the traditional learning and information access methodologies, including textual, hyper-textual and menu-based sources. We believe Q/A is one of the best ways to introduce a novice customer to a new domain without making him/her to obtain a prior knowledge of its overall structure. Under robust knowledge representation, vertical-domain natural language Q/A allows obtaining the specific portion of information independently on the accompanying topics and on the degree of familiarity with the domain terminology (that is rather extensive in such fields as finance, legal, psychological and medical).

In our previous studies, we have presented a wide manifold of financial domains (Galitsky 2001), including tax, mortgage, real estate, insurance, retirement, investment, banking, legal advises literature search, geography and politics, sports, psychology and others. Details on syntactic and semantic processing (Galitsky 1999), logical meta-programming implementation, performance evaluation and comparison with other academic and commercial question-answering systems can be found elsewhere (Galitsky 2000a). In this paper, we focus on the knowledge representation issues of an NL Q/A application.

Our knowledge representation component is quite different from such NL Q/A systems as a product choice domain that is based on database search, or a horizontal domain querying, usually based on syntactic match (See Galitsky 2000b for comparison). Furthermore, due to domain development and maintenance issues, traditional approaches to representation of semi-structural knowledge, (e.g. using mark-up languages), were found to be inconvenient. The reason is that under the above approaches, domain cannot be naturally split into the query-oriented and user details-oriented parts, required for Q/A. In addition to the pre-edited units of textual answers, represented via semantic headers and stored within the system, Q/A needs to involve the external data sources of various natures.

Because of the complicated nature of questions in the above domains (See Table 1 for the sample questions), knowledge representation means must satisfy the large-scale vertical domain requirements of logical complexity and high accuracy. The complexity raise of the structure of links between the entities and heterogeneity of information sources for the selected domains caused the necessity to extend the technique of semantic headers towards the semantic skeletons (SSK) methodology. It adds the advanced reasoning capabilities to the methods, employed in our earlier Q/A domains. SSK technique brings in the deductive links between the semantic headers, completing the commonsense knowledge representation on one hand and keeping the convenience of domain coding, inherent to SH technique, on the other hand. Deductive links between the semantic headers improve the content and query phrasing coverage of a domain, providing a more powerful way to associate a formal representation of a query with the existing semantic headers.

What is the difference between humans and other species in terms of genome?
What is the purpose of extracting the genes responsible for growth control?
Cam animals serve as organ donors for humans?
How does the biotech help to produce crops that resist insects and disease?
How does the gene in cows trigger the release of bovine growth hormone?
What organs are in use for DNA identification?
If my uncle has a Huntington disease, at what age do I have to undergo the genetic test?

Table 1: Sample questions of the genome domain.

To motivate the actual complexity of our knowledge representation system, we needed to demonstrate that keyword-based and syntactic match-based system would not function properly in our domains, unless a customer just uses the company or person name to initiate a search. While building the financial domains, we discovered that rather limited number of structural units (tenths of thousand of potential answers, thousands of answers) covered uniquely a logically complex and poorly formalized domain. It is well known, that the generic vocabulary of business communication does not exceed three to four thousand words. It would require 10-100 times higher number of syntactic structures (headers) to provide the Q/A access to the same set of answers, using syntactic match. We would have to explicitly enumerate the combination of keywords for a syntactic match instead of using the deductive capabilities of the SH approach, which allows us to automatically join entities in the formal representation of a query.

Semantic headers

The technique of semantic headers is intended to solve a problem of conversion an abstract textual document into form, ready for Q/A. The essence of our approach is to divide domain knowledge into two parts:

1) The formalized part. It includes knowledge that is explicitly matched against the querying expressions. This part does not constitute an answer.
2) Non-formalized knowledge in the way of human-readable means (original sources of information). This knowledge does not participate in the estimation whether it is relevant to a particular query, or not.

The first part includes essential knowledge only, and the second part is expected to contain all information valuable for a Q/A user. Suggested division is also based on observation that the level of details, contained in a query, is always much lower that the amount of details, expected in an answer.

There are two opposite approaches to the problem of domain knowledge division into formalized and non-formalized parts. With respect to the degree of formalization, the traditional methodologies are the highest degree of formalization (ontologies) and the least degree of formalization (syntactic match). The first one assumes that complete formal representation of any textual document is possible, and the second one assumes that the textual information is too tightly linked to natural language and it cannot be satisfactorily represented without it.

The first approach brings the advantage of the understanding, based on the full-range reasoning capabilities, but sacrifices the robustness of providing a marginally relevant answer and the necessity of NL generation. Attempts to provide an exact answer, inherent to the fully formalization approach, narrow down the overlap between a query and existing answers. (Under SH, query understanding is posed as a recognition problem: how to find the most relevant answer for a given query, even if there is no directly related answer (Fain&Rubanov 1996, Mirkin 1996)). Also, the condition of logical completeness makes it harder to build an ontology for a vertical domain. As our experience shows, if to build a projection of a full-scale ontology like CYC (Lenat 1998) onto our vertical domain, resultant knowledge would be insufficient and limited in spite of the huge total number of clauses in the whole CYC. The same observation is true for biological (genome) ontologies.

The SH technique is an intermediate approach in respect to the degree of knowledge formalization. Only the data, which can be explicitly mentioned in a potential query, occurs in semantic headers. The rest of information, which would unlikely occur in a question but can potentially form the relevant answer, does not have to be formalized. Pattern recognition methodology (Fain&Rubanov 1996, Mirkin 1996) has the significant influence on how we treat the answers. When we say that a document (or its fragment) serves as an answer to a question we mean that for this question (related to a domains) either (or) holds:

1) The fragments contains the information that is the subject of this question;
2) The fragment is marginally related to the question, but the other fragments have even lower correlation with the question (assuming the question indeed related to the given domain).

SH technique is based on logical programming, taking advantage of its convenient handling of semantic rules (Partee et al 1990, McCawley 1993) on one hand, and explicit implementation of the domain common-sense reasoning on the other hand. The declarative nature of coding semantic rules, domain knowledge and generalized potential queries introduces logical programming as a reasonable tool (Tarau et.al.1999, Dahl 1999). At the same time, the machinery of text annotation by the set of keywords has been proven to leverage the machine learning technique. Instead of using just the keywords as semantic means to represent the meaning of a short textual document (answer), we either use the logical formula where the keywords serve as atoms or apply the pragmatic processing to keyword SH's. Therefore, SH technique can be viewed as some way of merging potential results of statistical approach to Q/A with the logical programming way of matching the formal representation of a query with the formal representation of an answer (semantic header of this answer). What we call “statistical” here is indeed the intuition of a knowledge engineer who has manually looked through a manifold of answers to annotate a given one. In legal and financial domains, where the semantics of conversational language can only be ambiguously mapped into the semantics of the legal language, using just the statistical annotation by keywords does not lead to satisfactory results (Galitsky, 2001).

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. Your card will not be charged. Sellers must also have credit card verification. Items listed in the Adult-Only category are not included in the New Items page or the Hot Items section, and currently, are not available by any title search.* “

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 to adjust it to the potential questions; instead, we consider all the possible questions this paragraph can serve as an answer to by enumerating the semantic headers. Building the semantic headers of a textual document is based on the posing of query understanding problem as the recognition of the best pattern (document, answer). For example, if there is a question such that the paragraph above is a more appropriate answer than any other paragraph from the given domain, then the paragraph above should serve as an answer or at least a part of it. Evidently, knowledge of the other paragraphs and the semantic model of the whole domain are required to build the set of semantic headers for the given paragraph.

What kind of questions can this paragraph serve as an answer to?

What is the restricted-access auction? What kind of auctions sells adult-only items? How to avoid adult-rated products for my son? How do you sell adult items? When does a buyer need a credit card on file? Who needs to have a credit card on file? Why does a seller need credit card verification?

Below is the list of semantic headers for the answer above.

auction(restricted_access,_):-restrictedAuction.
product(adult,_):-restrictedAuction.
seller(credit_card(verification,_),_):-restrictedAuction.
credit_card(verification,_)):-restrictedAuction.
sell(credit_card(reject(_,_),_),_):-restrictedAuction.
bidder(credit_card(_,_),_):-restrictedAuction.
seller(credit_card(_,_),_):-restrictedAuction.
what_is(auction(restricted_access,_),_):-restrictedAuction.

Then the call to *restrictedAuction* will add the paragraph above to the answer.

As another example, we consider an answer from the tax domain:

"*The timing of your divorce could have a significant effect on the amount of federal income tax you will pay this year. If both spouses earn about the same amount of money, getting a divorce before the year ends will save taxes by eliminating the marriage penalty. However, if one spouse earns a significant amount more than the other, waiting until January will save taxes by taking advantage of the married filing jointly status for one last year*."

When we form the canonical questions, we start with the answer's phrasing. Then we try to compose more general questions in the terms, which are not necessarily present in the answer.

What are the tax issues of divorce?
How can timing of my divorce save a lot of federal income tax?
I am recently divorced; how should I file so I do not have a net tax liability?
Can I avoid a marriage penalty?
How to take advantage of the married filing jointly status when I am getting divorced?

Below is the list of semantic headers for the answer above. Note that the third canonical query is represented by a single term (not a conjunctive member) because the first part of the sentence is semantically insignificant (just *divorce, recently* is irrelevant here).

divorce(tax(_,_,_),_):-divorceTax.
divorce(tax(_,_,_), time):-divorceTax.
divorce(tax(file(_),liability,_), _):-divorceTax.
penalty(marriage,_):- divorceTax.
divorce(file(joint),_):-divorceTax.

A generic set of semantic headers for an entity *e*, expressed by a predicate with the same name, and its attributes $a_1, a_2, \ldots$ looks like the following:

e(A):-var(A), clarify($[a_1, a_2, \ldots]$*).* If the attribute of *e* is unknown, clarification procedure is initiated, suggesting choosing an attribute from the list.
e(A):-nonvar(A), $A = a_1$ *, answer(#).* The attribute is determined and the system outputs the answer, associated with the entity and its attribute (# is the answer id).
$e(e_1(A))$*:-nonvar(A),* $A= a_1$, $e_1(A)$.
$e(e_1(A),e_2)$*:-nonvar(A),* $A\neq a_1$, $e_2(\_)$. Depending on the existence and values of attributes, an embedded expression is reduced to its innermost entity that calls another SH.

e(A,#). This semantic headers serves as a constraints for the representation of a complex query *e(A,#), e_2(B,#)* to deliver just an *answer(#)* instead of all pairs for e_1 and e_2. It works in the situation where e_1 and e_2 cannot be mutually substituted into each other.

Before we proceed to the description of semantic skeletons, the final definition of semantic headers is presented. Semantic headers of an answer are the formal generalized representations of potential questions. These representations are built, taking into account the set of other semantically close answers and relevant semantically close questions. Therefore, semantic analysis under the SH technique consists in the formal representation of a question and relating it to a fixed set of answers.

Semantic skeleton of a domain

Evidently, the set of semantic headers represent 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 loose the commonsense links between the entities and objects used. It happens for the sole purpose of building the most robust and compact expression for matching with the query representations. Nevertheless, it seems reasonable to conserve the answer information that is not directly connected with potential questions, but useful for completeness of knowledge being queried. Semantic skeleton can be considered as a set of semantic headers, assigned to answers, together with *mutual explanations* of how they are related to each other (from the answer's prospective). Consult Fig. 1 for the illustration of the idea of a semantic skeleton concept.

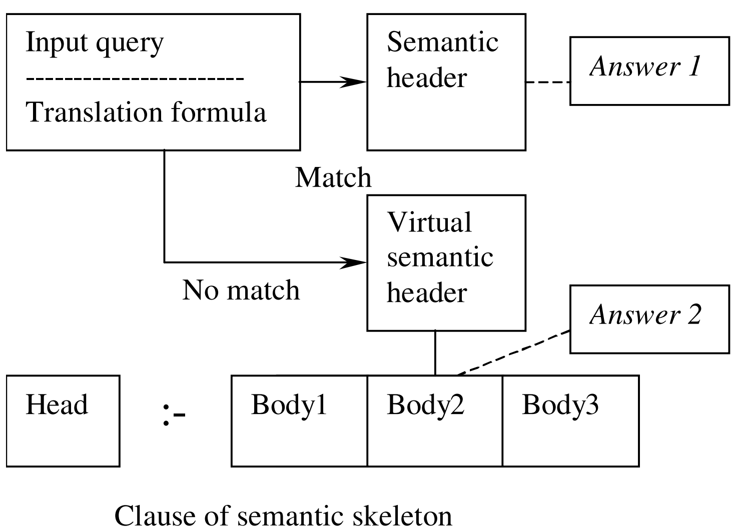

Fig. 1: Illustration for the idea of semantic skeletons. Input query is converted into TF and matched against the virtual semantic headers if there is no appropriate regular semantic header to match with. Virtual semantic headers are obtained given the terms of SSK clauses. The SHs are assigned with answers directly, however, vSHs assigned with answers via clauses. Both *Answer1* and *Answer2* may have other assigned regular and virtual semantic headers.

Semantic skeletons 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 assigned semantic headers, semantic skeleton derives an additional set of *virtual* headers to cover those questions that require a deductive step to be linked with this answer. In other words, semantic skeleton expands a set of questions that is covered by existing semantic headers towards the superset of questions, deductively connected with the former ones. This can be written as

$\forall a$ *SSK : {SH(a)}* $\rightarrow$ *{vSH(a)},* where {*SH(a)*} and {*vSH(a)*} are the sets of original and virtual semantic headers for an answer *a*. A virtual semantic header can be yielded by multiple answers. However, virtual semantic headers cannot be the regular headers for another answer (note that two semantic headers for different answers can be deductively linked): $\forall a'$ $vSH(a) \cap SH(a')=\varnothing$.

Hence, virtual semantic header 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 component. In the latter case, the terms of mentioned clause must not match with the negations of the components of that translation formula. For example, imagine a semantic header *tax(income)* that is intended to handle questions about tax brackets in TAX domain: how the *tax* amount depends on *income*. Evidently, this answer would be a right one for the question *What would be my tax if I lost my job last year?* Since *loosing job* is not directly related to *tax* (the former is deductively linked to the latter via *income*, *job(lost)*$\rightarrow$ *not income*). Therefore, expression *job(lost)* serves as a virtual semantic header in the TAX domain. At the same time, in the IRA domain *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*.

As well as approximation of meaning by semantic herders, semantic skeleton are capable of involving approximate semantic links. For example, various *forms of payment* questions are addressed to the *Internet retailer* domain that usually has an answer about *credit card payment*. How should we handle the questions mentioning *payment by check, money order, wiring* etc? “Pure”SH technique requires enumeration of SHs

payment(check):-credit_card_payment_answer.
payment(money_order) :-credit_card_payment_answer.
payment(wiring) :-credit_card_payment_answer.
payment(credit_card) :-credit_card_payment_answer.

Using SSK clause

payment(X):-member(X, [*check, money_order, wiring, credit_card*]).

One can use the fourth SH above as a regular SH and the first three ones as virtual SHs, involving the clause about forms of payment. The advantage of using SSK is the lower number of SHs to code, clearer semantic structure of a domain and reusability of encoded commonsense knowledge for the similar domains.

To form a semantic skeleton for an answer, the semantic headers need to be deductively linked via the clauses, involving the domain representation entities. The clauses below present the explanation of how a term *divorce* is linked to the terms *marriage, tax, file, separate, joint*. The first clause, completing the set of semantic headers above, introduces the commonsense fact that being *divorced* is an opposite entity to being *married*. The second clause is saying that if a couple was filing joint tax return before the divorce, it is filing separate tax return afterwards. Enumeration of terms within a clause may be used to express the temporal relationship between the respective entities (how they occur in time).

divorce(_):- not marriage(_). *tax(file(separate)):- file(joint), divorce(_).*

Using just the semantic headers we can answer the divorce questions without knowing that *divorce* stops *marriage*! Surprisingly, one does not need to know that fact to separate answers about *divorce*. Intuitively, a speaker would definitely need the basic commonsense facts to *talk* about a topic but can do *without them* answering questions about that topic, including rather specific and deep questions. Semantic skeletons come into play, in particular, to semantically link the basic domain entities. At the same time, semantic skeleton is a *least* knowledge required to have all the entities linked.

The clauses of semantic skeleton are not *directly* used to separate answers, so they can be built as complete as possible irrespectively on the knowledge correlation with the other answers. Furthermore, semantic skeletons for a pair of answers may overlap, having the common clauses.

Note that the predicates we use to describe the tax issues of marriage for Q/A are different from one we would use to better formalize the domain itself. SH-oriented predicate *divorce* ranges over its attributes, “squeezed” into a single argument, and an *extended* predicate *divorce* in a logical program would have two arguments ranging over the divorce parties and other arguments for the various circumstances of divorce action. Frequently, more semantically transparent (*extended*) predicates are required to form the semantic skeleton; analogous semantic header predicates should then be mutually expressed via the purified ones. These definitions do not usually bring in constraints for the attributes of semantic header predicates. Below is the semantic skeleton for the same answer in *extended* predicates. The first argument of the extended predicate *tax* (that is indeed a metapredicate) ranges over the formulas for a *taxpayer's states*; therefore, *tax* is a metapredicate.

```
divorce(Husband, Wife, Date, Place,…):-divorce(_).
marriage(Husband, Wife, Date, Place,…):-marriage(_).
tax( (pay(Husband)&pay(Wife)), file (separate),_):- tax(file(separate)).
tax( (pay(Husband)&pay(Wife)), file (joint),_):- tax(file(joint)).
divorce(Husband, Wife, Date, Place):- not marriage(Husband, Wife, Date1, Place1)
tax( (pay(Husband)&pay(Wife)), file (separate),_):-
    tax( (pay(Husband)&pay(Wife)), file (joint),_), divorce(Husband, Wife, Date, Place).
```

A particular case of what we call a non-direct link is the temporal one. If a pair of answers describe two consecutive states or actions, and a given query addresses a state before, in between or after these states (actions), the clauses of semantic skeleton are expected to link the latter states with the former ones and provide a relevant answer.

For a scenario, semantic skeleton may include alternating sequence of states, interchanging with actions intermediate states (we assume no forks for simplicity). Not alike the situation calculus considerations, states and actions can be merged into the same sequence in the perspective of being explicitly assigned with an answer. The set of these states and actions fall into subsets corresponding to the answers (based on expressions for these states and actions that are semantic headers as well). It can naturally happen that answers are not ordered by this sequence (they may be assigned be the expressions (semantic headers) for alternating states and actions). Then, if a question addresses some *unassigned states* or *actions*, those *answers* should be *chosen*, which are *assigned to the previous and following* elements of the sequence.

For the sequences being considered, we do not require their elements to be *deductively* linked: just their enumeration is sufficient. We follow the same idea of information approximation as while constructing semantic headers. To deductively link states and actions (more precisely, situations) in a real-world domain, it would be necessary to define all the used entities and provide a large number of details, irrelevant to querying the domain. However, some analogue of the frame axiom, optimizing representation of knowledge that only the objects affected by actions change their states, are applicable to our considerations to chose an optimal semantic header in case of the lack of direct match.

We present the state-action sequence for tax return preparation scenario:

file(tax(return)):-

```
    tax(minimize(speculate), _),              % state
    collect(receipts), calculate(deduction),  % action
    collect(form(_)), consult(accountant),    % action
    fill(form(_)),                            % action
    calculate(tax(property )),                % action
    calculate(tax(income)),                   % action
    estimate(tax(value)),                     % state
    send(form),                               % action
    tax(return(_), expect(_)).                % state
```

Semantic skeletons are helpful for queries with translation formula as conjunctions of multiple terms (It happens for complex queries, consisting from two or more components, for example *Can I qualify for a 15-year loan if I filed bankruptcy two years ago with my partner?)*. If a term can be neither matched against a semantic headers or delivers too many of them, this term can serve as a virtual one. In the Table 2 below we analyze various cases of the satisfaction of a translation formula with two terms.

In the complex sentence, we distinguish two parts: *leading* and *assisting*; these parts are frequently correlated with syntactic components of a sentence. In accordance to our informational model of a complex question, its principle part is usually more general than the dependent part. One of the canonical examples of a complex query is as follows (note that we rather accent its semantic structure than semantic one): *How can I do this* ***Action*** *with that* ***Attribute****, if I am* ***AdditionalAttribute1*** *and* ***AdditionalAttribute2****.* We enumerate the properties of our informational model above:

1. *Actions* and its *Attribute* are more important (and more general) than *AdditionalAttribute1* and *AdditionalAttribute2*.
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 likely 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 obtain the situation with marginally relevant or too many answers. In general, the number of assisting components is much higher than that of the leading ones; therefore, assisting components are more likely to be represented as virtual headers. Proper coding is intended to achieve the coverage of all leading components, so most of them are represented by semantic headers.

There are a few additional models for complex questions. When the questions obey neither informational model, semantic headers and semantic skeletons can be individually built to handle particular asking schema. However, if no special means have been designed for the deviated (semantically) questions, the resultant answers may be irrelevant.

To estimate the results of matching procedure without 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, when semantic skeletons are not used. How to compute the overall Q/A accuracy in case of SH+SSK? There is the similar criterion of the Q/A system accuracy for the keyword search and NL-based implementation (however, the values are dramatically different).

accuracy =< informative portion of answer total size of answer >→max*, averaging (<…>) through the set of sample questions.

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 that 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 assume 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. The higher confidence of the proper decision would be if the leading term matches with SH and the assisting- 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 yield 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 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 rather 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. Resultant confidence level is rather low and there is insufficient evidence to reduce the answers yield 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 that yield 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 that deliver most of vSH in $\{ \text{vSH}(a) \cap \text{vSH}(a') : a \in A, a' \in A' \}$

Table 2: 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.

Proposition 1: The rules (Table 2) for matching a translation formula as a conjunction with virtual semantic headers deliver the highest resultant accuracy, averaging through the respective set of queries (complex sentences).

It is very important for observing the optimal domain structure that the clauses of semantic skeleton "fill the gaps" in between the semantic headers, which enforce the domain taxonomy. For a Q/A domain, if the clauses between entities are introduces first and the expressions, which remain uncovered, serve as a basis for semantic headers, domain structure would be far apart from the answer taxonomy. Therefore, domain structure, derived in such a manner, would be inconsistent with our answer classification approach.

To conclude the Section, we mention that SSK approach adds the advanced reasoning capabilities to the technique of semantic headers. SSK technique brings in the deductive links between the semantic headers, completing the commonsense knowledge representation on one hand and keeping the convenience of domain coding, inherent to SH technique, on the other hand. Deductive links between the semantic headers improves the content and query phrasing coverage of a domain, providing a more powerful way to associate a formal representation of a query with the existing semantic headers.

The key requirements to a knowledge representation system for Q/A is easy extension capability. The normal procedure of building a knowledge base is continuous update with new answers and new links between existing entities, attributes and objects. Therefore, the set of semantic headers and semantic skeleton of a domain must be able to naturally accept whatever needs to be done to handle properly a new query. It is clear, for example, that it may not be straight-forward to add a new axiom to a (complete) axiomatic system. Indeed, the set of semantic headers is far from being complete in a sense that it stops separating answers much in advance approaching a state of being a complete axiomatic system.

At the same time, the set of semantic headers plus semantic skeleton is approaching the coverage of all possible questions that we can call the *coverage completeness.* Therefore, the logical system for Q/A must be incomplete as the axiomatic system and complete in terms of handling all potential queries.

Proposition 2 SH+SSK technique is logically incomplete but possess potential coverage completeness. The latter means that for any new query it is possible to add a semantic header, readjust the other ones in accordance to certain rules, add a clause of the semantic skeleton, and modify the existing clauses in accordance to certain clause rules such that it will be properly handled, as well as the other ones of the original system.

The proposition above is based on the existence of *semantic header modifications rules* and *semantic skeleton modification rules* under domain update. Existence of such rules for semantic headers means that to add a new semantic header, only a limited subset of the totality of semantic headers should be modified to obey the original separation of answers. Indeed, if a new semantic header introduces a new entity, there is no potential conflict with the other rules. If a new object or attribute is added with existing entity, all the semantic headers for this entity must undergo a possible modification of instantiation state or possible exception for the mentioned object or attribute.

The semantic header modification rules are discussed in details in connection with Domain Optimizer in Section 3.5. Concerning the clause modification, the following proposition is suggested:

Proposition 2: Such clause modification rules as the insertion and deletion of a predicate in the body of clauses are sufficient for Proposition 2.1.3.2 to hold.

In addition to semantic processing, leading to formation of the most precise formal representation of an input query, default reasoning (Antoniou 1997, Ourioupina & Galitsky 2001) is required to modify translation formulas. Default rules are intended to handle the situations, where an extra constraint needs to be eliminated or added in spite of the correct translation into the formal language. For example, the query *How can mutation of chromosomes in my genes occur?* is actually about the *chromosome mutations* and not about *genes*, which are much more general in this context. Therefore, the following default rule (schema) is expected to be applied:

$$\frac{chromosome(mutation,\_\ ) :\ gene(\_,\_)}{chromosome(mutation,\_\ )}$$

Methodology and tools for building knowledge representation for Q/A domain

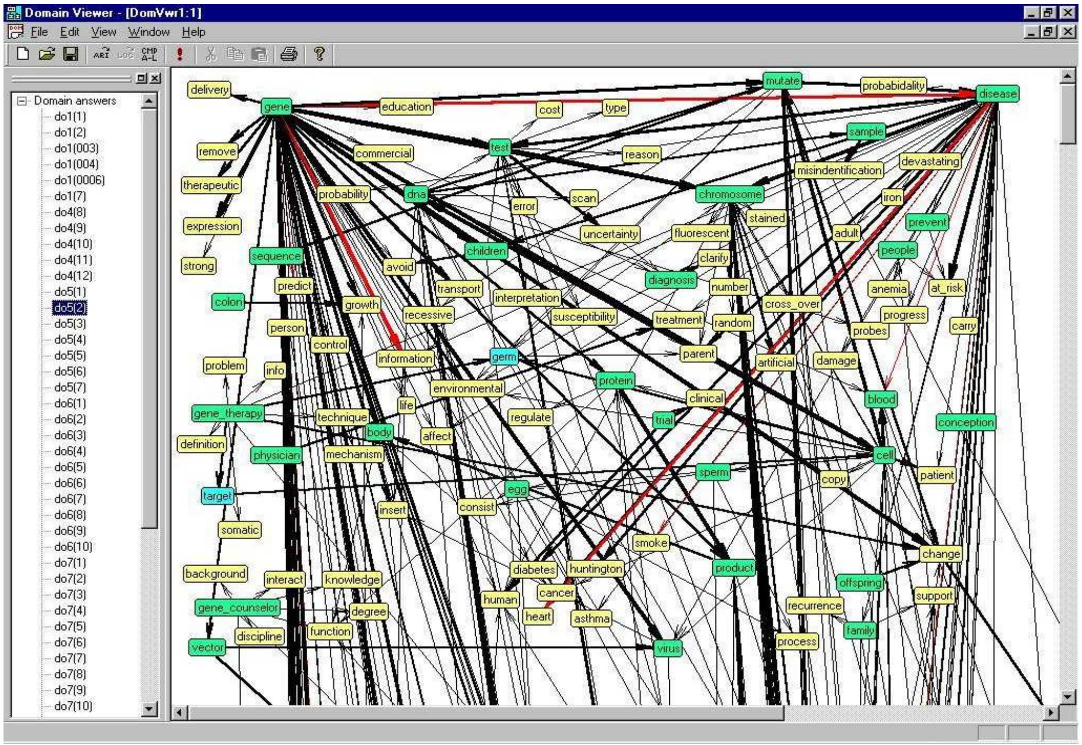

Fig 1: The domain restructuring tool (genome domain).

In this Section, we present the knowldge represenattion tools for SH-domain. Fig. 1 depicts the domain restructuring visualization by *DomViewer,* the Windows software for knowledge engineers, experts and domain testers. On the right, classification graph is displayed; the nodes are assigned with predicates and edges with their mutual substitution to form semantic headers. The thicker the edge, the more times the different paths for the whole domain contain this edge. In other words, the edge thickness visualizes how many times the corresponding sub formula occurs in the totality of semantic headers for the whole domain. Evidently, the majority of edges convert to the predicate *gene(_,_)*.

The toolkit automatically distributes the nodes throughout the rectangle to minimize the number of edge intersections. Built-in manual editing capabilities allow a knowledge engineer to distribute the principle entities in accordance to her view of the domain structure.

On the left, the answers are enumerated. The paths for given answer are shown in gray. In particular, the answer 5-2 (see the example above) has the paths *gene-disease, gene-predict, gene-disease-heart, gene-knowledge, disease-smoke*, etc. Judging on these entities, one can reconstruct the meaning of the answer, encoded using SH's.

The toolkit is in use to restructure the domain when additional answers have to be added or the information has to be redistributed between answers. Also, the tool helps to improve the overall domain structure approaching a tree-like domain graph, eliminating convergence of edges.

Knowledge representation for a given entity for TAX domain is depicted at Table 2.

Domain extension wizard (DEW, Table 3) allows a person, knowledgeable in a particular genome area, to share his/her experience in this domain. A customer can extend the natural language question-answering domain with the new advice, recommendation or intriguing story and to specify the major ways of asking about it. This toolkit processes the **combination** of the answer (the textual representation of advice with possible URL links to more detailed references) and a set of questions or statements. These questions or statements are supposed to be natural from the prospective of the domain extension author to generate the document as a reasonable answer (association). DEW does not directly support the addition of new facts or entities into the domain representation. However, the input of a set of questions, accompanied by an empty answer will mean that the pure formal facts are added (answers contain the parameters of the predicates). In the same manner, the input of an answer with an empty question window will switch DEW to the option of automatic annotation: formal questions will be generated independently. The (base) domain is ready for extension by DEW when its semantic model is designed: the essential entities and their attributes are specified, and the domain-related synonyms and multiword substitutions are defined.

Development and evaluation of Q/A domains

In this section, we consider GENOME and TAX domains. Human genome domain includes 200 answers and 2300 semantic headers. The size of the internal vocabulary is about 1000 atoms, and the NLP vocabulary is about 2000 words, including the synonyms and taking into account the multiword substitution. Domain of such complexity was verified by the experts to cover the majority of topics for the general audience. Desired accuracy can be achieved only by thorough domain adjustment in accordance to the query logs, processed by the domain experts. Usually, customers try to rephrase questions in case of the system's misunderstanding or failure to provide a response. Reiteration (rephrasing the question) is almost always sufficient to obtain the required information. At the beginning of evaluation period, the number of misunderstood question was significantly exceeded by the number of answers not known by the system. This situation is dramatically reversed later, however the number of misunderstood questions is monotonically decreasing in spite of the increase of overall represented knowledge.

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 70% accuracy, which is relatively low for traditional pattern recognition systems, over 95% of customers and quality assurance personnel agreed that tax Q/A system is the preferable way of accessing information for non-professional users. Using natural language Q/A system for advising is the fastest way to receive a reasonable advice, compare to Internet keyword search.

As an example of deployed domain, we can mention the TAX one with tens of thousand users monthly. Since the end of 2000, the major tax services provider H&R Block has used the tax adviser (please see www.hrblock.com/taxes/fast_facts/tax_search.html, Fig.2).

The Tax Advisor was subject to evaluation by about 6 thousand users per month during the season of filing returns for 1999. Though the answers usually do not exceed a page of text, the customers had more than 90% satisfaction with the system. The customers were attracted to our web site by the articles in "Boston Globe" newspaper and "Mass High-Tech" magazine, describing the Q/A system as an alternative to the traditional way of tax counseling. At that time, the answers usually do not exceed a page of text. 90% of the customers were satisfied with the system.

A lot of content improvement efforts dramatically improved the TAX engine for 2000 tax filing season. Obtaining the longer answers, users used hyper-references and related question links to obtain the relevant portion of information, and the NLP job itself was eased. The style of the presentation was much smoother than a year before that also contributed to the overall increase of user satisfaction.

```
gain(X,_,_):-var(X), clarify([home, remainder_interest,
       trade,transfer, installement_sale]), !, fail.
```

If a query contains gain without parameters (or with unknown parameters), the customer is suggested to clarify (to choose from the drop-down combo box).

```
gain(sell(P,_),_,77114):- nonvar(P), sell(P,77114).
```

The line above reduces one SH to another. In the TAX domain, document about the gain on a sale is the same document, containing the general information about sales. Argument of the predicate *sale* should be instantiated.

```
gain(sell(home(X,_,_),_),how(calculate,_),82103):-var(X),
do(82103).
gain(sell(home(X,_,_),_),how(calculate,_),82107):-
                              nonvar(X), home(X,_,82107).
```

If we speak about calculation of the gain for a sale of home, the situation depends on the attribute of this home. If there is no attribute (var(X)), then the answer "To figure the gain (or loss) on the sale of your main home…" is initiated. If there is a specific home attributes (a word, forming a specific meaning together with the entity *home*) such as *remainder_interest, improvement, inheritance, temporary, condominium, selling_price*, the answer is different. In contract to the previous SH above, the *gain-sell-home* is reduced to *home* and not to *sell*; in addition, instantiated argument of *home* is required to obtain the answer about "Foreclosure or repossession / Ordinary income".

```
gain(sell(home(_,_,_),_),_,82116).
```

SH above is required when we build the translation formula with two or more conjunctive members with the common answer id (here, 82116). This SH is design to be matched against one of these members. This match will not occur if another answer id is generated (verified) by the previous (following) conjunctive member (in the translation formula).

For example, the query "how to estimate the gain from sale of my house if I have used it for business" is represented as

```
  gain(sell(home(_,_,_),_),_,Id),
                      business(use(home(_,_,_),_),_Id);
```

SH for the second conjunctive member contains the reference to the answer `(do(82116))` and the first conjunctive member introduces the constraint via the answer id (SH above).

```
gain(sell(home(_,_,_),_),when,82121):- do(82121).
gain(sell(home(_,_,_),_),how(report,_),82122):- do(82122).
```

SH's above introduce the difference between the questions "when to do something" and "how to report something".

```
gain(sell(home(X,_,_),_),_,82128):-
nonvar(X),X=remainder_interest, do(82128).
```

The particular case of the family of SH above for *remainder_interest.*

```
gain(trade(home(_,_,_),_),how(calculate,_),82106):-
do(82106).
```

We have different answers for *sale* and *trade*, though their occurrence is symmetric.

```
gain(home(X,_,_),_,82107):- nonvar(X), home(X,_,82107).
```

"Gain for a home" is the same as "gain for a sale of home", if the predicate *home* is instantiated.

```
gain(home(transfer,_,_),_,82108):-do(82108).
```

Table 2 Representation of an entity *gain.*

Enter question(s): In this window, you can input all the questions or statements you think to be the natural ways of querying about the genome in the right window.	**Enter answer**: Here you can input the text you want to be available to your customer. You, as a question-answering system developer, have the control over how large the portions of text should be.
What is the difference between male and female chromosomes? How many chromosomes do humans have?	Chromosomes are… There are 46 chromosomes… Male chromosomes are…
Add to domain Press this button when you want to add a new portion of text (answer) with a list of questions	Compile domain When you press this button, the newly created source file will be appended to the existing domain and the updated source will be compiled to update the domain.

When the extended domain is compiled, you can see the formal (internal) representation for the questions. **Domain extension code:**

difference(chromosome(male,_), chromosome(female,_)):-do201.
chromosome(number,_):-do201.
do201:-iassert($Chromosomes are…$).

Domain is compiled. Ask a question to the updated domain

Ask Now you can ask the questions for the domain extension as well as for the base domain

Table 3: Domain Extension Wizard for the genome domain.

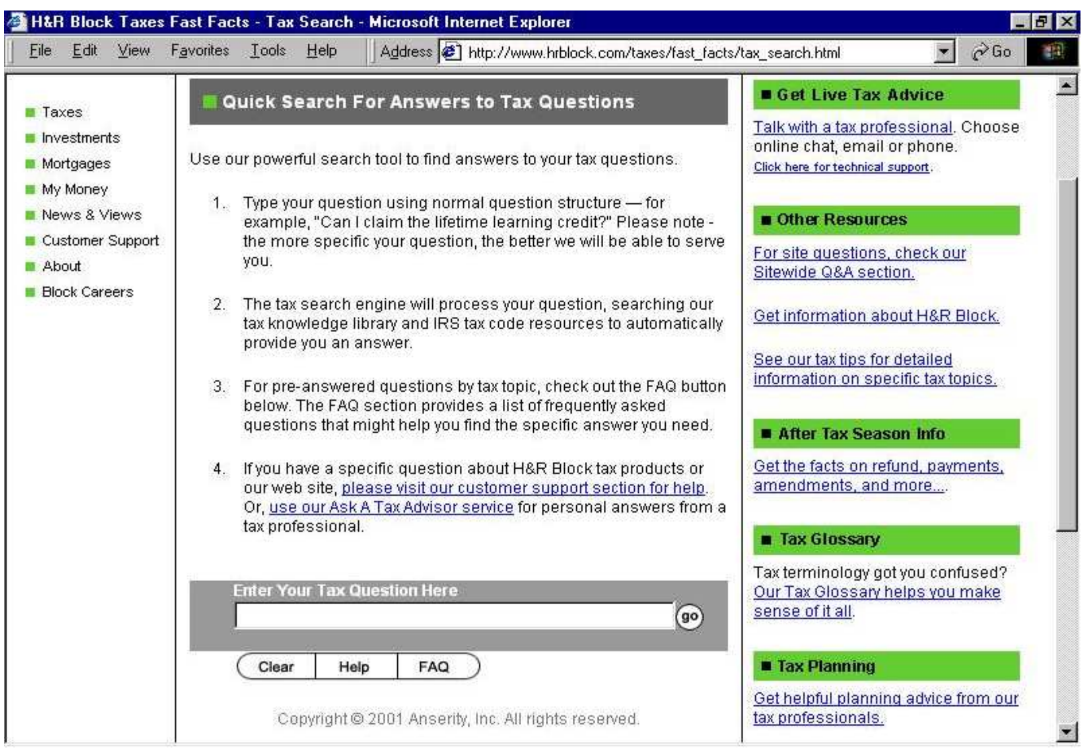

Fig. 2 TAX adviser at the H&R Block website.

Conclusions

1. SH and SSK-based knowledge representation is required to match the formalized queries with up to four entities, which is the case for a personalized advice request. Advanced semantic analysis and expressiveness of SSK means are sufficient to represent such logically and terminologically complex domains as financial, legal, psychological and genome. SH+SSK techniques are verified to be consistent to adding new knowledge, new semantic and syntactic phrasing. The system is capable of separation as semantically close natural language expressions as possible to distinguish the respective answers.

2. The SH+SSK technique brings in the following implementation and information management advantages:

- Easy training of knowledge engineers;
- Efficient and natural way of communication of knowledge engineers with domain experts;
- Robust and smooth capabilities of domain extension;
- Reusability of the domain components, including predicates and their semantic types, for the similarly structured domains;
- Applicability of the domain restructuring tools.

3. The chosen ratio of formalized and non-formalized data, determined by the technique of SH+SSK, allows optimizing the user satisfaction / domain coding efforts criterion. Only that knowledge has to be formalized that is in use for matching with a formal query.

Application of the semantic skeleton technique to NL Q/A shows the following. There is a superior performance over the knowledge systems based on a syntactic matching of NL queries with the previously prepared NL representation of canonical queries, and the knowledge systems based on fully formalized knowledge. Our approach gives a higher precision in answers than the former one because it involves the semantic information in higher degree. SH+SSK technique gives more complete answers, possesses higher consistency in context deviation and is more efficient than the latter approach because the full knowledge formalization is not required.

4. Evaluating the usage of vertical Q/A domain, we revealed that the majority of customers are satisfied having a 1000 answer domain (rather than the 10000-100000 answer ones). Therefore, all the "most wanted" answers can be thoroughly processed by the knowledge engineers manually. This observation allowed us to optimize the manual / automatic annotation efforts for the design of a vertical domain.

5. Extended logical programming approach to Q/A allows merging the knowledge sources of different nature: textual answers, associated with semantic skeletons, fully formalized ontologies (Kazik 2000, Karp 2000) and keyword-based annotation data to provide the complete answers, coming from the multiple prospectives.

References

1. Hirst, G. Semantic interpretation and ambiguity *AI* 34(2):131-177, (1988).
2. Gey, F.C., Chen, A.(1998) Phrase Discovery for English and Cross-language Retrieval at TREC-6, *Text Retrieval Conf*, NIST Special Publication.
3. McCallum, A., Nigam, K., (1999) Text Classification by Bootstrapping with Keywords, EM and Shrinkage. *Technical Report, Just Research* http://www.cs.cmu.edu/~mccallum.
4. Tarau,P. De Boschere, K., Dahl,V., and Rochefort, S. (1999) LogiMOO: an Extensible Multi-User Virtual World with Natural Language Control. Journal of Logic Programming, 38(3):331-353.

5. Ourioupina, O., Galitsky, B. (2001) Application of default reasoning to semantic processing under question-answering *DIMACS Tech. Report 01-16*, Rutgers University.
6. Galitsky, B. (1999) Natural Language Understanding with the Generality Feedback. *DIMACS Tech. Report 99-32*, Rutgers University.
7. Galitsky, B.(2000a) Technique of semantic headers: a manual for knowledge engineers *DIMACS Tech. Report #2000-29*, Rutgers University.
8. Galitsky, B. (2000b) Technique of semantic headers for answering questions in tax domain. *IASTED Intl Conf on Law & Technology,* San Francisco, CA, p.117.
9. Galitsky, B. (2001) Semi-structured knowledge representation for automated financial advisor *Industrial Applications of AI and Expert Systems.* Budapest, Hungary, June 2001.
10. Antoniou, G. (1997) Nonmonotonic reasoning. Chapter 14. MIT Press Cambridge, MA London England.
11. Dahl, V. (1999) The logic of language, in *The Logic Programming Paradigm,* Apt, Marek, Truszczynski, Warrens, eds, Springer-Verlag.
12. Fain, V.S. and Rubanov, L.I. (1996) Activity and Understanding *World Scientific Publishing*.
13. Partee, B.H., ter Meulen, A., and Wall R.E. (1990) Mathematical methods in linguistics. Kluwer, Dordrecht.
14. Kazic, T., (2000) Semiotes: a semantics for sharing. *Bioinformatics* v16 N12 1129-1144.
15. Karp, P.D.(2000) An ontology for biological functions based on molecular interaction Bioinformatics v.16 N3 269-285.
16. Lenat, D. (1998) The dimensions of context-space, *Cycorp technical report*, www.cyc.com.
17. McCawley, J.D. (1993) Everything that linguists have always wanted to know about logic but were ashamed to ask University of Chicago Press, Chicago IL.
18. Mirkin, B. (1996) Mathematical classification and clustering. Kluwer Academic Publishers. Dordrecht Boston London.