

Q&A - Natural Language Query and Command System



Dialogue Technologies – White Paper

Q&A , the Dialogue Technologies Natural Language Query and Command System.....	3
Abstract.....	3
Our Product.....	3
A short history of natural language query systems.....	5
The challenge.....	5
Keyword systems.....	5
Knowledge based systems.....	6
Leading question/answering systems.....	7
An overview of Q&A.....	8
The Conceptual Schema.....	10
Parsing and semantics.....	13
System requirements.....	15
Summary.....	15

Q&A , the Dialogue Technologies Natural Language Query and Command System.

Abstract

This White Paper describes the technology of the Dialogue Technologies Natural Language Query and Command System, Q&A. Q&A has accomplished a technology breakthrough, offering the user flexibility and accuracy without compromising performance. Q&A also offers unique productivity with languages and applications being exchangeable building blocks.

This paper describes the rationale for the design decisions that were made and how language fragments consisting of queries and some declarative sentences have been implemented in Q&A. We also explain how research in linguistics and data modelling has been integrated, providing unparalleled functionality to the customer.

With Q&A, Dialogue Technologies can offer a new versatile tool, suitable for applications such as automation of customer service or for intuitive user interfaces.

Our Product

Communication with a computer using a natural language, such as English or Swedish, is considered by many to be the ideal method for linking non-computer professionals to large and possibly complex information systems.

The main criticism of this approach is that the amount of computation to analyse a sentence given in natural language is large and subject to certain ill-defined constraints as variation of the human language is almost infinite. No system can be built that is able to cope with every possible situation.

For certain uses, such as question answering systems, there is no need for the computer to have details of anything other than a limited universe. Limiting the universe also limits the interpretation problem since the number of interpretations at any instance is reduced.

Dialogue Technologies – White Paper

A vital property of natural language is 'understanding' and understanding has been shown to depend upon a shared description of the application domain, the Universe of Discourse (UoD), between the participants.

Q&A is a solution that provides natural language understanding for the language fragment that is made up of queries and some declarative sentences. To provide 'understanding' it includes a conceptual model of the part of the world we are dealing with, our Universe of Discourse.

A short history of natural language query systems

The challenge

A few years ago, a researcher described a question-answering system where he showed a number of examples in the form of queries that his system could give the answers to. The queries were on several different subjects and quite complex. It all looked very impressive.

His system was implemented in a few dozen lines of code written in the programming language Perl. The system could cope with just the sentences he had shown and little else because the questions and the answers were hard coded into the application code!

If you know the questions, and there are not very many of them, it is not difficult to build a system to answer these questions. The above-mentioned system looks great at first sight but is useless! You have to look deep into a system to truly assess its capabilities - you cannot rely on examples.

The challenge is to build a system that gives the right answers for a large number of application-specific queries in natural language, each of which can be phrased in many different ways, with a response time the user finds acceptable, which often means a second or less. The answer must always be correct because otherwise we cannot trust the system and it becomes useless. It must also be easy to change applications and even languages in our Query System. And it is important to get the whole query analyzed – every word might be decisive.

Keyword systems

One development in the question-answering system history was the so-called keyword based systems. Here the input query was scanned and the existence of certain words in the sentence was noted. These words had to be defined in advance and mapped against data in tables and columns in a relational database. After scanning the input a report was created from the columns that were identified in the query.

Dialogue Technologies – White Paper

This method has severe drawbacks, though. Suppose we have a table, containing a column with the names of the US states, their capitals and a population number. If you ask “What is the population of Texas” or “What is the population of Austin”, you got the same answer. One must be incorrect. Austin is the capital of Texas.

These kinds of systems lost popularity and went out of fashion because they were notoriously unreliable. An unsuspecting user might accept the answer that Texas only has 550 000 inhabitants, if the table contains the population numbers for the US state capitals – or that 16 million people live in Austin, if the table contains figures for the population of the US states. You cannot require that the user is able to recognize and sort out incorrect answers to his queries, which he has to do here.

Knowledge based systems

An evident extension is to add the capability to recognize phrases and words. From this recognition we build an answer, or a query to a database, containing the answer. Pattern matching is often used here, in combination with some more or less capable parsing schema. You have to identify the pieces of a sentence and put them together in the right way, to determine what the answer should be. Again, this might work fine for limited domains and a limited number of objects and questions.

Many solutions have been tried. The complexities and the ad hoc solutions in pattern matching systems are often hidden in components called ‘knowledge base management system’. Often it is very difficult to get a coherent description from the designers of what goes on in this knowledge base. Usually the knowledge base contains some generalized application-dependent rules, complemented with large sets of specialized treatments of specific situations, encompassing the designer’s understanding of the part of the real world we want to deal with. The purpose is often to mimic the knowledge of an expert in these systems. New structures, like frames and semantic nets, are often used, which tend to highlight some characteristics and obscure others. All this makes the whole system expensive to maintain and complex to keep updated, making maintenance an expensive and ever-ongoing activity.

A knowledge-based system often contains embryos to components found in more capable systems, like conceptual models and various checking routines.

Leading question/answering systems

The design of Q&A is based on decades of research. The key research fields that were exploited were computational linguistics, conceptual data modelling and logic programming.

The following bullets summarizes the design goals that were met in order to build a leading-edge system.

1. Using a detailed analysis grammar, the input query is transformed into its predicate logic representation in a language called Conceptual Logical Form, CLF. A comprehensive set of syntactic and semantic checks are applied in this phase. From the representation in CLF, paraphrases, SQL and XML are generated
2. A hierarchy of classes, where each class inherits characteristics from its higher-level classes, is used with each entity assigned to a class. The class specifies what kind of queries we can ask involving each entity. The user can expand the predefined set of classes.
3. If the query has many interpretations, a paraphrase can be sent to the user to clarify which interpretation that represents his intentions. For the query "who sold the largest order" the user might be asked to specify if he means the largest by value or by number or items sold.
4. Anaphoric referencing, meaning that you can refer to earlier queries, is implemented. This makes the query session more natural to a human being. For example, the query "who received the highest bonus" might be followed by "who is *his* manager", in which anaphoric referencing is used.
5. A computer based tool call Customizer permits us to define the part of the real world we are going to deal with, our Universe Of Discourse, in the form of a conceptual model. See below for additional details. This effort is performed only once for every application and large parts of the model can usually be reused from earlier efforts.
6. All this information is used to make sure the queries are appropriate for the entities they are asked against. If all criteria are met, the SQL-code selecting the correct answer is generated

the way the system has been told to generate it. The answer is therefore always correct. If any criterion fails, the query is considered to be outside the scope of our UoD and a failure is reported.

- Both the language grammars and the model representing the application domain are part of a modular structure. A new language means a new grammar is plugged in and a new application means that a new application file is loaded.

An overview of Q&A

A picture of the system is shown in figure 1. The key component is the Engine. The user asks his questions through a query interface displayed in a web browser. The grammar of the selected language (English, Swedish, German, etc) is linked in and so is the description of the UoD of the selected application. Both grammars and application modules are independent functional units and handled as plug-in units.

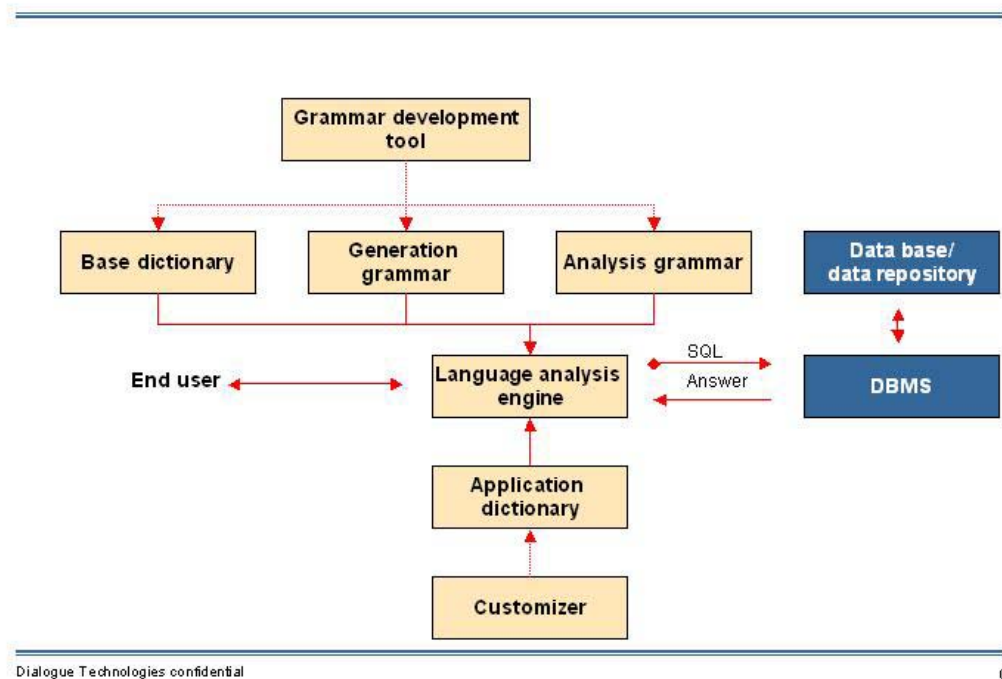


Figure 1

The application descriptions are generated with the Customizer, a graphical tool where the entities of the application and the relationships between them are specified.

Dialogue Technologies – White Paper

The translated query goes to a relational database (DBMS) to collect the requested information. This can be one or more data items in a table, a pointer to a picture, music or video file, etc. The DBMS can be replaced with other information sources, if required.

Some internal formats are shown, for illustrative purposes, below. After syntactic and semantic analysis of the input query, a representation of it is built in an internal representation called CLF (Common logical form), in predicate logic. From this a representation in DBLF (Data Base Logical Form) is generated and finally (if we are operating with a relational DBMS) into SQL. All these representations are language independent.

The Engine is implemented in the Prolog programming language, which still is the preferred implementation language for linguistic research in Europe. Other parts of the system are implemented in Java. Compilers for these two languages are available on a large number of platforms and operating systems, making it possible to run Q&A on all of these platforms.

The input query is first into CLF and later into a more database-oriented representation, DBLF, before it is transformed into SQL statement. It may be informative to look at what a query looks like in these three representations.

Query: "How do I send mail".

CLF:

```
Query(report,set(y2088,
  set(y2158,
    set(y2157,
      instance( e4,y2088)
      instance(e3,y2158)&
      instance(e31,y2157)&
      exist(y2159,
        instance(e30,y2159) &
        nom(e30,e3,y2159,y2158)&
        acc(e30,e31,y2159,y2157)&
        mod(e30,e4,y2159,y2088))))))
```

DBLF:

```
Query(report,
  set(y2088,
```

```
set(y2158,  
set(y2157,  
exist(y2384,  
exist(y2385,  
exist(y2386,  
relation('TT','INSTRUCTIONS',  
['URL'=y2158,  
'URL'=y2157,  
'FUNCTION'=y2386,  
'PART'=y2385,  
'URL'=y2088,  
'TYPE'=y2384])&  
y2385=([109,97,105,108])&  
y2386= ([115,101,110,100])&  
y2384=([95,104,111,119])))
```

```
SQL:  
SELECT DISTINCT X1.URL  
FROM TT.INSTRUCTIONS X1  
WHERE X1.PART='mail' AND  
X1.FUNCTION= 'send' AND  
X1.TYPE='_how'
```

The above shows a fairly simple example, working with only one table. Q&A can handle complex join operations between several tables. It is capable of dynamically generating up to six lines of SQL statements, which create temporary tables, insert values into them, makes selections from them and manipulates these selections. This means that it can handle complexities in the generated queries that are far beyond those the users of the system can handle.

The Conceptual Schema

The natural language system provides a capability to analyze NL expressions. A conceptual model of the world, relevant to the application at hand, is created (customizing the system) by the user and is stored as a conceptual schema. The schema is built of logical facts representing entities (concepts) and binary relationships between entities, forming a description of the universe of discourse or object system in question.

The entities and relationships of the universe of discourse are linked to corresponding natural language terms in the vocabulary. The schema

Dialogue Technologies – White Paper

is in principle completely language independent and contains only concepts and relationships between concepts.

The various grammatical forms for each entity are entered and each entity is classified in a hierarchy of classes, which determine what type of queries that can be asked about that entity. Classes in this hierarchy are "person", "address", "thing", "length", and many more. Also the prepositions used by each entity are specified. New classes may be added, as required.

In one of the implementation steps we specify what tables to work with in the DBMS and their interrelationships (joins, foreign keys, etc).

The main steps required to build a component in the conceptual schema is shown below. We define that someone can send mail. A variety of queries about this subject can then be asked.

Application building with DT 1.2

Step 1 – Naming Entities

Enter a unique name for each Entity



The screenshot shows a dialog box titled "Create Entity". It has three tabs: "Name", "Term", and "SQL". The "Name" tab is selected. Below the tabs is a text input field labeled "Enter name:" containing the text "e-mail". At the bottom of the dialog are three buttons: "OK", "Reset", and "Cancel".



The screenshot shows a dialog box titled "Create Entity". It has three tabs: "Name", "Term", and "SQL". The "Name" tab is selected. Below the tabs is a text input field labeled "Enter name:" containing the text "send". At the bottom of the dialog are three buttons: "OK", "Reset", and "Cancel".

Dialogue Technologies – White Paper

Application building with Phoenix 1.2

Step 2 – Defining Terms for Entities

Terms will identify Entities in the user query.

Enter all Terms that will be used for each Entity respectively.

Create Entity – e-mail		
Name	Term	SQL
Term:	mail	Add ...
Defined terms:		Change
e-mail message electronic mail mail message		
		Delete
OK	Reset	Cancel

Create Entity – send		
Name	Term	SQL
Term:	submit	Add ...
Defined terms:		Change
send post deliver		
		Delete
OK	Reset	Cancel

Customization 031/12/1

1

Application building with Phoenix 1.2

Step 3 – Define SQL for Entities

Enter the SQL statement that should be generated when the Entity is accessed.

The final SQL query is the sum of all SQL statements of the Entities that has been accessed by the user query.

Create Entity – e-mail		
Name	Term	SQL
Verify or change SQL statement		
select x1.url from ss.instructions x1 where x1.COLUMNX=VALUE1 and [...]		
OK	Reset	Cancel

Create Entity – send		
Name	Term	SQL
Verify or change SQL statement		
select x1.url from ss.instructions x1 where x1.COLUMNZ=VALUE2 and [...]		
OK	Reset	Cancel

Customization 031/12/1

2

Application building with Phoenix 1.2

Step 4 – Define relations between Entities

1. Drag and drop one Entity over another Entity.
2. Select relations between the Entities from the relation list
3. Relations will be graphically represented by a line between Entities.

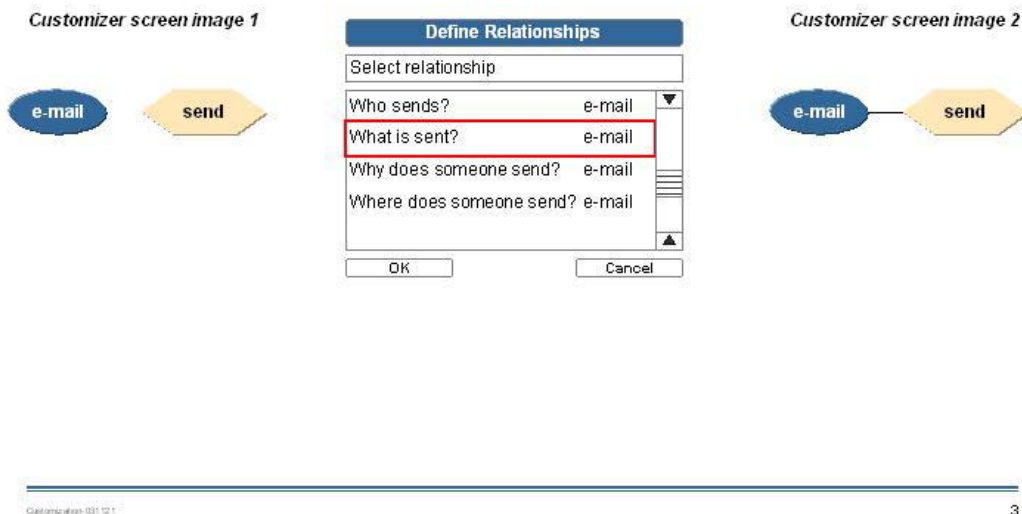


Figure 2

The schema itself is completely language independent, although the components of it may have 'names' expressed in a natural language such as English and some relationships may depend on the language. The schema is usually connected to the contents of a relational database so that each concept in the schema may or may not have a unique connection to a table in a database. The schema thus constitutes a link between the natural language and the database.

Parsing and semantics

The natural language engine (NLE) performs the actual analysis of the NL query expression, in cooperation with an analysis grammar and the schema. The analysis results in an intermediate language-independent logic form representation of the input. The input query is translated into a query language such as SQL.

Dialogue Technologies – White Paper

One extracts as much information as possible from syntax divorced from meaning but ultimately the syntax and semantics are inseparable, with meaning derived from both syntax and semantics.

Parsing routines are used on the input query to generate one or more syntactically valid parse tree for the input series of words, based on a vocabulary containing terms of a natural language. Terms and morphological rules about the terms have been defined. Based on the syntax rules of the grammar for the natural language, at least one syntactic parse tree containing semantic routines is built.

For each parse tree, an executable set of semantic routines is build, based on one or more semantic routines associated with at least on syntax rule and its associated semantic routines. A semantic parse tree is generated.

The semantic routines are executed, creating a language independent representation of the input series of words (query) using a conceptual model, having a set of language independent records of information, defining entities. Each entity has a connection to at least one term in the vocabulary and at least one entity has a connection to a database table. A vocabulary is defined for each entity as well as a set of relationships between the different entities. The execution of the semantic routines means that we check groups of one or more words in the parse tree against the conceptual model for conceptual validity. A language independent representation of the meaning of the input query is created in first order logic, complemented with higher order logic concepts.

A conceptual schema is stored in the system (see above). Since the schema is language independent there is a great advantage in that it is very easy to change analysis grammar and vocabulary and thus to switch between different natural languages. Grammars, dictionaries and conceptual schemas can be supplied as 'plug-in' modules.

The system has as a cornerstone the translation of input statements into a semantical representation in predicate logic. A set of semantic primitives has been developed and constitutes, in combination with the conceptual model, the nucleus of the 'understanding' in the chosen language fragment of the system.

The semantic description depends upon limiting the universe of discourse to a set of common relationships. Semantic structures are

Dialogue Technologies – White Paper

put together according to the rules of grammar to form larger semantic/logic structures that represent the meaning of a sentence.

The problem of specification can be approached through first-order logic since this makes no assumptions about the world. Once the constraints and requirements of formulating a situation are made in logic, it is possible to define the way the objects interact with each other. The methods of interpretation, manipulation and presenting of this knowledge are fixed and well understood. First order logic must be complemented with higher order logic to permit a wider range of questions to be asked.

The parts of speech of a word are in fact represented as binary relations. Verbs, for example, can require 0, 1 or more objects. The word 'believe' can be either transitive or intransitive and if intransitive can require the word 'in' as in 'I believe in something'. It is possible to associate given meanings of certain words, such as verbs, with a sentence pattern.

System requirements

The system is implemented in Java and SICStus Prolog. Q&A therefore runs under all HW/SW combinations where Java and SICStus Prolog code can be executed. This makes Windows and Linux as well as most UNIX systems our potential target systems. The minimum system consists of a 1 MHz machine with at least 256 MB of memory and at least 6 MB of disk storage for the application. The application workspace varies with the amount of detail in the application but 6 MB is a typical size.

Typical processing time for a query in a 3 MHz PC is 50 – 100 ms. If data is kept in a relational database this makes such a DBMS a requirement.

Summary

The Dialogue Technologies Query System is the leading Natural Language Query and Command System. It is based on decades of research and integrates research results primarily from linguistics, data modelling and logic programming. Applications and languages can be changed in a modular fashion.

Dialogue Technologies – White Paper

The product will help call centres and similar businesses to automate its operations and to reduce costs and improve service. The user may phrase his query in natural language, just the way he would do it to a human operator. And the system is available 24 hours a day, 7 days a week, providing answers and responses to commands that are consistent and have been validated for quality.

The system can be run on a wide range of hardware and software and has modest system requirements. All this adds up to a short payback period for the customer.