Natural Language Interfaces to Structured Data

Author 1, Author 2, Author 3

Abstract—Abstract will be written later...

Index Terms—Computer Society, IEEEtran, journal, LaTeX, paper, template.

1 Introduction

Natural Language (NL) is the most natural and easy way to express information need [**?**]. Moreover, it is not restricted to experts who have a good knowledge on how to formulate a formal query.

NL as an interface has been investigated by researchers in both Information Retrieval (IR) and Database communities. In the IR community however, the keyword paradigm has become the most popular, for sure because of the success of commercial search engines. NL interfaces for unstructured documents are refered to as Question Answering (Q&A) systems, that have received much researchers' attention as the Web had made avaialble huge amount of textual data, and recently as the famous *Jeopardy!* quiz was won by IBM's WATSON [**?**] system. WOLFRAMALPHA[1] is a popular search service based on structured data, that accepts keyword queries as well as some questions in natural language.

NL interfaces (to structured data) has focused much researchers' attention for decades, but the field seems to have known a renewed interest for a few years, probably thanks to the development of the semantic Web. **See also the quotation in [?]: users prefer NL interfaces than logic used in the Semantic Web.** However, the challenge of answering NL questions is far from being solved [**?**].

NL interface: not only in the context of search (information retrieval), for instance natural language query can be translated in goals / commands in the context of (household) appliances (see EXACT [**?**], not surveyed here).

In this survey, we try to point out the major dimensions of existing systems and to present the new trends and challenges of state-of-the-art systems. The history of NL interfaces can be developped as follows:

1) early years of domain-specific systems
2) complex question answering in a specific domain
3) rise of domain awareness in NL interfaces (through learning techniques)
4) data-driven systems with respect to
   - resources: almost domain-independent systems
   - large-scale data: domain-independent systems

1. http://www.wolframalpha.com/

We will present the "big picture" of the approaches in section **??**.

2 Main Dimensions

In this section, we introduce the main dimensions based on which NLI approaches can be distinguished. The input to every NLI is the (1) *data*, and (2) *user questions* representing information needs, which are translated to an (3) *internal representation* of the needs employed by the system and/or directly mapped to (4) *database queries* that finally, are executed by the underlying query engine to produce the final answers. The main problem tackled by a NLI approach is to transform the question to the internal representation and then to the database query ($2 \mapsto 3 \mapsto 4$), or to directly mapped the question to the query ($2 \mapsto 4$). This problem is hard when considering the large and rapidly evolving mass of structured data that have become available. The major challenges modern NLI systems are facing in this regard is to operate across domains (5, *domain-independence*) as well as to adopt to new domains (6, *portability*). For evaluating the quality of the NLI output, different (6) *metrics* based on the user questions the system can understand as well as the database queries it can produce, have been proposed.

——— TOBECONSIDERED Dimensions:

- What kind of query? (NL, controlled language)
- What are the data?
- Target structured query: select query? update query?
- What are the answers?
- How portable is the system? Does it improve its performance over time?
- What is the linguistic coverage of the system?
- Error feedback: how well is the user informed of misunderstanding of her question
- Predictability: how the system disambiguates / lets the user decide what should be the correct interpretation
- Performance: response time / evaluation metrics

———

2.1 Data

test Structured data:

- relational databases; data warehouses
- XML databases
- early data structures (Prolog databases; list-structured databases)
- ontologies; *linked data*

In the survey, we refer to *database elements* which stand for named properties of the database. For instance in a relational database, they stand for relation names, attributes names and values, ...

Data structures are associated with a data model or schema, which is explicitely defined in modern databases. Despite the effort of modeling and conceptualizing structured data, there is still a gap between the physical view of structured data (database administrators' view point) and the conceptual view of end users; interfaces concerned in this survey attempt to bridge this gap.

2.2 Question

Traditional database management systems provide an interface that allows users to query the data in a formal language (e.g., SQL).

2.2.1 Keyword query

A *keyword query* consists in a set of words traditionally used in the context of document search (i.e. searching relevant documents given a textual query). The classic idea is to represent documents in a vector model, where the vector is composed of terms of documents. Search engines have made this paradigm popular. [**?**] reports that the number of words in queries over Web search engines trends to increase (the experiment compared queries performed over a month in 2009 and in 2010): queries from 5 to 8 words increase up to 10% while queries from 1 to 4 words decrease up to 2%. It seems indeed that users are more and more aware of new capabilities of state-of-the-art search engines, and are less reluctant to express their information need in a more "natural" way.

2.2.2 Natural language query

Early systems understood some English words and English syntactic constructions; such language was called "English-like" by authors [**?**]. More recent systems try to go further and to capture as much as possible the regularities as well as the irregularities of NL.

Current systems handle only a subset of NL questions (the questions that the system can understand). This "subset" of NL is called controled NL. Some systems even go further, and are able to analyze why a question can not be answered. It could be out of coverage of linguistic understanding of the system (the question is not understood) or cannot be answered because the knowledge base does not contain the answer (even if the semantics of the question is fully captured).

Users prefer NL questions than keywords [**?**]. In this survey, we focus on natural language interfaces, and not on keyword search over structured data, even if the latter has become more popular lately[2].

2.3 Internal Representation

Syntactic question representation (also called *parse tree* because the tree graph structure is used most of the time) is an intermediate representation, before the creation of the internal semantic representation. In lots of systems, however, the syntactic representation also contain semantic pieces of information: nodes of the syntactic tree contain information about how to generate fragments of the target database query. The nodes of the tree representation contain information about words and relations between words of the question. Typical semantic information contained in a syntactic parse tree are the database elements that those words refer to. Those semantic information (also referred to as "meaning" in early systems) are kept in a lexicon. The syntactic parse tree usually does not try to resolve ambiguities, and keep all possible interpretations. Resolution of ambiguities is done afterward, when building the semantic representation out of the syntactic representation.

2.4 Database Query

The parse tree (which may or not contain semantic information in parse tree nodes) must then be *interpreted* in some internal semantic representation. Table **??** displays semantic meaning representations used in different systems. As shown in this table, the internal semantic representation can be or not the targetted query representation. The expressivity of the database query is discussed in section section **??**. The semantic representation is intented to capture as much as possible users' intent, and is sufficient to generate the final database query. While the syntactic representation may contain lots of ambiguities, the semantic representation does not. The semantic representation depends on the data structure. However, to demonstrate their portability features, some systems contain components that can translate the semantic representation for different database query languages.

TOBECONSIDERED ——-

2.4.1 Expressivity of database query

———

2.5 Results

NL interfaces produce database queries. Those queries must be then be executed by underlying DBMS. The execution of queries can lead to different failing states:

- the query execution fails (the generated database query is not valid)

2. Paper to be published at VLDB 2012: "SODA: Generating SQL for Business Users" from Blunschi et al.: Keyword-based interface to data warehouses

- the query execution lead to an empty result set

In these cases, the system may inform the user and/or suggest/rephrase the query. This is especially performed in systems belonging to the feedback approach (see section **??**).

2.6 Query-independence

2.7 Portability

2.8 Metrics

Several evaluation metrics have been introduced in various systems. We review them briefly below. We present the interesting figure **??** copied from Han et al. work [**?**]. It shows the tradeoff between linguistic coverage ("Expressions interpretable by a System") and the expressions that can be answered from a knowledge base. The goal of any interface would be that the linguistic coverage comprises all expressions that can be answered by the knowledge base.

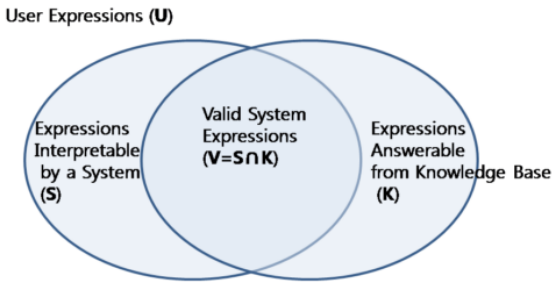

Fig. 1. Linguistic coverage vs. logical coverage within interfaces, copied from [**?**]

2.8.1 $Fluency_{Woods}$

Woods [**?**] defines *fluency* as "the degree to which virtually any way of expressing a given request is acceptable". Fluency measures roughly how easy to use is a system. Intuitively, a good fluency means a "natural" interface in the sense of natural interfaces [**?**]. This requires advanced natural language techniques, and would probably lead to systems that can interpret more expressions than those that can be actually answered by the knowledge base ($S \setminus K$ in figure **??** would not be negligeable).

2.8.2 $Completeness_{Woods}$

Completeness was first defined in Woods' work related to the LUNAR [**?**] system. It measures if there is a way of expressing any query which is logically possible from the database. In the end, it measures if the interface can answer all possible questions. In figure **??**, completeness would be represented by S comprising K ($K \subset S$).

2.8.3 $Soundness_{Popescu}$

An interface is said *sound* if "any SQL output is a valid interpretation of the input English sentence" [**?**]. In figure **??**, this evaluates the expressions of $V = S \cap K$.

2.8.4 $Completeness_{Popescu}$

An interface is said *complete* if it "returns all valid interpretations of input sentences" [**?**]. Yates et al. have reused this metrics in their EXACT system (not surveyed here). They do not claim that users should restrict their questions to query only tractable qestions; but they suggest that identifying classes of questions that are semantically tractable and measuring the prevalence of these questions is the direction of current research.

2.8.5 User's intent

Yates [**?**] makes an assumption, that if the interface is sound, complete and if a single SQL statement can be produced from a NL question, then the interface has unambiguously determined user's intent.

2.8.6 Predictability

Within user interfaces, predictability has been pointed out as the essential feature of user interfaces in the 90' by Norman [**?**] and Schneiderman et Maes [**?**]. Predictability and the feel of control seems however to be in contradiction with personalization, which is the one pillar of recent IR systems.

3 ANATOMY OF NLI SYSTEMS

The main problem to solve is to map user's intent expressed in a natural way (say unstructured way) to a database query, which is a very structured expression where there is no room for ambiguity unlike natural language. As a result, all interfaces must deal with words or terms "meaning". This "meaning" or semantics is in the end defined in terms of "database elements".

3.1 Lexicon

The lexicon is a data structure that is used to reduce the ambiguity in words when analyzing users' questions. NLI are usually composed of two lexicons: a domain-dependant lexicon and a domain-independant lexicon. The domain-dependant lexicon defines words in terms of semantic rules (see section **??**). The domain-independant lexicon defines how to interpret words independant from a given domain. For instance, *wh*-question words define constraints on the structure of the expected database query (for SQL, for the "WHERE" statement for instance).

3.2 Semantic rules

Semantic rules define the semantics of question words. The input in a node, or a part of the parse tree being constructed, and the output contains information on how to construct part of the final database query. The final database query will be generated from the parse tree, which contains semantic information in the nodes, in addition to lexical and syntactic information. The idea behind this process is a linguistic theory [missing ref], where the global meaning of a sentence is defined by

the individual meanings of words / expressions in the sentence, and by the syntactic relationships between the words / phrases.

3.3 Main problems to solve

The main problems to solve:

- have the boroadest linguistic coverage (let users use their own terminology)
- mechanisms that permit to query other domains with a low porting cost
- ensure that the resulting database query is valid

4 Taxonomy of main approaches

4.0.1 Classic translation approach

TOBECONSIDERED ———- The classic translation approach consists in going from (1) to (4). To reach step (3), semantic rules are needed to find out what database elements should be associated to question phrases.

4.0.2 Iterative approach

The iterative approach does not go directly from (1) to (3), but goes back to (1) several times before reaching step (3). Those iterations correspond to question reformulations, that involve user-feedback to interpret semantically the question in terms of a database query.

4.0.3 Configurable interfaces

Configurable interfaces let users improve the system's capabilities, in particular the linguistic coverage. Minock et al. [**?**] have identified three kinds of configuration applicable to NL interfaces:

- let users name database elements, so that phrases used in the question can be easily mathed with database elements
- offer a GUI that generates automatically semantic rules or a grammar for translating NL questions to database queries
- use machine learning techniques to induce semantic rules or a grammar from annotated corpora

The configuration is particularly important when porting the interface to other domains. However the three kinds of configuration mentionned above are not equally costly: for instance, the third one (machine learning techniques) can be hightly costly if it requires a huge volume of annotated data. The cheapest configuration is based on user interaction, where no initial configuration is needed, but domain-specific knowledge is learned based on user interaction. An example of such a system is NaLIX [**?**].

Most recent interfaces integrate configuration capabilities, since it allows domain portability which is a great improvement of interfaces. In the following we will emphasize the role of user interaction (*feedback-driven approaches*) and machine learning techniques (*learning-based approaches*).

———

The way translation from NL questions to formal query has evolved over years. Table **??** presents an overview of major systems that we consider in the survey.

4.1 Domain-dependent semantic parsing

Early systems belong to this class of approaches. The knowledge necessary to translate NL questions to database queries is encoded in lexicon. The systems allows users to use a subset of English (controlled language) to query databases. The lexicon, however, is a huge linguistic resource which defines how each word must be translated into database elements, and what semantic rules to trigger to get, in the end, the desired database query.

4.1.1 Lexicon

The lexicon is a resource that "defines" a set of words that belong to a domain. Those words are associated to a "meaning", which is a semantic rule that says how this word must be interpreted in the data domain and given the data structure. In addition, the lexicon also contains a list of specific rules that modify the global meaning of the sentence, given some words that are already defined in the lexion. The lexicon usually combines both syntactic with semantics information. For instance, the same word would have a different interpretation if it's a noun or a verb in a sentence.

4.1.2 Limitation

The domain dependence is however the great limitation of those systems; this is due to the cost of the lexicon. Indeed, porting such systems to other domain would mean provide a new lexicon and corresponding semantic rules, which is highly costly.

4.2 Complex question translation

The next generation of NL interfaces aims at increasing the linguistic coverage. This is performed with the distinction of both domain-dependent knoweldge and domain-independent knowledge. The domain-dependent knowledge base consists in semantic rules triggered by words, phrases or syntactic information encoded in a parse tree. Those rules produce fragments of the target query language (or of the intermediate query language) to be then combined and modified to generate the final qurey language. The domain-independent knowledge base is composed of lexical information in a dictionary (which might be completed with domain-dependent knowledge, like the most likely senses of words used in the application domain).

4.3 Feedback-driven approaches

This range of systems translate NL questions to database queries basing on users' feedback. We distinguish between the following kinds of feedback:

4.3.1 Authoring tool configuration

Authoring tools are GUI that permits users to edit domain-specific knowledge (e.g., the lexicon that is used to translate NL questions to internal queries). Domain-specific knowledge edited within such tools can be straightforward (like synonyms to be used in user's question to describe the same database elements) or more complex like semantic rules to translate user's terms to logical elements. Describing this knowledge is not an easy task for standard users, for instance semantic rules that define how to map words/phrases to logical elements, and rules that define how information from the parse tree must be combine to produce the final logical query. For that reason, recent advanced systems try to infer such rules basing on dialog-like interaction.

4.3.2 Interactivity

Authoring tools presented in the previous section are intended to be used as a preliminary task when porting the interface to other domains. Other systems do not explicitely ask users to answer question to get the domain-specific knowledge, but infer this knowledge basing on interaction when using it. This range of systems suggests users NL questions that could be reformulations of the current user's question. To the best of our knowledge, there are two different kinds of such interaction, when the system cannot interpret a question:

- the system tries to change words/phrases in user's question, so that the system is able to interpret the question
- the system also comprises a repository of successfully answered questions, and suggests one of those questions replacing some slots with terms used in the user's question, and ensuring that the generated question can be interpreted by the system

Recent work [**?**] investigate how to present similar questions in NL as interpretations of the current question. This is a way of making the user think she controls what is happening, which is one of the features of modern interfaces. This paraphrasing feature is also present in NALIX/DANALIX [**?**], [**?**] and C-PHRASE [**?**] works.

4.4 Learning-based approaches

Learning-based approach consists in learning a grammar or a set of rules that map NL sentences to logical forms. Learning approaches are popular among the NLP community. Indeed, learning techniques reduces the cost of linguistic resources, which is being learned over time. In the case of NL interfaces, this permits to port the system to other domains with limited cost. Most systems need a corpus of labelled examples, e.g., a set of sentences mapped to their corresponding logical form.

However, such corpora are rarely available, and are costly to produce. Some systems adopt strategies to address this (see WOLFIE section **??** for instance.)

System	Model
Miller et al. [**?**]	recursive transition network
Zettlemoyer et Collins [**?**]	log-linear model

TABLE 3
Different statistical models used by learning-based systems

System	Training data volume
Miller et al.	4000 sentences
Zettlemoyer et Collins [**?**]	600/500 training examples[3]

TABLE 4
Volume of training data of different systems

4.4.1 Statistical models

Table **??** summarizes the different statistical models used by learning-based systems. Miller et al. [**?**] use a probabilistic recursive transition network and consider the probability $P(T|W)$ of a parse tree T given a word string W:

$$P(T|W) = \frac{P(T) \times P(W|T)}{P(W)}$$

This model combines both *state transition probabilities* and *word transition probabilities*. The state transition probability concerns the labelling of a combination of semantic and syntactic information recursively (the label of a node is computed given the labels of the previous node in the syntactic order, and the parent node). The word transition probability is the probability of a word, given the previous syntactic word and a semantic information (attached to the parent node in the parse tree).

Zettlemoyer et Collins [**?**] use a log-linear model to learn a combinatory categorial grammar. This grammar, which also defines the domain-specific lexicon of the parser, contains semantic information (*i.e.* to which λ-calculus formula a given word/phrase should be associated).

4.4.2 Shortcomings of these approaches

These components rely on statistical models, that often require a huge amount of annotated data. For instance, a statistical parser require a significant number of questions annotated with corresponding parse tree. Table **??** shows the volume of training data corresponding to various systems. Besides, training data should also be composed of negative examples, whose availability is a strong requirement as pointed out by Giordani et Moschitti [**?**].

4.5 Schema-unaware approaches

A range of recent systems aggregate potential answers from different sources. This range of systems have emerged in parallel with the success of the semantic web technologies. The particularity of these approaches, is that they cannot rely on the schema of underlying

knowledge bases, because the different sources are potentially modelled in very different ways. Thus, the set of systems represented by these approaches try to bridge the gap between user terminologies and the different terminologies used in the different knowledge bases that the interface talks to. These approaches can be seen as an extension of several approaches presented before, namely "complex question translation" (because the question is further analyzed to map it with the terminology used in the different knowledge bases) and "learning-based approaches" (because some of the systems represented in these approaches comprise learning components).

4.5.1 Terminology mapping

In previous approaches, end-users were not aware of how data are modeled in the knowledge base. This requires advanced natural language processing to map user's terminology with database terminology. In schema-unaware appproaches, the systems do not talk to a single database, but to potentially an unlimited number of sources where the structured data are to be found. Each of those sources have its own logical schema, naming conventions, etc. Then, the system must know how to communicate with these knowledge bases, and what strategies to adopt to reduce the computation cost of generating distributed queries (see section below **??**). In some cases, it's also critical to aggregate results from different sources.

4.5.2 Shortcoming of these approaches

The major shortcoming is related to scalability and efficiency. In particular, in cases where the knowledge bases are searched over Internet (for instance Linked Data[4]), the computational complexity of mapping and expanding user query terms to the terminology of respective knowledge bases is a limiting factor. Internet latency in this case is also to be considered since the databases are not hosted.

5 DOMAIN-DEPENDENT SEMANTIC PARSING

The major systems are BASEBALL [**?**] and LUNAR [**?**]. Both systems are surveyed below.

5.1 BASEBALL [?]

BASEBALL aims at answering questions about baseball results. The main concepts in the data are games, teams, scores, day, month, place (of a game). The domain is quite close, which means that there is few ambiguity in terms of word meaning.

4. See http://linkeddata.org/.

5.1.1 Data structure

The data structure is a list structure called *specification list*. This is a hierarchical structure, where each level correspond to an attribute associated to a value, or a nested specification list. An attribute can also be modified. For instance, the city *Boston* is represented by "City=Boston"; an unknown number of games is represented by "$\text{Game}_{\text{number of}}$=?".

5.1.2 Semantic knowledge

The semantic knowledge necessary to map questions to the data structure is defined in the lexicon on the one hand, and in a set of semantic rules (called subroutines) on the other hand.

5.1.2.1 Lexicon: The lexicon maps words or idioms to their meaning in the same data structure presented section **??** as well as their POS (part-of-speech). Wh-words are also referenced in the lexicon.

5.1.2.2 Subroutines: Subroutines are a set of semantic rules that modify the query representation or make choices in cases of ambiguity. For instance, a word that can have two POS (noun and verb) is disambiguated with the help of some heuristic, like the fact that any sentence can only have one main verb. A meaning modification consists for instance in adding a modifier in an attribute. For example, the word "team" has the meaning "Team=(blank)". The word "winning" before the word "team" will lead to the modification "$\text{Team}_{\text{winning}}$=(blank)".

5.1.3 Question translation step by step

5.1.3.1 Dictionary lookup: The question is first tokenized in words, and empty words are left aside. The remaining words as well as adjoining words are looked up in the lexicon for the POS and the meaning. The output is a list of attribute/value pairs with extra information like the POS of each word and if the word is a wh-word.

5.1.3.2 Syntactic bracketting: POS of each word is used to syntactically analyze the question in term of phrases. Phrases are surrounded by brackets and the main verb is left aside. The parsing proceeds from right to left and bases on heuristics. For instance, prepositions are associted to the rearest right noun phrase to generate a prepositional phrase. Each phrase is then tagged with its functional role in the sentence (subject and object).

5.1.3.3 Subroutines activation: Some words trigger additional rules that modify the data structure of the query and/or disambiguate some words. For instance, the word *What* followed by the word *team* (whose meaning is "Team=(blank)") modify the latter meaning to "Team=?".

5.1.4 Shortcomings

The main shortcoming of the system is the cost that is recquired to produce the semantic knowledge base (the lexicon and the set of semantic rules). Authors

suggest an improvement for handling unknown words, where the meaning could be expressed basing on existing words in the lexicon. However, to port the system to another domain, one needs to rewrite entirely the knowledge base which represents a significant effort.

5.2 LUNAR [?]

LUNAR was published more than ten years after BASEBALL. LUNAR (unlike BASEBALL) has been experimented with scientific data and targets expert users.

5.2.1 Data structure

Authors do not give much details about the data (provided by the NASA). It looks almost like relational tables with a dedicated formal query language. The data are about the chemical analysis of lunar rocks of the Apollo 11 expedition. The application domain is again very closed but more complex than that of BASEBALL; some expertise is required to validate the answered provided by LUNAR.

5.2.2 Internal query representation

Woods [?] has defined a meaning representation language which is used to represent internally users' intent. This language is a combination of propositions (whose evaluation leads to a truth value) and commands (or actions to be performed by the DBMS). Propositions are composed of database objects (classes or table names; instances or variables). Propositions are combined together with logical predicates like OR, AND, etc. Commands are "TEST" (to test the truth of a proposition), "PRINTOUT" to print out the evaluation of a proposition and commands for loops ("FOR") to be used with a quantifier.

5.2.3 Question translation step by step

5.2.3.1 Syntactic parsing: the input question is first syntactically parsed using a general-purpose grammar (domain-independent grammar). The grammar in use bases on Augmented Transition Networks linguistic formalism (Chomsky), which is almost equivalent to the context-free grammar formalism. The syntactic parsing also needs a lexicon which contains terms in use in the domain. It contains for instance the technical names of samples recolted during the expedition. "S10046" is thus recognized as a proper noun by the parser [?].

5.2.3.2 Semantic mapping: a set of rules transform the syntactic parse tree into a meaning representation (see section ??). The rules are triggered both by the syntactic structure of the parse tree (the label of nodes like "NP" for noun phrase or "VP" for verb phrase) but also on the question words (like "S10046" which is a sample name in the lexicon, or "contain" which has a semantics also defined in the lexicon). The rule results in a database query pattern with slots to be filled with items from the lexicon. As the system also supports quantification, authors presents some heuristics on how to resolve the attachment in the generated propositions.

5.2.3.3 Query execution: The internal query representation is composed of commands that the database retrieval component understands and executes to retrieve data and display/print them.

6 COMPLEX QUESTION TRANSLATION

This class of approaches aims at increase the coverage of NL questions. In addition to the increased linguistic coverage, the systems of this class need domain knowledge, but the involved processing are independant from the unerlying DBMS.

As for domain-dependant systems, the domain-dependent knowledge base corresponds to a lexicon which defines the "meaning" of words/expressions. This meaning is expressed with a set of semantic rules that map words/expressions and/or syntactic information to database query fragments.

Besides, the domain-independant knowledge base is composed of the following components:

1) a syntactic parser which operates iteratively with the semantic rules
2) a set of NLP tasks that aim at resolve linguistic ambiguities (e.g., anaphora resolution and ellipsis resolution [?])

The parsing might be performed iteratively with the semantic component process as in IRUS [?].

6.1 CHAT-80 [?]

It is the basis of many future interfaces [?] like MASQUE [?]. The database is composed of facts about world geography (facts about ocans, seas, rivers, cities and relations, the largest one is "borders"). The database itself is implemented as ordinary Prolog. Questions are expressed in a subset of English. The subset of English qustions is a formal but user-friendly language [?]. The main difference with its predecessor (LUNAR) is that much effort has been spent on increasing the linguistic coverage. In particular, the system translates English determiners (*a*, *the*, *some*, *all*, *every*) and negation and focus on linguistic phenomena, like noun attachment and "transformational aspects" (that cannot be covered by context-free grammars).

6.1.1 Portability

The authors claim the system to be adaptable to other application [?]. In particular, it is composed of a small vocabulary of about 100 domains (excluding proper nouns), and a domain-independant knowledge base of about 50 words.

6.1.2 Lexicons

The system is composed of a small vocabulary of English words that are related to the database domain, plus a dictionary of about 50 domain-independent words. These lexicons consist in rules in the XG formalism from Pereira; those rules are processed by Prolog and output

Prolog clauses. In addition to these two lexicons used to parse the question, a dictionary is made up of semantic rules in the form of templates that define how a word associated to a predicate must be also associated with its arguments.

6.1.3 Question translation step by step

6.1.3.1 Parsing: The parser analyzes the syntactic categories of words; words like determiners (domain-independant lexicon) and database-related elements (domain-dependant lexicon): these elements are nouns or verbs that correspond to database predicates. Proper nouns are represented by logical constants; most verbs, nouns and adjectives are represented as predicates with one or more constants.

6.1.3.2 Interpretation: The interpretation step consists in "filling" the predicates identified in the previous step. This is performed using a set of templates.

6.1.3.3 Scoping: This step is used to define the scope of determiners and some operators (for instance, operator to count).

6.1.3.4 Planning: The output of the previous step is a logical expression. However, to avoid combinatory explosion when executing the Prolog query, some strategies to optimize the query have been implemented: reordering the "predications" in the Prolog query; putting braces arround independent "subproblems" to avoid too many backtracking procedures

6.1.4 Limitations

Constraints in NL are not convered (for instance "Which ocean..." presuppose there is only one right answer).

6.1.4.1 Query execution: Even relatively complex queries are answered in less than one second [**?**]. The Prolog expression is executed to retrieve the answer. Authors note however, that the answering process (query execution) is the limiting factor (while modern systems are limited by the question analysis task).

6.2 QWERTY [**?**]

The specificity of this system is that it is intended to interface temporal databases. This system has thus an increased linguistic coverage, a better temporal expressivity. Input questions are expressed in controlled NL. The grammar used in the system takes into account some aspects of temporality of NL: tenses and temporal PPs that modify sentences. The system produces queries in SQL/Temporal, a database query language dedicated to temporal databases.

6.2.1 Portability

The grammar used for parsing questions and translating it into the formal language is specifically designed for use with a particular database schema. Thus, this system cannot be considered as a portable system.

6.2.2 Question translation step by step

6.2.2.1 Semantic parsing: The NL question is parsed using the Type Grammar framework. While parsing, the question is being trasformed into a logical representation called L_{Allen}. This formal language is based on interval operators. The translation bases on a linguistic theory, that semantics of sentences is modified by temporal preposition phrases (PPs). In this work, PPs are considered as variants of standard generalized quantifiers, where the quantification is over time. The temporality in NL questions can be explicit (like "When", "Which year") or implicit ("Did Mary work in marketing?"). The quantification also allows iterations of PPs ("every year until 1992"). In addition to temporal quantifiers, the system recognized quantification over individuals ("some employees"), coordination and negation. The semantic mapping to logical expression is performed basing on a bottom-up approach, simultineously with the parsing.

6.2.2.2 Query translation: The logic expression is translated in SQL/Temporal, the database query language dedicated to temporal databases. Some logical query produce infinite SQL/Temporal queries. Some heuristics have been implemented to prevent such behaviour.

6.2.2.3 Query execution: The SQL/Temporal query is finally evaluated by the database engine to produce the answers.

6.3 IRUS [**?**]

The IRUS system processes the question independantly from the underlying domain and DBMS, which is a big change wrt previous systems. The meaning formalism to represent internally the query is the same as in the LUANR system (MRL) but its expression is domain and DBMS-independent. Besides, IRUS analyses linguistically the question and integrates state-of-the-art NLP components, such as anaphora and ellipsis resolution.

6.3.1 Internal query representation

The internal meaning representation language is a descendant of that of LUNAR The language has the following general form [**?**]:

$$(FOR < quant > X/ < class >: (p\ X); (q\ X))$$

where $quant$ is a quantifier such as EVERY, SOME, THREE, HALF, etc., X is the variable of quantification, $< class >$ is the class of quantification of X, $(P\ X)$ is a predicate that restricts the domain of quantification and $(q\ X)$ is an expression being quantified, or an action such as $PRINT\ Y$.

6.3.2 Question translation step by step

6.3.2.1 Syntactic parsing: The syntactic parsing of NL questions is performed using the ATN grammar formalism. The authors claim that the syntactic parser

can benefit from semantic mapping as well in the syntactic parsing, both evolving in a *cascaded system*. The output of the syntactic parsing is a partse tree, where nodes correspond to syntactic information about question words/phrases

6.3.2.2 Semantic mapping: The semantic mapping is done in interaction with the syntactic parser. It requires a domain lexicon, which defines the semantics of the words and expressions used in users' queries. The main subtasks involve disambiguation (pronouns and other anaphoric expression resolution, ellipsis resolution, references resolution through discource information).

6.3.2.3 Query execution: The MRL can be used to interface any database system, at the condition that there is a component responsible for the translation from the MRL to the target dtabase query language.

6.4 PRECISE [?], [?]

PRECISE maps questions expressed in natural language to SQL queries. The interesting approach in this system, is the introduction of *semantically tractable questions*. This class of questions is guaranteed to be mapped to the correct SQL query. This reduces then the classic gap between the query space and the data space.

6.4.1 Internal query representation

The internal query representation is original compared to previous systems. The system builds an "attribute-value graph" that maps words of the question to database elements, and a "relation graph" that maps "relation tokens" (some question words) to database relation names. Both graphs are then used to generate the final SQL query.

6.4.2 Lexicon

The lexicon defines the mapping between tokens and database elements. It is composed of 1) the tokens; 2) the database elements; 3) the binary relations that bind both tokens and database elements.

6.4.3 Question translation step by step

6.4.3.1 Syntactic parsing: A lexicon is composed of automatically extracted database elements, and is used to perform the matching with question tokens. NLP task are: tokenizing the question into words, categorizing those tokens into "syntactic markers" (empty words?) and "tokens" (which are words that can be potentially associated with database elements). In addition to the lexicon, the system expects a parser (implemented as a plug-in, it can thus be changed for experimentation purposes). Authors experimented it with a syntactic dependency parser which outputs a graph ("attribute/value graph") composed of paths linking database elements together. Those path will then be composed together (aggregation / foreign keys for relational databases). The lexicon is also composed of a set of restrictions corresponding to prepositions and verbs: those restrictions define the join paths connecting relation/attributes. The set of those restrictions defines the semantic part of the lexicon. A component is also responsible for classifying questions into tractable and not-tractable questions. This is done linking words of the question with a set of "compatible" elements of the database. PRECISE also implements strategies for correcting syntactic parsing errors (semantic over-rides) given the semantics (defined in the lexicon).

6.4.3.2 Semantic mapping: the interpretation consists in choosing paths between database elements. This choice is a constraint satisfying problem. Paths generate SQL fragments that are then aggregated. The aggregation is done using the join paths available in the restrictions.

6.4.4 Feedback component

In cases when no interpretation is possible (even trying to correct possible syntactic errors) the system asks the user to rephrase the question.

6.5 PANTO [?]

PANTO generates SPARQL queries, that can be executed to get answers to the information need expressed through a question in NL. The data are organized in a knowledge base, more specifically an ontology (RDF or OWL formalism). The most interesting aspect in PANTO is that its most important component (the linguistic component, the parser) is implemented as a "plug-in" component, that can be easily replaced by an other one. This permits thus to benefit from the improvements in terms of linguistic coverage, when integrating state-of-the-art parsers.

6.5.1 Portablity

"PANTO is designed to be ontology-portable". To ensure portability, a domain-independent component is comprised in PANTO. This component is a lexicon composed of WordNet entries. For portability purposes, users can collaborate and improve the domain-dependent component (the domain ontology) defining their own synonyms to be added in the lexicon used in the question parsing step.

6.5.2 Internal query representation

The internal query representation is a graph representation called "query triples". It is a representation of the parse tree, that is then mapped to database queries (i.e. RDF triples) thanks to the lexicon.

6.5.3 Question translation step by step

6.5.3.1 Syntactic parsing: words from the NL query are first mapped to entities (concepts, instances, relations) of the ontology. This corresponds to the "entity recognition" task, and several tools are used, such as WordNet and string metrics algorithms. Then, a syntactic

parser is used to recognize nominal phrases in the sentence (the question). Those phrases are represented by pairs in the parse treee. In addition, some NLP tasks are integrated in this step, such as the negation recognition.

6.5.3.2 Semantic mapping: Pairs of nominal phrases from the parse tree are associated to triples in the sense of the ontology, and composed of entities of the domain ontology. This association is performed using the domain knowledge (the database). Besides, two additonal components are involved in this step: the question target identifier which identifies the target in the parse tree, and a component responsible for the recognition of solution modifier in the sense of SPARQL (e.g., commands "FILTER" or "UNION"). Those components basically base on rules triggered by the recognition of some words (e.g., wh-words for the former component). The triples mentionned before along with the target and modifier information constitute the internal representation of the query.

6.5.3.3 Query execution: The internal representation of the query mentionned above is interpreted into SPARQL statements, that can then be executed to retrieve requested facts. The target item is used in this step to decide what to put after the "SELECT" command in the generated SPARQL statement. Query post-processing procedures are triggered, for example basing on the negation recognized in the parsing step. The negation is usually translated in the "FILTER" SPARQL clause to specificy the set of triples that must not appear in the result of the execution of the SPARQL statement.

7 FEEDBACK-DRIVEN APPROACHES

The systems presented in the previous section still require intensive configuration efforts, and thus cannot be considered as *portable*. As a result, several systems have arisen where semantic grammars are created automatically on the basis of user interaction. We distinguishes between configuration (authoring tools) and knowledge acquisition through user interaction in use mode.

7.1 TEAM [?]

The data are structured in a database about geographic facts like the largest cities in the world, the population of every countries etc. The TEAM system is intended to be used with two kinds of users: standard users and database experts, who engages dialogue to provide needed information to port the system to other application domains. This system belongs thus in *configurable* systems, where the required knowledge to port the system is provided by the expert user, through the interaction with an authoring tool.

7.1.1 Internal query representation

The meaning of users' queries are internally represented in formal logic.

7.1.2 Lexicon

A lexicon is used to map words and expressions of NL to their meaning in terms of database elements. Close classes of words are supposed to be domain-independent and a fix meaning. Open classes words, however, have a much more important frequency in users' queries, and their meaning is supposed to be domain-independent. The "meaning" is composed of both syntactic and semantic information. Entries for nouns are the instance to which their refer, or a class (or concept) in a type hierarchy; entries for adjectives and verbs correspond to the possible predicate and how to find arguments of the predicate in the NL question.

7.1.3 Database schema

In addition to the lexicon, a resource about how logical forms can be translated in terms of database elements must be available. This resource expresses for instance the link between predicates (that appear in the logical forms) and database relations and attributes or the definition of the class hierarchy in terms of database relations or fields.

7.1.4 Portability

Portability is performed in using the system in a different mode (knowledge acquisition). In this mode, the database/domain expert informs the system on how data are organized in the database, what are the database elements and what words and expressions from NL are used for those elements. The acquisition consists in a tool, where the expert must answer questions; the answers of those questions will impact the resources like the lexicon and the database schema, that are both used to interface the database.

7.1.5 Question translation step by step

The system is divided into two sub-systems: DIALOGIC that maps NL questions to formal expressions and a schema translator that translates formal logical queries in database queries.

7.1.5.1 Syntactic parsing: The parsing of the NL question is performed using an augmented-phrase structure grammar. The parser produces possibly several parse trees for one question; then one parse tree is selected basing on syntactic heuristics.

7.1.5.2 Semantic mapping: Several processings are responsible for resolving some domain-specific ambiguities like noun-noun combinations and "vague" predicates like *have* or *of* [**?**]. Finally, a quantifier determination process is triggered. In the end, a logical form of the question is identified.

7.1.5.3 Query generation: The logical form is then translated in the database query. This translation needs the conceptual schema as well as the database schema (which defines the structure of the database) to perform the translation.

7.2 MASQUE/SQL [?]

MASQUE is a NL interface to Prolog databases, and MASQUE/SQL is an extension of it that supports SQL query language. The system is entirely written in Prolog. The system is meant to be domain-portable. Users can indeed add new entity in the lexicon through a domain-editor.

7.2.1 Lexicon

The lexicon defines the semantic of words that are expected to appear in users' questions. The meaning of words are described in logic predicate. Possible argument types of predicates are organized using the hierarchical *is-a* relation.

7.2.2 Internal query representation

The internal query representation is a Prolog-like language called Logical Query Language. Question words are translated to predicates or predicate arguments. The types of those arguments are described in the *is-a* relation hierarchy

7.2.3 Portability

The system can be used with different domain databases, but this requires to edit the specific knowledge in an editor. This knowledge consists in entities linked with the hierarchical *is-a* relation. User also has to explicit links between words and corresponding logic predicates; the entities in the taxonomy are used to restrict the possible arguments of the predicates. Each predicate is also linked to the corresponding SQL statement.

7.2.4 Question translation step by step

7.2.4.1 Syntactic parsing: a dictionary consists of all English words that the system understands and is used by an extraposition grammar to parse the question.

7.2.4.2 Semantic mapping: the dictionary composed of lexical units also associates words to their meaning in the form of logic predicate. The internal formal representation is a Prolog-like meaning representation language (LQL).

7.2.4.3 Query execution: The internal representation is translated to SQL using an algorithm, and then SQL is executed in the underlying DBMS. The translation consists of rules triggered by the structure of the LQL expression; each unit LQL expression is associated with a SQL fragment; all fragments are then combined to produce the final SQL expression. The system has also been experimented with Prolog as query language in association with a Prolog database.

7.3 NALIX [?] and DANALIX [?]

NALIX is an interactive interface to XML databases for questions expressed in natural language.

7.3.1 Question history

The system is composed of a query history that keeps all successfully answered queries. Queries from this history are intended to be used as templates for formulating new queries. This feature also permits users better understand the linguistic coverage of the system.

7.3.2 Internal query representation

The internal query representation is the target database query, *i.e.* XQuery.

7.3.3 Portability

There is no internal representation of the query (the NL question is directly translated to XQuery). Thus, the translation is dependent on the database and the chosen query language (XQuery). In addition to the portability feature of NALIX, DANALIX takes advantage of domain-dependent knowledge which can also be automatically acquired from user interaction. When ported to a new domain, the system starts with a generic framework; then domain-dependent knowledge is learned from user interaction.

7.3.4 Question translation step by step

7.3.4.1 Syntactic parsing: The parsing consists in identifying words and phrases using MINIPAR which is a dependency parser (dependency among words and not hierarchical constituents). An other component is responsible for checking whether words/phrases identified in the previous step can be mapped to directives (e.g., "return" or "group by" clauses) in the target query language (XQuery). Each word/phrase that can match a directive is further typed, depending on the kind of directive. The output of the parser is a tree with those roles as labels of phrases of the initial query. The vocabulary mismatch is overcome using WordNet[5]. In DANALIX, an additional step consists in transforming the parse tree using domain knowledge, basing on relevant rules.

7.3.4.2 Semantic mapping: Each item from the parse tree is translated into a constituent of the target query language (XQuery). This is done using a series of processings; basically a set of rules define how to combine items of the parse tree to XQuery clauses. There are also further treatments like "nesting" and "grouping" developped in [?]. In cases when the system does not understand how to map a given phrase to a XQuery constituent, the system interacts with users and suggests potential reformulations. NALIX seems to be the first NL interfaces that introduces user interaction to select the correct parse interpretation.

7.3.4.3 Query execution: The question is directly translated into a XQuery expression, which is the data query language. The XQuery expression is then executed to retrieve answers. Answers are basically XML answers. Different visualization of the answers are possible (text

5. http://www.cogsci.princeton.edu/~wn/

view for simple answers, hierarchical list view or raw XML). Besides answers to queries, the system implements an advanced error manager that also supports users in rephrasing queries that were not parsed and translated correctly.

7.4 C-PHRASE [?]

C-PHRASE is a system that translate questions expressed in NL in a query for relational database. The system outputs expression in tuple calculus (FOL) that can be easily translated in a database query language like SQL.

7.4.1 Lexicon

Semantic information are encoded in a lexicon. It maps tokens with syntactic information (such as head/modifier in dependency analysis) and semantic information (translation in λ-calculus). It is represented as a set of rules that form the grammar of the parser. For instance, the rule

$$\text{HEAD} \rightarrow \langle \text{“cities”}, \lambda x.City(x) \rangle$$

In addition, the system comprises a set of sentence patterns, in the form of context-free rules. For instance:

$$\text{QUERY} \rightarrow \langle \text{“list the”}.NP, answers(x|NP(x)) \rangle$$

Authors say they “rarely” see users who do not use already defined sentence paterns. This set of rules are automatically created by an authoring tool. The tool explicitely asks for meaningful names of different database elements (relations, attributes and join paths). It also permits to define new concepts in NL language.

7.4.2 Internal query representation

The internal query reprsentation is λ-calculus.

7.4.3 Question translation step by step

7.4.3.1 Semantic parsing: The parsing bases on a context-free grammar, augmented with λ-calculus expressions. The framework (λ-SCFG) works with two parsing trees: one for parsing syntactically the NL sentence; the other one for expressing the semantics of the first one in λ-calculus. The input question is first tokenized and normalized. Then, the sequence of tokens is analyzed to idenfity database elements, numeric values, and proceed to some spelling corrections. The structure is then transformed in the form of tuple calculus queries. This is done based on a set of rules that map lexical/syntactic parses to tuples. Each rule has a plausibility; in the end, the product of plausitilities of all rules used in a parse is used to rank potential semantic interpretations. In case of ambiguity, the user is asked to rephrase the question, or to select some rephrasing propositions.

7.4.3.2 Query generation: The tuple query is converted in SQL.

7.5 ORAKEL [?]

ORAKEL [?]'s main feature is the portability. This portability is based on the use of subcategorization frames, which are linguistic structures (predicate + arguments). The feedback aspect consists of an authoring tool. This tool (FrameMapper) is used to generate the domain-specific knowledge, and is intended to be used by expert of the domain.

7.5.1 Role of lexicon

The system is composed of three kinds of lexicon: 1) domain-independant lexicon defining determiners, *wh*-pronouns and spatio-temporal prepositions; 2) domain-specific lexicon which defines the meaning of verbs, nouns and adjectives (see section **??**); and 3) ontological lexicon which is automatically created from the data, and maps ontology instances and concepts to proper nouns and standard nouns. Categories of the domain-independant lexicon are “generic” categories such as those of a generic ontology like DOLCE.

7.5.2 Internal query representation

The internal query representation is λ-calculus. The question is represented in a language which is an extension of FOL (in addition to FOL: quantifiers and operators).

7.5.3 Customization process / portability

Users can customize the system, *i.e.* create a domain-specific lexicon which maps subcategorization frames (arity n) to an ontology relation (arity n) (a definition in the database). The interesting thing is that binary relation where the range is an integer (for instance $height(mountain, int)$) can also be mapped to adjectives with the different possible degrees (base, comparative, superlative forms) and possibly the positive or negative scale (as in TEAM). This mapping is performed by users themselves through a front-end.

7.5.4 Question translation step by step

7.5.4.1 Semantic mapping: Both processes question parsing and semantic mapping are done in the same process. The parsing is based on the LTAG linguistic representation. The parsing operates in a bottom-up fashion: each question word is associated with an elementary tree. Then, all elementary trees are combined together to get the syntactic parse tree of the entire sentence.

7.5.4.2 Query execution: The FOL internal query representation is then translated in the database query language like SPARQL (for ontologies expressed in OWL) or the language used for ontologies expressed in F-Logic. This translation is peformed using a Prolog program.

7.5.5 Shortcomings

Supports only factoid questions (*wh*-questions plus questions starting with expressions like "How many" for counting) but not "complex" questions like those starting with *why* or *how*. Besides, there is a (strong) assumption that categories of the domain-independant lexicon should be aligned with those of the database (the domain ontology). The second one, is that the generation of the ontological lexicon assumes that the labels used in the database correspond to instance/concept names. The front-end used to improve the domain-specific knowledge permits to increase the linguistic coverage and is a nice tool for porting the interface; however it might not be user-friendly since the underlying concepts (like the semantic frames) are complex.

8 LEARNING-BASED APPROACHES

Learning-based approaches are approaches where machine learning algorithms are the core of the translation mechanism. These algorithms aims at learning domain-specific knowledge (e.g., a lexicon). This knowledge is used to parse the question and get clues on on how to translate it to a database query.

8.1 Miller et al. [?]

The main caracteristics of this system is that it is fully statiscal. It is composed of three components for parsing, semantic interpretation and discourse resolution and are associated with corresponding statistic models. Each component produces a set of ranked items, and the chosen interpretation of the question is the best one of the final component.

8.1.1 Question translation step by step

The different steps correspond to the different components of the system.

8.1.1.1 Syntactic parsing: The string of words W is searched for the n-best candidates of parse trees T basing on the measure $P(T)P(W|T)$. The parse tree contains syntactic information as well as semantic ones. The statistical model is based on the recursive transition network model. This model is more detailed section **??**. The model is trained with a set of questions annotated with the corresponding parse trees.

8.1.1.2 Semantic mapping: The semantic mapping step is composed of two sub-steps: first, a model associates a "pre-discourse meaning" to both a parse tree and the corresponding string of words. Second, a "post-discourse meaning" is retrieved from the pre-discourse meaning, the string of caracters, the parse tree and the history. Both pre-discourse meaning and post-discourse meaning are represented using frame semantics. The construction of the frames is integrated in the parse tree. Then, the statistical model is used to disambiguate the frames, for instance when there is no information about the frame type or when there is no information about a slot fill. The second phase (post-discourse meaning) correspond to ellipsis resolution. It takes the previous meaning (final meaning of previous questions) and the pre-discourse meaning (of the current question), and finds the best meaning. Previous meaning and pre-discourse meaning are represented as vectors, whose elements correspond to slots in the frame meaning representation. The statistical model applies different kinds of operations on those vectors (INITIAL, TACIT, REITERATE, CHANGE and IRRELEVANT). Those operations combine elemnts of both vectors to compute the vector which corresponds to the meaning of the current question.

8.1.1.3 Query generation: The paper does not expalin how is performed the query generation.

8.2 Zettlemoyer et Collins [?]

The system aims at translating NL sentences to λ-calculus expressions. The system is based on a lerning algorithm that needs a corpus of sentences labelled with λ-calculus expressions. The system induces then a grammar. The statistic model is a log-linear model. The paper focus only on the generation of λ-calculus expression from questions expressed in natural language.

8.2.1 Portability

The proposal has been tested for two application domains: a database of US geography and a database of job listings.

8.2.2 Internal query representation

The query is internally represented in λ-calculus.

8.2.3 Question translation

The syntactic parsing is performed using a combinatory categorial grammar. Resulting parse tree's nodes are composed of both syntactic and semantic information. The grammar of the parse operates with a domain-specific lexicon, which maps question words/expressions to a syntactic type as well as a semantic type. Several funtional rules define how syntactic types can be associated

8.2.4 Learning component

The learning algorithm takes as input a training set composed of pairs (S_i, L_i) where S_i is a string of words and L_i a logical form. The algorithm also takes as input a lexicon Λ_0. The learning algorithm will determine how a given string of words will be parsed to produce a parse tree, and what words from the lexicon are required to produce such trees. The algorithm involves then learning the lexicon and learning the distribution over parse trees for a given string of words.

8.2.5 Shortcomings

The experiments are based on database rewritten in λ-calculus. It would be interesting to implement an additional module that transforms λ-calculus expressions to any database query language and measure the overall performance.

8.3 WOLFIE [?]

WOLFIE is a system that learns a lexicon, that is used to translate NL questions in database queries.

8.3.1 Lexicon

The lexicon consists in a mapping from sentences to semantic representation (logical database queries). The mapping consists in two steps: first, phrases that compose the sentence are associated to database elements (or *symbols* in the representation). Second, it says how those intermediate representation must be combined to get the final database query.

8.3.2 Learning

Due to the nature of the lexicon, the number of possible interpretations given a NL sentence is very complex (the problem is computationally intractable). To cope with that, authors propose an active learning algorithm which selects only most relevant examples to be then annotated. The learning algorithm bases on a greedy approach. The generation of candidate meanings for each phrase consists in finding maximally common meaning for each phrase. But, the algorithm also aims at finding a general meaning for the whole sentence (not only meanings for the phrases that are part of the sentence).

- evaluation: corpus of 250 questions on US geography paired with Prolog queries

8.3.3 Shortcomings

The system has not been evaluated with long phrases; the paper describes how it works for phrases of maximum two words. Besides, "the algorithm is not guaranteed to learn a correct lexicon in even a noise-free corpus".

9 SCHEMA-UNAWARE APPROACHES

9.1 POWERAQUA [?]

POWERAQUA takes as input several semantic sources (heterogeneous ontologies) and a question (expressed in natural language). This system is an improvement of the system AQUALOG [?]. The latter system was able to answer questions related to one domain ontology only (which is a problem in the context of the semantic web composed of many heterogeneous ontologies). The geatest contribution of POWERAQUA is the method for mapping user terminology with ontology terminology (and the system is unaware of the structure of the semantic sources).

9.1.1 Domain portability

The portability cost is "negligible". Indeed, the system has a learning component responsible for the acquisition of domain-specific lexicon which maps users' relations (expressed in natural language) to knowledge represented in the domain ontology.

9.1.2 Internal query representation

The internal query representation is called *query triples*. It is triples like RDF triples used in the data (ontology), but these triples are based on terms of the query.

9.1.3 Lexicon

POWERAQUA (as well as AQUALOG) use domain-independant lexicon WordNet.

9.1.4 Question translation step by step

9.1.4.1 Syntactic parsing: Users' questions are first parsed syntactically, and questions are categorized (for instance on the basis of the *wh*-question word). Both the parse tree and the category of the question are used to generate the internal query representation, query triples. The sense of words used in the question are disambiguated using word sense disambiguation algorithms. Those query triples are composed of words of the question, and may be then modified so that they are *compatible* with the database. The linguistic component of POWERAQUA also consists in algorithms that make decisions, for instance for correctly interpreting the conjunction / disjunction terms (and/or). The question may also contain constraints, that are sub-questions to be anewered prior the actual question. Furthermore, the system provides a way to treat two instances (from two semantic sources) as equivalent, and thus ease the answering process (in particular in cases where more than one semantic source are required to answer the question). The mapping from users' terms to database elements is done using edition distance-like metrics.

9.1.4.2 Semantic mapping: the semantic mapping consists in processing query triples to retrieve the ontology triples that are associated to the query triples. First, the algorithm tries to filter the sources (the ontologies) in order to only consider those that contain all or most of the *query triples*. Query triples must then be filtered to take into account the potential high number of triples generated in the previous step, and the fact that a question can involve several different semantic sources. The mappings established in the previous step (query terms and ontology terms) are ranked basing on a sense disambiguation algorithm that uses WordNet and the is-a taxonomy. In this step, the equivalence between two terms is computed based on the label of the term (edition distance) but also on the position in the taxonomy (ancestors and descendants).

9.1.4.3 Query generation: Potential ontology terms (database elements) must be processed to generate the final ontology query. Terms identified in different relevant ontologies must be used to generate triples (first sub-step); these triples must then be linked, these triples belonging or not to the same ontology (second sub-step). In this step, relations will be created from terms identified in the previous step. A relation (in the query triple) does not always correspond to a relation (expressed in ontology triple): sometimes, a new triples must be generated.

9.1.5 Limitation

Authors make an assumption, that the Semantic Web provides an indexing mechanism.

9.2 DEEPQA **[?]**

The DEEPQA project is part of the well-known WATSON system. WATSON has been the major event in the Q&A community in 2010. Indeed, it is the first artificial system which has won the american *Jeopardy!* quizz. WATSON queries a wide range of different sources and aggregates the different results to produce answers.

9.2.1 Question

WATSON has been designed for *Jeopardy!* where questions are not standard questions, but *clues* that should help understanding what the question is about. For instance, in a classic Q&A process a question would be "What drug has been shown to relieve the symptoms of ADD with relatively few side effects?". In *Jeopardy!* the corresponding clue would be "This drug has been shown to erlieve the symptoms of ADD with relatively few side effects". The expected response would then be "What is the Ritalin?" (see section **??**) [**?**].

9.2.2 Answers

Another particularity of *Jeopardy!* is that an answer are not simply a phrase corresponding to what the clues are about, but the answer must be the expected question corresponding to the clues. For instance, a valid response can be "Who is Ulysses S. Grant?" but not "Ulysses S. Grant" [**?**].

9.2.3 Data

The data (sources) used by WATSON are mostly unstructured documents as in most Q&A systems. DEEPQA however also leverages databases and ontologies such as DBPEDIA[6] and the YAGO[7] ontology; this is why we consider this system in this survey.

9.2.4 Question translation

Question translation for databases involves different steps.

6. See http://dbpedia.org/About.
7. See http://www.mpi-inf.mpg.de/yago-naga/yago/.

9.2.4.1 Question analysis: it involves many tasks: shallow parses; deep parses; logical forms; semantic role labelling; coreference resolution; relations (attachment relation and semantic relations); named entities; ...Semantic relation detection is one of those tasks. For instance for the question "They're the two states you could be reentering if you're crossing Florida's northern border", the relation would be `borders(Florida, ?x, north)`.

9.2.4.2 Hypothesis generation: The module responsible for this step takes as input the analysis parses (see section **??**) and generates candidate answers for every system sources. Those candidate answers are considered as hypothesis to be then proved basing on some degree of confidence. To produce candidate answers, a large variety of search techniques are used. Most of them concern textual search, but some consist in searching knowledge bases, more specifically triple stores. The different search techniques lead to the generation of multiple search queries for a single question; then the result list is modified to take into account constraints identified in the question. The search is based on named entities identified in the clue. If a semantic relation has been identified in the question analysis step **??**, a more specific SPARQL query can be performed on the triple store. In this step, recall is preferred as precision, leading to the generation of hundreds of candidate answers to be then ranked according to the confidence of each candidate.

9.2.4.3 Soft filtering: Candidate answers are not directly scored and ranked, since those algorithms require lots of resources. Instead, the hundreads of candidate answers corresponding to a single question are first pruned, to produce a subset of initial candidate answers. The involved algorithms are lightweight in the sense that they do not require intensive resources (soft filtering). The candidate that do not pass the soft filtering threshold are routed directly to the final merging stage. The model used to filter the candidate in this step as well as the threshold are determined on the basis of machine learning over training data.

9.2.4.4 Hypothesis and evidence scoring: Candidate answers that successfully pass the previous step are scored again in this step. This involves a wide range of scoring analystics to support evidences for candidate answers. First, evidences for the candidate answers are being retrieved. In the context of triple stores, those evidences are triples related to entities and semantic relations identified in the question. Those evidences are then scored, to measure the degree of certainty of these evidences. For instance the score is determined on the basis of subsumption, geospacial and temporal reasoning. The DEEPQA framework suports a wide range of scorers (components) that provide scores for evidences wrt a given candidate answer. The evidences from different types of sources can also be combined (for instance from unstructured content and from triple stores) using a wide range of metrics. In the end, scores

of evidences are combined into an *evidence profile*. This profile "groups features into aggragate evidence dimensions that provide an intuitive view of the feature view".

9.2.4.5 Final merging and ranking: This step aims at ranking and merging "the hundreds of hypotheses based on potentially hundreds of thousands of scores to identify the single best-supported hypthesis given the evidence and to estimate its confidence" (or likelyhood).

9.2.4.6 Answer merging: To cope with candidate answers that might be equivalent (*e.g.*, leading to the same answer) but with different surface forms, authors propose to first find similar candidate answers to merge them in a single candidate answer.

9.2.4.7 Final ranking: After merging candidate answers, they must be ranked based on merged scores. This is done using a machine-learning approach. This approach requires a training dataset of questions with known answers with appropriate scores. In this step ranking scores and confidence estimation (estimation of the likelihood of a given candidate answer) must be computed in two separated processes in intelligent systems. For both processes, the set of scores can be grouped according to the domain and intermediate models specifically used for the task. The output of these intermediate models is a set of intermediate scores. Then, a "metalearner" is trained over this ensemble of scores. This approach allows for iteratively improving the system with more complex models, and adding new scorers. The metalearner uses multiple trained models to handle different question classes (for instance, scores for factoid questions might not be appropriate for puzzle questions).

9.2.5 Drawbacks

WATSON has been primarly designed for unstructured content: "Watson's current ability to effectively use curated databases to simply 'look up' the answers is limited to fewer than 2 percent of the clues".

10 CHALLENGES

From Hearst [**?**]

- speech input (dialog-like interaction: Siri). This means that the future systems must be able to understand ill-formed sentences and mispelled words.
- social search (collaboration, asking people, crowdsourcing). This could also comprise systems that broadcast the question to other systems expert in some fields, and then aggregate the answer

The next generation of interfaces will not focus on the user (personalized systems) but on non-textual information through non-textual input [**?**].

11 CONCLUSION

The conclusion goes here.

REFERENCES

[1] M. A. Hearst, "'natural' search user interfaces," *Commun. ACM*, vol. 54, no. 11, pp. 60–67, Nov. 2011. [Online]. Available: http://doi.acm.org/10.1145/2018396.2018414

[2] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty, "Building watson: An overview of the deepqa project," *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010.

[3] V. Lopez, V. S. Uren, M. Sabou, and E. Motta, "Is question answering fit for the semantic web?: A survey," *Semantic Web*, vol. 2, no. 2, pp. 125–155, 2011.

[4] A. Kotov and C. Zhai, "Towards natural question guided search," in *Proceedings of the 19th international conference on World wide web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 541–550.

[5] A. Yates, O. Etzioni, and D. Weld, "A reliable natural language interface to household appliances," in *Proceedings of the 8th international conference on Intelligent user interfaces*, ser. IUI '03. New York, NY, USA: ACM, 2003, pp. 189–196. [Online]. Available: http://doi.acm.org/10.1145/604045.604075

[6] W. A. Woods, "Progress in natural language understanding: an application to lunar geology," in *Proceedings of the June 4-8, 1973, national computer conference and exposition*, ser. AFIPS '73. New York, NY, USA: ACM, 1973, pp. 441–450.

[7] B. F. Green, Jr., A. K. Wolf, C. Chomsky, and K. Laughery, "Baseball: an automatic question-answerer," in *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, ser. IRE-AIEE-ACM '61 (Western). New York, NY, USA: ACM, 1961, pp. 219–224.

[8] D. H. D. Warren and F. C. N. Pereira, "An efficient easily adaptable system for interpreting natural language queries," *Comput. Linguist.*, vol. 8, pp. 110–122, Jul. 1982. [Online]. Available: http://dl.acm.org/citation.cfm?id=972942.972944

[9] B. J. Grosz, D. E. Appelt, P. A. Martin, and F. C. N. Pereira, "Team: an experiment in the design of transportable natural-language interfaces," *Artif. Intell.*, vol. 32, pp. 173–243, May 1987. [Online]. Available: http://dl.acm.org/citation.cfm?id=25672.25674

[10] R. Nelken and N. Francez, "Querying temporal databases using controlled natural language," in *Proceedings of the 18th conference on Computational linguistics - Volume 2*, ser. COLING '00. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 1076–1080. [Online]. Available: http://dx.doi.org/10.3115/992730.992808

[11] M. Bates and R. J. Bobrow, "Information retrieval using a transportable natural language interface," in *Proceedings of the 6th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '83. New York, NY, USA: ACM, 1983, pp. 81–86. [Online]. Available: http://doi.acm.org/10.1145/511793.511804

[12] A.-M. Popescu, O. Etzioni, and H. Kautz, "Towards a theory of natural language interfaces to databases," in *Proceedings of the 8th international conference on Intelligent user interfaces*, ser. IUI '03. New York, NY, USA: ACM, 2003, pp. 149–157.

[13] I. Androutsopoulos, G. Ritchie, and P. Thanisch, "Masque/sql - an efficient and portable natural language query interface for relational databases," in *Proceedings of the 6th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*. Gordon and Breach Publishers Inc, 1993, pp. 327–330.

[14] Y. Li, H. Yang, and H. V. Jagadish, "Nalix: an interactive natural language interface for querying xml," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '05. New York, NY, USA: ACM, 2005, pp. 900–902.

[15] Y. Li, I. Chaudhuri, H. Yang, S. Singh, and H. V. Jagadish, "Danalix: a domain-adaptive natural language interface for querying xml," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '07. New York, NY, USA: ACM, 2007, pp. 1165–1168. [Online]. Available: http://doi.acm.org/10.1145/1247480.1247643

[16] M. Minock, "C-phrase: A system for building robust natural language interfaces to databases," *Data Knowl. Eng.*, vol. 69, pp. 290–302, March 2010. [Online]. Available: http://dx.doi.org/10.1016/j.datak.2009.10.007

[17] C. Wang, M. Xiong, Q. Zhou, and Y. Yu, "Panto: A portable natural language interface to ontologies," in *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, ser. ESWC '07. Berlin, Heidelberg: Springer-Verlag,

2007, pp. 473–487. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72667-8_34

[18] P. Cimiano, P. Haase, and J. Heizmann, "Porting natural language interfaces between domains: an experimental user study with the orakel system," in *Proceedings of the 12th international conference on Intelligent user interfaces*, ser. IUI '07. New York, NY, USA: ACM, 2007, pp. 180–189. [Online]. Available: http://doi.acm.org/10.1145/1216295.1216330

[19] S. Miller, D. Stallard, R. Bobrow, and R. Schwartz, "A fully statistical approach to natural language interfaces," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ser. ACL '96. Stroudsburg, PA, USA: Association for Computational Linguistics, 1996, pp. 55–61. [Online]. Available: http://dx.doi.org/10.3115/981863.981871

[20] L. S. Zettlemoyer and M. Collins, "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars," in *UAI*. AUAI Press, 2005, pp. 658–666.

[21] C. A. Thompson and R. J. Mooney, "Acquiring word-meaning mappings for natural language interfaces," *J. Artif. Int. Res.*, vol. 18, no. 1, pp. 1–44, Jan. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622420.1622421

[22] V. Lopez, E. Motta, and V. S. Uren, "Poweraqua: Fishing the semantic web," in *ESWC*, ser. Lecture Notes in Computer Science, Y. Sure and J. Domingue, Eds., vol. 4011. Springer, 2006, pp. 393–410.

[23] Y.-J. Han, T.-G. Noh, S.-B. Park, S. Y. Park, and S.-J. Lee, "A natural language interface of thorough coverage by concordance with knowledge bases," in *Proceedings of the 15th international conference on Intelligent user interfaces*, ser. IUI '10. New York, NY, USA: ACM, 2010, pp. 325–328. [Online]. Available: http://doi.acm.org/10.1145/1719970.1720022

[24] D. A. Norman, "How might people interact with agents," *Commun. ACM*, vol. 37, no. 7, pp. 68–71, Jul. 1994. [Online]. Available: http://doi.acm.org/10.1145/176789.176796

[25] B. Shneiderman and P. Maes, "Direct manipulation vs. interface agents," *interactions*, vol. 4, no. 6, pp. 42–61, Nov. 1997. [Online]. Available: http://doi.acm.org/10.1145/267505.267514

[26] A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates, "Modern natural language interfaces to databases: composing statistical parsing with semantic tractability," in *Proceedings of the 20th international conference on Computational Linguistics*, ser. COLING '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004.

[27] G. Koutrika, A. Simitsis, and Y. E. Ioannidis, "Explaining structured queries in natural language," in *ICDE*, F. Li, M. M. Moro, S. Ghandeharizadeh, J. R. Haritsa, G. Weikum, M. J. Carey, F. Casati, E. Y. Chang, I. Manolescu, S. Mehrotra, U. Dayal, and V. J. Tsotras, Eds. IEEE, 2010, pp. 333–344.

[28] A. Giordani and A. Moschitti, "Syntactic structural kernels for natural language interfaces to databases," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, ser. ECML PKDD '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 391–406.

[29] W. A. Woods, "Readings in natural language processing," B. J. Grosz, K. Sparck-Jones, and B. L. Webber, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1986, ch. Semantics and quantification in natural language question answering, pp. 205–248. [Online]. Available: http://dl.acm.org/citation.cfm?id=21922.24336

[30] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases - an introduction," *CoRR*, vol. cmp-lg/9503016, 1995.

[31] Y. Li, H. Yang, and H. V. Jagadish, "Constructing a generic natural language interface for an xml database," in *EDBT*, ser. Lecture Notes in Computer Science, Y. E. Ioannidis, M. H. Scholl, J. W. Schmidt, F. Matthes, M. Hatzopoulos, K. Böhm, A. Kemper, T. Grust, and C. Böhm, Eds., vol. 3896. Springer, 2006, pp. 737–754.

[32] V. L. Garcia, E. Motta, and V. Uren, "Aqualog: an ontology-driven question answering system to interface the semantic web," in *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume: demonstrations*, ser. NAACL-Demonstrations '06. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006, pp. 269–272. [Online]. Available: http://dx.doi.org/10.3115/1225785.1225790

System	Internal representation	DB query
BASEBALL [?]	specification list	√
LUNAR [?]	meaning representation language	X
CHAT-80 [?]	logic expression	X
TEAM [?]	logic expression	X
QWERTY [?]	logic expression	X
IRUS [?]	meaning representation language	X
PRECISE [?]	graph representation	X
MASQUE/SQL [?]	meaning representation language (LQL)	X
NALIX & DANALIX [?], [?]	XQuery	√
C-PHRASE [?]	λ-calculus	X
PANTO [?]	graph representation (query triples)	X
ORAKEL [?]	λ-calculus	X
Miller et al. [?]	semantic frames	X
Zettlemoyer et Collins [?]	λ-calculus	√
WOLFIE [?]	Prolog	√
POWERAQUA [?]	graph representation (query triples)	X
DEEPQA [?]	semantic triples	√

TABLE 1
Semantic meaning representations

Approach	System	Dimensions					
		Query	Data	Answers	Portability	Linguistic coverage	Error feedback
Domain-dependent semantic parsing	BASEBALL [?]						
	LUNAR [?]						
Complex question translation	CHAT-80 [?]						
	IRUS [?]						
	QWERTY [?]						
	PRECISE [?], [?]						
	PANTO [?]						
Feedback-driven approaches	MASQUE/SQL [?]						
	TEAM [?]						
	NALIX & DANALIX [?]						
	C-PHRASE [?]						
	ORAKEL [?]						
Learning-based approaches	Zettlemoyer et al. [?]						
	Miller et al. [?]						
	WOLFIE [?]						
Schema-unaware approaches	POWERAQUA [?]						
	DEEPQA [?]						

TABLE 2
Overview of major NL interfaces to structured data