

Computational Linguistics: Human Computer Interaction

RAFFAELLA BERNARDI

KRDB, FREE UNIVERSITY OF BOZEN-BOLZANO

P.ZZA DOMENICANI, ROOM: 2.28, E-MAIL: BERNARDI@INF.UNIBZ.IT

Contents

1	Human-Computer Interaction via Natural Language	4
2	Natural Language Interfaces to Data Bases	5
2.1	Advantages & Disadvantages	6
2.2	Experiments	7
2.3	Linguistic problems	8
2.4	Architecture	9
2.5	Parsers: query analysis	9
2.6	Sample architecture	11
2.7	Which approach	12
2.8	Response generation	13
2.9	Restricted NL input	14
2.10	NLIDBs as intelligent agents	15
2.11	ACE: Attempto Control Natural Language	16
2.12	Online Demos	17
3	Dialogue System	18
3.1	Practical Dialogues	19
3.2	Practical Dialogues Complexity	20

3.3	Example	21
3.4	Observations	22
3.5	Sample Architecture	23
3.6	Computational Approaches	24
3.7	Conventional Approach	25
3.8	Protocols as Finite State Automata	26
3.8.1	Failure of simple DFA-protocols	29
3.8.2	Follow-up	30
3.8.3	Protocol with memory	31
3.8.4	Protocols with stacks	32
3.8.5	Failure of protocols with stacks	33
3.8.6	Failure of protocols with stacks	34
3.9	Summary: Expressive power	35
4	Conclusion	37

1. Human-Computer Interaction via Natural Language

In the '50 Machine Translation work pointed out serious problems in trying to deal with unrestricted, extended text in open domains. This led researchers in the '60 and early '70 to focus on question-answering dialogues in restricted domain.

Attention shifted from developing NL systems to solving individual language-related problems, e.g., to develop faster, and more efficient parsers.

Now, researchers are back to deal with unrestricted extended text and dialogues.

1. NLDB
2. Dialogue Systems,
3. QA
4. IQA

All of them aim at assisting users to access data from some source. Today we speak of NLDB and Dialogue Systems, next time of QA and IQA.

2. Natural Language Interfaces to Data Bases

NLDB refers to systems that allow the user to access information stored in a database by typing requests in some natural language. Its history (see Androutsopoulos for more details):

'60/'70 they were built having a particular DB in mind. No interest in portability issues. E.g., LUNAR

late '70 Dialogues; large DB; semantic grammars (domain dependent - no portable). E.g. LADDEqR

early '80 From English into Prolog evaluated against Prolog DB. Eg., CHAT-80

mid '80 popular research area. Research focused on portability issues. E.g. TEAM

'90 NLIDBs did not gain the expected commercial acceptance. Alternative solutions were successful (graphical or form-based interface). Decrease in the number of papers on the topic.

2.1. Advantages & Disadvantages

▶ Advantages:

- ▶ NLDB should be easier to use. But: currently only limited subsets of NL. Hence, training is needed.
- ▶ It supports anaphoric and elliptic expressions.

▶ Disadvantages:

- ▶ The NL coverage is not clear to the user. False positive expectation and False negative expectation
- ▶ It is not clear to the user whether the rejected question is outside the system's linguistic coverage or the system's conceptual coverage. Need of diagnostic messages.
- ▶ User assume intelligence of the NLDBs.
- ▶ NL is verbose and ambiguous.
- ▶ Tedious configuration.

2.2. Experiments

- ▶ Training of the interfaces (graphical, SQL, NL). Then ask queries most of which are similar to the ones used in the training period.

Results: NLIDBs seem to be better in queries where data from many tables have to be combined and in queries that were not similar to the ones the users had encountered during the training period.

- ▶ NL is an effective method of interaction for casual users with a good knowledge of the DB, who perform question-answering tasks in a restricted domain.
- ▶ Another approach: Wizard of Oz experiment.

2.3. Linguistic problems

- ▶ Quantifier scoping. Ambiguous, ad hoc solutions (e.g., choose only one reading as possible, give different weights to QPs.)
- ▶ Conjunction and Disjunction: Sometime conjunctions in NL are actually interpreted as disjunction. E.g., List all applicants who live in California and Arizona. There are also cases of ambiguous use of “and”, e.g., Which minority and female applicants know Fortran?”
- ▶ Nominal compound problem: E.g., “research department” vs. “research system”. In the first case, the department carries out the research, in the second the system is used in research.
- ▶ Anaphora: Use of pronouns and possessive determiners or noun phrases to denote entities mentioned in the discourse. Solution: keep list of all entities, use the most recent one as link to the anaphora. Use of world knowledge.
- ▶ Elliptical sentences. E.g., U1: “Who is the manager of the largest department?” U2: “The smallest department?” Need of discourse model.

2.4. Architecture

2.5. Parsers: query analysis

Pattern-matching Simple, but bad failures.

Syntax Based The user's question is parsed and (each node of) the parse tree is mapped directly to an expression in some DB query language. (eg. LUNAR). The DB query language is designed to facilitate the mapping, since it is hard to map the parse tree into e.g. SQL.

Semantic Grammar Difficult to port to other knowledge domain.

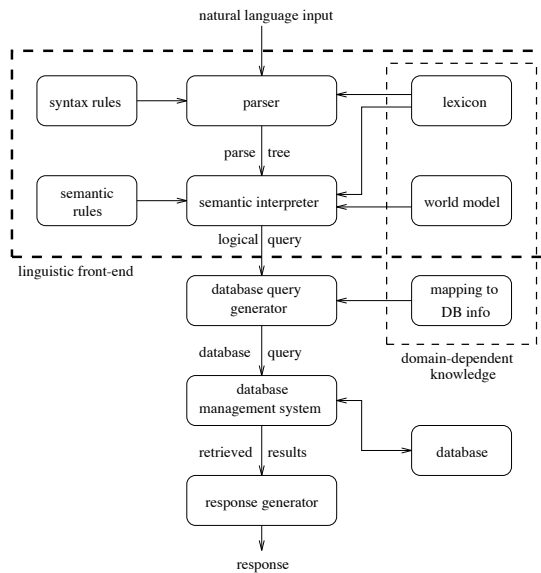
Intermediate representation languages NL question is transformed into an intermediate logical query –in terms of high level world concepts independent of the DB structure. This query is then translated into a DB query language supported by the underlying DBMS

E.g. JANUS: the semantic interpretation is built compositionally –syntactic rules correspond to semantic rules.

E.g. LOQUI: linguistically based formal grammar. Importance of lexicon, where the logical expressions corresponding to the words (possible leaf nodes) are stored. Lexicon is domain dependent whereas the rule-to-rule correspondence is not.

E.g. MASQUE: world models are used to specify the possible arguments of some predicate . This info can be used to give the diagnostic message to the user and specify whether there is a problem of conceptual coverage.

2.6. Sample architecture



2.7. Which approach

Advantages of the last approach: modularity of the architecture

- ▶ the linguistic front-end is independent of the underlying DBMS
- ▶ domain knowledge is separated from the rest of the front end
- ▶ reasoning modules can be added between the semantic interpreter and the DB query generator.

2.8. Response generation

Failure Explain cause of failure to retrieve answer.

False Presupposition The system should report the false presupposition about the DB world.

Literal answers some time a literal answer would be “yes/no” but it won’t be an acceptable answer. Cooperative answers can help. Sometime important to reason about the user’s goal.

Misunderstandings translate the SQL query back to NL, (paraphrase modules)

2.9. Restricted NL input

Currently systems use limited subsets of NL.

Limitation user doesn't know which is this subset. Has to rephrase the question, does not know which questions could be handled.

Long term aim to broad the linguistic coverage.

Alternative approach deliberately and explicitly restrict the set of NL the user is allowed to input (controlled natural language.)

syntactic pattern E.g PRE.

menu-based E.g. NLMENU,

ontology-driven See Paolo Dongilli's work.

complexity of NL fragments See Ian Pratt (in two weeks here!) and Camilo Thorne works

2.10. NLIDBs as intelligent agents

External modules could be used to reason

on the world To answer questions that cannot be answered by the DBMS

about user's goal Dialogue-oriented systems driven by a reasoning module which reasons about the goals and beliefs of both the user and the system.

2.11. ACE: Attempto Control Natural Language

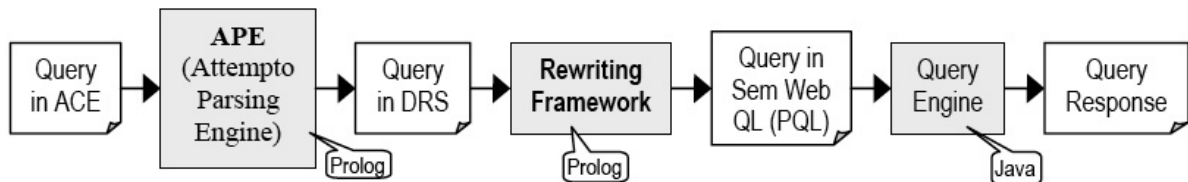


Figure 1: Overall data flow of the controlled English query front-end

2.12. Online Demos

Examples of today NLDBs:

- ▶ ACE: <http://attempto.ifi.unizh.ch/site/tools/>
- ▶ Geo <http://www.cs.utexas.edu/users/ml/geo-demo.html>
- ▶ PENG: <http://www.ics.mq.edu.au/~peng/PengEditor.html>
- ▶ PRECISE

3. Dialogue System

Two uses of Dialogue Systems:

- ▶ Dialogues as a way of controlling and restricting the interaction.
- ▶ Dialogues based on human conversation to enhance the interaction with the machine.

We focus not on human-human conversation rather on human-machine interactions where a concrete task must be achieved. We call these dialogues **practical** (Allen 2001)

3.1. Practical Dialogues

Practical Dialogues can be divided into: task-oriented dialogues, information-seeking dialogues, advice and tutoring dialogues, and command and control dialogues.

The conversational competence required for practical dialogues, while still complex, is significantly simpler to achieve than general human conversational competence. . . . Within the genre of practical dialogue, the bulk of the complexity in the language interpretation and dialogue management is independent of the task being performed.

3.2. Practical Dialogues Complexity

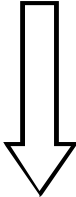
Technique Used	Example Task	Task Complexity	Dialogue Phenomena handled
Finite-state Script	Long-distance dialing	 <p>least complex</p> <p>most complex</p>	User answers questions
Frame-based	Getting train arrival and departure information		User asks questions, simple clarifications by system
Sets of Contexts	Travel booking agent		Shifts between predetermined topics
Plan-based Models	Kitchen design consultant		Dynamically generated topic structures, collaborative negotiation subdialogues
Agent-based Models	Disaster relief management		Different modalities (e.g., planned world and actual world)

Figure 1: Dialogue and Task Complexity

3.3. Example

Finite State Script and Frame Based systems are handled by robust pattern matching techniques and do not use deep language understanding whereas the others do. A simple interaction containing several challenges is the following from Allen et al.

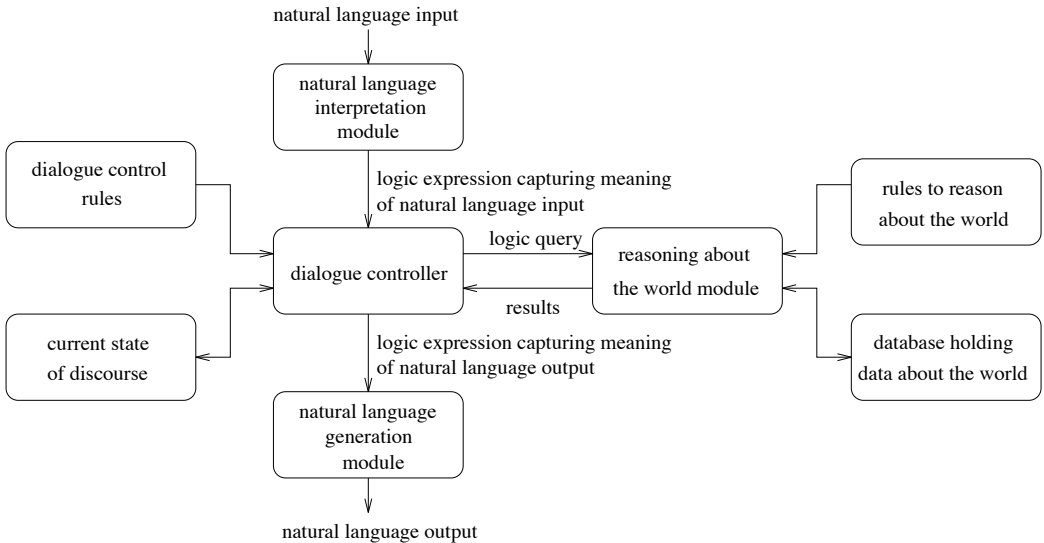
1. USR: We need to get the woman in Penfield to Strong
2. SYS: OK
3. USR: What vehicles are available?
4. SYS: There are ambulances in Pittsford and Webster
5. USR: Ok. Use one from Pittsford
6. SYS: Do you know that Route 96 is blocked due to construction?
7. USR: Oh
8. SYS: Let's use the interstate instead
9. USR: OK. I'll dispatch the crew

3.4. Observations

1. Utterance (1) “We need to get the woman in Penfield to Strong” is needed to establish a joint objective
2. Utterance (2) “OK” confirms the joint objective
3. Utterance (3) “What vehicles are available?” initiate a problem-solving act.
4. (4) “There are ambulances in Pittsford and Webster” only those available ambulances near to the place are listed.
5. (5) “Ok. Use one from Pittsford” it’s an attempt to specify a solution.
6. (6) “Do you know that Route 96 is blocked due to construction?” is a clarification question asked by the system.
7. ...

Many challenges! See also the course “Introduction to Linguistics” (e.g., speech Acts)

3.5. Sample Architecture



3.6. Computational Approaches

From Fernandez and Endriss (2007)

- ▶ **Conventional Approach** focused on public and conventional aspects of communication. A dialogue is seen as a conversational scoreboard that keeps track of the **state** of the conversation.
- ▶ **Plan Based Approach** This approach is also known as mentalistic approach. Attention is focused on the mental attitudes such as knowledge, belief, desire and intention. The most well known framework based on this is BDI (Cohen and Levesque 1990).

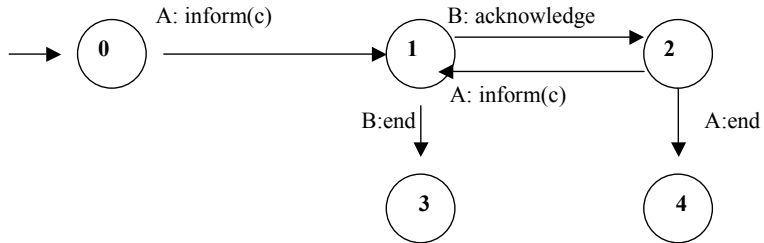
3.7. Conventional Approach

- ▶ The main idea is that each participant's contribution determines a set of preferred options for follow-up in the dialogue.
- ▶ A few standard interaction patterns are identified, viz. Questions are usually followed by answers and proposals are usually either accepted, rejected or countered.
- ▶ Hence, protocols can be seen as formal constructs modelling the public conventions behind these interaction patterns.
- ▶ Conventional protocols are used in multiagent systems to characterize coherent interaction between software agents. In this case, protocols describe the set of allowed or legal dialogue continuations.
- ▶ In natural language dialogues, protocols characterize the range of preferred or less-marked follow-ups in particular dialogue situations. The violation of a protocol can also be informative, as it can be seen as signalling a topic or task change (topic shift).

3.8. Protocols as Finite State Automata

Deterministic Finite State Automata (DFA) have been used to represent communication protocols.

Continuous update protocol specifies a class of dialogues between two agents A and B where A continuously updates B on the value of some proposition.



Definition for DFA-based protocol A DFA-base protocol is a quintuple $\langle Q, q_0, F, \mathcal{L}, \delta \rangle$ consisting of finite set of dialogue states Q , including an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$, a communication language \mathcal{L} , and a transition function $\delta : Q \times \mathcal{L} \rightarrow Q$.

The elements of the communication language \mathcal{L} are utterances and are constructed from a finite set \mathcal{A} of agents (or dialogue participants), a finite set \mathcal{M} of dialogues moves (or illocutory acts) , and a content language \mathcal{C} . We assume that every utterance has the structure $i : m(c)$ with $i \in \mathcal{A}, m \in \mathcal{M}$ and $c \in \mathcal{C}$. We consider only two participants dialogues, hence $\mathcal{A} = \{A, B\}$

3.8.1. Failure of simple DFA-protocols In natural language dialogue it's not uncommon to find embedded pairs of questions and answers (particularly in information oriented interaction). Eg.

- (1) A: Who should we invite? [Q₁]
- (2) B: Should we invite Bill? [Q₂]
- (3) A: Which Bill? [Q₃]
- (3) A: Jack's brother. [A₃]
- (3) A: Oh, yes. [A₂]
- (3) A: Ok, the we should invite Gill as well. [A₃]

Replying to a question with another question (2) and asking for clarification (3) are very common phenomena in natural language dialogue.

Simple DFA-protocols as in Definition 1 fails to model these kinds of dialogues.

To overcome these limitations we can provide DFA-protocols with **memory**. But first let's look at a definition of "follow-up" utterance.

3.8.2. Follow-up

Definition 2: Possible follow-up Given the current dialogue state q , an utterance u constitutes a possible follow-up of the dialogue iff there exists a state $q' \in Q$ such that $\delta(q, u) = q'$ holds.

Before a dialogue start we are in the initial state q_0 . The dialogue state then gets updated whenever an utterance is performed, following the transition function δ . A complete dialogue conforms to a given protocol iff it is accepted by the DFA, i.e. iff each utterance in the dialogue is a possible follow-up and the final utterance leads to a final state in F .

3.8.3. Protocol with memory Unlimited memory could be added by “abstract data type” (ADTs) such as “stack’s”, queues, sets or lists.

Every ADT comes with a set of basic operations, e.g. stacks come with $push(x)$, pop and top .

Definition 3: protocols with memory A protocol with memory based on a given ADT is a sextuple $(Q, q_0, F, \mathcal{L}, \mathcal{L}', \delta)$ consisting of a finite set of dialogues states Q , including an initial state q_0 and a set of final states $F \subseteq Q$, a communication language \mathcal{L} , a memory alphabet \mathcal{L}' , and a transition function $\delta : Q \times \Gamma \times \mathcal{L} \rightarrow Q \times \Gamma$, where Γ denotes the set of all possible configurations of the memory component.

Definition 4: possible follow-up Given the current dialogue state q and the current configuration of the memory component x , an utterance u constitutes a possible follow-up of the dialogue iff there exists a state $q' \in Q$ and a configuration $x' \in \Gamma$ such that $\delta(q, x, u) = (q', x')$.

A complete dialogue conforms to a given protocols with memory iff it is accepted by the automaton in question.

3.8.4. Protocols with stacks A stack allows us to store arbitrarily large amounts of information that are accessible in a last-in-first-out (LIFO) manner.

Such information can be manipulated by means of the function *top*, which returns the top element on the stack, and the operation *push*(*x*), which pushes element *x* onto the stack, and *pop*, which removes the top element from the stack.

Example: QUD Questions under discussion (Ginzburg 1996, 2007): the last question asked is the more salient question under-discussions which will be the first one to be answered. This can be modeled by a stack.

3.8.5. Failure of protocols with stacks

- | | | |
|---------|---|-------------------|
| (1) A: | Where were you on the 15th? | [Q ₁] |
| (2) A: | Do you remember talking to anyone after the incident? | [Q ₂] |
| (3) B: | I didn't talk to anyone | [A ₂] |
| (4) B: | I was at home | [A ₁] |
| (3') B: | I was at home | [A ₁] |
| (4') B: | I didn't talk to anyone | [A ₂] |

The questions under discussion could be answered in any order, they are in what has been called a “coordinate structure” (Asher 1998). When this is the case, a protocol based on a DFA plus a stack would not be appropriate to handle this phenomenon.

Protocols with stacks of sets This can be modelled by using a DFA together with a finite stack of sets.

3.8.6. Failure of protocols with stacks When successive questions are asked by a single speaker, the DFA-protocols with stacks fails.

- | | | |
|--------|---|-------------------|
| (1) A: | Who will you be inviting? | [Q ₁] |
| (2) A: | And why? | [Q ₂] |
| (3) B: | Mary and Bill, I guess | [A ₁] |
| (4) A: | Aha | [Ack] |
| (5) B: | Yeah, (because) they are very undemanding folks | [A ₂] |

The first question asked seem to take precedence over the last one –only after the first question is answered does the second question get addresses. This suggest that the order in the QUD is not determined solely by conventional means, but is also guided by semantic relation that may hold between its elements.

Ginzubrg (2007) identifies differences in terms of the relation that are said to hold between the questions: coordination in case questions can be answered in any order, and query extension in case they are more naturally answered in the order in which they have been posed.

To account for this, complex relations between the elements of the content language C would have to be encoded as part of the definition of the transition function δ .

3.9. Summary: Expressive power

1. DFA with a stack corresponds to a push-down automaton. Hence, we can claim that the **class of dialogues conforming to protocols with a stack strictly includes the class of dialogues conforming to DFA-based protocols.**
2. DFA enriched with a stack of sets is not more expressing than that of a push-down automaton (with a simple stack). Hence, we can claim that the **class of dialogues conforming to protocols with a stack is the same as the class of dialogues conforming to protocols with a stack of sets.**
3. Adding a set to a DFA does in fact not increase the expressive power, because the range of all possible configurations of the set component could be encoded into a larger DFA. Hence, we can conclude that **the class of dialogues conforming to DFA-based protocols is the same as the class of dialogues conforming to protocols with a set.**

Abstract Models	Examples
Shallow rules	simple communication protocols (Endriss et al. 2003)
DFA	finite state model of grounding (Traum, 1994)
DFA + set (=DFA)	commonly accepted facts (Ginzburg, 1996)
DFA + stack (=pushdown automata)	simplified questions under discussion (Ginzburg, 1996)
DFA+stack of sets (=pushdown automata)	questions under discussion (Ginzburg, 1996)
DFA+list (=Turing machine)	explicit representation of dialogue history

3.10. Online Demo

DORIS <http://www.coli.uni-saarland.de/bos/doris/>

4. Conclusion

Do you want to know more about the topic?

See Allen or Jurafsky's textbooks.

Come to today LCT Colloquium on “ From Natural Language to Databases via Ontologies” Leonardo Lesmo. (17:00-18:00, Seminar Room)

Next LCT colloquia on the 24th and 28th of May are on this topic.

No KBDB (David Toman) class this morning!