Decalog 2007: Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue, pages 83–90.
Trento, Italy, 30 May – 1 June 2007. Edited by Ron Artstein and Laure Vieu.

A Chatbot-based Interactive Question Answering System

Silvia Quarteroni
The University of York
York, YO10 5DD
United Kingdom
silvia@cs.york.ac.uk

Suresh Manandhar
The University of York
York, YO10 5DD
United Kingdom
suresh@cs.york.ac.uk

Abstract

Interactive question answering (QA) systems, where a dialogue interface enables followup and clarification questions, are a recent field of research. We report our experience on the design, implementation and evaluation of a chatbot-based dialogue interface for our open-domain QA system, showing that chatbots can be effective in supporting interactive QA.

1 Introduction

Question answering (QA) systems can be seen as information retrieval systems which aim at responding to natural language queries by returning answers rather than lists of documents.

Although QA differs from standard information retrieval in the response format, both processes share a lack of interactivity. In the typical information-seeking session the user submits a query and the system returns a result; the session is then concluded and forgotten by the system.

It has been argued (Hobbs, 2002) that providing a QA system with a dialogue interface would encourage and accommodate the submission of multiple related questions and handle the user's requests for clarification. Indeed, information-seeking dialogue applications of QA are still at an early stage and often relate to close domains (Small et al., 2003; Jönsson and Merkel, 2003; Kato et al., 2006).

In this paper, we report on the design, implementation and evaluation of the dialogue interface for our open-domain QA system, YourQA (Quarteroni and Manandhar, 2006). The system is able to provide both factoid and complex answers such as definitions and descriptions. The dialogue interface's role is to enable an information seeking, cooperative, inquiry-oriented conversation to support the question answering component.

Section 2 introduces the design of our interactive QA system; Section 3 describes an exploratory study conducted to confirm our design assumptions. The implementation and evaluation of our prototype are described in Sections 4 and 5. Section 6 briefly concludes on our study.

2 System Design

Our open domain QA system, YourQA, takes the top 20 Google results for a question, retrieves the corresponding Web pages and analyzes them to extract answers and rank them by relevance to the question. A non-interactive interface exists for the system where users enter a question in a text field, and obtain a list of answers in the form of an HTML result page (Quarteroni and Manandhar, 2006).

We now describe the dialogue scenario and management model for the interactive version of the system, where the core QA component is mediated by a dialogue interface.

2.1 Dialogue scenario

In the dialogue scenario we are modelling, a typical QA session consists of the following *dialogue moves*:

1. An initial greeting (*greet* move), or a direct question *q* from the user (*ask(q)* move);

2. *q* is analyzed to detect whether it is related to previous questions;

    (a) If *q* is unrelated to the preceding questions, it is submitted to the QA component;

(b) If *q* is related to the preceding questions (i.e. *q* is a followup question), and is elliptic, i.e. contains no verb (*"Why?"*), the system uses the previous questions to complete *q* with the missing keywords and submits a revised question *q'* to the QA component;

(c) If *q* is a followup question and is anaphoric, i.e. contains references to entities in the previous questions, the system tries to create a revised question *q"* where such references are replaced by their corresponding entities, then checks whether the user actually means *q"* (move *ground(q")*); if the user agrees, query *q"* is issued to the QA component. Otherwise, the system asks the user to reformulate his/her utterance (move *sysReqClarif*) until finding a question which can be submitted to the QA component;

3. As soon as the QA component results are available, an answer *a* is provided (*answer(a)* move);

4. The system enquires whether the user is interested in a *followup* session; if this is the case, the user can enter a query (*ask* move) again. Else, the system acknowledges (*ack*);

5. Whenever the user wants to terminate the interaction, a final greeting is exchanged (*quit* move).

At any time the user can issue a request for clarification (*usrReqClarif(r)*) in case the system's utterance is not understood.

2.2 Dialogue Moves

The dialogue moves with which the interactive QA scenario above is annotated are summarized in Tables 1 and 2. Such moves are a generalization of the dialogue move sets proposed for other information-oriented dialogue models such as GoDiS (Larsson et al., 2000) and Midiki (MITRE Corporation, 2004).

We now discuss the choice of a dialogue management model to implement such moves.

User move	Description
greet	conversation opening
quit	conversation closing
ask(q)	user asks question *q*
ack	acknowledgement of previous utterance, e.g. *"Thanks."*
usrReqClarif(r)	clarification request (*r* = reason)

Table 1: User dialogue moves

System move	Description
greet	conversation opening
quit	conversation closing
answer(a)	answer (*a* = answer)
ack	acknowledgement of previous utterance, e.g. *"Ok."*
sysReqClarif	clarification request
ground(q)	grounding (*q* = question)
followup	proposal to continue session

Table 2: System dialogue moves

2.3 Choosing a dialogue manager

When designing information-seeking dialogue managers, the simplest approaches appear to be finite-state (FS) models. Here, each phase of the conversation is modelled as a separate state, and each dialogue move encodes a transition to a subsequent state (Sutton, 1998).

However, FS models allow very limited freedom in the range of user utterances. Since each dialogue move must be pre-encoded in the models, these are not scalable to open domain dialogue.

A more complex dialogue management model is the Information State (IS) approach inspired by Ginzburg's dialogue gameboard theory (Ginzburg, 1996). The topics under discussion and common ground in the conversation are part of the IS and continually queried and updated by rules fired by participants' dialogue moves. The IS theory, introduced in the TRINDI project (Larsson and Traum, 2000), has been applied to a range of closed-domain dialogue systems (e.g. travel information, route planning).

Although it provides a powerful formalism, the IS infrastructure appears too voluminous for our QA application. We believe that the IS approach is primarily suited to applications requiring a planning component such as in closed-domain dialogue systems and to a lesser extent in our open-domain dialogue system as we currently do not make use of planning. Moreover, in our system the context is only used for question clarification purposes.

2.3.1 The chatbot approach

As a solution joining aspects of both FS and IS approaches, we studied the feasibility of a conversational agent based on an AIML interpreter.

AIML (Artificial Intelligence Markup Language) was designed for the creation of conversational robots ("chatbots") such as ALICE[1]. It is based on pattern matching, which consists in matching the last user utterance against a range of dialogue patterns known to the system ("categories") in order to produce a coherent answer following a range of "template" responses.

Designed for chatting, chatbot dialogue appears more natural than in FS and IS systems. Moreover, since chatbots support a limited notion of context, they seem to offer the means to support followup recognition and other dialogue phenomena not easily covered using standard FS models.

To assess the feasibility of chatbot-based QA dialogue, we conducted an exploratory Wizard-of-Oz experiment, described in Section 3.

3 Wizard-of-Oz experiment

A Wizard-of-Oz (WOz) experiment is usually deployed for natural language systems to obtain initial data when a full-fledged prototype is not yet available (Dahlbaeck et al., 1993). The experiment consists in "hiding" a human operator (the "wizard") behind a computer interface to simulate conversation with the user, who believes to be interacting with a fully automated prototype.

3.1 Assumptions

In addition to the general assumption that a chatbot would be sufficient to successfully conduct a QA conversation, we intended to explore whether a number of further assumptions were founded in the course of our experiment.

First, users would use the system to obtain information, thus most of their utterances would be questions or information requests.

Then, users would easily cope with the system's requests to rephrase their previous utterances should the system fail to understand them.

Finally, the user's clarification requests would be few: as a matter of fact, our answer format provides more information than explicitly required and this has been shown to be an effective way to reduce the occurrence of clarification requests (Kato et al., 2006; Hickl and Harabagiu, 2006).

3.2 Task Design

We designed six tasks, to be proposed in groups of two to six or more subject so that each task was performed by at least two different users. These reflected the intended typical usage of the system (e.g. *"Find out who painted Guernica and ask the system for more information about the artist"*).

Users were invited to test the supposedly completed prototype by interacting with an instant messaging platform, which they were told to be the system interface.

Since our hypothesis was that a conversational agent is sufficient to handle question answering, a set of AIML categories was created to represent the range of utterances and conversational situations handled by a chatbot.

The role of the wizard was to choose the appropriate category and utterance within the available set, and type it into the chat interface to address the user. If none of these appeared appropriate to handle the situation at hand, the wizard would create one to keep the conversation alive and preserve the illusion of interacting with a machine. The wizard asked if the user had any follow-up questions after each answer (*"Can I help you further?"*).

3.3 User Feedback Collection

To collect user feedback, we used two sources:

- the *chat logs*, which provided information about the situations that fell above the assumed requirements of the chat bot interface, the frequency of requests for repetition, etc.;
- a *questionnaire* submitted to the user immediately after the WOz experiment, enquiring about the user's experience.

The questionnaire, inspired by the WOz experiment in (Munteanu and Boldea, 2000) consists of six questions:

Q_1 *Did you get all the information you wanted using the system?*

Q_2 *Do you think the system understood what you asked?*

Q_3 *How easy was it to obtain the information you wanted?*

Q_4 *Was it difficult to reformulate your questions when you were invited to?*

[1] http://www.alicebot.org/

Q_5 *Do you think you would use this system again?*
Q_6 *Overall, are you satisfied with the system?*

Questions Q_1 and Q_2 assess the performance of the system and were ranked on a scale from 1= "Not at all" to 5="Yes, Absolutely". Questions Q_3 and Q_4 focus on interaction difficulties, especially relating to the system's requests to reformulate the user's question. Questions Q_5 and Q_6 relate to the overall satisfaction of the user. The questionnaire also contained a text area for optional comments.

3.4 WOz experiment results

The WOz experiment was run over one week and involved one wizard and seven users. These came from different backgrounds and native languages, were of different ages and were regular users of search engines. All had used chat interfaces before and had an account on the platform used for the experiment, however only one of them doubted that they were confronted to a real system. The average dialogue duration was 11 minutes, with a maximum of 15 minutes (2 cases) and a minimum of 5 minutes (1 case).

From the chat logs, we observed that as predicted all dialogues were information seeking. One unexpected result was that users often asked two things at the same time (e.g. *"Who was Jane Austen and when was she born?"*). To account for this case, we decided to handle double questions in the final prototype, as described in Section 4.

The *sysReqClarif* dialogue move proved very useful, and sentences such as *"Can you please reformulate your question?"* or *"In other words, what are you looking for?"* were widely used. Users seemed to enjoy "testing" the system and accepted the invitation to produce a followup question (*"Can I help you further?"*) around 50% of the time.

Our main observation from the user comments was that users seemed to receive system grounding and clarification requests well, e.g. *" ... on references to "him/it", pretty natural clarifying questions were asked."*

The values obtained for the user satisfaction questionnaire, reported in Table 3, show that users tended to be particularly satisfied with the system's performances and none of them had difficulties in reformulating their questions (Q_4) when this was requested (mean 3.8, standard deviation .5, where 3 = "Neutral" and 4 = "Easy"). For the remaining questions, satisfaction levels were high, between 4±.8 (Q_3) and 4.5±.5 (Q_5).

Question	judgment	Question	judgment
Q_1	4.3±.5	Q_2	4.0
Q_3	4.0±.8	Q_4	3.8±.5
Q_5	4.1±.6	Q_6	4.5±.5

Table 3: Wizard-of-Oz questionnaire results: mean ± standard deviation

4 System Architecture

The dialogue manager and interface were implemented based on the scenario in Section 2 and the outcome of the WOz experiment.

4.1 Dialogue Manager

Chatbot dialogue follows a pattern-matching approach, and is therefore not constrained by a notion of "state". When a user utterance is issued, the chatbot's strategy is to look for a pattern matching it and fire the corresponding template response.

Our main focus of attention in terms of dialogue manager design was therefore directed to the dialogue moves invoking external modules such as the followup recognition and QA component.

We started from the premise that it is vital in handling QA dialogue to apply an effective algorithm for the recognition of followup requests, as underlined in (De Boni and Manandhar, 2005; Yang et al., 2006). Hence, once a user utterance is recognized as a question by the system, it attempts to clarify it by testing whether it is a double question or a followup question.

4.1.1 Handling double questions

For the detection of *double* questions, the system uses the OpenNLP chunker[2] to look for the presence of "and" which does not occur within a noun phrase. If it is found, the system simply offers to answer one of the two "halves" of the double question (the one containing more tokens) as the QA component is not able to handle multiple questions.

4.1.2 Handling followup questions

The types of followup questions which the system is able to handle are elliptic questions, questions containing third person pronoun/possessive adjective anaphora, or questions containing noun

[2] `http://opennlp.sourceforge.net/`

phrase (NP) anaphora (e.g. "the river" instead of "the word's longest river").

Detection of followup questions For the detection of followup questions, the algorithm in (De Boni and Manandhar, 2005) is used. This is based on the following features: presence of pronouns, absence of verbs, word repetitions and similarity between the current and the n preceding questions. The algorithm is reported below:

```
Followup_question ($q_i$, $q_i..q_{i-n}$)
is true if

  1. $q_i$ has pronoun and possessive
     adjective references to $q_i..q_{i-n}$

  2. $q_i$ contains no verbs

  3. $q_i$ has repetition of common or
     proper nouns in $q_i..q_{i-n}$
     or
     $q_i$ has a strong semantic
     similarity to some $q_j \in q_i..q_{i-n}$
```

Following the authors, we apply the above algorithm using $n = 8$; at the moment the condition on semantic distance is not included for the sake of processing speed.

Resolution of followup questions If a question *q* is identified as a followup question, it is submitted to the QA component; otherwise the following reference resolution strategy is applied:

1. if *q* is *elliptic* (i.e. contains no verbs), its keywords are completed with the keywords extracted by the QA component from the previous question for which there exists an answer. The completed query is submitted to the QA component.

2. if *q* is *anaphoric*:

   (a) in case of *pronoun/adjective anaphora*, the chunker is used to find the first compatible antecedent in the previous questions in order of recency. The latter must be a NP compatible in number with the referent.

   (b) in case of *NP anaphora*, the first NP containing all of the referent words is used to replace the referent in the query.

   In both cases, when no antecedent can be found, a clarification request is issued by the system until a resolved query can be obtained and submitted to the QA component.

Ellipsis and reference resolution is useful not only for question interpretation but also to optimize the retrieval phase: it suggests to extract answers from the same documents collected to answer the antecedent question (De Boni and Manandhar, 2005). Hence, if a clarified followup question is submitted to the QA component, the QA system extracts answers from the documents retrieved for the previous question.

When the QA process is terminated, a message directing the user to the HTML answer page is returned and the followup proposal is issued. We must point out that such solution implies that the clarification and followup abilities of YourQA are limited to the questions. Indeed, it is not possible to handle *answer* clarification at the moment: it would be impossible for the system to conduct a conversation such as:

User$_n$: *Who was Shakespeare married to?*
System$_n$: *Anne Hathaway.*
User$_{n+1}$: *what was her profession?*

Data-driven answer clarification in the open domain is an open issue which we would like to study in the future, in order to make the dialogue component more tied into the structure of the QA system.

4.2 Implementation

Following the typical implementation of a pattern-matching conversational agent, we designed a set of patterns to cover the dialogue scenarios elaborated in the design stage and enriched with the WOz experiment.

4.2.1 AIML interpreter and context

First, we grouped the AIML categories in different .aiml files, each corresponding to one of the dialogue moves in Table 2.

We used the Java-based AIML interpreter Chatterbean[3], which allows to define custom AIML tags and allows a seamless integration between the QA module and the chat interface.

We augmented the Chatterbean tag set with two AIML tags:

- `<query>`, to invoke the YourQA question answering module;
- `<clarify>`, to support the tasks of clarification detection and reference resolution.

The Chatterbean implementation of the conversation context (in a dedicated `Context` class) al-

[3] `http://chatterbean.bitoflife.cjb.net/`

lows to instantiate and update a set of variables, represented as context *properties*. We defined several of these, including:

- the user's *name*, matched against a list of known user names to select a profile for personalized answer extraction (this feature of YourQA is not discussed here);

- the current *query*, used to dynamically update the stack of recent user questions. The stack is used by the clarification request detection module to perform reference resolution, following the algorithm exposed in Section 4.1;

- the *resolved* question, i.e. the current query as resolved during followup handling (identical to the current query if no followup is detected);

- the *topic* of conversation, i.e. the set of keywords of the last question issued by the user which received an answer (e.g. "What is the world's longest river?" => {`world, longest, river`}). The topic keywords are used to clarify elliptic questions, by augmenting the current query keywords (e.g. "In what country?"=> {`country, world, longest, river`}).

To illustrate a typical case of use of the implementation we start with the AIML category:

```
<category>
<pattern>DO YOU KNOW *</pattern>
<template><srai>CLARIFY *</srai>
</template>
</category>
```

Here, the system attempts to clarify the content of the request following the cue words "Do you know". The template:

```
<template><srai>CLARIFY *</srai>
</template>
```

invokes a category calling the newly introduced AIML tag `<clarify>`:

```
<category>
<pattern>CLARIFY *</pattern>
<template>
<think><set name="clarif">
<clarify></star><clarify>
</set></think>
```

The `<clarify>` tag in turn invokes the Java implementation dedicated to clarification recognition on the text matching the `"*"` expression, which follows the strategy exposed in Section 4.1. Finally, the clarified query (assigned to the context property `"clarif"`) as well as the output of the followup detection algorithm (e.g. `"ELLIPTIC"`) are used in a conditional branch to call the appropriate routine:

```
<condition name="clarif"
value="ELLIPTIC"> ...</condition>
<condition name="clarif"
value="DOUBLE"> ...</condition>
...
</template>
</category>
```

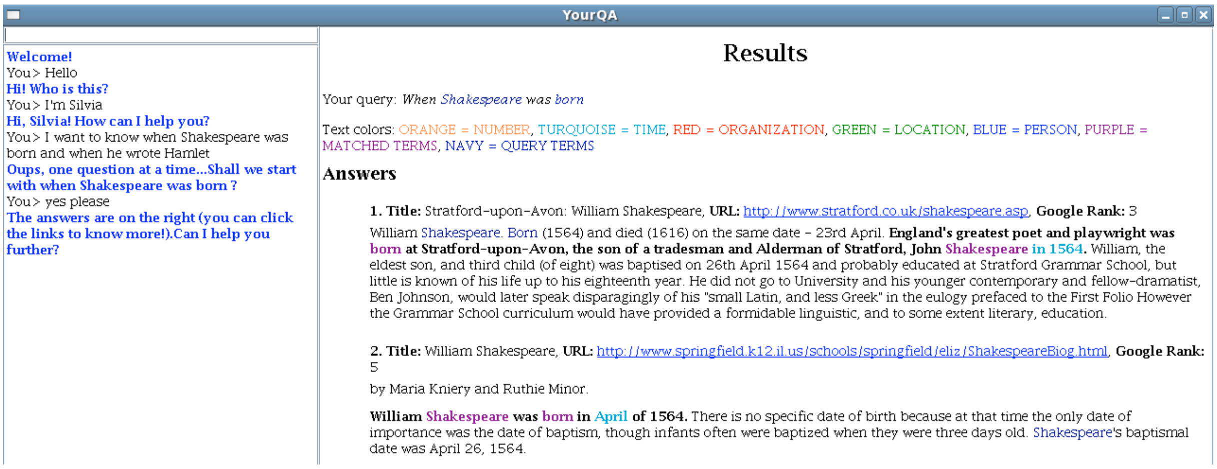
4.2.2 Dialogue Interface

YourQA's interactive interface consists of an applet window with a left panel where the chat takes place and a right panel where results are visualized when ready (see Figure 1). As in a normal chat application, users enter text in a text field and the current session history as well as the system replies are visualized in an adjacent text area.

An example dialogue deriving from a real interaction with the system is the following:
System$_0$*: Welcome!*
User$_1$*: Hi*
S$_1$*: Hi! How can I help you?*
U$_2$*: what is a thermometer?*
S$_2$*: Possible answers are in the panel on the right. Are you happy with these answers?*
U$_3$*: yes!*
S$_3$*: Good!*
U$_4$*: How does it measure the temperature?*
S$_4$*: Do you mean how does a thermometer measure the temperature?*
U$_5$*: yes*
S$_5$*: I found the answers in the panel on the right. Can I help you further?*
U$_6$*: ...*

5 Evaluation

While the accuracy of standard QA systems can be evaluated and compared using quantitative information retrieval metrics (Voorhees, 2003), dialogue interfaces pose complex evaluation challenges as they differ in appearance, intended application and target users.



Welcome!
You> Hello
Hi! Who is this?
You> I'm Silvia
Hi, Silvia! How can I help you?
You> I want to know when Shakespeare was born and when he wrote Hamlet
Oups, one question at a time...Shall we start with when Shakespeare was born ?
You> yes please
The answers are on the right (you can click the links to know more!).Can I help you further?

Results

Your query: *When Shakespeare was born*

Text colors: ORANGE = NUMBER, TURQUOISE = TIME, RED = ORGANIZATION, GREEN = LOCATION, BLUE = PERSON, PURPLE = MATCHED TERMS, NAVY = QUERY TERMS

Answers

1. Title: Stratford-upon-Avon: William Shakespeare, **URL:** http://www.stratford.co.uk/shakespeare.asp, **Google Rank:** 3

William Shakespeare. Born (1564) and died (1616) on the same date - 23rd April. **England's greatest poet and playwright was born at Stratford-upon-Avon, the son of a tradesman and Alderman of Stratford, John Shakespeare in 1564.** William, the eldest son, and third child (of eight) was baptised on 26th April 1564 and probably educated at Stratford Grammar School, but little is known of his life up to his eighteenth year. He did not go to University and his younger contemporary and fellow-dramatist, Ben Johnson, would later speak disparagingly of his "small Latin, and less Greek" in the eulogy prefaced to the First Folio However the Grammar School curriculum would have provided a formidable linguistic, and to some extent literary, education.

2. Title: William Shakespeare, **URL:** http://www.springfield.k12.il.us/schools/springfield/eliz/ShakespeareBiog.html, **Google Rank:** 5

by Maria Kniery and Ruthie Minor.

William Shakespeare was born in April of 1564. There is no specific date of birth because at that time the only date of importance was the date of baptism, though infants often were baptized when they were three days old. Shakespeare's baptismal date was April 26, 1564.

Figure 1: The chat interface (partial view)

Indeed, these are often evaluated using qualitative metrics such as user satisfaction and perceived time of usage (Walker et al., 2000). Similarly, user satisfaction questionnaires and interaction logs appear to be effective tools to evaluate interactive QA systems (Kelly et al., 2006).

5.1 Experiment design

To quickly conduct a preliminary evaluation of our prototype, we designed three scenarii where users had to look for two different items of infomation relating to the same topic (e.g. Shakespeare's date of birth and when he wrote Hamlet), as in the previous WOz experiment. Users had to choose one or more topics and use first the non-interactive Web interface of the QA prototype (handling questions in a similar way to a search engine) and then the interactive version depicted in Figure 1 to find answers.

After using both versions of the prototype, users filled in a questionnaire about their experience with the *chat version* which comprised the same questions as the WOz questionnaire and the following additional questions:

Q_7 *Was the pace of interaction with the system appropriate?*

Q_8 *How often was the system sluggish in replying to you?*

Q_9 *Did you prefer the chat or the Web interface and why?*

Questions Q_7 and Q_8 could be answered using a scale from 1 to 5 and were taken from the PARADISE evaluation questions (Walker et al., 2000). Q_9 was particularly interesting to assess if and in what terms users perceived a difference between the two prototypes. All the interactions were logged.

5.2 Evaluation results

From the initial evaluation, which involved six volunteers, we gathered the following salient results:

1. in the chat logs, when pronominal anaphora was used by the users, the system was able to resolve it in seven out of nine cases;
2. no elliptic queries were issued, although in two cases verbs were not spotted by the system causing queries to be completed with previous query keywords;
3. due to the limited amount of AIML categories of the system, the latter's requests for reformulation occurred more frequently than expected;
4. Users tended not to reply to the chatbot offers to carry on the interaction explicitly, directly entering a followup question instead.

From the questionnaire (Tab. 4), we collected sightly lower user satisfaction levels (Q_1 to Q_6) than in the WOz experiment (Section 3).
Users felt the system to reply slowly to the questions (Q_7 and Q_8). This is mainly because the system performs document retrieval in real time, hence it heavily depends on the network download speed.
All but one user (i.e. 83.3%) said they preferred the chat interface of the system (Q_9), because of its liveliness and ability to understand pronominal anaphora.

Question	judgment	Question	judgment
Q_1	3.8±.4	Q_2	3.7±.8
Q_3	3.8±.8	Q_4	3.8±.8
Q_5	4.0±.9	Q_6	4.3±.5
Q_7	3.5±.5	Q_8	2.3±1.2

Table 4: Questionnaire results: mean±standard deviation

6 Conclusions

This paper reports the design and implementation of a chatbot-based interface for an open domain, interactive question answering (QA) system. From our preliminary evaluation, we draw optimistic conclusions on the feasibility of chatbot-based interactive QA.

In the future, we will study more advanced strategies for anaphora resolution in questions, e.g. (Poesio et al., 2001) and conduct a more thorough evaluation of our dialogue interface.

As mentioned earlier, we are also interested in data-driven answer clarification approaches for the open domain to further integrate the dialogue component into the QA system.

Acknowledgement We are indebted to the anonymous reviewers for their valuable comments on the initial version of this paper.

References

N. Dahlbaeck, A. Jonsson, and L. Ahrenberg. 1993. Wizard of Oz studies: why and how. In *Proceedings of IUI '93*, pages 193–200, New York, NY, USA. ACM Press.

M. De Boni and S. Manandhar. 2005. Implementing clarification dialogue in open-domain question answering. *Nat. Lang. Eng.*, 11.

J. Ginzburg, 1996. *Interrogatives: Questions, Facts and Dialogue*. Blackwell, Oxford.

A. Hickl and S. Harabagiu. 2006. Enhanced interactive question answering with conditional random fields. In *Proceedings of IQA*.

J. R. Hobbs. 2002. From question-answering to information-seeking dialogs.

A. Jönsson and M. Merkel. 2003. Some issues in dialogue-based question-answering. In *Working Notes from AAAI Spring Symposium*, Stanford.

T. Kato, J. Fukumoto, F.Masui, and N. Kando. 2006. Woz simulation of interactive question answering. In *Proceedings of IQA*.

D. Kelly, P. Kantor, E. Morse, J. Scholtz, and Y. Sun. 2006. User-centered evaluation of interactive question answering systems. In *Proceedings of IQA*.

S. Larsson and D. R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Nat. Lang. Eng.*, 6(3-4):323–340.

S. Larsson, P. Ljunglöf, R. Cooper, E. Engdahl, and S. Ericsson. 2000. GoDiS—an accommodating dialogue system. In C. Sidner, editor, *ANLP/NAACL Workshop on Conversational Systems*, pages 7–10, Somerset, New Jersey. ACL.

MITRE Corporation, 2004. *MIDIKI User's manual.*

C. Munteanu and M. Boldea. 2000. MDWOZ: A Wizard of Oz Environment for Dialog Systems Development. In *Proceedings of LREC*.

M. Poesio, U. Reyle, and R. Stevenson, 2001. *Justified sloppiness in anaphoric reference – Computing meaning*, chapter 3. Kluwer.

S. Quarteroni and S. Manandhar. 2006. User modelling for adaptive question answering and Information Retrieval. In *Proceedings of FLAIRS*.

S. Small, T. Liu, N. Shimizu, and T. Strzalkowski. 2003. HITIQA: an interactive question answering system- a preliminary report. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and QA*, pages 46–53, Morristown, NJ, USA. ACL.

S. Sutton. 1998. Universal speech tools: the CSLU toolkit. In *Proceedings of the International Conference on Spoken Language Processing*.

E. M. Voorhees. 2003. Overview of the TREC 2003 question answering track. In *Text REtrieval Conference*.

M. A. Walker, C. Kamm, and D. Litman. 2000. Towards Developing General Models of Usability with PARADISE. *Nat. Lang. Eng. Special Issue on Best Practice in Spoken Dialogue Systems*.

F. Yang, Z. Feng, and G. Di Fabbrizio. 2006. A data driven approach to relevancy recognition for contextual question answering. In *Proceedings of IQA*.