

Addressing How-to Questions using a Spoken Dialogue System: a Viable Approach?

Silvia Quarteroni
University of Trento
38050 Povo (Trento), Italy
silviaq@disi.unitn.it

Patrick Saint-Dizier
IRIT
Toulouse, France
stdizier@irit.fr

Abstract

In this document, we illustrate how complex questions such as procedural (how-to) ones can be addressed in an interactive format by means of a spoken dialogue system. The advantages of interactivity and in particular of spoken dialogue with respect to standard Question Answering settings are numerous. First, addressing user needs that do not necessarily arise in front of a computer; moreover, a spoken or multimodal answer format can often be better suited to the user's need. Finally, the procedural nature of the information itself makes iterative question formulation and answer production particularly appealing.

1 Introduction

Question answering (QA) is nowadays an established technology, advancing information retrieval to the point of allowing queries to be formulated in natural language and to return actual answers (in the form of sentences/phrases).

While the first QA systems (Simmons, 1965) mainly dealt with factoid questions, i.e. questions about names, dates and all that can be reduced to a fact, a number of systems in the last decade have appeared with the aim of addressing non-factoid questions (Voorhees, 2003). In particular, the problem of addressing definition questions has received great attention from the research community (Chen et al., 2006; Moschitti et al., 2007), while less research has been conducted so far on other types of non-factoid QA, such as *why*-questions (Verberne et al., 2007; Pechsiri et al., 2008) and procedural (also called *how-to*) questions (Yin, 2006; Delpech and Saint-Dizier, 2008).

Another recent trend in QA is interactivity, i.e. the use of a dialogue interface to better support the user, e.g. by resolving anaphoric and elliptic

expressions in his/her queries (Webb and Strzalkowski, 2006). Indeed, the dialogue community has been addressing the problem of information seeking for decades, often with very satisfying commercial products able to interact not only in text but especially via spoken interfaces (Gupta et al., 2006; Traum, 1996). However, also in this field the information retrieval task has mainly focused on a limited domain (travel planning, telecom rates) and on returning database values rather than cooperatively solving problems or providing complex information.

In this paper, we focus on handling procedural questions, not as commonly researched as definitional QA but for which a number of resources are available on the Web. Indeed, although portals dedicated to how-to questions exist (eHow.com), where stereotyped questions are presented together with a few responses, QA would allow a broader approach to intelligently respond to how-to questions.

Our main claim is that joining the existing QA technology for complex procedural questions with the potentials of spoken conversation would provide an excellent testbed for the integration of these two technologies. Indeed, understanding and answering procedural questions requires a high level of cooperation between the user and the system: a procedure is a complex answer to return and would better be provided and received step by step than “dumped” in a text-to-speech generator or a text file.

In the rest of this document, we outline the main features of procedural QA and the approach we propose to address it via dialogue. We illustrate the potentials of our approach with two use cases of different complexity.

2 Procedural Question Answering

Procedural text contains not only step-by-step instructions, but also additional content such as

warnings, recommendations and advice. Due to the argumentative nature of such text, procedural QA is a complex task. Indeed, the main challenges offered by procedural QA can be summarized as:

1. Acquiring procedural data:
 - (automatically) obtaining the data, filtering out text with little procedural content;
 - tagging relevant structures in procedures (such as warnings, advice, step-wise instructions);
 - efficiently indexing texts based on their title and content;
2. Answering procedural questions:
 - recognizing and interpreting procedural questions (question classification and analysis);
 - pinpointing answer passages (answer retrieval);
 - generating answers to procedural questions and supporting interaction spanning over more than one Q/A pair, such as step-by-step procedural descriptions.

To our knowledge, little extensive work exists in this field; an example is the TextCoop project (Delpech and Saint-Dizier, 2008) that produced a procedural tagger able to recognize and segment the main units found in French procedural text (titles, instructions, prerequisites, warnings and advice) via an *ad hoc* markup convention (see Table 1). In addition, QA technology was used for the resolution of elliptic titles and their indexing for answer matching (titles often express goals).

Although automatic procedure tagging and analysis appears as a necessary step towards an efficient treatment of procedural questions, we argue that an accurate choice of the format and modality in which their answers are returned would be a vital advantage. In particular, we propose to return the response to a procedural QA under the form of oral instructions rather than text to read. Indeed, besides the advantages of oral communication in terms of expressiveness, the latter solution may be inappropriate in some situations such as when walking around or driving.

In Section 3, we discuss our dialogue-based approach to procedural QA.

3 Dialogue-based Procedural QA

We believe that the integration of QA research with a Spoken Dialogue System (SDS) is a promising approach to procedural Question Answering. Indeed, work on procedural QA so far accounts for the textual structures of written documents; since procedural texts are in general highly interactive, it is clear that the pairing with a spoken dialogue system is of much interest. In addition, a spoken interface enables to go far beyond a mere enumeration of instructions (as found in Web pages), achieving cooperation between the service provider (SDS) and the user.

A first step towards this is the (automatic or semi-automatic) annotation of procedural texts via an *ad hoc* markup in order to distinguish subtexts that can yield to dialogues, such as conditions (texts containing “if you are under 20 . . .”, “if you are aged between . . .” may be translated in the question: “how old are you?”). Similarly, in warnings, terms bearing the illocutionary force (“Remember”, “Caution”, “Notice”) can be marked in order to be stressed.

During system execution, instructions can be uttered one after the other, waiting for an acknowledgement from the user, but the system can also provide more information about the task at hand upon request. Moreover, the system can provide alternative solutions when users have trouble carrying out an instruction (“I cannot pay by credit card”), or make an instruction more explicit by splitting it into simpler ones (automatically generating another how-to question for the subgoal at hand).

Finally, in addition to speech and dialogue, multimodal aspects of interactivity can be considered, such as displaying a map when providing an itinerary, or a 3D picture related to an instruction.

Translating a procedural text into speech is a challenge that requires intensive NLP processing, a strong and accurate domain model and an ability for reasoning. In order to address this challenge, we propose the following approach:

1. Obtain domain-related data from the Web;
2. Represent domain knowledge and reasoning. While most of the factual knowledge of a domain can be captured by means of an enriched ontology, other types of knowledge (know-how, domain constraints, etc.) and

reasoning procedures need to be defined on other grounds, optionally manually;

3. Devise a Dialogue Manager able to interact about procedural information using markup in the procedural data representation;
4. Define how the procedural data representation can be rendered by a Natural Language Generator;
5. Use existing technology for Automatic Speech Recognition and Text-To-Speech.

Evidently, the difficulty of answering procedural questions via dialogue varies depending on the availability and format of answers. We distinguish between two types of questions:

Type 1: a procedural text corresponding to the question is already available on the Web; in this case, the user’s query can be answered by tagging such text using a tagger such as TextCoop and enriching it with dialogic and prosodic markers to be rendered by an off-the-shelf TTS module;

Type 2: there is no direct answer to the user’s query on the Web; for instance, the answer may be dependent on information which the user has not yet provided. In this case, the query must first be formulated, and procedural tagging/TTS intervene later.

In Sections 4 and 5, we report two case studies reflecting type 1 and type 2 situations, respectively: the first relates to the University helpdesk domain, the second to the tourist advice domain.

4 Text-to-Speech from a Web page

To illustrate type 1 questions, we study a well-known domain, universities, where helpdesks must provide various kinds of procedural information (dealing with e.g. paperwork, student life and infrastructure). Let us consider the question: “How to get a student card in Wolverhampton?”. In Fig. 1, we report an extract of the top Web page obtained by typing such question into a search engine. It can be noted that in this case, the top search engine result contains the procedural answer sought by the question, hence procedural tagging can be performed on the text.

A possible procedural annotation has been (manually) applied to the same text in Figure 2,

Get your student ID card

Once you have enrolled, collect your student ID card from your home campus.

- ▶ New UK and international students should follow the [new students instructions](#).
- ▶ Returning UK and international students should follow the [returning student ID card instructions](#).

APPLYING FOR AN ID CARD

When you receive a firm offer from the University, you can upload your photo for your student ID card, and can.

WHAT YOU'LL NEED

- ▶ Your student number, a seven-digit number which will be on your offer letter
- ▶ A digital photo that meets the requirements outlined below
- ▶ Access to a computer with Internet access to send your photo to the University using the [Photo Upload](#)

If you don't have a digital photo or a computer with internet access, you can go to one of the webcam station Centres over the summer, or when you arrive, and we'll help you to do it.

REQUIREMENTS FOR STUDENT ID CARD PHOTOS

You can upload your photo from any computer, phone or mobile device with Internet access.

We need a good quality passport style photo which must meet the following criteria:

1. The photo should be of you alone
2. Colour, clear with good contrast

Figure 1: Extract from the top Web hit for: “How to get a student card in Wolverhampton?” (source: [wlv.ac.uk](#))

following the conventions used in the TexCoop tagger (see Tab. 1) to denote the abilities of a procedural tagger. While some of the HTML objects in the Webpage, such as title, headers and enumerations, are directly converted in their equivalent tags (`item`, `subtitle`), additional markup appears, such as warnings and prerequisites.

Table 1: TextCoop procedural markup (extract)

Label	Example
<code>title</code>	“Get your student ID card”
<code>subtitle</code>	“What you’ll need”
<code>cond</code>	“if you are a UK student”
<code>objective</code>	“in order to get your ID”
<code>instr</code>	“Head to the Uni info service.”
<code>prerequisite</code>	“You’ll need 3 passport photos”
<code>warning</code>	“Format MUST be passport!”
<code>aim</code>	“to get good photos”
<code>advice</code>	“try the photobooth next to ...”

At this point, using a dialogue system to simply “read out” the above passage (even if split into their main components) would result in ineffective, close-to intonation free speech. Indeed, in order to provide instructions to the Natural Language Generator and Text-to-Speech modules of a dialogue system for verbalizing such text, dialogue-level markup must be added to the above procedural annotation.

In some cases, direct mapping rules can be devised to directly translate procedural markup into dialogue patterns. For instance, step-by-step instructions (`item`) contained in the `itemize` environment can be rendered as a sequence of *inform*

```

<subtitle> Applying for an ID card </subtitle>
<inst-compound>
When you receive a firm offer from the University, you can upload your photo
for your student ID card, <warning> and you should do this as soon as
you can. </warning>
</inst-compound>

<prerequisite> What you'll need
<itemize>
<item:1> Your student number, a seven-digit number which will be on your
offer letter </item:1>
<item:2> A digital photo that meets the requirements outlined below
</item:2>
<item:3> Access to a computer with Internet access to send your
photo to the University using the Photo Upload facility. </item:3>
</itemize></prerequisite>

<inst-compound> <cond> If you don't have a digital photo or a com-
puter with internet access, </cond> ...

```

Figure 2: Procedural annotation of a Web page

dialogue acts, expecting acknowledgments (*ack*) from the user. In addition, conditions can be rendered as yes-no questions (“If you don’t have a digital photo” becomes *ask*(digital photo));

In other cases, such as verbalizing warnings and advice, specific salient words should be marked with prosodic information as to how to pronounce them. Specific lexical patterns can be matched by rules to provide such annotations, such as “Remember” or “as soon as possible”. Finally, part of the procedural annotation could be excluded from the dialog when redundant or implicit. For instance, titles (*title*) could be skipped or mentioned separately by the dialogue system (e.g. “Would you like to hear about how to *Get your student ID card?*”).

Figure 3 illustrates the dialog act and prosodic annotation enriching the procedural one of Figure 2. Such generic markup can then be converted in a specific commercial voice markup languages, e.g. VXML or SALT, via simple rules.

5 Integrating Scenarios and QA

Besides improving access to procedures via direct interactions by spoken dialogue, it is often necessary to interact with the user to get more precise information about his query, so that the response can be accurate enough. Furthermore, a number of procedural questions do not get any direct response via Web queries. This is the case of type 2 questions, as introduced in Section 3. There are several reasons to this situation. First, a number of these questions are both complex and very specific. Next, most of them involve various forms of reasoning and of elaboration. Other questions require the integration of several simpler procedures, e.g. via concatenation. Finally, others re-

```

<subtitle> Applying for an ID card </subtitle>
<inst-compound> When you receive a firm offer from the University,
you can upload your photo for your student ID card, <warning> and
you should do this </prosody:emphasize> as soon as you can.
</prosody:emphasize> </warning> </inst-compound>

<prerequisite> What you'll need
<itemize>
<item:1> <dialog:inform-ack> Your student number, a seven-
digit number which will be on your offer letter </dialog:inform-ack>
</item:1>
<item:2> <dialog:inform-ack> A digital photo that meets the re-
quirements outlined below </dialog:inform-ack> </item:2>
<item:3> <dialog:inform-ack> Access to a computer with Inter-
net access to send your photo to the University using the Photo Upload facility.
</dialog:inform-ack> </item:3>
</itemize> </prerequisite>

<inst-compound> <dialog:ask> <cond> If you don't have a dig-
ital photo or a computer with internet access, </cond> </dialog:ask>
...

```

Figure 3: Dialog act and prosodic annotation of a Web page

quire a substantial adaptation of existing procedures: adaptation to a different context, generalizations (e.g. knowing how to register in a university may lead to a generalization so that it is globally acceptable for other universities).

This is in particular the case for non-trivial itineraries. For example, looking on the Web for ways to go from Toulouse to Trento does not lead to any solution. Search engines return partial and often local information, e.g. description of Verona airport, train schedules going via Trento, etc. We need in this case to define a very generic scenario, which is a procedure, of type ‘travel’ and, for a given trip, to construct the details from simpler procedures or factual data available on the Web.

To overcome these limitations and to be able to offer a real QA service, we propose the following approach:

- Creating a general scenario, in our case for itinerary construction, involving dialogue to get necessary (departure/arrival location and dates, etc.) and optional (budget, comfort, etc.) information from the user.
- Including reasoning procedures and preferences related to transportation: e.g. it is preferable to fly above a certain distance or if there are obstacles (sea, mountains), or elaborate compromise between cost and transportation length. Itinerary construction also involves a planner, that operates over any kind of transportation means, paired with an optimizer. The planner should be flexible so that it can propose alternatives (e.g. train or

renting a car, stops at different places) while the optimizer should take user preferences into account.

- Submitting queries to a search engine to get detailed information on precise points: flight schedules, airport transportation, bus routes, etc. Such queries are triggered by the different functions of the scenario to fill in information slots. From search engine results, it is necessary to process the text segments so that the correct information is found. This includes either getting precise data or selecting a text portion (e.g. that describes services, schedules, etc.).
- Summarizing the information and generating a response in natural language under the form of a procedure, possibly with schemas, maps, etc. and then producing a vocal output. As shown above, parts of this scenario may be vocal or multimedia. As in most natural language generation systems, this involves a planner that operates of various types of input data (text, words, structured sequences of the scenario) and a language generation component which, in this type of application, can be based on predefined word sequences and gaps to be filled in for the query at stake.

This approach has its roots in the frames and scripts of cognitive science and AI in the 70s (Schank and Abelson, 1977). However, in our case we include a QA component to get information and a planner to construct the itinerary based on the results of the queries which also outputs a procedure in natural language. In addition, the proposed approach supports cooperative dialogues and provides explanations to the user when there is no direct answer to his request.

6 Perspectives

We have proposed a model of procedural QA system conducting cooperative spoken dialogue with the user. Indeed, we argue that the advantages of spoken communication channel to address procedural QA are mainly twofold. On the one hand, procedural information can be returned to the user in a more efficient way compared to the textual format. On the other hand, cooperative dialogue allows the system to understand and refine the user's information needs and to account for the

cases when information is not directly available on the Web.

Our proposed approach has currently only been validated through case studies and a long process is required in order to achieve spoken procedural QA. However, we believe that using existing resources to address procedural information, such as procedural taggers, as well as state-of-the art QA and spoken dialogue technology, fulfilling our objectives is a feasible task.

References

- Y. Chen, M. Zhou, and S. Wang. 2006. Reranking answers from definitional QA using language models. In *Proc. ACL*.
- E. Delpech and P. Saint-Dizier. 2008. Investigating the structure of procedural texts for answering how-to questions. In *Proc. LREC*.
- N. Gupta, G. Tur, D. Hakkani-tur, G. Riccardi, S. Bangalore, M. Rahim, and M Gilbert. 2006. The AT&T spoken language understanding system. *IEEE transactions on speech and audio*, 14:213–222.
- A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proc. ACL*.
- C. Pechsiri, P. Sroison, and U. Janviriyasopa. 2008. Know-why extraction from textual data. In *Proc. KRAQ*.
- R. C. Schank and R. P. Abelson. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Erlbaum.
- R. F. Simmons. 1965. Answering english questions by computer: a survey. *Comm. ACM*, 8(1):53–70.
- D. Traum. 1996. Dialogue management in conversational agency: The TRAINS-93 dialogue manager. In *Proc. TWLT*, pages 1–11.
- S. Verberne, L. Boves, N. Oostdijk, and P. Copen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proc. SIGIR*, pages 735–737.
- E. M. Voorhees. 2003. Overview of TREC 2003. In *Proc. TREC*.
- N. Webb and T. Strzalkowski, editors. 2006. *Proc. HLT-NAACL Workshop on Interactive Question Answering*.
- L. Yin. 2006. A two-stage approach to retrieving answers for how-to questions. In *Proc. EACL (Student Session)*.