Dialogue Games for Crosslingual Communication

Paul Piwek
NLG group
Computing Department
The Open University
Walton Hall
Milton Keynes, UK
P.Piwek@open.ac.uk

David Hardcastle
NLG group
Computing Department
The Open University
Walton Hall
Milton Keynes, UK
D.W.Hardcastle@open.ac.uk

Richard Power
NLG group
Computing Department
The Open University
Walton Hall
Milton Keynes, UK
R.Power@open.ac.uk

Abstract

We describe a novel approach to *crosslingual dialogue* that supports *highly accurate* communication of *semantically complex* content between people who do not speak the same language. The approach is introduced through an implemented application that covers the same ground as the chapter of a conventional phrase book for food shopping. We position the approach with respect to dialogue systems and Machine Translation-based approaches to crosslingual dialogue. The current work is offered as a first step towards the innovative use of dialogue theories for the enhancement of human–human dialogue.

1 Introduction

> **Original Dutch text:** Daar achter staat een doos met appels. Kan ik daar een een halve kilo van hebben?
>
> **Translation into English by human:** *Back there, there is a box with apples. Can I have half a kilo of those?*
>
> **Translation into English by Altavista Babelfish (April 17, 2007):** *There behind state a box with apples. Am I possible of it a half kilo have?*

The example above illustrates some of the shortcomings of Machine Translation (MT). Apart from many other errors in the translation, note that Babel Fish incorrectly uses singular 'it' to refer to the plural 'apples'. Babel Fish does not model how sentences both change the context and depend on it for their interpretation; consequently 'apples' does not lead to the introduction of a representation for a (plural) set of apples that can subsequently be referred to. This is a symptom of a more general issue: Much of MT is still grounded in the classical transmission model in which a speaker communicates a message *m* by encoding *m* in a natural language sentence and the hearer subsequently decodes it. MT typically maps sentences from source to target *one at a time*, treating each sentence as separate problem. In this paper, we will put forward an approach to crosslingual dialogue that fits better with contemporary semantic theory, in which meanings of natural language expressions are conceived of as 'programs' that change information states, rather than static representations (of the world or what is in the mind of the speaker).

From a practical point of view, it is worthwhile to compare MT-based crosslingual dialogue systems with spoken dialogue systems. Even for relatively simple domains, such as travel planning, large and extremely large-scale research projects such as the Spoken Language Translator (Rayner et al., 2000) and Verbmobil[1] have, despite making substantial contributions to various areas of speech and language processing, not yet delivered systems for practical deployment. In contrast, spoken dialogue systems are nowadays deployed in many countries for tasks ranging from providing travel information to call routing. The apparent intractability of human–human crosslingual dialogue, as opposed to human–machine dialogue, is partly a result of the fact that whereas in the latter it is straightforward to influence the human dialogue participant's contributions, through system

[1] See http://verbmobil.dfki.de/

initiative, it is less obvious how to do so in human–human dialogue. When a system tracks human–human dialogue, it cannot influence the utterances of the human interlocutors (e.g., by asking questions such as 'From where to where would you like to travel?').

In short, both in theoretical and practical terms, the current state-of-the-art of tools for supporting crosslingual human–human dialogue lags behind other areas of dialogue research. The current work is an attempt to close the gap. We will present an approach to crosslingual dialogue that allows both for better transfer of knowledge from contemporary theories of semantics and dialogue to crosslingual dialogue technology, and has potential for practical applications.

The basic idea is to take the conception of dialogue as a game in which contributors take turns that result in updates on their information states (Traum and Larsson, 2003) quite literally. Although we aim to leverage insights regarding the foundations of human–human dialogue, we will not direcly mimick it in all its details. The aim is to exploit contemporary insights (from speech act theory, theories of common ground in dialogue, conversational sequencing, etc.) to build computational artifacts that *enhance* human–human dialogue.

In the next section, we introduce the underlying technology, Conceptual Authoring, and describe how it can be adapted to facilitate crosslingual dialogue. Details of the underlying system architecture are described in Section 3. Section 4 summarizes the benefits of the proposed approach and compares it with Machine Translation-based approaches. Finally, in Section 5 we provide a number of research goals that we intend to pursue in future, using the current work as a starting point.

2 From Conceptual Authoring to Crosslingual Dialogue

Conceptual Authoring (CA) was pioneered by Power and Scott (1998). A number of CA-applications were developed at the University of Brighton and, subsequently, the Open University[2]. At Harvard, Nickerson (2005) investigated CA for reference specification, and Xerox Research Centre Europe has explored a similar approach, which they call Multilingual Document Authoring

[2]See `http://mcs.open.ac.uk/nlg/research/Conceptual_Authoring.html`

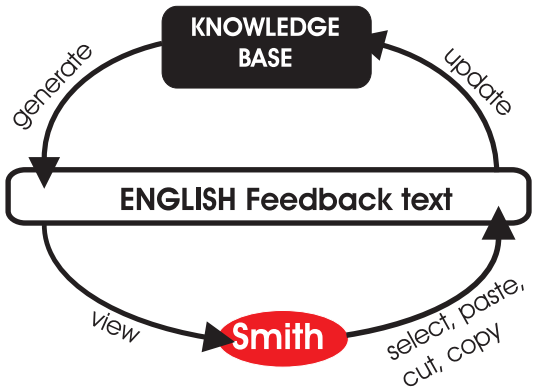

Figure 1: Conceptual Authoring (CA) editing cycle

(Dymetman et al., 2000).

The key principle underpinning CA is presented by the editing cycle in Figure 1: Given a Knowledge Base (KB), the system generates a description of the KB in the form of a feedback text containing anchors (coloured spans of text) representing places where the content in the KB can be extended. In Figure 1, the user is Mr. Smith and he interacts with an English feedback text. Each anchor is associated with pop-up menus, which present the possible extensions of the KB at that point. These are computed by consulting an ontology that underlies the KB. More precisely, the KB consists of two components:

1. an ontology, also known as the terminological box (*T-box*) which specifies the set of available concepts and their attributes, and

2. an assertion box (*A-box*) in which instances of concepts/classes are introduced. It is the A-box that is updated, and the T-box which specifies the set of possible updates.

On the basis of the user's selection, the KB is updated and a new feedback text (reflecting the updated content) is generated. Additionally, spans of feedback text representing an object in the KB can be selected using the mouse to move or remove the object to or from a location in the KB. After each action, a new feedback text is generated representing the updated KB.

The potential of this approach is evidenced by its successful deployment in query formulation (Piwek et al., 2000; Evans et al., 2006; Hallett et al., 2007). For example, Hallett et al. (2007) showed that the method enables untrained users

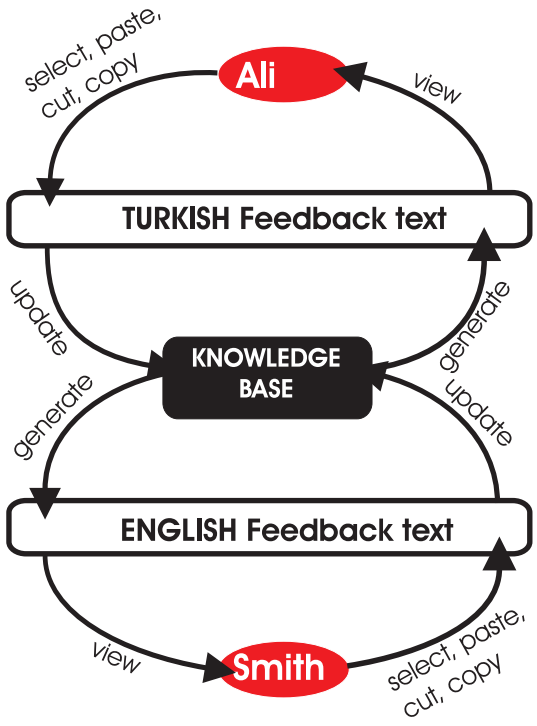

Figure 2: CROCODIAL Multi-person Conceptual Authoring (CA) editing cycle

Figure 3: Screen capture of Conceptual Authoring (CA) Interface for English-speaking Customer. Construction of contribution is in progress in feedback pane.

to successfully and reliably formulate complex queries, while avoiding the standard pitfalls of free text queries.

Here, we discuss an extension – CROCODIAL (for Crosslingual Computer-mediated Dialogue) – of CA to dialogue that was first proposed in Piwek & Power (2006). The extension rests on the idea of taking CA from single-person authoring to multi-person authoring. This is visualized in Figure 2. Here, we have a second editor (Ms. Ali) with access to the same underlying KB as Mr. Smith. Crosslingual dialogue is made possible because although each editor has access to the same KB, their views of it are different: Ali looks at it through 'Turkish glasses' (a language generator for Turkish) and Smith through English ones. Of course such a multi-person editing does not necessarily lead to interactions that qualify as dialogues. To approximate dialogue behaviour we introduce some constraints:

1. The jointly edited structure has to be interpreted as representing the *dialogue history*, progressively built up.

2. Only the most recent turn in the history can be modified, although material can be *copied from preceding turns to establish anaphoric links*.

3. Interlocutors construct turns one at a time.

Figures 3, 4 and 5 are screen captures of our implemented CROCODIAL prototype system. The system supports for conversations between between a shopkeeper and a customer. In our examples, we have a Turkish-speaking shopkeeper and an English-speaking customer.

CROCODIAL allows both for use of the system similar to chatroom internet applications, and on a single portable device (see section 3). For this particular scenario, the scenario is running on a single portable device (e.g., PDA or Tablet PC). The interface consists of three panes:

1. a history pane (top left) that shows a record of the conversation so far,

2. a feedback editing pane (bottom left) where

the current 'speaker' can edit his or her turn, and

3. a pane (right-hand side) with several icons and buttons running from the top to the bottom of the pane:

   (a) an icon representing the current role of the speaker (either shopkeeper or customer),

   (b) an icon representing the language of the current speaker (the icon is clickable and allows the current user to change their language, i.e., the language in which the KB is depicted in the history and feedback panes),

   (c) a button to exit from the application,

   (d) an 'undo button',

   (e) a button that allows the current speaker to add further speech acts/messages to their turn, and

   (f) a button that allows the current speaker to release the turn to the other speaker. When this button is pressed, a number things happen: Firstly, in the KB the representation underlying the feedback text is added to the history. Secondly, fresh underlying representation is created for the feedback text that allows formulation of a new turn. Thirdly, the language of the history and feedback panes are changed to that of the next 'speaker'. Finally, the righhand side pane is changed accordingly, i.e., the icon of the current role is changed, and the icon for the current language is changed also to that of the next speaker.

In Figure 3, it is the English-speaking customer's turn. The history pane shows the preceeding conversation. In the feedback pane, the state of the turn that is under construction is represented by the text 'I would like *some quantity* of *something*'. The anchors are in grey italicized text. They indicate options for extending the current turn. The user has selected the anchor '*some quantity*' and is presented on a menu with several options ('half', 'quarter' and 'one third').

The next figure (Figure 4) shows the state of the feedback text after the user has made selections for both anchors, resulting in the text 'I would like half a kilo of melons'.

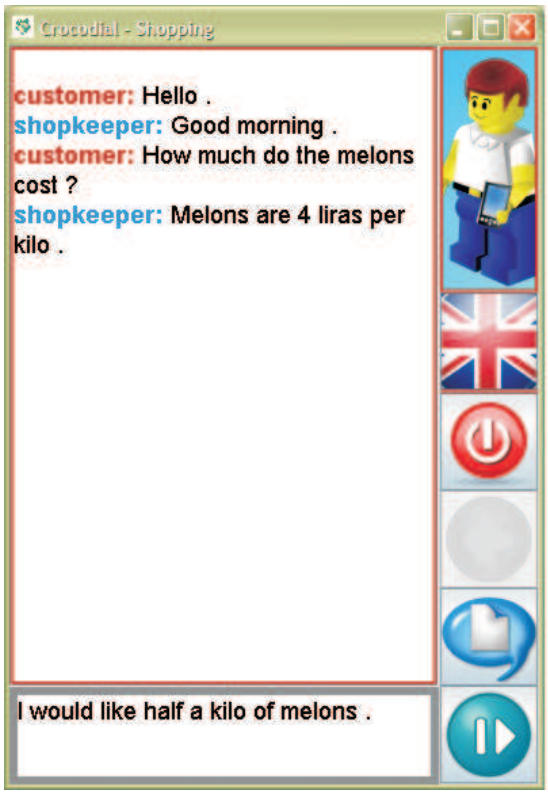

Figure 4: Screen capture of Conceptual Authoring (CA) Interface for English-speaking Customer. Contribution has been completed.

Figure 5: Screen capture of Conceptual Authoring (CA) Interface for Turkish-speaking shopkeeper. State of the Interface after the customer has released the turn to the shopkeeper.

The result of the current user yielding the turn is depicted in Figure 5. Now, it is the Turkish shopkeeper's turn. The language of the history and feedback panes and the right-hand pane icons have been changed accordingly. The feedback pane provides the starting point for constructing a new turn; the anchor '*Konusma*' is equivalent to the English anchor '*Some speech act*'.

The current prototype implements one chapter (food shopping) from a traditional English-Turkish phrase book. Further work will include extending this to further chapters (such as traveling and restaurants/bars). Each chapter is viewed as a self-contained 'dialogue game' that allows the users to construct certain locutions that are appropriate in the corresponding setting.

It took the first author approximately three days to develop the ontology and English resources for the food shopping dialogue game. It took another two days to add the language sources for Turkish. This involved consulting a native speaker of Turkish.

3 System Architecture and Implementation

CROCODIAL is implemented as a chat room: users log in and are authenticated against a server, and each user can see who else is logged in and initiate a dialogue with them. The difference is that the users must agree a dialogue game (such as a shopping encounter) and decide the role within that dialogue game that each is to play (for example the customer and the shop assistant).

The chat window that each user sees is similar in layout to most chat interfaces. It contains a history of the conversation with each entry labelled and colour-coded to identify the speaker, some navigation controls and an input pane to receive text input. This input pane is a CA feedback text interface, allowing the user to interact with the underlying Knowledge Base to develop each utterance that they wish to contribute to the conversation. In the current implementation, the CA application which deals with operations on the KB and generation of feedback texts is implemented in Prolog, running as a shared process on the server.

The chat application is implemented in Java and sits on top of a newly developed framework that makes it easy to develop user interfaces to our CA applications. The architecture – see Figure 6 – is broadly MVC (Model-View-Controller) with the task of updating the model delegated by the controller to the Prolog CA core system. Since the interlocutors are both extending the same underlying KB, the Prolog system is single-threaded, with each new utterance extending the same A-Box. To turn this single-threaded application into a CA chat room the View component is replaced with a multi-threaded session object that allows each chat window to send commands to Prolog and receive updates to its current model as appropriate. To ensure that users do not simultaneously extend the A-Box in mutually inconsistent ways the users are forced to take turns.

Bandwidth requirements are kept down by transmitting only the most recent turn as the model - this means that the history of the conversation shown to each user must be stitched back together by the Java session from the sequence of partial models returned by Prolog.

At any point in the dialogue each user can switch to a different language, choosing from any of the languages supported by the system. Because the text in the chat window is a conceptually

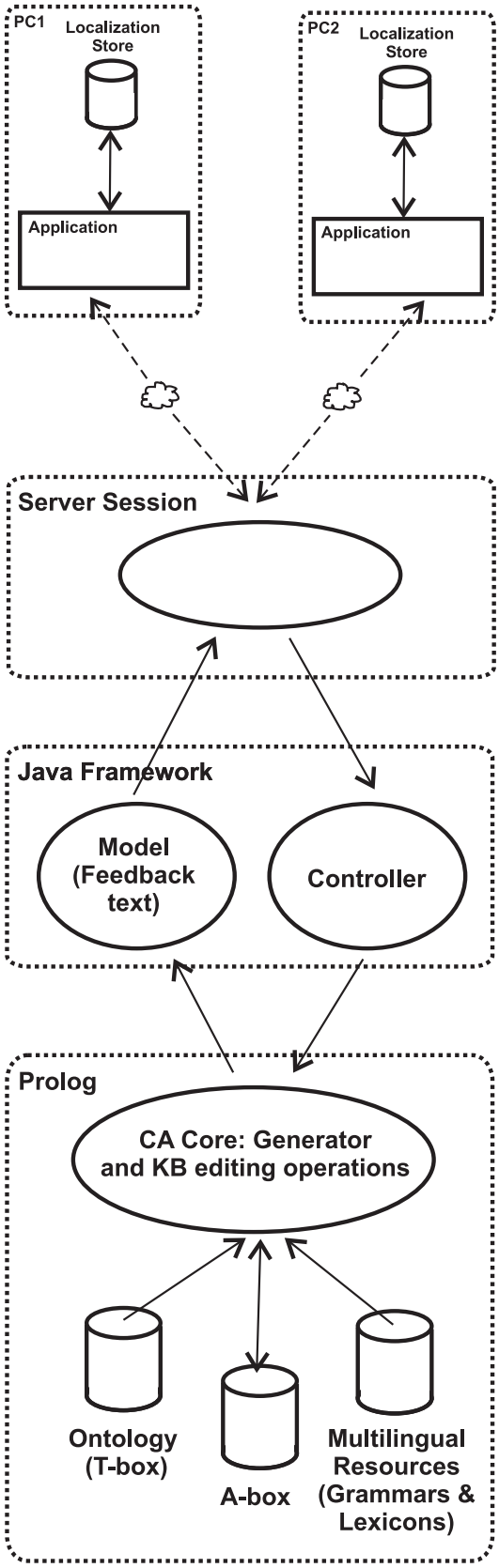

Figure 6: CROCODIAL architecture diagram

authored feedback text this action is functionally equivalent to any other interaction with the text: a command is sent from the window to the session on the server, it delegates the request to a Prolog executable and receives a new CA model which it then returns to the chat window which made the request.

The server itself is multi-sessional, and so a single server can support many simultaneous dialogues, and a single user logged onto the server can participate in many different dialogues at the same time. Each of these dialogues is conceptually logged: by which we mean that each state change made to the A-Box is recorded in a log file. This allows a record of the conversation to be regenerated in any language supported by the system, even if the language was not used in the original dialogue, and also allows us to analyse the use of the system, for example by analysing the time taken to formulate particular questions or responses, or analysing how often speakers back track and correct their utterances.

We have also implemented CROCODIAL as a single-user standalone system which launches a single chat window that switches role and language each time the user completes an utterance. We are planning to migrate the Prolog generation system to Java and produce a single-user Java-only version for use in mobile computing contexts, such as on a PDA in an actual shop in a foreign country.

4 Discussion

In our introduction, we attributed the success of dialogue systems for tasks such as travel planning partly to the fact that the tasks that are involved allow the initiative to reside with the system. The system determines what the main topic will be of the user's next turn, and can thus build up fairly reliable expectations concerning what the user is about to say next. The difficulties facing Machine Translation-based crosslingual dialogue systems for facilitating human–human dialogue can be traced back to the absence of opportunities for such system initiative: each interlocutor is free to say whatever they want at any given time in the conversation. The system has no reference point such as its own utterances, with respect to which it can 'anchor' the users's utterances.

The CROCODIAL system can be viewed as addressing this problem. The multi-person CA technology forces each interlocutor to construct their