Tools for hierarchical annotation of typed dialogue

Myroslava O. Dzikovska, Charles Callaway, Elaine Farrow
Human Communication Research Centre, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW, United Kingdom,
{mdzikovs,ccallawa,efarrow}@inf.ed.ac.uk

1 Introduction

We discuss a set of tools for annotating a complex hierarchical and linguistic structure of tutorial dialogue based on the NITE XML Toolkit (NXT) (Carletta et al., 2003). The NXT API supports multi-layered stand-off data annotation and synchronisation with timed and speech data. Using NXT, we built a set of extensible tools for detailed structure annotation of typed tutorial dialogue, collected from a tutor and student typing via a chat interface. There are several corpora of tutoring done with such chat-style communication techniques (Shah et al., 2002; Jordan and Siler, 2002), however, our annotation presents a special problem because of its detailed hierarchical structure. We applied our annotation methodology to annotating corpora in two different tutoring domains: basic electricity and electronics, and symbolic differentiation.

2 Data Structures

Our corpus has two sources of overlapping annotations: the turn structure of the corpus and situational factors annotation. The data are naturally split into turns whenever a participant presses their "submit" button. Timing information is associated with individual turns, representing the time when the entire message was sent to the other participant, rather than with individual words and sounds as it would be in spoken corpora.

However, turns are too large to be used as units in the annotation for dialogue phenomena. For example, the single turn "Well done. Let's try a harder one." consists of two utterances making different dialogue contributions: positive tutorial feedback for the previous student utterance and a statement of a new tutorial goal. Thus, turns must be segmented into smaller units which can serve as a basis for dialogue annotation. We call these utterances by analogy with spoken language, because they are often fragments such as "well done" rather than complete sentences.

Thus, the corpus has two inherently overlapping layers: the turn segmentation layer, grouping utterances into turns, and the dialogue structure layer built up over individual utterances. The NXT toolkit supports such overlapping annotations, and we built two individual tools to support corpus annotation: an utterance segmentation tool and a tutorial annotation tool.

Additionally, the corpus contains annotation done by the tutor herself at collection time which we call "situational factors". The tutors were asked to submit a set of these factors after each turn describing the progress and state of the student, such as answer correctness, confidence and engagement. The factors were submitted separately from dialogue contributions and provide another layer of dialogue annotation which has to be coordinated with other annotations. The factors are typically related to the preceding student's utterance, but the link is implicit in the submission time.[1] Currently we include the factors in the tool's transcript display based on the submission time, so they are displayed after the appropriate turn in the transcript allowing the annotators to visually synchronise them with the dialogue. We also provide an option to annotators for making them visible or not. In the future we plan to make factors a separate layer of the annotation linked by pointers with the preceding student and tutor turns.

[1] The factor interface was designed to be quick to use and minimally impact the dialogue flow, so the submission timings are generally reliable.

3 Utterance Segmentation

We process the raw data with an automatic segmenter/tokenizer which subdivides turns into individual utterances, and utterances into tokens, providing an initial segmentation for the annotation. However, perfect automatic segmentation is not possible, because punctuation is often either inconsistent or missing in typed dialogue and this task therefore requires human judgement. The output of our automatic segmentation algorithm was verified and corrected by a human annotator.

A screen-shot of the interface we developed for segmentation verification is displayed in Figure 1. With the aid of this tool, it took 6 person-hours to check and correct the automatically segmented utterances for the 18 dialogues in our corpus.

4 Tutorial Annotation

To provide a detailed analysis of tutorial dialogue and remediation strategies, we employ a hierarchical annotation scheme which encodes the recursive dialogue structure. Each tutorial session consists of a sequence of tasks, which may be either teaching specific domain concepts or doing individual exercises. Each task's structure includes one or more of the following: giving definitions, formulating a question, obtaining the student answer and remediation by the tutor.

Generally speaking, the structure of tutorial dialogue is governed by the task structure just as in task-oriented dialogue (Grosz and Sidner, 1986). However, the specific annotation structure differs depending on the tutoring method. In our basic electricity and electronics domain, a tutorial session consists of a set of "teach" segments, and within each segment a number of "task" segments. Task segments usually contain exercises in which the student is asked a question requiring a simple (one- or two-line) answer, which may be followed by a long remediation segment to address the conceptual problems revealed by the answer.

In contrast, in our calculus domain the students have to do multi-step procedures to differentiate complex math expressions, but most of the remediations are very short, fixing the immediate problem and letting the student continue on with the procedure. Thus even though the dialogue is hierarchically structured in both cases, the annotation schemes differ depending on the domain. We developed a generic tool for annotating hierarchical dialogue structure which can be configured with the specific annotation scheme.

The tool interface (Figure 2) consists of a transcript of a session and a linked tree representation. Individual utterances displayed in the transcript are leaves of the tree. It is not possible to display them as tree leaves directly as would be done in syntactic trees, because they are too large to fit in graphical tree display. Instead, a segment is highlighted in a transcript whenever it is selected in the tutorial structure, and a hotkey is provided to expand the tree to see all annotations of a particular utterance in the transcript.

The hierarchical tree structure is supported by a schema which describes the annotations possible on each hierarchical tree level. Since the multi-layered annotation scheme is quite complex, the tool uses the annotation schema to limit the number of codes presented to the annotator to be only those consistent with the tree level. For example, in our basic electricity domain annotation described above, there are about 20 codes at different level, but an annotator will only have "teach" as an option for assigning a code to a top tree level, and only "task" and "test" (with appropriate subtypes) for assigning codes immediately below the teach level, based on the schema defined for the domain.

5 Transcript Segmentation

We had to conduct several simpler data analyses where the utterances in the transcript are segmented according to their purpose. For example, in tutorial differentiation the dialogue concentrates on 4 main purposes: general discussion, introducing problems, performing differentiation proper, or doing algebraic transformations to simplify the resulting expressions. In another analysis we needed to mark the segments where the student was making errors and the nature of those errors.

We developed a generic annotation tool to support such segmentation annotation over the utterance layer. The tool is configured with the name of the segment tag and colours indicating different segment types. The annotator can enter a segment type, and use a freetext field for other information. A screenshot of the annotation tool with utterance purposes marked is given in Figure 3.

6 Data Analysis

The NITE query language (NQL) enables us to access the data as a directed acyclic graph to correlate simple annotations, such as finding out the

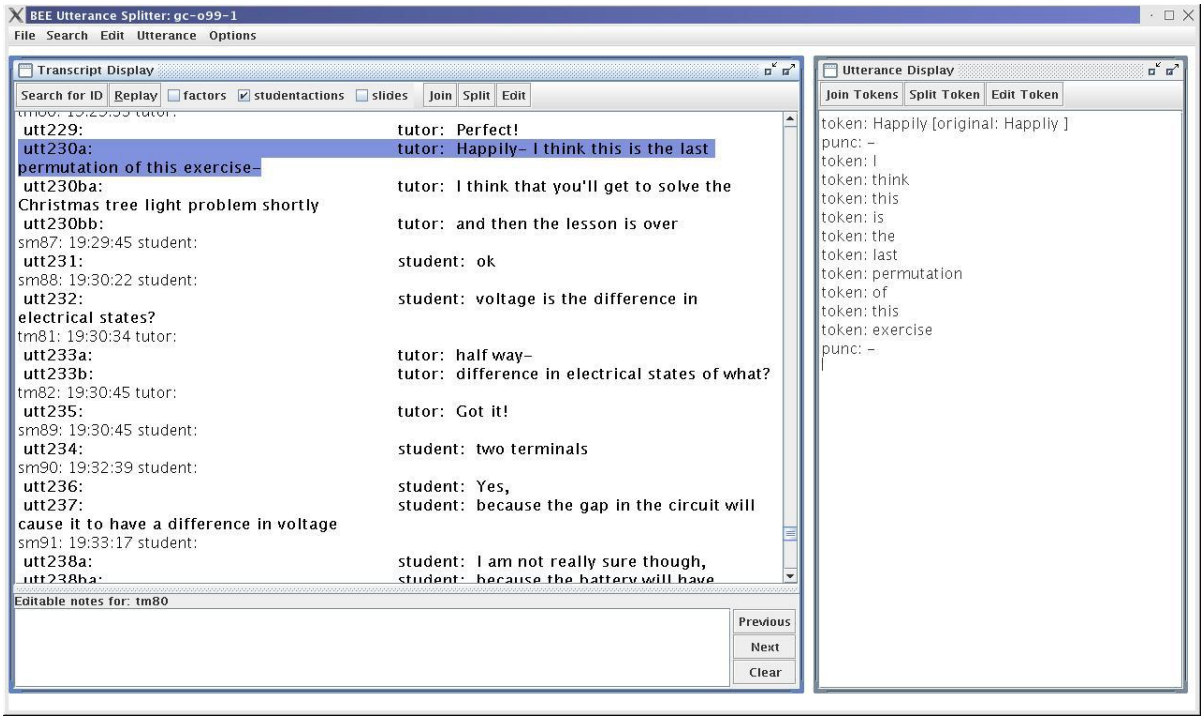

Figure 1: Utterance Segmentation Tool.

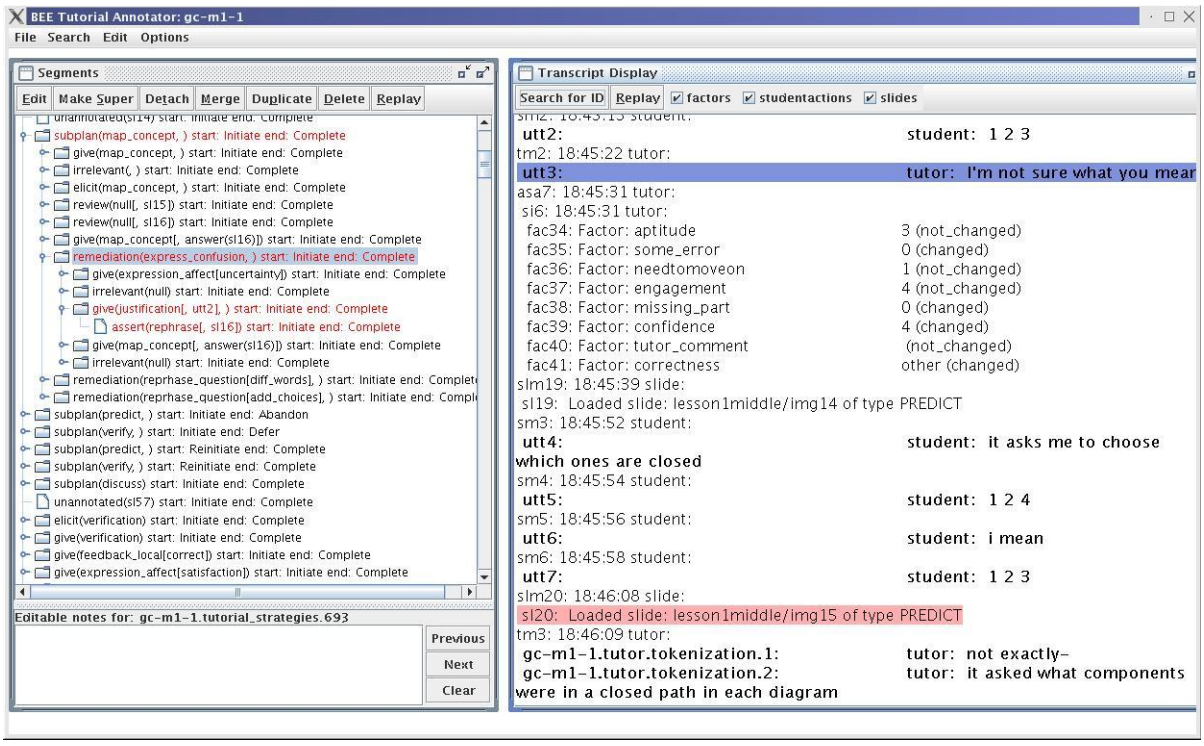

Figure 2: Tutorial Strategy Annotation Tool.

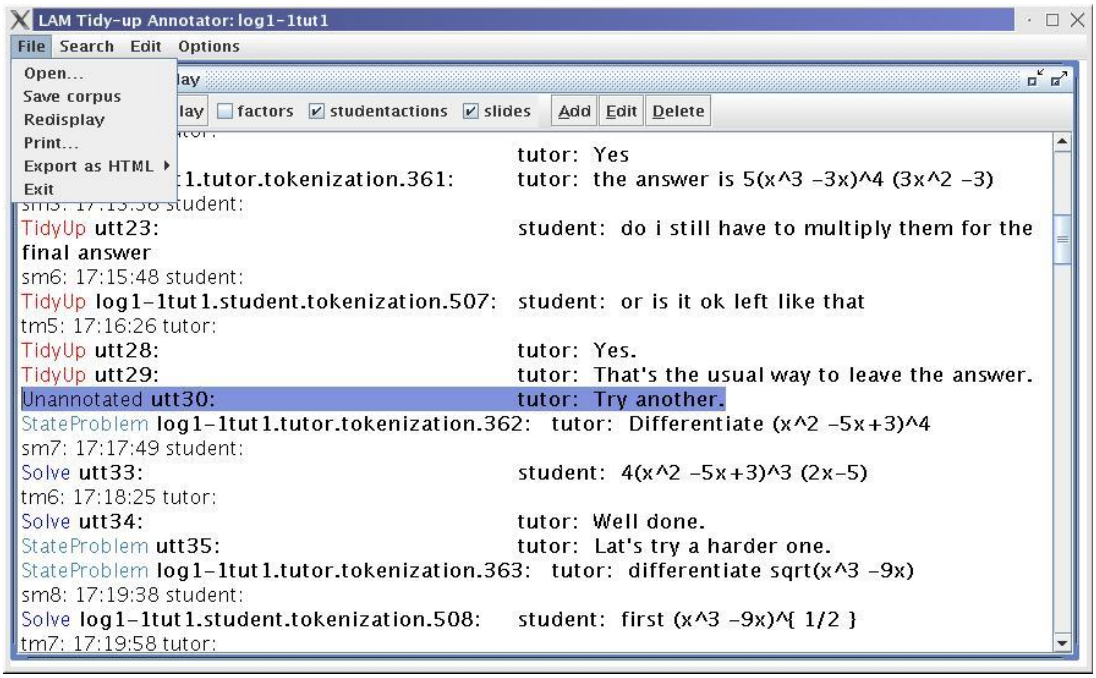

Figure 3: Segmentation tool. The segment labels are shown on the left.

number of turns which contain only mathematical expressions but no words. We use the NITE query interface for simpler analysis tasks such as finding all instances of specific tags and tag combinations.

However, we found the query language less useful for coordinating the situational factors annotated by tutors with other annotation. Each set of factors submitted is normally associated with the first student turn which precedes it, but the factors were not linked to student utterances explicitly. NQL does not have a "direct precedence" operator.[2] Thus it is easier derive this information using the JAVA API. To make the data analysis simpler, we are planning to add a pointer layer, generated automatically based on timing information, which will use explicit pointers between the factor submissions and preceding tutor and student turns.

7 Conclusions

We presented a set of tools for hierarchically annotating dialogue structure, suitable for annotating typed dialogue. The turns in these dialogues are complex and overlap with dialogue structure, and our toolset supports segmenting turns into smaller utterance units and annotating hierarchical dialogue structure over the utterances, as well as providing simpler segmentation annotation.

[2]It's possible to express the query in NQL using its precedence operator "$\ll$" as "`($f factor) ($u utterance) (forall $u1 utterance) : (($f` $\ll$ `$u) && ($f` $\ll$ `u1)) → (u` $\ll$ `u1)`". However, this is very inefficient since it must check all utterance pairs in the corpus to determine direct precedence, especially if it needs to be included as part of a bigger query.

Acknowledgements

This material is based upon work supported by a grant from The Office of Naval Research number N000149910165 and European Union 6th framework programme grant EC-FP6-2002-IST-1-507826 (LeActiveMath).

References

Jean Carletta, J. Kilgour, T. O'Donnell, S. Evert, and H. Voormann. 2003. The NITE object model library for handling structured linguistic annotation on multimodal data sets. In *Proceedings of the EACL Workshop on Language Technology and the Semantic Web*.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, 12(3):175–204.

Pamela Jordan and Stephanie Siler. 2002. Student initiative and questioning strategies in computer-mediated human tutoring dialogues. In *Proceedings of ITS 2002 Workshop on Empirical Methods for Tutorial Dialogue Systems*.

Farhana Shah, Martha W. Evens, Joel Michael, and Allen Rovick. 2002. Classifying student initiatives and tutor responses in human keyboard-to-keyboard tutoring sessions. *Discourse Processes*, 33(1).