IORelator: A Graphical User Interface to Enable Rapid Semantic Annotation for Data-Driven Natural Language Understanding

David DeVault and **Susan Robinson** and **David Traum**
USC Institute for Creative Technologies
13274 Fiji Way
Marina del Rey, CA 90292
{devault,robinson,traum}@ict.usc.edu

Abstract

This paper describes a new annotation GUI, called IORelator, which is designed to facilitate rapid semantic annotation in support of high-performance data-driven NLU for spoken dialogue systems. We summarize our requirements for rapid NLU annotation, and discuss how the GUI views and operations that IORelator provides meet these needs by enabling thousands of natural utterances to be quickly annotated with their correct semantics.

1 Introduction

This paper introduces a new annotation GUI, called IORelator, that is designed to improve the process of annotating the semantics of large collections of natural language utterances. This new GUI has a number of design features which work together to enable an annotator to focus on relevant parts of a growing annotated corpus, and to add new annotations quickly and in a consistent manner. These design features include:

- The annotator can create one or more views of the data using several kinds of filters. The filters enable similar utterances or similar semantics to be grouped together within a view (a sub-window) in the GUI.
- Several of these filtered views can be positioned simultaneously on screen, and can be connected in a way that allows the annotator to choose the best semantics by highlighting and exploring previous annotations of similar or related utterances.
- Together, these features allow groups of similar utterances to be identified, selected, considered, and quickly annotated.

We begin by motivating our need for rapid semantic annotation.

$$\begin{bmatrix} mood : declarative \\ sem : \begin{bmatrix} type : event \\ agent : captain-kirk \\ event : deliver \\ theme : power-generator \\ modal : \begin{bmatrix} possibility : can \end{bmatrix} \\ speech-act : \begin{bmatrix} type : offer \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 1: Example utterance semantics (AVM).

```
<s>.mood declarative
<s>.sem.type event
<s>.sem.agent captain-kirk
<s>.sem.event deliver
<s>.sem.theme power-generator
<s>.sem.modal.possibility can
<s>.sem.speechact.type offer
```

Figure 2: Example utterance semantics (frame).

2 Motivation and Background

In this paper, we consider an annotation task that arises for the natural language understanding (NLU) component in SASO-EN (Traum et al., 2008), a virtual human dialogue system. This system allows users to interact in interactive spoken dialogue with virtual human characters, and is designed to serve as a negotiation training tool, where users learn about negotiation tactics in the context of the culture and social norms of a particular community. The semantic representation for NLU, also used in several previous virtual human systems (Traum, 2003), is an attribute-value matrix (AVM), where the attributes and values are linked to a domain ontology and task model (Hartholt et al., 2008). Figure 1 shows an example AVM for an utterance such as "we can provide you with power generators". To facilitate statistical NLU, the AVMs are linearized using a path-value (or *key-value*) notation, as shown in Figure 2. We call these linearized AVMs *frames*. In SASO-EN, the NLU module takes a spoken utterance (as rec-

Training set size	100	500	1000	2000	3000	3826
NLU F-score	0.31	0.63	0.70	0.74	0.75	0.79

Table 1: NLU performance vs. training set size.

ognized by ASR) as input and emits an appropriate semantic frame as output.

The NLU module is data-driven, and requires a set of annotated *(transcribed utterance, frame)* pairs as training data. Over the years, a variety of machine learning frameworks have been explored to try to increase NLU performance. Table 1 shows how NLU performance for one recent implemented NLU algorithm (Sagae et al., 2009) increases as the number of annotated training utterances grows. In this figure, NLU performance is measured with F-score, computed as the harmonic mean of the precision and recall of the individual key-values in the NLU output frames for 449 held-out test utterances. In the figure, note first that NLU performance generally increases as the training set grows. For SASO-EN, this improvement is especially pronounced as the training set grows to include around 2000 annotated examples.

Incorrect NLU output, which can create misunderstandings that disrupt the training process, is one of the most problematic issues in the current SASO-EN system. Unfortunately, the need to annotate and maintain an NLU training corpus of thousands of utterances to achieve high-performing NLU introduces considerable costs and challenges into the development process. The IORelator GUI has been developed with the aim of increasing both the speed at which utterances can be annotated as well as the size of annotated corpus that can be effectively explored and maintained when changes in the semantic annotation scheme are necessitated by ongoing development.

3 Related Work

Rather than developing a new GUI, we could use, or adapt, a general-purpose linguistic annotation tool such as GATE (Cunningham et al., 2002) for SASO-EN annotation. However, we perceived a need for rapid browsing, filtering, and linking operations, as described in the next section, which we judged were easier to implement, for initial evaluation purposes, in a standalone GUI. We are currently integrating our new GUI into a plug-in for Protégé (Knublauch et al., 2004), a general purpose ontology editor used for SASO-EN.

Previous work has illustrated that special-purpose tools (e.g. (Geertzen, 2007)) and using annotators with strong expertise are important to achieving a high annotation throughput, with experts able to annotate much more quickly than non-experts (Geertzen et al., 2008). Our effort here is motivated by the desire to improve the annotation speed that non-experts as well as experts can attain (though we do not address the differences between annotator types in this paper).

The specific functionality implemented in different annotation GUIs shows differences in emphasis and assumptions, such as support for adding semantic annotations to existing syntactic structure (Burchardt et al., 2006), or providing detailed pop-up or drop-down options and other guidance for structured semantic annotations (Geertzen, 2007). We discuss the emphasis and assumptions of our own GUI in the next section.

4 The IORelator GUI

The IORelator (**I**nput **O**utput **Relator**) GUI, which is implemented in Java and depicted in Figure 3, is designed to support the rapid definition and maintenance of links from an input domain $\mathcal{I}$ (such as a set of utterances) to an output domain $\mathcal{O}$ (such as a set of possible SASO-EN frames). The links that have been defined determine an input-output relation $\mathcal{R} \subseteq \mathcal{I} \times \mathcal{O}$.

The IORelator GUI allows the annotator to create any number of *views* of the input domain (see *Utterance View 1* and *Utterance View 2* in the Figure), the output domain (see *Frame View*), and the annotated links (see *Links View*). Each view is a resizable, dockable sub-window that can be positioned as desired by the annotator to make efficient use of screen real-estate for the task at hand. Each view has its own separate *filters, connectedness, item selection*, and *toolbar*.

A *filter* uses some criterion to limit the items that are visible in a view. Based on guidance from our annotators, IORelator supports filters that include or exclude items using keywords or keyphrases, regular expressions, or whether an item participates in any links. For example, in the *Frame View* in the figure, the user has introduced a keyphrase filter (see *Frame Text Filter*) that includes only frames containing the text

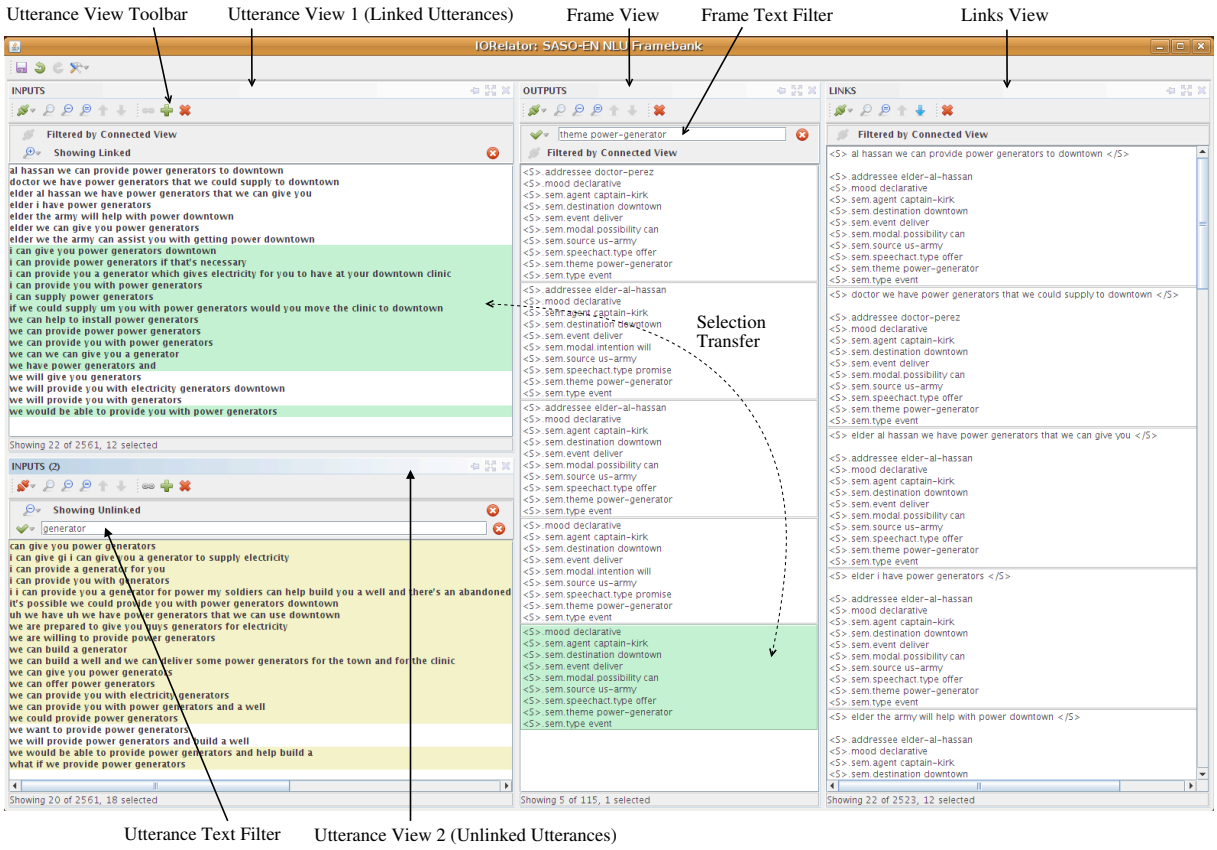

Figure 3: The IORelator GUI.

`theme power-generator`.[1] Because only 5 of 115 frames in the output domain match this filter, only 5 frames are visible in the view. The annotator can thereby focus on only these frames. In the figure, the user has used filters to create *Utterance View 1* as a view that shows only utterances which have already been linked, and *Utterance View 2* as a view that shows only utterances which have not yet been linked. In this way, *Utterance View 2* serves as a "todo list" for utterances requiring annotation; the annotator can consult the previous annotations of already linked utterances in *Utterance View 1* when desired. Additionally, the user has added a keyword filter (see *Utterance Text Filter*) to *Utterance View 2*, so that only the 20 unlinked utterances containing the word `generator` are displayed. Together, these filters allow the annotator to focus attention on the 20 utterances including the word "generator" and the 5 possibly appropriate frames for these utterances.

Each view in IORelator can be either *connected* or *not connected*. When a view is *connected*, items selected with the mouse cause related items in other connected views to become highlighted. In the figure, the user has marked all the views as connected except for *Utterance View 2*, which contains unlinked utterances. In this screenshot, the user has selected the bottom-most frame in the *Frame View*; this selection has been transferred (*Selection Transfer*) to *Utterance View 1* by automatically selecting the utterances in that view that are already linked to that frame. This helps the annotator to consider this frame as a possible annotation for some of the unlinked utterances, as they can be quickly compared to other utterances already linked to that frame.[2] A second aspect of connectedness between views is that filters applied in connected views automatically filter out unrelated items in other connected views.

The *item selection* in a view v is determined

[1] To support filtering with regular expressions, IORelator requires that all items in its input and output domains be convertible to plain text. This does not preclude the use of structured semantic annotations such as SASO-EN frames in the output domain. Any semantic annotation can be used so long as it can (1) be depicted on screen in views and (2) converted to plain text so that filtering operations can be performed.

[2] A long-term motivation for this functionality is to lower the learning curve associated with the NLU annotation task, by allowing novice annotators to consult previous expert annotations for similar utterances.

by mouse-based selection in v, or, if v is connected, by mouse-based selections in other connected views. Finally, each view has a *toolbar* (see *Utterance View Toolbar* for example) providing access to relevant operations for the kind of items in the view. These operations can be customized for specific applications, and can have effects that depend on the selected items in the different views. For example, in the figure, the user has selected a range of unlinked utterances in *Utterance View 2*. By clicking the link icon in that view's toolbar, the user can link all of these utterances to the frame selected in the *Frame View*.

5 Evaluation

We have performed an initial evaluation of the IORelator interface for SASO-EN NLU annotation. We began with a corpus of 4,678 token user utterances which had been captured as audio and subsequently transcribed. In this domain, some common transcripts such as *yes*, *no*, *okay*, and others recur frequently; however, we require that only one example of each utterance type be annotated. Factoring out duplicates left 2,561 unique utterance transcripts in need of annotation. We then had a system developer (one of the authors of this paper) annotate each of these 2,561 utterances with its correct NLU output frame using IORelator. This task was completed in 7.1 hours, or an average of about 10 seconds per unique utterance.

6 Discussion and Future Work

We have found using IORelator to be substantially faster than prior approaches to SASO-EN NLU annotation, which have included linking utterances to frames inside the Protégé GUI (using custom dialogue boxes and forms), or alternatively, using a simple text editor for annotation.[3] This has made it easier to annotate enough training data that the NLU module can achieve the high-performance region of Table 1. Our annotators attribute the speed-up to the rapid browsing, filtering, and linking operations discussed in Section 4.

Future directions include evaluation of additional design goals for IORelator. These include evaluating throughput for novice annotators or hybrid teams of novices and experts, efficiency of revision and maintenance as opposed to initial annotation, inter-annotator agreement, and the extent to which the throughput observed here transfers to other NLU annotation schemes.

[3] While there has not been a formal study to quantify exact annotation time for these previous approaches, we estimate that the previous approach to annotation in Protégé, where navigation and searching frames is more cumbersome, required at least 1 minute per utterance for an expert annotator. Several schemes for annotation in a text editor have also been implemented, but we estimate that even the fastest of these required an expert about 20 seconds/utterance.

Acknowledgments

We thank our anonymous reviewers, Ron Artstein, Anton Leuski, Arno Hartholt, Tom Russ, and Kenji Sagae for helpful discussions. The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

A. Burchardt, K. Erk, A. Frank, A. Kowalski, and S. Pado. 2006. SALTO: a versatile multi-level annotation tool. In *Proceedings of LREC*.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of ACL*.

J. Geertzen, V. Petukhova, and H. Bunt. 2008. Evaluating dialogue act tagging with naive and expert annotators. In *Proceedings of LREC*.

J. Geertzen. 2007. Ditat: a flexible tool to support web-based dialogue annotation. In *IWCS-7*.

A. Hartholt, T. Russ, D. Traum, E. Hovy, and S. Robinson. 2008. A common ground for virtual humans: Using an ontology in a natural language oriented virtual human architecture. In *Proceedings of LREC*.

H. Knublauch, R.W. Fergerson, N.F. Noy, and M.A. Musen. 2004. The Protégé OWL plugin: An open development environment for semantic web applications. In *Proceedings of the Third International Semantic Web Conference*, Hiroshima, Japan.

K. Sagae, G. Christian, D. DeVault, and D. R. Traum. 2009. Towards natural language understanding of partial speech recognition results in dialogue systems. In Short Paper Proceedings of NAACL HLT.

D. R. Traum, W. Swartout, J Gratch, and S Marsella. 2008. A virtual human dialogue model for non-team interaction. In L. Dybkjaer and W. Minker, editors, *Recent Trends in Discourse and Dialogue*. Springer.

D. R. Traum. 2003. Semantics and pragmatics of questions and answers for dialogue agents. In *Proceedings of IWCS*, pages 380–394.