NOMOS: A Semantic Web Software Framework for Annotation of Multimodal Corpora

John Niekrasz[*], **Alexander Gruenstein**[†]

[*] Center for the Study of Language and Information
Stanford University
niekrasz@stanford.edu

[†] Spoken Language Systems
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
alexgru@csail.mit.edu

Abstract

We present NOMOS, an open-source software framework for annotation, processing, and analysis of multimodal corpora. NOMOS is designed for use by annotators, corpus developers, and corpus consumers, emphasizing configurability for a variety of specific annotation tasks. Its features include synchronized multi-channel audio and video playback, compatibility with several corpora, platform independence, and mixed display of temporal, non-temporal, and relational information. We describe NOMOS from two perspectives. First, we present its software architecture, highlighting its principal difference from comparable systems: its use of an OWL-based semantic annotation back-end which provides automatic inference capabilities and a well-defined method for layering datasets. Second, we describe how the system is used. For corpus development and annotation we present a typical use scenario involving the creation of a schema and specialization of the user interface. For processing and analysis we describe the GUI- and Java-based methods available, including a GUI for query construction and execution, and an automatically generated schema-conforming Java API for processing of annotations. Additionally, we present some specific annotation and research tasks for which NOMOS has been specialized and used, including topic segmentation and decision-point annotation of meetings.

1. Introduction

Progress in the field of human language technology has brought about a great increase in the complexity and depth to which multimodal language resources are being annotated and used by humans and machines. An excellent example of this is the research being done as part of the European Augmented Multi-party Interaction (AMI) project[1]. One major aim of this project is to enhance the value of multimodal recordings by providing tools for their dissemination, summarization and browsing. In order to accomplish this goal, a large corpus of multi-party meetings is being collected and annotated for an extensive range of inter-connected properties, including dialogue acts, emotion, head and hand gestures, speech transcription, topics, named entities, and many others (McCowan et al., 2005). Of course, this kind of endeavor has not been exclusive to AMI. The multi-party meeting domain has become a major focus in the fields of speech recognition and automatic understanding of natural communication, and the AMI Meeting Corpus is one of many meeting corpora which have been collected and annotated in this manner (Janin et al., 2003; Garofolo et al., 2004; Burger et al., 2002). Producing rich, multi-tiered annotations of multimodal spoken language resources such as these is difficult for many reasons: distribution of the data and annotation work is difficult, the annotation tasks require a tight integration between signal and symbol, and the design of annotation tools appropriate for specific tasks is non-trivial.

In our own research as part of the DARPA Cognitive Assistant that Learns and Organizes (CALO) project[2] (see (Niekrasz et al., 2005) for background information), we too have been confronted with similar problems to those faced by the AMI project and have designed an annotation and processing framework to address them. The CALO project has essentially two scenarios in which producing multi-tiered annotations are an important problem: 1) during system development and research, and 2) in the deployable CALO assistant. In the development scenario, meetings are recorded at several different sites, distributed, and processed by both humans and machines. For example, the recorded audio is processed at one site by a speech-to-text system to produce transcriptions which are then delivered to a site which parses this text which in turn provides input to a decision detection algorithm. The recordings may also be manually transcribed or annotated by humans at any step in the process. Currently, manual annotations are being done for features like topics, action items, and physical gestures, and they are being used for training the algorithms which detect these phenomena automatically. In the deployable CALO assistant scenario, meetings are recorded by the system, automatically analyzed, and summarized for actual users who then use this information as part of their daily work and provide feedback to the system by means of their interaction with the system. In this latter scenario, extracted information resides centrally in a shared ontology-based knowledge base that uses an ontology of multimodal discourse (Niekrasz and Purver, 2006) and is the focal point

---

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

[2] http://www.ai.sri.com/project/CALO

for knowledge sharing in the deployable system. In the former case, however, the methods for sharing data and annotations are not as clearly defined nor as easily managed, since both humans and machines are doing the processing and the process is distributed over time and space. It is this problem for which the NOMOS software has been designed: to establish a flexible framework in which CALO researchers can produce, share, and process annotations during system development and research, and to have this framework use a knowledge representation back-end which is similar to that which the deployable system employs, facilitating the transition from development to deployment.

In the remainder of this paper, we describe the NOMOS framework in detail. In Section 2., we present its software architecture, highlighting its principal difference from comparable systems like the NITE XML Toolkit (Carletta et al., 2003): its use of an OWL-based semantic annotation back-end which provides automatic inference capabilities and a well-defined method for layering datasets. In Section 3., we describe how the system is used. For corpus development and annotation we present a typical use scenario involving the creation of a schema and specialization of the user interface. For processing and analysis we describe the GUI- and Java-based methods available, including a GUI for query construction and execution and an automatically generated schema-conforming Java API for processing of annotations. In Section 4., we present some specific annotation and research tasks for which NOMOS has been specialized and used, including topic segmentation and decision-point annotation of meetings.

2. Software Architecture

The NOMOS software architecture has been designed to meet many key requirements identified in the literature as being critical to effective linguistic annotation (Ide and Romary, 2004), including expressive adequacy, media independence, semantic adequacy, incrementality, separability, extensibility, and processability. It has also been constructed in accordance with many of the principles of annotation tool design collected and summarized in (Reidsma et al., 2005): the software is open-source, flexible, platform-independent, and extensible, and it supports layered annotation, transcription, configurable visualization, querying, and a programming API. In the remainder of this section, we discuss how the NOMOS software is implemented to address these requirements, beginning with an introduction to the semantic annotation technology on which the NOMOS annotation framework is built.

2.1. Semantic Annotation

NOMOS is unique among related annotation frameworks in that it is rooted in a W3C-recommended standard for knowledge representation called the Web Ontology Language (OWL)[3], part of the Semantic Web framework. The OWL language is a description logic that provides a generic and expressive formal semantics for annotation which is not offered by languages like XML and RDF. It is designed and supported by a large community and a rich set of Semantic Web technologies. Building on OWL technology,

NOMOS is able to provide capabilities similar to other notable frameworks like the NITE XML Toolkit (Carletta et al., 2003), ATLAS (Bird et al., 2000), and AGTK (Maeda et al., 2002) including layered annotation sets, a connection between symbolic linguistic features and temporal signal segmentation, and a query language. However, the use of a formal logic as an annotation schema allows NOMOS to provide mechanisms for aiding annotation tool developers and annotators such as the ability to check the logical consistency of annotations and to use reasoners to infer logically implied annotations. It also facilitates use and integration of a growing number of existing OWL ontologies that are being used for annotation of web documents or for linguistic annotation, e.g. the General Ontology for Linguistic Description (Farrar and Langendoen, 2003).

The OWL-based semantic annotation core of NOMOS is implemented using a mature and well-supported open-source back-end Semantic Web implementation called the Jena Semantic Web Framework[4]. The Jena framework provides the core mechanisms for storing, reasoning with, and querying the data. The Jena architecture also abstracts the storage method for annotations, allowing them to be stored in a database or in files.

2.2. The Ontology Programming Interface

As an intermediate layer between the Jena Semantic Web core architecture described above and the NOMOS system, we have created our own ontology API called the Ontology Programming Interface (OPI). This layer is used to simplify the Jena ontology API and to provide a programming interface for annotation tool developers and corpus consumers who wish to process the annotations directly. The OPI layer works by transforming the OWL ontology into automatically-generated Java interfaces which encode the classes and properties in the ontology as classes and methods in Java. This facilitates coding, enables compile-time type-checking and guarantees ontology conformance in the code.

The OPI framework is an important element in the configurability of the NOMOS framework. As will be shown below, the OPI is used as the basis for creating Java plugins for importing and processing of annotations as well as the specialization of the user interface for particular annotation tasks. The OPI is the sole Java interface through which annotations may be created, managed, layered, and processed in NOMOS.

Following the Jena API on which it is built, at the core of the OPI framework is the notion of a *model*. A model consists of a set of statements which use the terminology established in the ontology to make assertions about the data. For example, a statement might relate one object to another using a property from the ontology, or it may simply state the class membership or existence of a particular object. These statements are called triples in OWL/RDF parlance, and they are produced by the annotator (whether human or machine), formulated in OWL, and brought together into a single file (or database table) to form a model. To produce layered annotations, these models are made to be dependent

---

[3]http://www.w3.org/TR/owl-features/

[4]http://jena.sourceforge.net/

on other models which are said to be "imported". For example, an annotator who is annotating a meeting for dialogue acts will produce a new model which will be assigned a dependency on the other models which contain the transcripts and other supporting information. The necessity for this dependency comes from the fact that every object (i.e., event, entity, relation, etc.) is assigned a unique identifier which is shared across models. In this case, an utterance event would have been assigned a unique identifier in a previously available model, and the new statement assigning its dialogue act property must use the same identifier. This dependency structure between annotation layers can be used by the corpus designer to manage multiple annotations of the same features or to maintain control over versioning of annotations. It also facilitates the selective use of particular subsets of data.

3. Capabilities

In this section, we describe what NOMOS can *do* and how it may be *used* by corpus developers, annotators, and corpus consumers. We present these capabilities by describing a typical use scenario which follows the process of importing a corpus or input layer, creating an annotation ontology (schema), designing a specific user interface, making the annotations, and then processing them for the purpose of research or dissemination. Specific examples of NOMOS specialization, for which this process has been followed, are reserved for the subsequent section.

3.1. Importing the Input Layer

For all annotation tasks, the process begins with an input dataset. This may consist of unprocessed information such as media files or documents, or it may consist of the output layers of previously produced annotations. It is most common that the input actually includes both of these. For example, to annotate a meeting for dialogue acts, one requires the raw audio (and perhaps video), a speaker diarization, utterance segmentation, and transcripts of the segments. While it may be the case that if NOMOS is used for previous annotation tasks that some of these data will already be accessible to NOMOS, we assume for the purpose of illustration that they are not.

In the NOMOS framework, the importation of these data is performed through the use of an *importer plugin*. To date we have created importer plugins for the ICSI (Janin et al., 2003), ISL (Garofolo et al., 2004), NIST (Burger et al., 2002), and AMI (McCowan et al., 2005) meeting corpora, the Meeting Recorder Dialog Act Corpus (Shriberg et al., 2004), and the CALO dataset. Importer plugins are Java methods which translate source information such as the location of media files or the information contained in an XML-formatted transcript into NOMOS annotations using the OPI framework API. As mentioned previously, an OPI Java API is generated from and embodies a particular ontology. It is therefore the case that any information which needs to be imported must have an ontology which accommodates that information. NOMOS comes with an ontology for the most common concepts used in multimodal discourse research such as *Recording*, *Person*, and *Meeting*, which may be used as a basis for annotation. However, if the input layer also consists of information not adequately represented by the existing default NOMOS ontology, these classes and properties must be added, and they are added in the same manner that is done for new annotations that will be produced as part of the annotation task. We now describe that process.

3.2. Creating an Annotation Ontology

To create an annotation ontology which contains the classes, subclass (taxonomic) relationships, and properties of objects that will be the subject of annotation, one may use one of several well-supported graphical tools designed for creating OWL ontologies, e.g. Protégé-OWL[5] (Knublauch et al., 2004). Optionally, OWL code may also be created directly, or a simpler syntax which has been designed for the NOMOS framework may also be used. New ontologies may also simply extend existing ontologies using the model importing mechanism as they too are considered models in the OWL/RDF framework. Upon completion of an ontology design, a command within the NOMOS application is used to generate the corresponding OPI Java code. This code is then available for use by importer plugins as described above, by user interface plugins, or by processing plugins (each described below). For example, from a class *Utterance* and property *start-time* in the ontology, Java code for an interface `Utterance` with method `Utterance.getStartTime()` would be generated into the OPI for that ontology.

3.3. Designing a User Interface

As pointed out in previous surveys of annotation tools (Reidsma et al., 2005), specific designs for specific problems produce greater accuracy and efficiency of annotation. With this in mind, the NOMOS user interface is a highly configurable and extensible plugin-based framework. The core interface is highly general and composed of a media-playback interface, temporal tracks, non-temporal panels, and objects that may be placed on those tracks and panels called labels. Each of these elements are implemented in the Java Swing framework and may be specialized for the particular annotation task. In the NOMOS framework, these are called *label plugins*, *track plugins*, and *panel plugins*. Additionally, there is an *interaction plugin* which manages interaction between these components. A comprehensive manual and API documentation is available to developers, and some example plugins are shown in Section 4.. Using this framework, the full set of capabilities provided by Swing are still available to the designer, with a very small set of required plugin points which connect the interfaces with the NOMOS architecture. This gives tool developers the option of leveraging existing NOMOS user interface plugins or creating something completely new.

Once any needed user interface specialization plugins have been created, the user then employs *queries* and *perspectives* to determine which data are displayed and which interface plugins should be used to display them. Queries

---

[5]For applications which will not fully employ the logical properties of the OWL language, one may consider this for instructional purposes simply as creating a typed relational database schema.
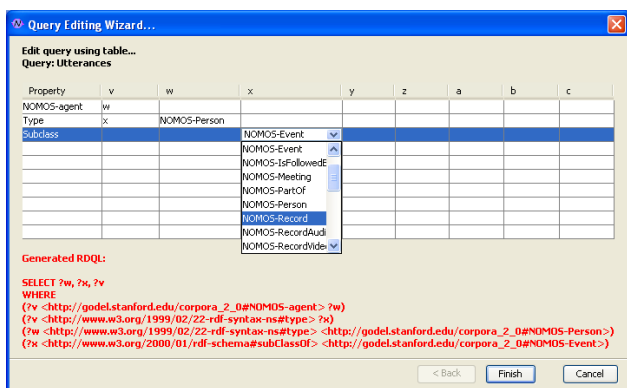
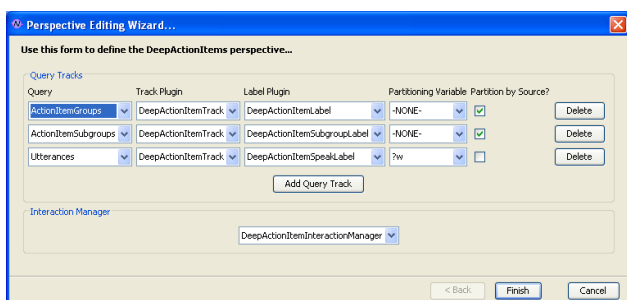Figure 1: The NOMOS interface for editing queries.

Figure 2: The NOMOS interface for editing perspectives.

extract the desired information from the annotations and may be created using a graphical query-construction interface included in the NOMOS application (see Fig. 1), or using the SPARQL[6] query language. Perspectives then configure the manner in which the query results are displayed by assigning them to interface plugins, as seen in Fig. 2, which shows the NOMOS perspective-editing interface. We present examples of the user interface plugins, queries, and perspectives in Section 4..

3.4. Performing Annotations

The manner in which annotations are performed will be highly dependent on the specific user interface that has been designed for the task. However, NOMOS does provide a shared interface for common or generic interactions. For example, media playback control employs a shared user interface in which actions may be performed using menu items, a toolbar, or keystrokes, depending on the user's preference. The NOMOS API also provides several common actions which may be used when designing the annotation interface, including generic property editing for individual objects, drag-and-drop creation of relations between objects, and various mechanisms for modifying the temporal properties of event objects.

3.5. Querying and Processing

For consumers of annotations, the most common interest is for statistical processing, and NOMOS provides two methods for this. In the first, the user selects a query to run and a set of models on which to run it. The query is then

---

[6]http://www.w3.org/TR/rdf-sparql-query/

automatically executed iteratively on each model, producing flat tab-separated files containing the results. This technique facilitates processing by other programs or by statistical analysis applications. The other technique available for processing annotations is the use of *processor plugins*. Processor plugins perform an operation that is essentially the inverse of importer plugins: they read NOMOS annotations and produce some output. As with importer plugins, processor plugins may be run directly from the NOMOS interface or executed as standalone Java processes which use the OPI Java API to access the annotation data. This technique circumvents the need to translate annotations and allows the consumer to generate output of any kind using Java. For example, Java algorithms are able to add a layer of annotations automatically using this method, without the need for intermediate translation steps.

4. Example Applications

Following the typical use scenario described above, the NOMOS framework has been employed effectively for a number of tasks as part of the CALO project and others. For example, NOMOS specializations have been designed for the annotation of topic segments, decision points, and action items in meetings (Gruenstein et al., 2005). These specific annotations have been done in accordance with a rich ontology of multimodal discourse (Niekrasz and Purver, 2006) which we have employed as part of the CALO system ontology. This has facilitated the use of our NOMOS-annotated data in research toward the deployable CALO system. The following sections describe how the NOMOS framework has been specialized for these specific tasks and briefly describes the datasets which have been produced.

4.1. Topic Segmentation and Decision-points

In Figure 3, a screenshot of a NOMOS implementation for the segmentation of meetings by topic and marking of decision-relevant utterances is shown. In the implementation, a track plugin has been used to show the coarse segmentation of the discourse and provide a specific set of mouse actions which allows the annotator to manipulate those segments and their names. Additionally, the implementation uses a label plugin to specialize the manner in which utterances are displayed such that they are colored based on their association to an identified action item. These action items are also shown as a tree structure in a panel plugin on the left side of the window. Using this tool, 56 meetings from the ICSI and ISL meeting corpora were annotated, and the annotations have been made publicly available (see (Gruenstein et al., 2005) for full details on the design of the tool and the resulting dataset).

4.2. Action Items

In Figure 4, a NOMOS implementation for the annotation of meeting action items is shown. In this scheme, utterances are grouped according to how they contribute to the participants' agreement on the description, ownership, and due-date of an assigned action item. In the screenshot, relations between temporal and non-temporal objects are shown. This implementation uses a track plugin which assigns a display time to non-temporal entities (action items)

based on their relationship to temporal entities (utterances). Label plugins are used to provide additional coloring and to specialize the text that is displayed in the labels based on the class of object they represent.

In both of the above implementations, a query to extract utterance and transcription information has been made part of the perspective that has been defined for the task. In the latter example, however, additional queries have been added to the perspective in order to extract the objects which are being displayed on the action item tracks.

5.   Future Work

We are just beginning to work with several other researchers on annotation tasks which use NOMOS, and a new version of NOMOS, version 2.0, has been made publicly available for download[7]. It is being used to collect more action-item annotations, topic segmentations, and importance labeling of utterances in meetings. In the future, to support a broader audience for NOMOS, we hope to concentrate on simplifying the documentation for non-technical users. This will principally involve the integration of more plugins in order to provide more leverage for those interested in creating their own annotation tools. Also, an alternate implementation of the media player for Direct X is also being undertaken to support large-format multiple-channel video (the current implementation uses JMF).

6.   Acknowledgments

7.   References

Steven Bird, David Day, John Garofolo, John Henderson, Christophe Laprun, and Mark Liberman. 2000. ATLAS: A flexible and extensible architecture for linguistic annotation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, Athens, Greece.

Susanne Burger, Victoria MacLaren, and Hua Yu. 2002. The ISL Meeting Corpus: The impact of meeting type on speech style. In *Proceedings of the 6th International Conference on Spoken Language Processing (IC-SLP 2002)*, Denver, Colorado.

Jean Carletta, Stefan Evert, Ulrich Heid, Jonathan Kilgour, Judy Robertson, and Holger Voorman. 2003. The NITE XML Toolkit: Flexible annotation for multimodal language data. *Behavior Research Methods, Instruments, and Computers*, 35(3):353–363.

Scott Farrar and Terry Langendoen. 2003. A linguistic ontology for the semantic web. *Glot International*, 7(3):97–100.

John S. Garofolo, Christophe D. Laprun, Martial Michel, Vincent M. Stanford, and Elham Tabassi. 2004. The NIST Meeting Room Pilot Corpus. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.

Alexander Gruenstein, John Niekrasz, and Matthew Purver. 2005. Meeting structure annotation: Data and tools. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, Lisbon, Portugal.

Nancy Ide and Laurent Romary. 2004. International standard for a linguistic annotation framework. *Journal of Natural Language Engineering*, 10(3–4):211–225.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The ICSI Meeting Corpus. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003)*.

Holger Knublauch, Mark A. Musen, and Alan L. Rector. 2004. Editing description logic ontologies with the Protégé OWL plugin. In *Proceedings of the 2004 International Workshop on Description Logics (DL 2004)*, Whistler, British Columbia.

Kazuaki Maeda, Steven Bird, Xiaoyi Ma, and Haejoong Lee. 2002. Creating annotation tools with the annotation graph toolkit. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, Paris, France.

I. McCowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, W. Post, D. Reidsma, and P. Wellner. 2005. The AMI Meeting Corpus. In *Proceedings of Measuring Behavior 2005, the 5th International Conference on Methods and Techniques in Behavioral Research*, Wageningen, The Netherlands.

John Niekrasz and Matthew Purver. 2006. A multimodal discourse ontology for meeting understanding. In Steve Renals and Samy Bengio, editors, *Machine Learning for Multimodal Interaction: 2nd International Workshop, MLMI 2005, Edinburgh, UK, July 11-13, 2005, Revised Selected Papers*, volume 3869 of *Lecture Notes in Computer Science*, pages 162–173. Springer-Verlag.

John Niekrasz, Matthew Purver, John Dowding, and Stanley Peters. 2005. Ontology-based discourse understanding for a persistent meeting assistant. In *Persistent Assistants: Living and Working with AI: Papers from the 2005 AAAI Spring Symposium*, Stanford, California.

Dennis Reidsma, Dennis Hofs, and Natasa Jovanović. 2005. Designing focused and efficient annotation tools. In *Proceedings of Measuring Behavior 2005, the 5th International Conference on Methods and Techniques in Behavioral Research*, Wageningen, The Netherlands.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The ICSI Meeting Recorder Dialog Act Corpus. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, Cambridge, Massachusetts.
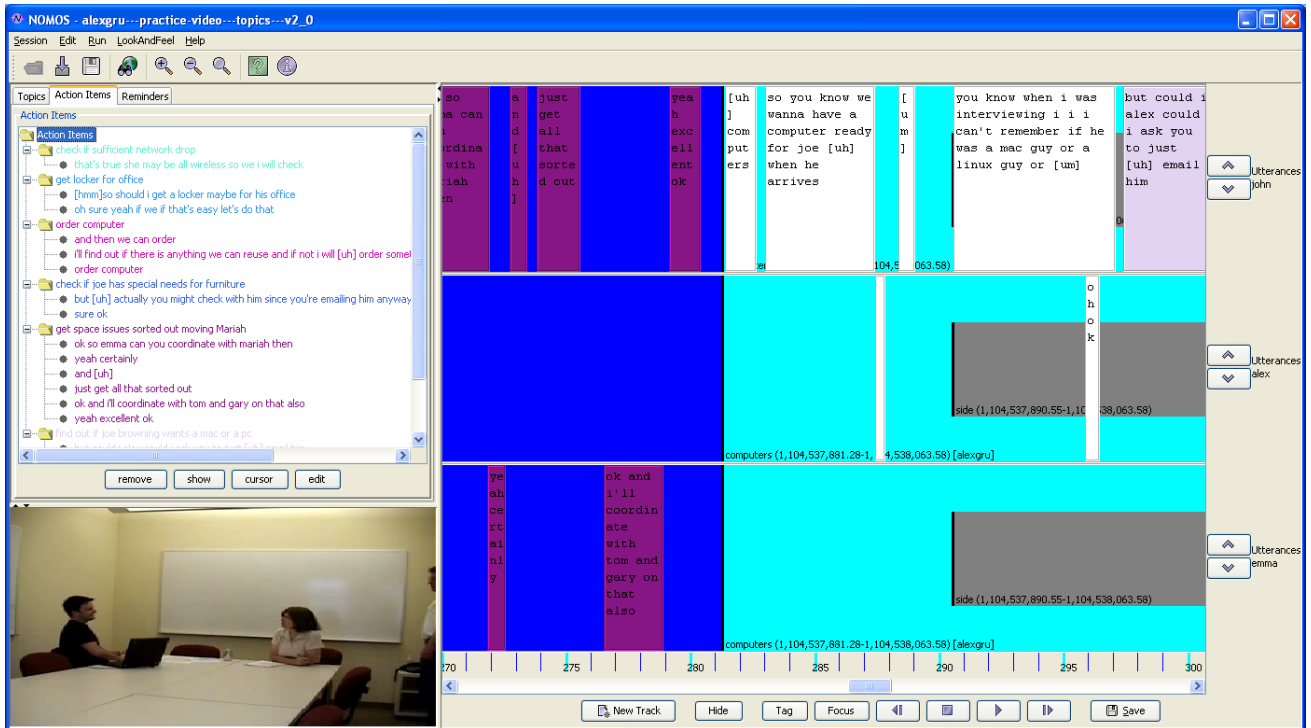
---

[7]http://godel.stanford.edu/

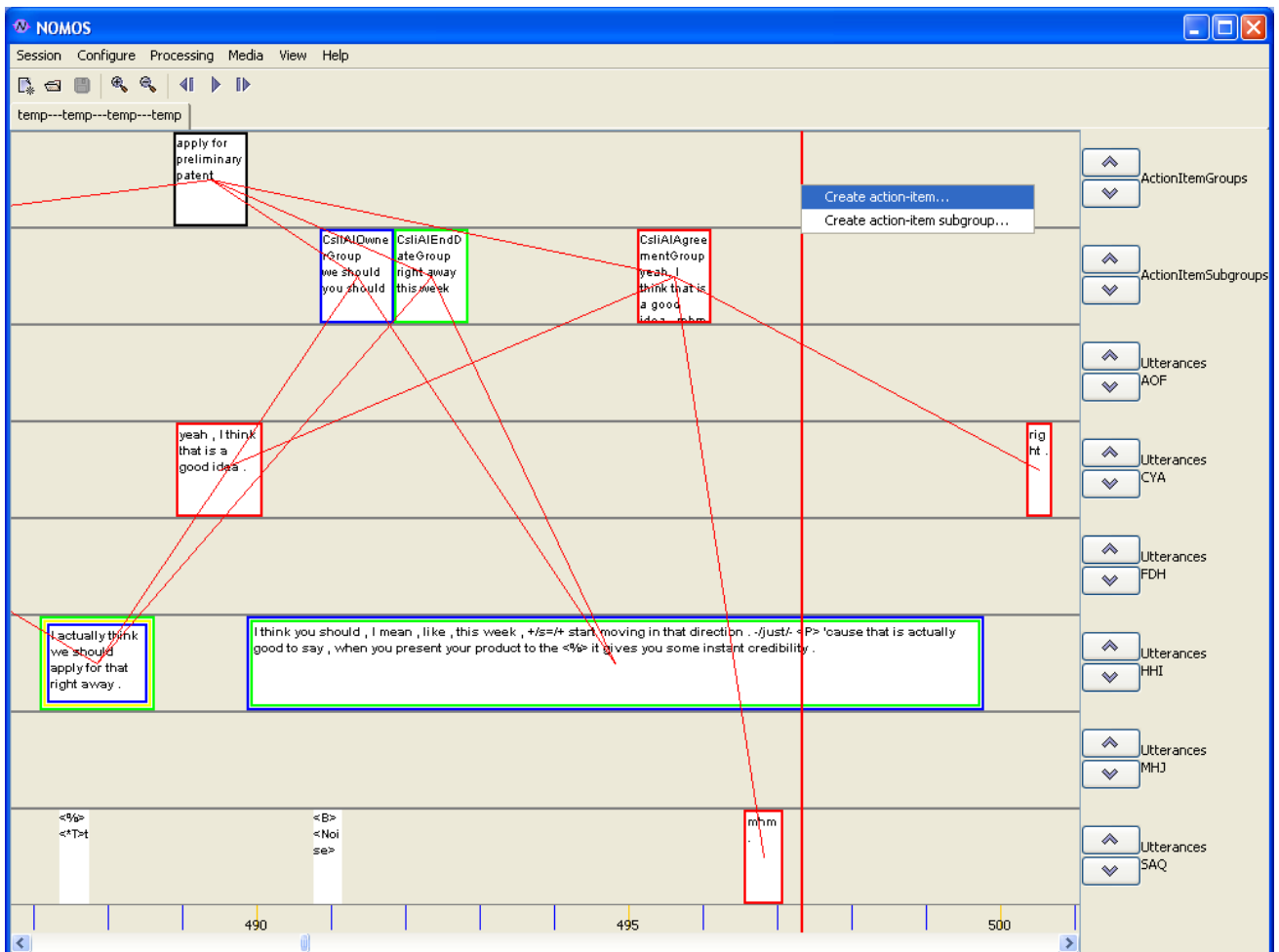Figure 3: A topic segmentation and action-item annotation tool built with NOMOS.

Figure 4: An action-item annotation tool built with NOMOS.

498