MMAX: A Tool for the Annotation of Multi-modal Corpora

Christoph Müller and **Michael Strube**
European Media Laboratory GmbH
Villa Bosch
Schloß-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
{christoph.mueller, michael.strube}@eml.villa-bosch.de

Abstract

We present a tool for the annotation of XML-encoded multi-modal language corpora. Non-hierarchical data is supported by means of stand-off annotation. We define base level and supra-base level elements and theory-independent markables for multi-modal annotation and apply them to a cospecification annotation scheme. We also describe how arbitrary annotation schemes can be represented in terms of these elements. Apart from theoretical considerations, however, the development of a fast, robust and highly usable annotation tool was a major objective of the work presented.

1 Introduction

In a corpus of multi-modal language data, each modality can be envisaged as a stream of *signals* of a particular type [Isard *et al.*, 1998]. Possible signal types are words, gestures, gazes, but also user actions like the operation of buttons, etc. Each signal type is communicated through a different channel. Different signals can and in most cases will occur simultaneously in different streams. Signals will thus be embedded into and overlap each other. The relations embedding and overlap can be defined with respect to a time line which is shared by all streams. Consider the following utterances taken from data collected within the project EMBASSI[1] which is concerned with multi-modal interfaces for consumer electronic devices (see also turns 26-28 in Figure 3):

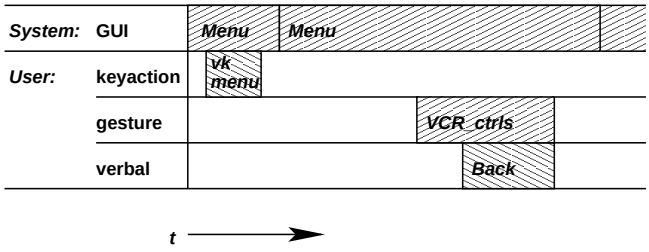

Figure 1: Signal streams

[1] http://www.embassi.de

The tool we present in this paper utilizes the signal-based approach to aid in the annotation of multi-modal corpora. This approach consists of a set of base level transcription elements, each of which can represent signals of a particular type, and two supra-base level annotations which build on the base level elements and put a structure over them. Supported supra-base level annotations are the formally defined *turns* and the pragmatically defined *utterances*. These two annotations are not mutually exclusive, but view the data from different perspectives and thus complement each other (see Figure 2).

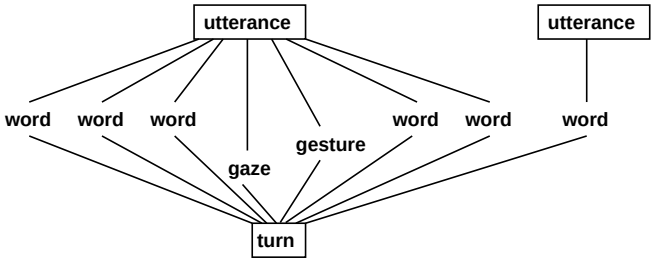

Figure 2: Supra-base level annotations

The annotation proper is represented in terms of *markables* and labelled relations between these. While our approach is in principle extendable to cover arbitrary levels of linguistic description, we apply it here to the annotation of cospecification [Sidner, 1983] in multi-modal dialogues. The tool processes non-hierarchical XML. Since non-hierarchical relations cannot be represented in XML directly, we implement them by means of so-called *standoff annotation*. In this technique, overlap between elements is resolved by explicitly linking elements via their *href* attributes, thus permitting (possibly crossing) pointers. [Isard *et al.*, 2000].

2 Motivation

The motivation for the work presented here is twofold: The growing need for the annotation of multi-modal corpora on the one hand and the lack of tools that are productively usable for this task on the other.

2.1 The importance of multi-modal annotation

Given the recent interest in multi-modal user interfaces and the growing need for robust multi-modal recognition and integration techniques, multi-modal corpora and their annotation are becoming increasingly important for several reasons.

- Manually produced annotations provide the standard against which the performance of automated systems is measured and evaluated. Therefore, the availability of large volumes of manually annotated corpora is a prerequisite to progress in this field.
- When conducting empirical research on the acceptance and ergonomics of non-verbal modes of interaction (e.g. in the context of dialogue systems), huge amounts of raw data (like audio, video, etc.) will be generated. These have to be encoded in a way that allows for efficient storage and retrieval and facilitates automated analysis.
- In the field of machine learning, learning algorithms which are to induce rules and discover correlations between signals in different modalities depend on the availability of huge amounts of (usually complex) data. This data must be easily accessible to all kinds of quantitative methods[2].

2.2 Existing tools

There already exist numerous tools for cospecification annotation. We reviewed some of these with respect to the question whether they are possibly extendable to cover input from additional modalities.

The first tool we considered is the Discourse Tagging Tool (DTTool) [Aone and Bennett, 1994], a Tcl/Tk tool for tagging multilingual texts with antecedent-anaphor relations and view them graphically. The tool accepts as input SGML encoded texts, into which cospecification tags are inserted directly during annotation. This approach to annotation (which is quite common, cf. below) is a drawback because it mixes the basic data and the annotation proper and thus is not optimal in cases where e.g. alternative annotations of the same data are to be compared. Since the tool accepts one text file only at a time, data from additional modalities would have to be included into this text file. This we wanted to avoid for reasons of data maintenance. Finally, DTTool does not provide an easy way of extending the set of cospecification types, which is a necessary precondition to cover cospecification phenomena in multi-modal corpora.

The Alembic Workbench[3] is an annotation tool which, among other tasks, directly supports cospecification annotation. In contrast to DTTool, it also allows for the extension of the tag set, so that in principle the handling of additional coreference phenomena is possible. The tool processes SGML files. Again, however, distributing data from different modalities into different files and merging these for annotation and display is not possible.

DAT[4] is a Perl/Tk tool for dialogue act tagging which processes files in SGML format. It directly supports dialogue structures (turns and utterances) and would thus in theory be amenable to the inclusion of cospecification data from different modalities. According to its original application, however, the smallest unit that can be tagged in this tool is a single utterance. In addition, tags and possible attributes are hardcoded in the tool and thus cannot be easily extended or modified.

CLinkA[5] [Orăsan, 2000] is a more recent Java tool for the annotation of coreference chains. Although the set of tag attributes appears to be extendable by the user, only a single file can be used as input, which rules out this tool for our application as well.

Among the tools that we considered for the task of multimodal annotation, the MATE Workbench[6] is the most ambitious one. We discuss it in more detail, since it did have a strong influence on the work presented here. The MATE Workbench is an open source tool (written in Java) for the display and annotation of XML-encoded linguistic corpora. It has been developed as a highly customizable tool for the parallel annotation of arbitrary and possibly non-hierarchical levels of linguistic description. In order to model non-hierarchical and overlapping relations within XML, the concept of standoff annotation [Thompson and McKelvie, 1997] has been applied and extended. We believe that this is one of the major theoretical achievements of the MATE project, and our own work is strongly indebted to it. On the practical side, however, the performance of the MATE Workbench leaves a lot to be desired.

Consider the following example, which is indicative of possible reasons for the performance problems. [Isard *et al.*, 2000]: Within MATE, user interaction with objects on the display (i.e. words, in most cases) is realized by means of event handlers which, when associated with a particular object, execute some code as a reaction to events like mouse clicks. This code is written in a special declarative action language which is interpreted by the workbench every time a particular event occurs, thus causing rather long response times. In principle, the same is true for the stylesheet processor: Since it is not only used initially to transform a given set of data files into objects for display, but also for refreshing the display during annotation, poor performance on its part is also a serious problem. We found the workbench to be practically unsuable as soon as a certain stylesheet complexity and a certain real-world corpus size was reached. Apparently, it is the striving for flexibility and customizability which can be identified as the cause of this problem, since it was with these features in mind that the action language and the powerful but very demanding stylesheet processor were devised. We believe that while these features are in principle very desirable in an annotation tool, they must under no circumstances be at the expense of performance. We argue that speed is in fact a

[2] This is true not only for machine learning algorithms: Algorithms for computing inter-annotator agreement like the kappa statistic [Carletta, 1996] also profit from this kind of data format. We provide an add-on for our annotation tool which computes the kappa statistic directly from the annotation XML files.

[3] http://www.mitre.org/technology/alembic-workbench/

[4] http://www.cs.rochester.edu/research/cisd/resources/damsl/

[5] http://www.wlv.ac.uk/sles/compling/software/

[6] http://mate.nis.sdu.dk

limiting factor here, which means that poor performance (experienced by the user as long response times) will make it a poor tool, no matter what it could have achieved if the user had only been more patient. We conclude from this that for our own tool limiting the range of application is required.

3 The signal model

Within our tool, language data is described in terms of signals. Different signal types as well as structural units above the signal level are defined in XML document type definitions (DTDs) which stipulate which structure a well-formed element of a particular type must have.

In the following, we give DTD fragments and discuss their semantics.

3.1 Base level elements

Each signal occurring in a dialogue is an instantiation of a particular signal type which defines the attributes relevant to this type. Attributes relevant to every signal type are

- the *ID* attribute, which uniquely identifies the signal and
- the *starttime* and *endtime* attributes, which specify he temporal extension of the signal.

At present, the following signal types are supported. Additional types (like e.g. gazes, which can function like pointing gestures) can be added.

Word signals. A word is the most straightforward type of signal in this context. It is the orthographic transcription of an originally spoken word from a dialogue.

```
<!ELEMENT words (word*)>
<!ELEMENT word (#PCDATA)>
<!ATTLIST word id ID #REQUIRED>
<!ATTLIST word starttime CDATA #IMPLIED>
<!ATTLIST word endtime CDATA #IMPLIED>
```

Gesture signals. Gestures, except in cases where they represent commands, can be identified with the object or objects they specify. Therefore, for the time being, we do not distinguish in our approach between different types like pointing or moving gestures. Rather, we supply with each gesture a textual representation of the object(s) it specifies.

```
<!ELEMENT gestures (gesture*)>
<!ELEMENT gesture EMPTY>
<!ATTLIST gesture id ID #REQUIRED>
<!ATTLIST gesture starttime CDATA #IMPLIED>
<!ATTLIST gesture endtime CDATA #IMPLIED>
<!ATTLIST gesture specifies CDATA #REQUIRED>
```

Keyaction signals. This signal type is supplied in order to handle signals like the operation of buttons (e.g. on a remote control) which can, e.g. in the context of a human-machine dialogue, constitute signals relevant to the particular communication. Keyaction signals are specified with respect to the key that was operated and to the kind of action that was performed on it. Note that the list of possible values for the *action* attribute is not complete and given for illustrative purposes only: Depending on the kind of control devices available (e.g. sliders), additional actions will have to be added.

```
<!ELEMENT keyactions (keyaction*)>
<!ELEMENT keyaction EMPTY>
<!ATTLIST keyaction id ID #REQUIRED>
<!ATTLIST keyaction starttime CDATA #IMPLIED>
<!ATTLIST keyaction endtime CDATA #IMPLIED>
<!ATTLIST keyaction key CDATA #REQUIRED>
<!ATTLIST keyaction action (press) #REQUIRED>
```

3.2 Supra-base level annotation elements

The sequence of signals which as a whole constitute a complete dialogue can be divided with respect to two criteria that are possibly crossing each other, i.e. a formal and a pragmatic one: Each signal is at the same time part of a particular (formally defined) turn and a particular (pragmatically defined) utterance. Moreover, utterances can be discontinuous, e.g. in cases where one speaker interrupts the other and the latter resumes his utterance later in a new turn. The two elements *turn* and *utterance* have the following attributes:

- the *ID* attribute and
- a *span*[7] attribute which is a pointer to a signal or a sequence of signals.

Turns. On the formal side, a dialogue can be divided into turns, a turn-break being marked by a change of speaker. Accordingly, each turn has a *speaker* attribute specifying which speaker uttered the turn in question. In addition, a running number is introduced for easy reference to single turns.

```
<!ELEMENT turns (turn*)>
<!ELEMENT turn EMPTY>
<!ATTLIST turn id ID #REQUIRED>
<!ATTLIST turn speaker CDATA #REQUIRED>
<!ATTLIST turn number CDATA #REQUIRED>
<!ATTLIST turn span CDATA #REQUIRED>
```

Utterances. On the pragmatic side, a dialogue can be divided into single utterances on grounds of their content or function in the discourse. With each utterance, a particular dialogue act may be accomplished.

```
<!ELEMENT utterances (utterance*)>
<!ELEMENT utterance EMPTY>
<!ATTLIST utterance id ID #REQUIRED>
<!ATTLIST utterance dialogue_act CDATA
          #IMPLIED>
<!ATTLIST utterance span CDATA #REQUIRED>
```

4 Annotation by means of *markables*

It is the aim of multi-modal annotation to make explicit the correlations and interdependencies that exist between the separate signals occurring in each signal stream in the dialogue. To be more precise, it is not between signals, but between higher-order entities that these relations hold. In the case of cospecification, a *sequence* of words (constituting a definite noun phrase, e.g.), could be such an entity, because the phrase

[7] We use our own attribute name here because the semantics of the *span* attribute as we define it differs from the *href* attribute in XPointer.

as a whole stands in a cospecification relation to some other entity.

In order to express this necessary abstraction, the concept of *markables* is introduced. This concept is of major importance because it is in terms of markables (with attributes) and labelled relations between markables that the annotation is expressed.[8]

An annotation tool should be flexible enough to be usable within different theoretical frameworks and for different tasks. In particular, the tool must not impose any theoretical connotation on the phenomena it deals with: This should be left to the annotation scheme employed. Therefore, the way in which markables are implemented in an annotation tool is of major importance.

In the case of cospecification annotation, markables represent *discourse entities*. It is important to note, however, that *markable* is a formally defined notion only: If one is interested in e.g. dialogue act tagging, markables could be used to represent utterances. If the task is part-of-speech tagging, on the other hand, each word would be a markable. This means that the markable obtains its interpretation from the annotation scheme for which it is defined. To be exact, it is the set of attributes and their respective possible values that add meaning to the formally defined notion: For the annotation of utterances,e.g., a dialogue-act attribute is needed, which is useless for the annotation of part-of-speech on the word level.

In our approach, therefore, the XML elements representing markables possess a set of attributes which is only partly pre-defined: A closed set of fixed system attributes is complemented by an open set of user-definable attributes which depend on the annotation scheme used.

4.1 System attributes

As for the system attributes, each markable has an *ID* attribute which uniquely identifies it. In addition, a *span* attribute is needed as well which maps the markable to one or more signals. Finally, we introduce a *type* attribute the meaning of which will be described in the next subsection. Two additional system attributes serve to express the relations between markables. We argue that two basic relations are sufficient here.

The first is an unlabelled and unordered relation between arbitrarily many markables, which can be interpreted as set-membership, i.e. markables standing in this relation to each other are interpreted as constituting a set. Note that the interpretation of this relation is not pre-defined and needs to be specified within the annotation scheme. In order to express a markable's membership in a certain set, a *member* attribute is introduced which has as its value some string specification. Set membership can thus be established/checked by unifying/comparing the *member* attribute values of two or more markables.

The second relation is a labelled and ordered relation between two markables, which is interpreted as one markable pointing to the other. Note that here, again, the nature of this pointing is not pre-defined. However, there is a structural constraint imposed on the pointing relation which demands that each markable can point to at most one other markable. Initially, this resulted from the fact that the annotation scheme we used did not demand more than one pointing relation between markables. Easy implementation as well as a limitation of annotation complexity were additional arguments. Moreover, we believe that the constraint does not restrict the applicability and versatility of our tool too much. A *pointer* attribute is required for the expression of the pointing relation. The range of possible values for this attribute is the range of existing markables' IDs, with the exception of the current markable itself.

The DTD fragment for markables and their system attributes looks as follows:

```
<!ELEMENT markables (markable*)>
<!ATTLIST markable id ID #REQUIRED>
<!ATTLIST markable span CDATA #REQUIRED>
<!ATTLIST markable type CDATA #REQUIRED>
<!ATTLIST markable member CDATA #IMPLIED>
<!ATTLIST markable pointer IDREF #IMPLIED>
```

4.2 User-definable attributes

It is by means of its user-definable attributes that a markable obtains its semantic interpretation within an annotation scheme. But even within a single scheme, it may be required to discriminate between different types of markables. In cospecification annotation, e.g., one needs to distinguish words from gesture and keyaction signals. There may be attributes which pertain to a particular signal type only: Linguistic attributes like grammatical role or agreement obviously are meaningful only when applied to verbal signals. It would be highly desirable, therefore, to constrain their use to markables of the correct type. This would significantly simplify the annotation process and increase annotation reliability by preventing meaningless attribute combinations.

In our approach, the type attribute is introduced to serve as a discriminator for different markable types. This attribute does not have any pre-defined possible values. Instead, a list of these has to be supplied by the annotation scheme. For each of these values, in turn, a list of relevant attributes and possible values has to be defined by the user. Depending on which of the mutually exclusive type attributes is assigned to a given markable, only the attributes relevant to this type will be offered during annotation.

5 A signal-based annotation tool

In this section, we briefly describe some implementation details and design features of our annotation tool. The tool itself will be made available via our website[9] along with annotation guidelines and samples.

5.1 Implementation

The tool is written in Java (Version 1.3) because we found platform-independence important. The availability of the Apache[10] implementations of an XML parser and stylesheet

[8] In fact, it seems that *any* annotation can be expressed in this way. Cf. the notion of *annotation graphs* [Bird and Liberman, 1999].

[9] http://www.eml.villa-bosch.de/english/research/nlp/

[10] http://www.apache.org

processor (Xerces-J and Xalan-J) were additional reasons for this decision. We limited the tool's system requirements by using a text-only display control (as opposed to an HTML display). While this control offers but a fraction of the formatting possibilities of HTML, (i.e. merely font size, underlining, italicizing, bold print and different colors), we believe that these are sufficient. The display offers simple and efficient methods for directly modifying the formatting of any desired part of the displayed text. Since the calls to these methods are hard-coded in the tool, we achieve very good performance (i.e. low response time) even on a standard PC. Identification of clicked or selected words and markables is done by mapping the respective display position to element IDs via hashtables, which is is much faster than invoking generic event handlers on each element.

5.2 Internal representation

Our tool processes multi-modal corpora which consist of a set of files which adhere to their respective DTDs described above. For each corpus, there is a setup file which contains references to all files comprising the corpus.

Apart from different XML files for each type of base level and supra-base level elements as well as markables, each corpus contains an XSL style sheet and an attribute file in which user-defined attributes and their possible values (according to the annotation scheme) are specified.

When the tool processes a given corpus, an internal representation is generated first. This representation consists of an XML DOM object (i.e. a tree structure representing a hierarchical XML document) in which the data of the supplied XML files (i.e. word, gesture and keyaction signals) is merged. If utterance annotation for the corpus is supplied, this information is included into the internal representation as well. The DOM thus created is then passed to an XSL stylesheet processor, which uses the supplied XSL style sheet to transform it into a display string. In this style sheet, any information defined in the corpus (i.e. all elements and their respective attributes, like speakers, turn numbers, but also time attributes) is accessible and can be used to design the display. Information about which element underlies a certain display element is automatically inserted in the display string by means of a pre-defined style sheet template. In addition, users have at their disposal a number of simple markup tags (for underlining, italics, etc.) which can be inserted at this stage. The display string returned by the stylesheet processor is then re-parsed, and the markup and identification tags are removed, the respective information being stored in a number of hashtables. The style information is then applied to the display string. After the markables XML file has been parsed as well, the string is finally displayed.

5.3 User interaction

At this time, user interaction includes the creation and deletion of markables, the setting and modification of the user-defined attributes, the selection of markables and the annotation and display of relations between these. Once a markable is created, its attributes can be set in a separate window. The only attribute that is always present in this window is the *type* attribute. This attribute offers the possible values that have been defined in the attributes file. Depending on the value that the user selects for this attribute, the attribute window changes and contains only the attributes that are defined for this type.

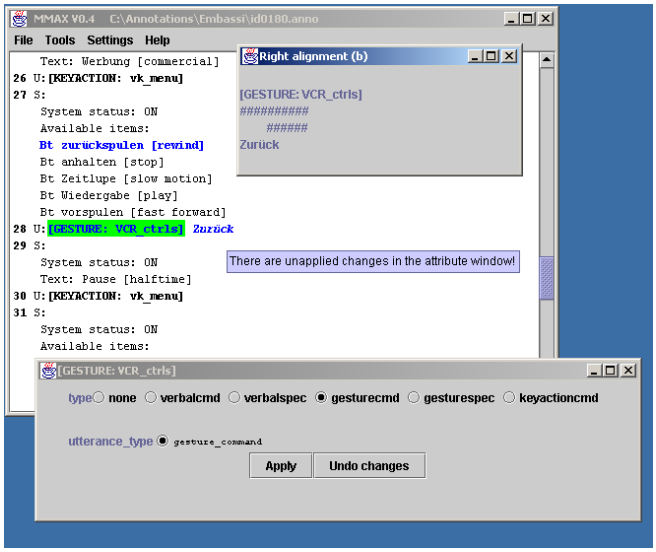

Figure 3: Screen capture of MMAX

Since markables can be embedded into each other (e.g. a pronoun in a larger noun phrase), mouse clicks can be ambiguous, and a popup menu is provided in cases like these from which a user can unambiguously select the desired markable. Pointing relations between markables as well as equivalence relations (via the *member* attribute) are annotated by selecting markables by left- resp. right-clicks and choosing the desired annotation from a popup menu. Markables pointing at each other as well as sets of markables can be highlighted in user-definable colours. Annotations can then be saved back to XML files for further processing.

6 Cospecification annotation

In this section, we will demonstrate how an annotation scheme for cospecification can be formulated and applied with our tool. Cospecification is a relation that holds between two or more discourse entities that specify (i.e. are used to refer to) [Sidner, 1983] the same entity. The member attribute is used to represent cospecification, i.e. cospecifying markables share the same value in this attribute. This value can be interpreted as what has been called *universe entity* elsewhere, e.g. in the coreference section of the Mate Dialogue Annotation Guidelines[11]. Within a set of cospecifying markables, pairs can be identified in which one member is the *antecedent* and the other one an *anaphor* relating back to it. This "relating back" (or forward, in which case the expression is a *cataphor*), is easily represented by means of the markable's pointer attribute. While the relation of antecedence is obviously linked in some way to the concept of cospecification (a pair of antecedent and anaphor is always cospecifying), we

[11] http://www.ims.uni-stuttgart.de/projekte/mate/mdag/

decided to keep the two annotations independent. We did so for the following reasons:

- Given that determining the antecedent of a discourse entity can contain a considerable amount of interpretation on the part of the annotator, it is not clear if the antecedence relation can be reliably annotated at all.
- Common algorithms for evaluating cospecification annotations [Vilain *et al.*, 1995] do not depend on antecedence annotation, but treat sets of cospecifying discourse entities as equivalence classes.
- Finally, with regard to multi-modal corpora, it is not yet clear how (if at all) the concept of antecedence is to be applied to discourse entities that co-occur in different modalities. More research is needed here in order to describe phenonema like cross-modal anaphors.

The scheme described here represents a first attempt to the classification of discourse entities in multi-modal dialogues. It should be clear at this point that our tool does support this kind of rapid prototyping of annotation schemes very well by means of the user-definable attributes. Markables are distinguished along two lines: First, according to the type of signal that they consist of (i.e. word, gesture, keyaction), and second according to whether they represent a specification or a command. Specifications are those that can be used to specify a certain entity. Both words and gestures can function as specifications. Commands, on the other hand, do not specify any object. They can be realised by any of the three (and possibly even more) signal types. The following example serves to illustrate the above definition: If one considers a pointing gesture to a TV set and the accompanying verbal utterrance *switch on*, the gesture is a specification, while the verbal utterance is a command. A pointing gesture to a *control* on the TV set, on the other hand, corresponds to a command, equivalent to the physical operation of the control. This differentiation leaves us with five types of markables (see the attribute window in Figure 3), all of which (we believe), can take part in cospecification relations. For each type, a set of different attributes and possible values has been defined. At this time, however, only the verbal specification type has any non-trivial attributes (like grammatical role), because it is not yet clear which attributes may be relevant for the other types.

7 Conclusions and future work

This paper described a tool for the annotation of multi-modal corpora. We argued that in cases where there is a tradeoff between the performance of an annotation tool on the one hand and high customizability and suitability for a wide range of applications on the other, the former must be given preference. Otherwise, the tool can neither be practically used nor evaluated. In our tool, annotations are represented in terms of a small set of well-defined relations between markables. This permits an efficient implementation while at the same time not imposing any theoretical bias on the annotation. In particular, we showed how different annotation schemes can be supported in a single tool by allowing the user to define arbitrary attributes. Finally, we demonstrated that a relevant real-world annotation scheme for cospecification annotation can be formulated within our approach. Future work will include: The tool will be evaluated in the course of the large-scale annotation of a multi-modal corpus. The annotations thus produced will be used as training data for a machine learning approach to automatic cospecification resolution.

Acknowledgements. We would like to thank Lutz Wind for giving feedback on previous versions of MMAX. The work presented here has been partially funded by the German Ministry of Research and Technology under grant 01 IL 904 D/2 (EMBASSI) and by the Klaus Tschira Foundation.

References

[Aone and Bennett, 1994] Chinatsu Aone and Scott W. Bennett. Discourse tagging tool and discourse-tagged multilingual corpora. In *Proceedings of the International Workshop on Sharable Natural Language Resources (SNLR)*, Ikoma, Nara, Japan, 10–11 August, 1994, pages 71–77.

[Bird and Liberman, 1999] Steven Bird and Mark Liberman. Annotation graphs as a framework for multidimensional linguistic data analysis. In *Proceedings of the ACL '99 Workshop Towards Standards and Tools for Discourse Tagging*, College Park, Md., 21 June, 1999, pages 1–10.

[Carletta, 1996] Jean Carletta. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.

[Isard *et al.*, 1998] Amy Isard, David McKelvie, and Henry S. Thompson. Towards a minimal standard for dialogue transcripts: A new SGML architecture for the HCRC Map Task corpus. In *Proceedings of the 5th International Conference on Spoken Language Processing*, Sydney, Australia, December 1998.

[Isard *et al.*, 2000] Amy Isard, David McKelvie, Andreas Mengel, and Morten Baun Møller. The MATE workbench annotation tool, a technical description. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece, May, 2000.

[Orăsan, 2000] Constantin Orăsan. ClinkA a coreferential links annotator. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece, May, 2000, pages 491–496.

[Sidner, 1983] Candace L. Sidner. Focusing in the comprehension of definite anaphora. In M. Brady and R.C. Berwick, editors, *Computational Models of Discourse*, pages 267–330. Cambridge, Mass.: MIT Press, 1983.

[Thompson and McKelvie, 1997] Henry S. Thompson and David McKelvie. Hyperlink semantics for standoff markup of read-only documents. In *Proceedings of SGML Europe '97*, Barcelona, Spain, May 1997.

[Vilain *et al.*, 1995] Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings fo the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, Cal., 1995. Morgan Kaufmann.