Answering natural language queries on spoken dialogs in meeting discussions

Hatem Ghorbel, Marita Ailomaa and Martin Rajman

Centre for Global Computing (CGC)
School of Computer and Communication Sciences EPFL
In Ecublens, 1015, Switzerland

Abstract. In this article, we focus on the analysis of human-human multimodal dialogue and aim at structuring human discussions in meetings in order to develop a computational environment to answer user natural language queries related to the content of these discussions. We first define an annotation schema that describes from an argumentative point of view how segments of human discussions are coherently organized and related to each other to expose the content and the dynamics of the dialogue. Second, we describe a rule-based computational environment to semantically analyze natural language queries in the restricted domain of the meetings argumentative description. Finally, we show results of excerpts of selected queries and we discuss the entailed issues.

1 Introduction

Interaction through meetings is among the richest human communication activities. Multimodal multi-party dialogs can be audio-visually recorded and stored in a multimedia repository. Recording meetings therefore implies the storage and the structuring of a large set of heterogeneous information scattered over time and media. The raw data format from the various recording devices is not directly usable for the creation of indexes, or for the content-based access to the relevant parts of the meetings. The data needs to be then analyzed and annotated in order to provide thematic access to the meeting recordings and support for the process of querying for relevant information.

In order to access to the content of a recorded meeting so as to find out needed information, user may want to write a natural language query (or he may interact with vocal modality, in all the cases a textual lattice of words is obtained). In such a situation, we first need to linguistically analyze the user's input, second we need to produce a semantic representation that aims at interpreting the query within a restricted domain ontology and thenceforward mapping it to structured queries within the database.

In the framework of the Interactive Multimodal Information Management (IM2)[1] project, we aim at developing a robust computational dialogue model

[1] The National Center of Competence in Research (NCCR) on Interactive Multimodal Information Management, in brief IM2, <http:///www.im2.ch/>, is aimed at the advancement of research, and the development of prototypes, in the field of man-machine interaction.

which should be exploited for the representation and access to the information contained in discussions occurring during meetings. In this paper, we propose an annotation schema that allows to elicit the argumentative structure of meeting discussions. The goal of such an annotation is to enrich the data model of meetings by means of higher-level description which provide a restricted semantic framework for rule-based user query analysis. The output of the query analysis module is a set of semantic constraints which should be translated afterwards to SQL queries.

The paper is organized as follows: in the second section, we describe the current annotation levels applied to the audio recording and we argue for the need of an argumentative description that ensures higher efficiency in the process of query answering. In the third and forth section, we introduce and detail the proposed argumentative annotation schema for meetings. In the fifth section, we detail the rule-based natural language query analysis module and we provide results of analyzed example queries. We finally discuss the entailed issues.

2 Meeting annotation

After the transcription[2] of the audio recordings, the first level of annotation concerns the description of various meta-data such as date, location, participants, exchanged documents, etc. The second level is the discursive annotation which structures the dialogues into utterances and assigns to each utterance a set of dialogue acts highlighting its communicative function. The set of dialogue act labels we use for this annotation is defined in the MALTUS guidelines [2] based on the ICSI-MRDA labels [1].

The resulting discursive annotation is therefore composed of a sequence of utterances described by one or more dialogue acts; however there is no structure that groups together several utterances sharing a same content. For instance, proposals related to the same issue are represented as separate utterances, which makes the extraction process quite difficult in the case an end-user wishes to visualize all proposals uttered when discussing a certain issue.

Moreover, speech acts [3] describe elementary illocutionary acts of communication without taking into account their inter-relations or contextual organization in the dialogue. Such a level of description is therefore not adequate in the perspective of meeting content querying as it does not allow to extract sequences of utterances that describe complex acts such as a disagreement on a certain issue, or a rejection of some proposed solution for an issue. Instead, we can simply extract elementary utterances describing separate acts such as proposals and rejections without being able to relate them to a specific context. Therefore, if the goal is the extraction of complex sequences of speech acts within a precise context of the discussion, we need to have a structured representation as to model the dynamics of the meetings (i.e describe how the discussion goes on) and

[2] Ideally, a speech recognition engine should generate such meeting transcriptions. This work is currently under development by several teams in IM2.

participant interaction; this is what we denote hereafter by the argumentative structure of the meeting.

3 Argumentative structure

3.1 Background

We may consider human dialogue from a structural perspective inspired from discourse theories [4, 5] where we describe the way participants take part in the discussion and argue their standpoints. Within such a perspective, several applied studies have proposed meta-descriptions of the argumentative structure according to different application domains. Delannoy [6] provided an Argumentation Mark-Up language consisting of a set of XML tags to pre-process monologues focusing on argumentative rhetorical relations, in order to build summaries. A simple model of an argumentative structure is the "Issue Based Information Systems" (IBIS), proposed by Kunz and Rittel [9] and adopted as a foundational theory in several computer-supported collaborative argumentation (CSCA) systems such as HERMES [8], Questmap [10], Compendium [7], AMI Project [12], and CALO [14]. IBIS captures and highlights the main lines of a discussion in terms of what *issues* have been discussed, what *alternatives* have been proposed to solve issues, and finally on what positions (accept or rejected) were taken by participants on the stated proposals. The present work stems from the first attempts on structuring meeting discussion using argumentation proposed by Pallotta and Ghorbel [20, 21].

3.2 Basis for the argumentative structure

When using the IBIS mark-up labels, a meeting is decomposed into several stages such as *issues*, *proposals*, and *positions*, each stage being possibly related to specific aggregations of elementary speech acts. Moreover, argumentative interactions may be viewed as specific parts of the discussion where several speech acts are combined to build such an interaction; as for instance, a disagreement could be seen as an aggregation of several acts of reject and accept of a same proposal. From this perspective, we elaborate the argumentative structure taking into account the different stages defined by the IBIS model and based on the concept of adjacency pairs [11] to relate these stages to each other and to the corresponding speech acts.

For example, when searching for the answers to the following query: *Was there any disagreement during the discussion on the choice of colors?* We need to find adjacency pairs (reject or accept acts related to proposal acts) within the part of the meeting dedicated to the discussion on the issue whose subject is “choice of colors”.

In short, we aim at enriching the meeting data with an additional level of description that fills the gaps of the discursive annotation and better organizes the dialogue segments in order to provide a structured environment for content querying. We present in the next section an annotation schema built upon the main elements of the IBIS model but specifically adapted to our needs[3] [13]. The final goal is to produce annotation guidelines that define a set of mark-up labels and the rules for their application. These guidelines will be the basic reference for human annotators to generate coherent annotations.

4 Meeting description schema (MDS)

In order to produce the argumentative structures of discussions, we propose an annotation schema consisting of a collection of mark-up labels and rules describing the different stages and related act sequences [13]. The labels are hierarchically organized so that upper levels highlight general descriptions and lower ones highlight more specific descriptions. The terminal labels are elementary speech acts, such as question, answer, accept, reject, etc. The non-terminals describe various stages of the discussion, such issues or complex speech acts such as propose which may embed a question and an answer acts, for instance.

4.1 Argumentative classes

We denote the (terminal or non terminal) labels as *argumentative classes*. In our approach, non-terminal classes, which are formed by complex speech acts, generally embed sub-acts that are related to them or occur within their scope. Formally, the scope of an argumentative class is defined as the time interval during which the considered dialogue segment is the focus of the discussion. During its scope, the segment could be decomposed into more detailed sub-segments to recursively form a hierarchical structure of argumentative segments. For example, if we consider the following meeting situation: a participant proposes a solution for a certain issue, then someone asks for a clarification of this proposal, and finally a clarification is provided. Although the meeting segment is characterized by three different speech acts (proposal, question, and answer), in our MDS we consider that the answer is embedded in the question which is itself embedded in the proposal. This scenario is therefore described as an argumentative class called *propose* whose scope covers the whole meeting segment and that embeds a further class *Ask_for* which itself embeds the elementary class *provide*.

4.2 Basic elements of MDS

In the MDS, the argumentative structure of a meeting is composed of an *agenda* segment (interval of time during which participants are fixing the agenda of the meeting) followed by a set of *discussion issues* segments.

An issue is generally a local problem to be discussed and solved. It is generally related to a specific topic. Participants propose alternatives, solutions,

[3] We do not claim to produce a generic argumentative annotation model, but we focus our research on the manual production of simple structures that lead to an efficient environment for answering content-related queries in meetings.

opinions, ideas, etc. in order to achieve a satisfactory solution. Meanwhile, participants either express their positions and standpoints through acts of accepting or rejecting proposals, or ask questions related to the current proposals. Hence, for each issue, there is a corresponding set of proposals (solutions, alternatives, ideas, etc.) that are composed of a certain number of related positions (for example a rejection to a proposed alternative in a discussing issue) or questions and answers.

The following rules illustrate the definition of an issue.

$Discuss(issue) := Propose(x)*, Decide$ `where`
$x \in \{issue, solution, idea, alternative, opinion\}$

$Propose(x) := Ask\_for(y)*, Accept(x)*, Reject(x)*$ `where`
$x \in \{issue, solution, idea, alternative, opinion\}$ `and`
$y \in \{explanation, justification, argument\}$

$Ask\_for(x) := Provide(x)*$

$Provide(x) := Accept(x)*, Reject(x)*$

The manual annotation is supported by an annotation tool which offers a user-friendly interface to segment the dialogue, assign argumentative class to each segment and organize their structure. The tool checks gradually the consistency of the annotated argumentative structure with the rules fixed within the MDS so as to assist the user to generate coherent annotations.

Finally, from an annotator's point of view, more general categories can be refined into sub-categories if the annotator is able to identify finer-grained episodes. This means that the annotation can be done in several stages, following either a top-down (from general to specific categories) or bottom-up strategy.

4.3 An example of argumentative annotation

Let's consider the ISSCO35[4] meeting excerpt (see table 1). In this segment the issue of the choice of the color for the furniture of a reading room was discussed. We distinguish two sub-segments:

- *Propose(issue)* : the segment where the issue is introduced for discussion.
- *Propose(idea)* : the segment containing a proposal relative to the issue under discussion. It is composed of three sub_segments: an *Accept(idea)*, a *Reject(idea)*, and an *Ask_for(clarification)* segment. The latter is itself composed of a further sub_segment that stands for a *Provide(clarification).*

[4] This is a recorded discussion about the choice of a suitable furniture for a reading room.

Table 1. Tabular representation of the annotation.

child	start_time	end_time	class	parent
1	718.912	814.112	Discuss(issue)	0
2	718.912	761.616	Propose(issue)	1
3	759.365	814.112	Propose(idea)	1
4	770.833	771.968	Accept(idea)	3
5	806.752	807.456	Reject(idea)	3
6	806.818	814.112	Ask_for (clarification)	3
7	812.368	814.112	Provide(clarification)	6

4.4 Evaluation

In the framework of the evaluation of the proposed annotation schema (MDS), two different annotations of the meeting ISSCO35 were performed separately by two different annotators. The kappa ratio [22] is then calculated to measure the degree of agreement of the annotators. The agreement on a class is counted positive when the annotators agree on both of the class name and the parent class name, we didn't however check for the whole structure to simplify the evaluation process. The kappa-ratio has reflected satisfactory agreement on the lower level of the structure i.e. the elementary classes of the model, however weak agreement on the upper level of the structure in particular on the *issue* class. Table 2 summarizes the kappa ratio for the attribution of the most relevant argumentative classes.

The main disagreement between annotators was on the level of the *issue* class. The nature of the disagreement was basically related to the segmentation problem: an issue, for a first annotator, may be seen as a sequence of issues (or sub-issues) by a second annotator. This may result in some ambiguous cases where the *propose(issue)* class is confused with the *propose(solution)* class. A similar agreement problem is seen on the level of the class *Ask_for*; one of the annotators forces the existing of such a class in the dialogue (through a brief time interval) in order to induce a *provide* class whereas the other builds the rest of the annotation assuming implicit *Ask_for*. These problems could be fixed once the annotation guidelines are more detailed.

5 Querying the meeting records

Suppose that someone did not attend the ISSCO35 meeting, but needs information about what happened during that meeting. In this situation the user might want to make queries, in natural language, about the meeting participants, the discussed issues, and the proposals and the decisions made. For a system to be able to provide appropriate answers to such queries, each query needs to be analyzed correctly by a natural language understanding engine, and translated into a representation that describes a coherent structure within the argumentative annotation model [13].

Table 2. The kappa ratio calculated for the annotations of two different annotators

class	kappa-ratio
Issue	0.42
Propose	0.51
Ask_for	0.33
Provide	0.69
Accept	0.79
Reject	0.46
Decide	0.05

Several steps, such as speech recognition, syntactic analysis and semantic analysis of the input queries are required. The focus of this section is on the last step of analysiswhere we will describe how natural language queries can be processed in a model-driven fashion to produce the desired representation. We first introduce the concept of "semantic constraints", then we detail the different levels of processing in the semantic analysis module. We also show, with several examples, the precise results that are obtained when implementing these ideas.

5.1 Semantic constraints

The different levels of meeting annotation are stored in a database structured according to a set of relational schemes, what we call the *data model*. The role of the query analysis module is to produce the mapping between the natural language queries and the information that is stored in the database. The semantic analyzer receives the natural language query as a sequence of words, together with a linguistic (lexical and syntactic) analysis of the sequence, e.g. a parse tree or a logical form. The aim of the analyzer is to extract, from the linguistic representation, the relevant information for producing a domain-specific representation of the query, which can then be easily translated into an SQL-query [15]. We call this second representation *semantic constraints*.

Semantic constraints are attribute-value pairs that describe an instance of a concept in the data model or a relation between two concepts. In the data model there are for example the three concepts: 'Person', 'Argumentative segment' and 'Subject' (of a topic). A person is characterized by attributes such as name and address, an argumentative segment by the type (e.g. propose, accept or reject) and scope (parent and child segments) and a subject by one or several subject items. To give an intuition about what the semantic constraints can be, consider the query: *Were there any proposals when discussing about the choice of colours?*. The mapping between this query and the data model is the following: there was a *discussion* about a subject (or topic) which was characterized by the items *choice* or *color*. And within this discussion there was a *proposal*. Expressed as a set of semantic constraints, this results in:

```
argseg1.Class = 'discuss'
```

```
subject1.item = ‘choice’
subject1.item = ‘color’
argseg1.issue = subject1
argseg2.Class = ‘propose’
argseg2 within argseg1
```

5.2 Processing level 1: Extracting concepts

In many natural language interfaces, the mappings between sentences and domain-specific representations are done with mapping rules [16, 17]. Such rules apply on selected linguistic terms in the query and produce a domain-specific interpretation, given that certain conditions hold, e.g. that the term stands in a certain local context. Nevertheless, when faced to a complex data model, like the one for the database of annotated meeting records, the terms have not only domain-specific interpretations but also domain-specific relations with each other, so the mapping rules are required to be more sophisticated. Thus, we make the distinction between two types of rules: (1) *concept rules*, which map the linguistic terms to concepts in the data model, in the traditional fashion, and (2) *relational rules*, which produce semantic constraints describing the relations between pairs of concepts. This section focuses on concept rules.

Formally, a concept rule consists of three elements:

1. Input word: `W`
2. Lexical condition: `L(W)`
3. Semantic constraint: `sc(C,CID,A,W)`

One rule corresponds to one attribute of a concept, e.g. first name. The lexical condition L(W) verifies that word W is a possible value of the lexical type L, e.g. firstname(Susan). The semantic constraint is a tuple with four variables where C is the name of a concept, CID is a unique concept identifier (id), which is generated each time a rule applies on a word and enables to distinguish between different instances of the same concept in the query[5]. ‘A’ is the name of an attribute, and finally W is the word in the query that refers to the attribute A of concept C, for example the ‘firstname of a person is Susan’ is written as follows:

`sc(person,1,firstname,Susan)`

Some words can refer to concepts in general, without pointing to a specific attribute, for instance ‘who’, ‘people’ and ‘participant’, which all refer to the concept Person. Such words are mapped to ‘unspecified concepts’ which are tuples of three variables:

`uc(C,CID,W)`

Unspecified concepts are not restrictive constraints (they don’t narrow down the search in the database), but they are important when extracting concept relations. This will be shown briefly in section 5.3.

[5] The concept id is not related to a key in the database

Basically, the linguistic resource used by the concept rules, is a restricted lexicon that associates words to semantic classes, based on the structure of the data model and the variability in user vocabulary. Concretely, for each linguistic term that is relevant to the data model (decision, proposal, etc.), we use WordNet [18] to extract the synonyms from the different synsets containing the term, and we assign them a common semantic class.

The hyponyms of the relevant synsets, such as specific types of discussion (negotiation, argumentation, debate) also need to be included in the lexicon, however, they can not be handled in a systematic way. In fact, some of these specific terms are mapped to argumentative templates rather than argumentative classes; for instance 'argumentation' can be interpreted as the template of repeated occurrences of a the class *proposal* followed by a *rejection*.

Special rules are required for concepts like *Subject*, which cannot always be identified through an explicit lexical label (any word can be a subject item). Instead, subject items are detected through the context in which they appear. For instance, in *Who rejected Agnes proposal about a coffee machine?*, the subject item is *coffee machine*, and this is comes from the fact that *coffee machine* occurs in a phrase of type *about X*.

5.3 Processing level 2: Extracting relations between concepts

A relational rule is triggered when two concepts identified in the query have a given relation in the data model. Figure 1 shows an example of identified concepts and relations for the query.

Relational rules have the following elements:

- Input: `sc(C1,CID1,A1,W1) sc(C2,CID2,A2,W2)`
- Syntactic condition: `syntactic_relation(W1,S,W2)`
- Output: `sc(C1,CID1,R,CID2)`

The input semantic constraints have been produced during the first processing stage (one or both can be unspecified constraints). The syntactic condition verifies that the two words W1 and W2 have the syntactic relation S in the linguistic analysis of the user query. The semantic constraint, that the rule produces to describe the relations in the data model, has four variables: C1 is the name of the first concept, CID1 is its concept id, R is the name of a relation in the data model and CID2 is the concept id of the second concept that participates in the relation. For instance, in the query *Who rejected Agnes proposal?*, the segment *Agnes proposal* triggers the following rule:

- Input: `sc(person,CID2,A2,W2) sc(argseg,CID1,class,W1)`
- Syntactic condition: `syntactic_relation(W1,possessor_of,W2)`
- Output: `sc(C1,CID1,speaker,CID2)`

Depending on the type of output given by the syntactic analyzer, the roles can be explicit in the analysis (logical forms), or can be translate to a set of possible *syntactic patterns* in a parse tree. Checking the syntactic relations is necessary in order to prevent non-valid relations from being extracted, e.g. that *Agnes* is the *speaker* of the *rejection*.

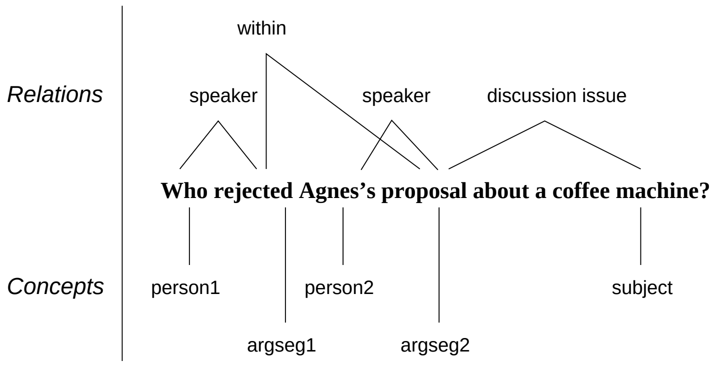

Fig. 1. Example of extracted concepts and relations

5.4 Examples of queries

A prototype of the semantic analyzer was implemented in Sicstus Prolog with 25 concept rules and 30 relational rules, verifying 11 different syntactic relations and covering the entire data model (including concepts related to meta-data annotation). The (written) user query was first parsed with a unification-based grammar and semantic lexicon in Regulus [19], specialized for the domain. The parser returned a single nested logical form of the query, which was post-processed to a flat quasi-logical form to simplify the verification of syntactic relations.

In the following, we give some examples of user queries referencing the argumentative model and the corresponding semantic constraints returned by the semantic analyzer.

Query 1. *Who rejected Agness proposal about a coffee machine?*

- *uc(person,1,who)*
- *sc(argseg,2,class,reject)*
- *sc(person,3,firstname, Agnes)*
- *sc(argseg,4,class,proposal)*
- *sc(subject,5,item,coffee_machine)*
- *sc(argseg,2,speaker,1)*
- *sc(argseg,4,speaker,3)*
- *sc(argseg,4,discussionissue,5)*

Query 2. *Were there any proposals when discussing the choice of colours?*

- *sc(argseg,1,class,proposal)*
- *sc(argseg,2,class,discuss)*
- *sc(subject,3,item,choice)*
- *sc(subject,3,item,colour)*
- *sc(argseg,1,within,2)*
- *sc(argseg,2,discussionissue,3)*

6 Discussion

The generated semantic constraints as previously showed are then translated into SQL queries to select the relevant parts of the dialogue to retrieve. The main problem encountered with the results of the SQL queries when applied to the manually annotated meeting ISSCO35 is the rate of precision of answers related to the fact that the retrieved candidate segment frequently contain utterances that are not relevant to the current query (for instance backchannels, interruptions, or embedded segments). Therefore, the retrieved answer may contain extra information that is not in the scope of the query. An additional filtering mechanism is hence needed to eliminate detailed and satellite information. For this purpose, we are currently elaborating a set of filtering rules expressed in terms of the characteristics (nuclear element position) of each class of argumentation and of the segments speech acts [5].

The query analyzer is based on a set of mapping rules that extract concepts and relations only from fragments of the linguistic query that are relevant to the data model, hence a *model-driven* approach to natural language understanding. In the current setup, the semantic analyzer receives a single logical form of the query, from which the mapping rules extract a more or less unambiguous interpretation.[6] The weakness of this approach is that the syntactic analyzer only provides analyses for queries that are fully described by the grammar. Queries that are beyond the scope of the grammar will have no analysis at all, and this is likely to occur frequently in the domain of meeting content querying. Speech recognition errors add another level of complexity to the task.

7 Conclusions

The argumentative annotation proposed in this contribution defines an additional level of description in the domain model used in our approach. The description is based on previous studies in dialogue analysis and argumentative theories, but restricted to the main elements considered as relevant to our case, more precisely to the specification resulting from the user query analysis [13].

The present work showed that the argumentative annotation is useful in answering user queries about the content of the meetings. A query analyzer was implemented, that automatically defines for each user query the right constraints that are adapted to argumentative structure. Nevertheless, a great deal of work is still needed to deal with issues such as robustness to cope with speech recognition errors and extra-grammaticality in the user query as well as the integration of automatic modules to the task of the argumentative annotation in order to aid in the process of attributing argumentative categories to dialogue segments.

[6] Some ambiguity is imposed by subject rules, which can accept any word as a topic item, but this can easily be solved with priority.

References

1. Morgan, N. et al.: Meetings about meetings: research at ICSI on speech in multiparty conversations. In Proc. of ICASSP. Hong Kong (2003)
2. Popescu-Belis, A.: Dialogue Acts: One or More Dimensions? WP 62 UNIGE (2005)
3. Searle, J. R.: Speech Acts: An essay in the philosophy of language. Cambridge University Press (1969)
4. Mann W. C.: Dialogue games: conventions of human interaction. Argumentation **2(4)**(1988) 511–532
5. Mann W. C. and Thompson S. A.: Rhetorical Structure Theory: A Theory of Text Organization. Text **8(3)** (1988) 243–281.
6. Delannoy, J-F.: Argumentation Mark-Up: A Proposal. In Proc. of ACL Workshop on Towards standards and tool for discourse tagging. Maryland (1999)
7. Selvin, A., et al. Compendium: Making Meetings into Knowledge Events, In Proc. of Knowledge Technologies. Austin TX (2001)
8. Karacapilidis, N., and Papadias, D.: Computer Supported Argumentation and Collaborative Decision Making: The HERMES system. Information Systems **26(4)** (2001) 259–277
9. Kunz, W., and Rittel, H. W. J.: Issues as elements of information systems. Working Paper 131, Universität Stuttgart, Institüt für Grundlagen der Planung (1970)
10. Conklin, J., Selvin, A., Buckingham Shum, S., and Sier-huis, M.: Facilitated Hypertext for Collective Sensemaking: 15 years on from gIBIS. In Proc. of the twelfth ACM conference on Hypertext and Hypermedia. Århus, Denmark (2001) 123–124
11. Schegloff EA. and Sacks, H.: Opening up closings. Semiotica **7(4)** (1973) 289–327
12. Reidsma, D., Rienks, R.J., and Jovanovic, N.: Meeting Modelling in the Context of Multimodal Research. In Proc. of the MLMI'04 (2004)
13. Pallotta, V., Ghorbel, G., Ballim, A., Lisowska, A., Marchand-Maillet, S.: Towards Meeting Information Systems: Meeting Knowledge Management. ICEIS **3** (2004) 464–469
14. Niekrasz, J., Purver, M., Dowding, J. and Peters, S.: Ontology-Based Discourse Understanding for a Persistent Meeting Assistant. In: Proc. of the AAAI Spring Symposium on Persistent Assistants (2005)
15. Stratica, N., Kosseim, L., Desai, B.C.: NLIDB Templates for Semantic Parsing. In: Proc. of Applications of Natural Language to Databases, NLDB. (2003) 235–241
16. Rayner, M.: Abductive Equivalential Translation and its Application to Natural Language Database Interfacing. PhD thesis, Royal Institute of Technology/Stockholm University (1993)
17. Allen, J.F.: Natural Language Understanding. Benjamin Cummings. Second Edition (1994)
18. Fellbaum C. (ed.): WordNet: An electronic lexical database. MIT Press (1998)
19. http://sourceforge.net/projects/regulus
20. Pallotta, V. and Ghorbel, H.: Argumentative segmentation and annotation guidelines. Technical report IM2.MDM-08 (2003)
21. Pallotta, V.: A computational dialectics approach to meeting dialogues tracking and understanding. Linguistic and new professions, Giacalone Ramat A., Special Issue of Materiali Linguistici **1095.41** (2003)
22. Carletta, J.: Assessing agreement on classification tasks: the kappa statistic. Computational Lingustics **22(2)** (1996) 249–254

This article was processed using the LaTeX macro package with LLNCS style