Media Interpretation and Companion Feedback for Multimedia Annotation

Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, Sebastian Wandelt

2011

Abstract. Companion technology supports, for instance, context-specific dialogues between a user and a computational system. We explore this technology in the context of cooperative computer-aided semantic annotation of multimedia (CASAM). Reasoning-based media interpretation exploits user input and multimedia analysis results to propose high-level media annotations (called interpretations). In this paper it is shown how queries are generated that are addressed to users of the system, with the goal to exploit answers such that the number of internal interpretations is substantially reduced. Queries are associated with so-called importance values specifying the expected degree of disambiguation for interpretation alternatives.

STS Institute for Software, Technology & Systems

TUHH Hamburg University of Technology

Media Interpretation and Companion Feedback for Multimedia Annotation

Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, and Michael Wessel

Institute for Software Systems, Hamburg University of Technology
21073 Hamburg, Germany

1 Introduction

To speed up the manual multimedia annotation process, the CASAM project[1] (Computer-Aided Semantic Annotation of Multimedia) investigates the collaboration of human annotators and machine intelligence. Initially, multimedia analysis results provided by a Knowledge-Driven Multimedia Analysis component (KDMA) as well as text provided by the user of the system are sent to the Reasoning-based Media Interpretation component (RMI) in the form of description logics assertions. RMI explains the received observations in terms of a set of possible interpretations (additional assertions) based on formalized background knowledge about domain of interest. Maintaining (large) sets of assertions as interpretation alternatives is a very resource-consuming task for RMI, especially because interpretations have to be recomputed if new observations are processed. Therefore, there is an interest to disambiguate interpretation alternatives and to delete them from the so-called agenda, in particular if interpretations are considered to hold with similar probability. Thus, the idea is to obtain the extra information in a controlled dialogue with the user.

Recently, a new technology emerged accompanied with the vision of providing new dimensions to the field of human-computer interaction. The so-called *companion technology* is targeted to be adaptable, collaborative and persistent. Instead of helping a user to achieve a clearly defined goal, systems using companion technology are capable to guide the users also in more complex tasks or even in tasks that are not clearly definable at all [1]. This technology is based on multimodal interaction with the system, context-specific dialogues between the user and the system and the ability to transform information regarding the user to a symbolic representation of knowledge.

In the context of companion technology the main target of CASAM is to enable a dialogue between the system and the user in order to disambiguate interpretation alternatives. This dialogue is performed by asking *queries* to the user of the system. In order to generate queries, the system has to reason about its own reasoning processes, i.e., it has to perform so-called meta-reasoning, with the objective to provide queries to the user, such that, given the user provides answers, it expects the largest degree of disambiguation.

The query mechanism of CASAM – as other systems using companion technology – is able to solve complex tasks in cooperation with the user, in our case a video annotation task. In order to support the Human Computer Interaction (HCI) component in this way, RMI provides queries, i.e., disjunctions of assertions. Given the user responds to these queries, RMI can eliminate possible interpretations from the internal agenda. Queries displayed to a user have certain "costs" since they distract the user from the main annotation task. In order to specify the value of an answer to a particular query, each generated query is associated with a so-called importance value. Summarizing, the challenges being investigated in this paper are the following:

1. Generation of context-specific disambiguation queries.
2. Computation of expected degrees of disambiguation for queries (importance values).
3. Processing of answers to queries.

2 Preliminaries

2.1 Preliminaries on Description Logic

One of the main targets of the CASAM project is to support human annotators during their work in producing elaborate symbolic descriptions for video shots. Annotations are used for later information retrieval and require a representation language. We assume that a less expressive description logic (DL) should be applied

[1] www.casam-project.eu

Fig. 1. CASAM prototype displaying query choices

to facilitate fast computations. We decided to represent the domain knowledge with the DL $\mathcal{ALH}_f{}^-(\mathcal{D})$ (restricted attributive concept language with role hierarchies, functional roles and concrete domains). For details see [2].

In logic-based approaches, atomic representation units have to be specified. The atomic representation units are fixed using a so-called signature. A DL *signature* is a tuple $\mathcal{S} = (\mathbf{CN}, \mathbf{RN}, \mathbf{IN})$, where $\mathbf{CN} = \{A_1, ..., A_n\}$ is a set of concept names (denoting sets of domain objects) and $\mathbf{RN} = \{R_1, ..., R_m\}$ is a set of role names (denoting relations between domain objects). The signature also contains a component **IN** indicating a set of individuals (names for domain objects).

In order to relate concept names and role names to each other (terminological knowledge) and to talk about specific individuals (assertional knowledge), a knowledge base has to be specified. An $\mathcal{ALH}_f{}^-$ *knowledge base* $\Sigma_\mathcal{S} = (\mathcal{T}, \mathcal{A})$, defined with respect to a signature $\mathcal{S}$, is comprised of a terminological component $\mathcal{T}$ (called *Tbox*) and an assertional component $\mathcal{A}$ (called *Abox*). In the following we just write Σ if the signature is clear from context. A Tbox is a set of so-called *axioms*, which are restricted to the following form in $\mathcal{ALH}_f{}^-$:

(I) Subsumption	$A_1 \sqsubseteq A_2$, $R_1 \sqsubseteq R_2$
(II) Disjointness	$A_1 \sqsubseteq \neg A_2$
(III)Domain and range restrictions for roles	$\exists R.\top \sqsubseteq A$, $\top \sqsubseteq \forall R.A$
(IV)Functional restriction on roles	$\top \sqsubseteq (\leq 1\, R)$
(V) Local range restrictions for roles	$A_1 \sqsubseteq \forall R.A_2$
(VI)Definitions with value restrictions	$A \equiv A_0 \sqcap \forall R_1.A_1 \sqcap ... \sqcap \forall R_n.A_n$

With axioms of form (I), concept (role) names can be declared to be subconcepts (subroles) of each other. Axioms of form (II) denote disjointness between concepts. Axioms of type (III) introduce domain and range restrictions for roles. Axioms of the form (IV) introduce so-called *functional* restrictions on roles, and axioms of type (V) specify local range restrictions (using value restrictions, see below). With axioms of kind (VI) so-called definitions (with necessary and sufficient conditions) can be specified for concept names found on the lefthand side of the $\equiv$ sign. In the axioms, so-called *concepts* are used. Concepts are concept names or expressions of the form $\top$ (anything), $\bot$ (nothing), $\neg A$ (atomic negation), $(\leq 1\, R)$ (role functionality), $\exists R.\top$ (limited existential restriction), $\forall R.A$ (value restriction) and $(C_1 \sqcap ... \sqcap C_n)$ (concept conjunction).

Knowledge about individuals is represented in the Abox part of Σ. An Abox $\mathcal{A}$ is a set of expressions of the form $A(a)$ or $R(a, b)$ (concept assertions and role assertions, respectively) where A stands for a concept name, R stands for a role name, and a, b stand for individuals. Aboxes can also contain equality $(a = b)$ and inequality assertions $(a \neq b)$. We say that the unique name assumption (UNA) is applied, if $a \neq b$ is added for all pairs of individuals a and b.

In order to understand the notion of logical entailment, we introduce the semantics of $\mathcal{ALH}_f{}^-$. In DLs such as $\mathcal{ALH}_f{}^-$, the semantics is defined with interpretations $\mathcal{I} = (\triangle^\mathcal{I}, \cdot^\mathcal{I})$, where $\triangle^\mathcal{I}$ is a non-empty set of domain objects (called the domain of $\mathcal{I}$) and $\cdot^\mathcal{I}$ is an interpretation function which maps individuals to

objects of the domain ($a^{\mathcal{I}} \in \triangle^{\mathcal{I}}$), atomic concepts to subsets of the domain ($A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$) and roles to subsets of the cartesian product of the domain ($R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$). The interpretation of arbitrary $\mathcal{ALCH}_f{}^-$ concepts is then defined by extending $\cdot^{\mathcal{I}}$ to all $\mathcal{ALCH}_f{}^-$ concept constructors as follows:

$$\begin{array}{ll}
\top^{\mathcal{I}} & = \triangle^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = \emptyset \\
(\neg A)^{\mathcal{I}} & = \triangle^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(\leq 1\, R)^{\mathcal{I}} & = \{u \in \triangle^{\mathcal{I}} \mid (\forall v_1, v_2)\,[((u, v_1) \in R^{\mathcal{I}} \wedge (u, v_2) \in R^{\mathcal{I}}) \rightarrow v_1 = v_2] \\
(\exists R.\top)^{\mathcal{I}} & = \{u \in \triangle^{\mathcal{I}} \mid (\exists v)\,[(u, v) \in R^{\mathcal{I}}]\} \\
(\forall R.C)^{\mathcal{I}} & = \{u \in \triangle^{\mathcal{I}} \mid (\forall v)\,[(u, v) \in R^{\mathcal{I}} \rightarrow v \in C^{\mathcal{I}}]\} \\
(C_1 \sqcap ... \sqcap C_n)^{\mathcal{I}} & = C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}
\end{array}$$

In the following, the *satisfiability* condition for axioms and assertions of an $\mathcal{ALCH}_f{}^-$-knowledge base Σ in an interpretation $\mathcal{I}$ are defined. A concept inclusion $C \sqsubseteq D$ (concept definition $C \equiv D$) is satisfied in $\mathcal{I}$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (resp. $C^{\mathcal{I}} = D^{\mathcal{I}}$) and a role inclusion $R \sqsubseteq S$ (role definition $R \equiv S$), if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ (resp. $R^{\mathcal{I}} = S^{\mathcal{I}}$). Similarly, assertions $C(a)$ and $R(a, b)$ are satisfied in $\mathcal{I}$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ resp. $(a, b)^{\mathcal{I}} \in R^{\mathcal{I}}$. If an interpretation $\mathcal{I}$ satisfies all axioms of $\mathcal{T}$ resp. $\mathcal{A}$ it is called a *model* of $\mathcal{T}$ resp. $\mathcal{A}$. If it satisfies both $\mathcal{T}$ and $\mathcal{A}$ it is called a model of Σ. Finally, if there is a model of Σ (i.e., a model for $\mathcal{T}$ and $\mathcal{A}$), then Σ is called satisfiable.

We are now able to define the entailment relation $\models$. A DL knowledge base Σ *logically entails* an assertion α (symbolically $\Sigma \models \alpha$) if α is satisfied in all models of Σ. For an Abox $\mathcal{A}$, we say $\Sigma \models \mathcal{A}$ if $\Sigma \models \alpha$ for all $\alpha \in \mathcal{A}$.

2.2 Substitutions, Conjunctive Queries, and Rules

Sequences, Variable Substitutions and Transformations A *variable* is a name of the form *String* where *String* is a string of characters from {A...Z}. In the following definitions, we denote places where variables can appear with uppercase letters.

Let V be a set of variables, and let $\underline{X}, \underline{Y_1}, \ldots, \underline{Y_n}$ be sequences $\langle \ldots \rangle$ of variables from V. The notation $\underline{z}$ denotes a sequence of individuals. We consider sequences of length 1 or 2 only, if not indicated otherwise, and assume that $(\langle X \rangle)$ is to be read as (X) and $(\langle X, Y \rangle)$ is to be read as (X, Y) etc. Furthermore, we assume that sequences are automatically flattened. A function *as_set* turns a sequence into a set in the obvious way.

A *variable substitution* $\sigma = [X \leftarrow i, Y \leftarrow j, \ldots]$ is a mapping from variables to individuals mentioned in an Abox. The application of a variable substitution σ to a sequence of variables $\langle X \rangle$ or $\langle X, Y \rangle$ is defined as $\langle \sigma(X) \rangle$ or $\langle \sigma(X), \sigma(Y) \rangle$, respectively, with $\sigma(X) = i$ and $\sigma(Y) = j$. In this case, a sequence of individuals is defined. If a substitution is applied to a variable X for which there exists no mapping $X \leftarrow k$ in σ then the result is undefined. A variable for which all required mappings are defined is called *admissible* (w.r.t. the context).

Grounded Conjunctive Queries Let $\underline{X}, \underline{Y_1}, \ldots, \underline{Y_n}$ be sequences of variables, and let $Q_1, \ldots, Q_n$ denote concept or role names. A query is defined by the following syntax: $\{(\underline{X}) \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$. The sequence $\underline{X}$ may be of arbitrary length but all variables mentioned in $\underline{X}$ must also appear in at least one of the $\underline{Y_1}, \cdots, \underline{Y_n}$: $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \cdots \cup as\_set(\underline{Y_n})$.

Informally speaking, $Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$ defines a conjunction of so-called *query atoms* $Q_i(\underline{Y_i})$. The list of variables to the left of the sign | is called the *head* and the atoms to the right are called the query *body*. The variables in the head are called distinguished variables. They define the query result. The variables that appear only in the body are called non-distinguished variables and are existentially quantified. Answering a query with respect to a knowledge base Σ means finding admissible variable substitutions σ such that $\Sigma \models \{\sigma(Q_1(\underline{Y_1})), \ldots, \sigma(Q_n(\underline{Y_n}))\}$. We say that a variable substitution $\sigma = [X \leftarrow i, Y \leftarrow j, \ldots]$ introduces *bindings* $i, j, \ldots$ for variables $X, Y, \ldots$. Given all possible variable substitutions σ, the *result* of a query is defined as $\{(\sigma(\underline{X}))\}$. Note that the variable substitution σ is applied before checking whether $\Sigma \models \{Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\}$, i.e., the query is *grounded* first.

For a query $\{(?y) \mid Person(?x), hasParticipant(?y, ?x)\}$ and the Abox $\Gamma_1 = \{HighJump(ind_1), Person(ind_2), hasParticipant(ind_1,\ ind_2)\}$, the substitution $[?x \leftarrow ind_2, ?y \leftarrow ind_1]$ allows for answering the query, and defines bindings for $?y$ and $?x$.

A *boolean* query is a query with $\underline{X}$ being of length zero. If for a boolean query there exists a variable substitution σ such that $\Sigma \models \{\sigma(Q_1(\underline{Y_1})), \ldots, \sigma(Q_n(\underline{Y_n}))\}$ holds, we say that the query is answered with

true, otherwise the answer is *false*. Later on, we will have to convert query atoms into Abox assertions. This is done with the function *transform*. The function *transform* applied to a set of query atoms $\{\gamma_1, \ldots \gamma_n\}$ is defined as $\{transform(\gamma_1, \sigma), \ldots, transform(\gamma_n, \sigma)\}$ where $transform(P(\underline{X}), \sigma) := P(\sigma(\underline{X}))$.

Rules A rule r has the following form $P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$ where P, $Q_1, \ldots, Q_n$ denote concept or role names with the additional restriction (safety condition) that $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \cdots \cup as\_set(\underline{Y_n})$. Rules are used to derive new Abox assertions, and we say that a rule r is *applied* to an Abox $\mathcal{A}$. The function call $apply(\Sigma, P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}), \mathcal{A})$ returns a set of Abox assertions $\{\sigma(P(\underline{X}))\}$ if there exists an admissible variable substitution σ such that the answer to the conjunctive query

$$\{() \mid Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\}$$

is *true* with respect to $\Sigma \cup \mathcal{A}$.[2] If no such σ can be found, the result of the call to $apply(\Sigma, r, \mathcal{A})$ is the empty set. The application of a set of rules $\mathcal{R} = \{r_1, \ldots r_n\}$ to an Abox is defined as follows:

$$apply(\Sigma, \mathcal{R}, \mathcal{A}) = \bigcup_{r \in \mathcal{R}} apply(\Sigma, r, \mathcal{A})$$

The result of $forward\_chain(\Sigma, \mathcal{R}, \mathcal{A})$ is defined to be $\emptyset$ if $apply(\Sigma, \mathcal{R}, \mathcal{A}) \cup \mathcal{A} = \mathcal{A}$ holds. Otherwise the result of *forward_chain* is determined by the recursive call $apply(\Sigma, \mathcal{R}, \mathcal{A}) \cup forward\_chain(\Sigma, \mathcal{R}, \mathcal{A} \cup apply(\Sigma, \mathcal{R}, \mathcal{A}))$. For some set of rules $\mathcal{R}$ we extend the entailment relation by specifying that $(\mathcal{T}, \mathcal{A}) \models_{\mathcal{R}} \mathcal{A}_0$ iff $(\mathcal{T}, \mathcal{A} \cup forward\_chain((\mathcal{T}, \emptyset), \mathcal{R}, \mathcal{A})) \models \mathcal{A}_0$.

2.3 Probabilistic Formalism

An observation is an assertion received by the RMI component with an associated certainty value representing the degree of belief in that particular assertion. All certainty values are considered as the probability that the corresponding assertion is true. In [3] it is shown how probabilities of observation Aboxes can be computed given specific explanations for the observations and the background knowledge that is comprised of a set of rules $\mathcal{R}$, a set of weighted rules $\mathcal{WR}$, and the Tbox $\mathcal{T}$. Note that rules in $\mathcal{R}$ are either applied in a forward- or in a backward chaining way (see [3] for more details). The determination of the probability is performed based on the Markov logic formalism [4].

Henceforth, we use $P(\mathcal{A})$ as an abbreviation for the probability of an Abox with respect to the background knowledge.

3 Companion-Oriented Media Interpretation

The overall CASAM system can also be seen as an agent environment in which the three main components HCI, KDMA, and RMI act together as agents. This section describes a Media Interpretation Agent (MI_Agent) which represents the RMI module. It receives low-level analysis results from KDMA as observations and builds high-level interpretations on it. These interpretations are then communicated to HCI, in order to be displayed on the user interface, and to KDMA that takes those interpretations as input to refine its analysis processes (see also [3] for details). As a new feature, in this paper we address how the agent acquires addtional information for disambiguation purposes. If multiple interpretations are possible, the MI_Agent is interested in disambiguating between the different possibilities. This feature is supported by the ability to generate disambiguation queries that are sent out to the other agents. Responses to queries, mainly given by a human annotator through the user interface on which the queries are displayed by HCI, are then taken into account for the ongoing interpretation process. While the general functionality of the agent is described in 3.1 and the following figure, the generation of queries is explained in Section 3.2 and the way the responses are processed is shown in Section 3.3.

[2] We slightly misuse notation in assuming $(\mathcal{T}, \mathcal{A}) \cup \Delta = (\mathcal{T}, \mathcal{A} \cup \Delta)$. If $\Sigma \cup \mathcal{A}$ is inconsistent the result is well-defined but useless. It will not be used afterwards.

Function MI_Agent($\mathcal{Q}_\Gamma$,$\mathcal{Q}_\Upsilon$, *partners*, *die*, $(\mathcal{T},\mathcal{A}_0)$, $\mathcal{FR}$, $\mathcal{BR}$, $\mathcal{WR}$, k, ϵ)
Input: a queue of observations $\mathcal{Q}_\Gamma$, a queue of responses to queries $\mathcal{Q}_\Upsilon$, a set of partners *partners*, a termination function *die*(), a background knowledge base $(\mathcal{T},\mathcal{A}_0)$, a set of forward chaining rules $\mathcal{FR}$, a set of backward chaining rules $\mathcal{BR}$, a set of weighted rules $\mathcal{WR}$, a parameter k indicating the top k Aboxes on the agenda, and a threshold control parameter ϵ
Output: –

$currentI = \emptyset$, $\mathfrak{A} = \{\emptyset\}$;
$startThread(\lambda().$
repeat
$\Gamma := extractObservations(\mathcal{Q}_\Gamma)$;
$W := MAP(\Gamma, \mathcal{WR}, \mathcal{T})$;
$\Gamma' := select(W, \Gamma)$;
$\mathfrak{A}' := filter(\lambda(\mathcal{A}).consistent_\Sigma(\mathcal{A}),$
$map(\lambda(\mathcal{A}).\Gamma' \cup \mathcal{A} \cup \mathcal{A}_0 \cup$
$forwardChain(\Sigma, \mathcal{FR}, \Gamma' \cup \mathcal{A} \cup \mathcal{A}_0),$
$\{select(MAP(\Gamma' \cup \mathcal{A} \cup \mathcal{A}_0, \mathcal{WR}, \mathcal{T}),$
$\Gamma' \cup \mathcal{A} \cup \mathcal{A}_0) \mid \mathcal{A} \in \mathfrak{A}\}));$
$(\mathfrak{A}, newI, \Delta^+, \Delta^-) := interpret(\mathfrak{A}', currentI, \Gamma', (\mathcal{T}, \mathcal{A}_0),$
$\mathcal{FR}, \mathcal{BR}, \mathcal{WR} \cup \Gamma, \epsilon);$
$currentI := newI$;
$communicate(\Delta^+, \Delta^-, partners)$;
$\mathfrak{A} := manageAgenda(\mathfrak{A})$;
$Q := generateDisambiguationQuery(\mathfrak{A}, k)$;
if $Q \neq \emptyset$ **then** $ask(Q, partners)$ **end**;
until *die*();
);
$startThread(\lambda().$
repeat
$\Upsilon := extractQueryAnswer(\mathcal{Q}_\Upsilon)$;
$\mathcal{A}_a := find(\Upsilon, \mathfrak{A})$;
$update(\mathcal{A}_a)$;
$\mathfrak{A} := shift(\mathcal{A}_a, \mathfrak{A})$;
$newI := head(\mathfrak{A})$;
$(\Delta^+, \Delta^-) := AboxDiff\,(newI, currentI)$;
$currentI := newI$;
$communicate(\Delta^+, \Delta^-, partners)$;
until *die*();
);

3.1 Functionality of the Agent

The *MI_Agent* uses a set of standard functional programming patterns such as *map*, *filter*, and *zip*. Furthermore, a function *select* is defined that uses the latter two functions. For convenience reasons they are included in this document.

$$filter(f, X) = \bigcup_{x \in X} \begin{cases} \{f(x)\} & \text{unless } f(x) = false \\ \emptyset & \text{else} \end{cases}$$

The function *filter* takes as parameters a function f and a set X and returns a set consisting of the values of f applied to every element x of X.

$$zip(X, Y) = \bigcup_{x \in X, y \in Y} \{(x, y)\}$$

The function *zip* produces as output a set of tuples by taking as input two sets X and Y and pairing their successive elements.

To select elements y from an ordered set Y using a bit vector that is also represented by an ordered set X, the function *select* is defined as follows.

$$select(X,Y) := \lambda(X,Y).filter(\lambda((x,y)).\textbf{if } x \textbf{ then } y \textbf{ else } false, \\ zip(X,Y))$$

In the MI_Agent function, the current interpretation $currentI$ is initialized to empty set and the agenda $\mathfrak{A}$ to a set containing empty set. The agent performs an incremental process and uses two distinct threads for its operation, each one containing a repeat-loop. The first thread will be described below, whereas the second one, responsible for handling and processing answers to queries, will be explained in 3.3. In case the agent receives a percept result Γ, it is sent to the queue Q_Γ. In order to take the observations Γ from the queue Q_Γ, the MI_Agent calls the $extractObservations$ function. The function $MAP(\Gamma, \mathcal{WR}, \mathcal{T})$, explained in [3], determines the most probable world of observations Γ' with respect to a set of weighted rules $\mathcal{WR}$ and the Tbox $\mathcal{T}$. It returns a vector W which consists of ones and zeros assigned to indicate whether the ground atoms of the considered world are true (positive) or false (negative), respectively. The function $select(W, \Gamma)$ then selects the positive assertions in the input Abox Γ using the bit vector W as described above. The selected positive assertions are the assertions which require explanations. The *select* operation returns as output an Abox $\Gamma' \subseteq \Gamma$.

Next, iterating over all agenda entries $A \in \mathfrak{A}$ the determination of the most probable world by the MAP function and the selection of the positive assertions is carried out on $\Gamma' \cup \mathcal{A} \cup \mathcal{A}_0$. Then, a set of forward chaining rules $\mathcal{FR}$ is applied to $\Gamma' \cup \mathcal{A} \cup \mathcal{A}_0$. The generated assertions in this process are added to $\Gamma' \cup \mathcal{A} \cup \mathcal{A}_0$. In the next step, only the consistent Aboxes are chosen and the inconsistent Aboxes are removed. Afterwards, the *interpret* function is called to determine the new agenda $\mathfrak{A}$, the new interpretation Abox $newI$ and the Abox differences Δ^+ and Δ^- for additions and omissions among $currentI$ and $newI$. Afterwards, the Abox $currentI$ is assigned to $newI$ and the MI_Agent function communicates the Abox differences Δ^+ and Δ^- to $partners$. In CASAM, $partners = \{KDMA, HCI\}$ applies.

Subsequently, the *manageAgenda* function is called. It incorporates several self-protective techniques, e.g. the elimination of interpretation Aboxes from $\mathfrak{A}$, the agent can apply in order to remain operable if too many interpretation possibilities exist. This function will be described in detail in a subsequent report. The last two functions in the processing loop for observations are $generateDisambiguationQuery$ and ask. As stated above, they are used to calculate disambiguation queries and communicate them to the partners. The termination condition of the MI_Agent function is that the *die* function returns true. We assume that the function calls $extractObservations(Q_\Gamma)$ and $extractQueryAnswer(Q_\Upsilon)$ wait for the respective results to become available.

3.2 Generation of Disambiguation Queries

As stated at the beginning of this section, queries for disambiguation, or queries for short, are sent to the HCI agent in order to better control the interpretation process. Interpretations are represented by Aboxes and selecting the Abox $\mathcal{A}_i$ from $\mathfrak{A}$ with the maximum score is an essential step in the *interpret* function. In fact, there could be multiple Aboxes which satisfy this criterion or, in a slightly weakened condition, do not differ much. In this case, the generation of queries for disambiguation between preferable Aboxes is performed by the agent.
First, we syntactically define a query in Backus-Naur form as

$$\begin{aligned} Q &::= Q'\ Iv \mid Q''\ Iv \\ Q' &::= \alpha \mid \alpha\ \textbf{OR}\ Q' \\ Q'' &::= \alpha \mid \alpha\ \textbf{XOR}\ Q'' \end{aligned}$$

where α is an assertion and $Iv \in (0, 1]$ is a real value denoting a so-called *importance value* of a query. The assertions are connected by a logical operator, called *compound operator*.
Using the previously defined syntax, an example query could be

$$Q = visits(ind_1, ind_2)\ \textbf{XOR}\ visits(ind_2, ind_1)\ 0.8.$$

This query asks if either the role assertion $visits(ind_1, ind_2)$ or the role assertion $visits(ind_2, ind_1)$ applies to the interpreted content. Because the role *visits* is defined as an asymmetric role in the domain knowledge, only one of the assertions can be consistent which is why **XOR** is chosen as the logical compound operator. Based on the probabilities of the possible interpretations, we assume that the importance value for this query is computed to be 0.8. Now that we defined the query syntax, we can take a closer look at the implementation of the query generation mechanism. It is realized by the function *generateDisambiguationQuery* which is defined as follows.

Function generateDisambiguationQuery($\mathfrak{A}$, k)
Input: an agenda $\mathfrak{A}$ and a parameter k indicating the top k Aboxes on $\mathfrak{A}$
Output: a disambiguation query Q
$\mathfrak{I} := \{\mathcal{A}_i \in \mathfrak{A} \mid i = 1 \dots k\}$;
$\mathcal{D} := \emptyset$, $Q := \emptyset$;
foreach $\mathcal{A}_i \in \mathfrak{I}$ **do**
  $\mathcal{D}_{\mathcal{A}_i} := \bigcap_{1 \leq j \leq k, i \neq j} \mathcal{A}_i \setminus \mathcal{A}_j$;
  $d_{\mathcal{A}_i} := selectAssertion(\mathcal{D}_{\mathcal{A}_i})$;
  $\mathcal{D} := \mathcal{D} \cup d_{\mathcal{A}_i}$;
end
if $\mathcal{D} \neq \emptyset$ **then**
  $Op := computeLogicalCompound(\mathcal{D})$;
  $Iv := computeImportanceValue(\mathcal{D}, \mathfrak{I})$;
  $Q := concat(buildQuery(\mathcal{D}, Op), Iv)$;
end
return Q;

Function buildQuery($\mathcal{D}$, Op)
Input: a set of assertions $\mathcal{D}$ and a logical operator Op
Output: assertions concatenated with a logical operator
if $\mathcal{D} = \{d_{\mathcal{A}_i}\}$ **then**
  return $d_{\mathcal{A}_i}$;
else
  return $concat(d_{\mathcal{A}_i}, Op, buildQuery(\mathcal{D} \setminus d_{\mathcal{A}_i}, Op))$;
end

The function *generateDisambiguationQuery* takes as input an agenda $\mathfrak{A}$ of ordered interpretation Aboxes (the order is determined by a scoring function, see Chapter 2.3) and a parameter k that indicates the top k Aboxes on $\mathfrak{A}$. This is possible because the Aboxes $\mathcal{A}_i$ are ordered by a scoring function. The selected k Aboxes are those with the highest scores, denoted as $\mathfrak{I}$. Additionally, a set of difference assertions $\mathcal{D}$ is initialised to empty set. After the Aboxes were selected from $\mathfrak{A}$, each of them is processed in a foreach loop. First, the set $\mathcal{D}_{\mathcal{A}_i}$, the intersection of all Abox differences $\mathcal{A}_i \setminus \mathcal{A}_j$ between the currently chosen Abox $\mathcal{A}_i$ and all other Aboxes $\mathcal{A}_j$ is computed. Compared to all other Aboxes, these assertions are unique to $\mathcal{A}_i$.

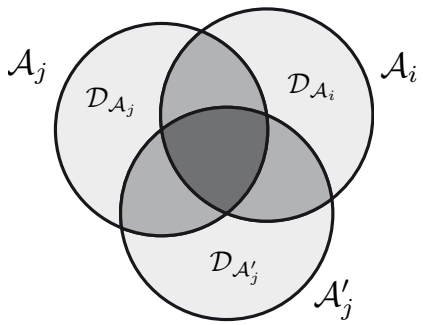

Fig. 2. Schema of Abox difference

Considering Aboxes as sets, Figure 2 shows a schematic diagram of the Abox difference operation. Aboxes are represented by light gray circles and intersections of them are marked as darker gray areas. Assuming the Abox $\mathcal{A}_i$ is the current selected Abox and the area $\mathcal{D}_{\mathcal{A}_i}$ represents the assertion that is unique to $\mathcal{A}_i$. If this set contains more than one assertion, one of them is chosen randomly, because it does not matter which assertion represents the uniqueness. This step is realized by the function *selectAssertion*. The resulting assertion $d_{\mathcal{A}_i}$ is then added to $\mathcal{D}$. After the loop terminates, $\mathcal{D}$ contains a set of difference assertions that are unique to every Abox among the top k Aboxes on the agenda $\mathfrak{A}$. By using logical conjunction, these assertions build a query that

is sent out by the agent. The logical compound Op that is told by the agent to connect the assertions is **OR** if there is no pair of assertions that is inconsistent with the background knowledge and **XOR** otherwise.

Importance Value Queries are asked to users in order to reduce the large space of abducibles to the most appropriate explanations for the observations.

In order to specify the degree of disambiguation that is expected by RMI when a particular query is answered, each generated query is associated with an importance value. In the following, a first proposal is presented to compute these values. For answering queries, the user has to invest an amount of time as well as some cognitive ressoures ("costs") such that this proposal is not based on simply considering the number of query disjuncts that are asked to the user.

In order to obtain a degree of disambiguation, queries can be favoured in which there is a pair of disjuncts $(d_{\mathcal{A}_i}, d_{\mathcal{A}_j})$ where the probabilities of the corresponding Aboxes $(\mathcal{A}_i, \mathcal{A}_j)$ are most similar. According to this, an importance value is the reciprocal of the minimum $min(|P(\mathcal{A}_i) - P(\mathcal{A}_j)|)$ of all pairs of disjuncts.

However, in addition, queries can be ranked according to the similarity of the disjuncts itself, since the more similar the disjuncts are, the more they cannot hold in parallel (i.e., according to the current scene in the video), and, following to this, the more disjuncts probably will be disambiguated by the user. For example, the concepts $Drought$ and $Pollution$ are less probable to hold in parallel than $Drought$ and $Microphone$ (being less similar) such that an answer to the former pair is believed to provide more information for disambiguation than an answer to the latter pair. An estimation for the similarity of a pair of disjuncts is the reciprocal of their distance with respect to the underlying taxonomy. For computing the taxonomical distance of a pair of disjuncts (A_i, A_j), the least common subsumer $lcs(A_i, A_j)$ [5] of these concepts is computed. Then, $dist(A_i, A_j)$ is the sum of the distances of A_i to $lcs(A_i, A_j)$ and A_j to $lcs(A_i, A_j)$ (cf. [6]). In the case that $A_i = lcs(A_i, A_j)$ resp. $A_j = lcs(A_i, A_j)$ one of the disjuncts subsumes the other such that no disambiguation is needed and $dist(A_i, A_j)$ is set to the maximal taxonomic distance of pairs of disjuncts occurring (or to the maximal depth of the underlying taxonomy, if there is no such maximum). The same holds for queries with a single disjunct, for pairs of query disjuncts that include a role, or for disjuncts that refer to the same concept, i.e., differ only in the associated individual.

Let $pairs$ be the set of all unordered pairs $(d_{\mathcal{A}_i}, d_{\mathcal{A}_j})$ of query disjuncts. An importance value for a set $\mathcal{D}$ of query disjuncts in $(0, 1]$ is defined by

$$importance(\mathcal{D}) = \frac{1}{(1+min(|P(\mathcal{A}_i)-P(\mathcal{A}_j)|))\cdot\frac{1}{2|pairs|}\sum_{pairs} dist(pairs)}$$

The minimal distance of two concepts is 2, since pairs of disjuncts subsuming each other are excluded. In order to guarantee that the maximal importance of a query is 1, the sum of distances is divided by $2\,|pairs|$.

3.3 Processing of Query Responses

In the beginning of this section the updated *MI_Agent* was presented. As mentioned before, the agent now makes use of two distinct threads. In the following we present how the agent reacts to user responses, buffered in the queue Q_Υ. Each assertion in $\mathcal{D} = \{d_{\mathcal{A}_1}, ..., d_{\mathcal{A}_i}\}$, computed by the *generateQueries* function, has also a certainty value c_i. This certainty value represents how confident the agent is about this assertion. After presenting the queries in an human readable form on the user interface, the user is able to select one or more assertions as an answer. A confirmation of an assertions results in a response where the certainty is increased to 1 and a rejection of the user, given by a non selected answer possibility, lowers the certainty value to 0.

As soon as the agent receives a response, it extracts the query answer from Q_Υ by calling the function $extractQueryAnswer(Q_\Upsilon)$. Afterwards it has to look up all the Aboxes which contain the assertions in the response. This functionality is provided by the function $find(\Upsilon, \mathfrak{A})$ which returns a set of all interpretation Aboxes $\mathfrak{I}_a = \{\mathcal{A}_a \in \mathfrak{A} \mid \upsilon \in \mathcal{A}_a \wedge \upsilon \in \Upsilon\}$ that are affected by the answer. Then these Aboxes have to be updated according to the new certainty value of these assertions given by the answer from the user. This is done by the function $update(\mathfrak{I}_a)$. Changes of the certainty value of a particular assertion can have an influence on the certainty value of structures built upon that assertion as well as on the probability of the whole Abox. Therefore, the certainty values of the assertions and the probability of the Abox has to be recomputed. If the probability of an Abox changes, also the ranking of that Abox in the agenda might have changed and the position has to be updated. This is done by the function $shift(\mathfrak{I}_a, \mathfrak{A})$ which sorts all Aboxes from $\mathfrak{I}_a$ such that $\{\mathcal{A}_1, \ldots, \mathcal{A}_i, \mathcal{A}_a, \mathcal{A}_j, \ldots, \mathcal{A}_n \mid P(\mathcal{A}_{\leq i}) \geq P(\mathcal{A}_a) \geq P(\mathcal{A}_{\geq j})\}$ holds. As a consequence of the *shift* operation the most probable Abox, ranked at the first position of the agenda, might have changed.

To handle this possibility, the eventually new most probable Abox is selected by *selectHead* and assigned to *newI*. Afterwards, the Abox-Difference (see [3]) between this Abox and the former best Abox *currentI* (and vice versa) is computed. This operation results in additions Δ^+ as well as in omissions Δ^-. The next step consists of assigning the newly found most probable Abox to *currentI*. Finally, the additions and omissions are communicated to the partners.

3.4 Complete Example

In this section, we discuss an example which shows how the disambiguation queries are generated and how the user response affects the interpretation process. Let us use a Tbox $\mathcal{T} := \{CarEntry \sqsubseteq \neg CarExit, CarEntry \sqsubseteq Movement, CarExit \sqsubseteq Movement, \ldots\}$ expressing concept disjointness and concept subsumption. Furthermore, we assume that

$\Gamma = \{1.3\ Car(c_1), 1.2\ DoorSlam(ds_1), causes(c_1, ds_1)\}$ indicates the set of observation results. Consider the set of backward chaining rules is

$\mathcal{BR} = \{$

$causes(x,y) \leftarrow CarEntry(z), Car(x), DoorSlam(y), hasObject(z,x), hasEffect(z,y)$

$causes(x,y) \leftarrow CarExit(z), Car(x), DoorSlam(y), hasObject(z,x), hasEffect(z,y)$

$causes(x,y) \leftarrow PickUp(z), Car(x), DoorSlam(y), hasObject(z,x), hasEffect(z,y),$
$Person(u), Person(w), hasParticipant(z,u), hasParticipant(z,w),$
$hasPart(z,v), House(v), owns(u,x), owns(w,v)$

$\cdots\}$

Further assume that the set of forward chaining rules $\mathcal{FR} = \emptyset$. In addition to the above mentioned sets, we require a set of weighted rules

$\mathcal{WR} = \{$

$5\ \ \forall x,y,z\ \ CarEntry(z) \wedge hasObject(z,x) \wedge hasEffect(z,y) \rightarrow$
$Car(x) \wedge DoorSlam(y) \wedge causes(x,y),$

$5\ \ \forall x,y,z\ \ CarExit(z) \wedge hasObject(z,x) \wedge hasEffect(z,y) \rightarrow$
$Car(x) \wedge DoorSlam(y) \wedge causes(x,y)$

$1\ \ \forall x,y,z,u,v,w\ \ PickUp(z) \wedge hasObject(z,x) \wedge hasEffect(z,y) \wedge$
$hasParticipant(z,u) \wedge hasParticipant(z,w) \wedge hasPart(z,v) \rightarrow$
$Car(x) \wedge DoorSlam(y) \wedge causes(x,y) \wedge Person(u) \wedge$
$Person(w) \wedge House(v) \wedge owns(u,x) \wedge owns(w,v)\}$

By applying $\mathcal{BR}$ to Γ based on the MI_Agent function, the agenda is $\mathfrak{A} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \cdots, \mathcal{A}_6\}$ which is sorted in descending order based on the scoring values of the interpretation Aboxes. Due to space constraints, only the first three interpretation Aboxes with the highest scoring values are presented here:

$\mathcal{A}_1 = \Gamma \cup \{CarEntry(ind_{42}), hasObject(ind_{42}, c_1), hasEffect(ind_{42}, ds_1)\}$

$\mathcal{A}_2 = \Gamma \cup \{CarExit(ind_{42}), hasObject(ind_{42}, c_1), hasEffect(ind_{42}, ds_1)\}$

$\mathcal{A}_3 = \Gamma \cup \{PickUp(ind_{42}), hasObject(ind_{42}, c_1), hasEffect(ind_{42}, ds_1),$
$hasParticipant(ind_{42}, ind_1), hasParticipant(ind_{42}, ind_2),$
$Person(ind_1), Person(ind_2), hasPart(ind_{42}, ind_3), owns(ind_1, c_1),$
$owns(ind_2, ind_3), House(ind_3)\}$

The above Aboxes have the following scoring values $P(\mathcal{A}_1) = 0.913$, $P(\mathcal{A}_2) = 0.908$, and $P(\mathcal{A}_3) = 0.71$. The scoring value of each interpretation Abox $P(\mathcal{A}')$ which is an abbreviation for $P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$ characterizes the probability that the conjunction of observations is true, namely:

$P(Car(c_1) = true \wedge DoorSlam(ds_1) = true \wedge causes(c_1, ds_1) = true)$

For the determination of the set $\mathfrak{I}$ in the *generateQuery* function, there is a variable k which indicates the top k interpretation Aboxes from $\mathfrak{A}$ with the maximum scoring value. Let us assume $k = 2$. Consequently, $\mathfrak{I} = \{\mathcal{A}_1, \mathcal{A}_2\}$. This means that the top two interpretation Aboxes namely, $\mathcal{A}_1$ and $\mathcal{A}_2$ are considered for the generation of the first disambiguation query. At this step the Abox differences are determined: $\mathcal{D}_{\mathcal{A}_1} = \{CarEntry(ind_{42})\}$ and $\mathcal{D}_{\mathcal{A}_2} = \{CarExit(ind_{42})\}$. Consequently, $\mathcal{D} = \mathcal{D}_{\mathcal{A}_1} \cup \mathcal{D}_{\mathcal{A}_2}$. Since the concepts $CarEntry$ and $CarExit$ are disjoint and the individual name in these assertions is the same, the appropriate operator for this query is **XOR**. The importance value of this query is $Iv = \frac{1}{(1+0.005)\times\frac{1}{2}\times 2} = 0.99$. This shows that asking this query is very important. Consequently, the first generated query is:

$$CarEntry(ind_{42})\ \mathbf{XOR}\ CarExit(ind_{42})\ 0.99$$

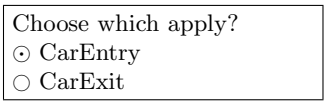

Fig. 3. The first generated query

At the user interface the query is shown as indicated in Figure 3.

Let us assume the user response to this query is $CarEntry$, or, to be more precise, $CarEntry(ind_{42})$ and $\neg CarExit(ind_{42})$. At this step, we have:

$\mathcal{A}_2 = \Gamma \cup \{\neg CarExit(ind_{42}), hasObject(ind_{42}, c_1), hasEffect(ind_{42}, ds_1)\}$

The above change reduces the scoring value to $P(\mathcal{A}_2) = 0.7$. The next step is sorting the interpretation Aboxes of the agenda $\mathfrak{A}$ based on the new scoring values. Let us assume $P(\mathcal{A}_2) > P(\mathcal{A}_4)$. Consequently, the new order is $\mathfrak{A} = \{\mathcal{A}_1, \mathcal{A}_3, \mathcal{A}_2, \mathcal{A}_4, ...\}$.

The top two interpretation Aboxes at this step are in the following set $\mathfrak{I} = \{\mathcal{A}_1, \mathcal{A}_3\}$. In order to generate the next query, the Abox differences are calculated as follows:

$\mathcal{D}_{\mathcal{A}_1} = \{CarEntry(ind_{42})\}$

$\mathcal{D}_{\mathcal{A}_3} = \{PickUp(ind_{42}), hasParticipant(ind_{42}, ind_1), Person(ind_1),$
$hasParticipant(ind_{42}, ind_2), Person(ind_2), hasPart(ind_{42}, ind_3),$
$owns(ind_1, c_1), owns(ind_2, ind_3), House(ind_3)\}$

At this step, $d_{\mathcal{A}_3}$ has to be selected from $\mathcal{D}_{\mathcal{A}_3}$. Since we would like to have a query with a high importance value, we prefer to select concept assertions from $\mathcal{D}_{\mathcal{A}_3}$. The following set contains only the concept assertion pairs. $\mathcal{D}'_{\mathcal{A}_3} = \{PickUp(ind_{42}), Person(ind_1), Person(ind_2), House(ind_3)\}$. There are four possible pairs namely,

$\{(CarEntry(ind_{42}), PickUp(ind_{42})), (CarEntry(ind_{42}), Person(ind_1)),$
$(CarEntry(ind_{42}), Person(ind_2)), (CarEntry(ind_{42}), House(ind_3))\}$

Additionally, in order to have a high importance value for the query, we have to search for the concept assertion pair with minimum distance.

$(CarEntry(ind_{42}), PickUp(ind_{42}))$ is the pair with the minimum distance. The importance value of the new query is calculated as follows: $Iv = \frac{1}{(1+0.203)\times\frac{1}{2}\times 2} = 0.83$. The second disambiguation query is as follows:

$$CarEntry(ind_{42})\ \textbf{XOR}\ PickUp(ind_{42})\quad 0.83$$

4 Summary and Remarks

The query mechanism of CASAM can be seen as companion technology in the context of multimedia annotation. The RMI component of CASAM is able to provide queries as a prerequisite for communication with users. The objective of this communication is to disambiguate interpretation alternatives. The three challenges posed in the introduction are handled as follows:

Generation of Queries. An enhanced version of the MI_Agent, originally introduced in [3], is presented. It is shown how disambiguation queries are generated. We have defined two different query types, namely **OR** and **XOR** queries.

Computation of Importance Values for Queries. The general idea behind this score is to prefer queries in which there is a pair of disjuncts where the probabilities of the corresponding Aboxes are most similar. The score is also based on the computation of a taxonomical distance.

Processing of Query Answers. Responses of users to queries have the objective to disambiguate different interpretation possibilities. It is explained which influence the answers have on the agenda of the MI_Agent.

References

1. Webb, N., Benyon, D., Hansen, P., Mival, O.: Evaluating human-machine conversation for appropriateness. In: Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, European Language Resources Association (ELRA) (2010)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
3. Gries, O., Möller, R., Nafissi, A., Rosenfeld, M., Sokolski, K., Wessel, M.: A probabilistic abduction engine for media interpretation. In Alferes, J., Hitzler, P., Lukasiewicz, T., eds.: Proc. of the 4th International Conference on Web Reasoning and Rule Systems. (2010)
4. Domingos, P., Richardson, M.: Markov logic: A unifying framework for statistical relational learning. In Getoor, L., Taskar, B., eds.: Introduction to Statistical Relational Learning, pp. 339–371. Cambridge, MA: MIT Press (2007)
5. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: Proc. of AAAI. (1992)
6. Wolter, K., Smialek, M., Hotz, L., Knab, S., Bojarski, J., Nowakowski, W.: Mapping MOF-based requirements representations to ontologies for software reuse. In: Proc of the 2nd International Workshop on Transformation and weaving ontologies in model driven engineering. (2009)