Probabilistic Dialogue Models for Dynamic Ontology Mapping

Paolo Besana and Dave Robertson

Centre for Intelligent Systems and their Applications
University of Edinburgh

Uncertainty Reasoning for the Semantic Web Workshop 2006
Athens (GA), 5th November 2006

Introduction

- Agents communicate to perform tasks that they cannot accomplish alone.
- To communicate means to exchange messages that convey meanings encoded into signs for transmission.
- To understand a message, a receiver should be able to map the signs in the messages to meanings aligned with those intended by the transmitter.
- Agents should agree on the terminology used to describe the domain of the interaction:
  - ontologies specify the terminology
- Having a shared ontology can be a strong assumption in an open environment:
  - agents may come from different backgrounds, and have different ontologies, designed for their specific needs

Introduction

- In this sort of environment, communication implies ontology mapping.
- In an open environment, it is impossible to know which agents will take part in the interactions.
- Agents have to map ontologies dynamically when needed.
- However, agents may meet infrequently and only for interactions on specific topics.
- A full ontology mapping would be a waste of resources:
    - only the terms that are needed for the interaction should be mapped

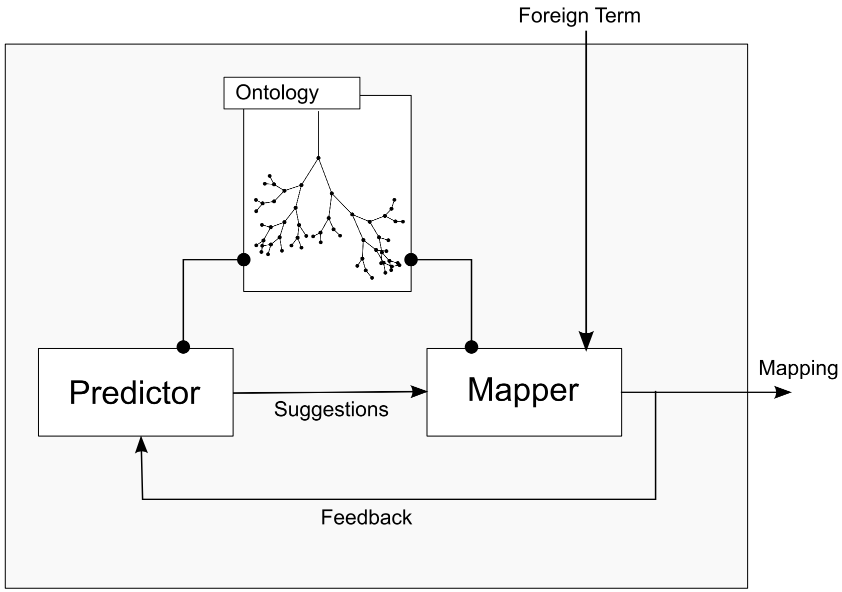
Ontology Mapping and interactions

- Usually, a term $t_j \in O_1$ is mapped to another term $w_i \in O_2$ comparing the term t_j with all the terms in O_2
- However, during an interaction it is often possible to avoid all these comparisons.
- A term in a received message is unlikely to refer to an entity completely unrelated with the context of the dialogue.
- Intuitively, the type of interaction, the specific topic and the messages already exchanged bind the content of a message to a set of possible expected entities.

Probability and Uncertainty in Ontology Mapping

- Probabilities are often used in ontologies to express the uncertainty of static relation between entities
- In this work, probability is used to *predict* the content of received messages in specific interactions.
- The repetition of similar interactions provide the information needed to compute the probability distribution of the entities.
- The predictions are used as *suggestions* for an Ontology Mapping system that must match foreign terms with local ones.
- The use of suggestions:
  - Reduce the number of comparisons to perform, improving its efficiency
  - Reduce the ambiguities, excluding terms that are unrelated to the interaction.

General Structure



- The *predictor* learns the probabilities of terms received as feedback from the Mapper
- The predictor provides suggestions to the Mapper

Predicting the terms in messages

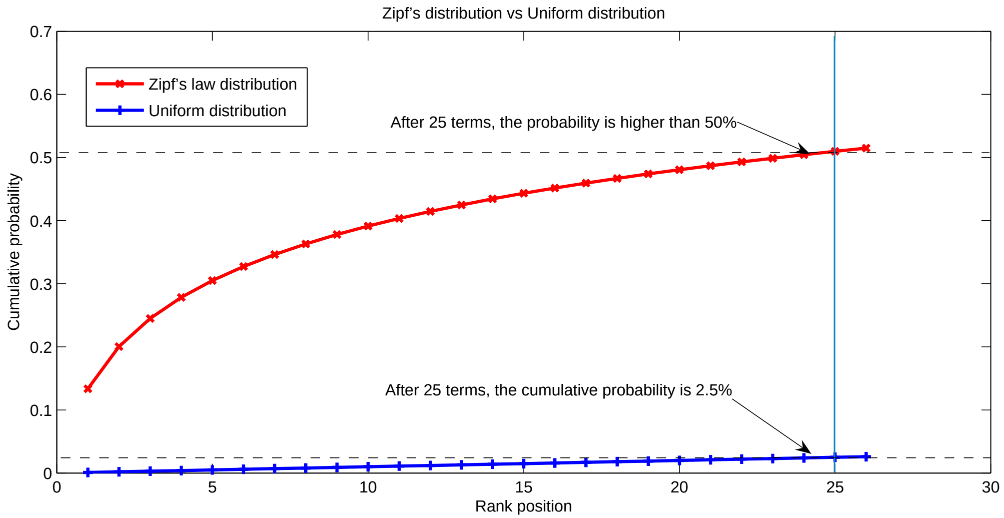
- Suppose an agent receives a message $m_k(\ldots, w_i, \ldots)$, where $w_i \notin L_a$, where L_a is the agent's ontology.
- The mapping algorithm (the "oracle") must find what entity, represented in the agent's ontology by t_m, was encoded in w_i by the transmitter.
- *The aim of this work is to specify a method for choosing the smallest set* $\Gamma \subseteq L_a$ *of terms to compare with* w_i*, given a probability of finding the matching term* $t_m \in L_a$.
- We assume that t_m exists in L_a.
- Let $p(t_j)$ be the probability that the entity represented by $t_j \in L_a$ was used in the i^{th}slot inside m_k.
- The oracle will find t_m if $t_m \in \Gamma$, am event that has a probability:

Definition

$p(t_m \in \Gamma) = \sum_{t_j \in \Gamma} p(t_j)$

- The core issue is how to find the probabilities of the entities that can be used in a message $m_k(\ldots, w_i, \ldots)$.

Exploiting term distribution



- If all terms are equiprobable, then $p(t_m \in \Gamma)$ will be proportional to $|\Gamma|$, as in the blue line in the graph.
- If the probability is distributed unevenly, we can obtain a higher probability for smaller Γ.
- For example, if $p(t_j)$ is distributed according to Zipf's law, the probability of finding t_m follow the red line in the graph.
- For $|L_a| = 1000$, then $p(t_m \in \Gamma) \geq 0.5$ for $|\Gamma| = 25$.

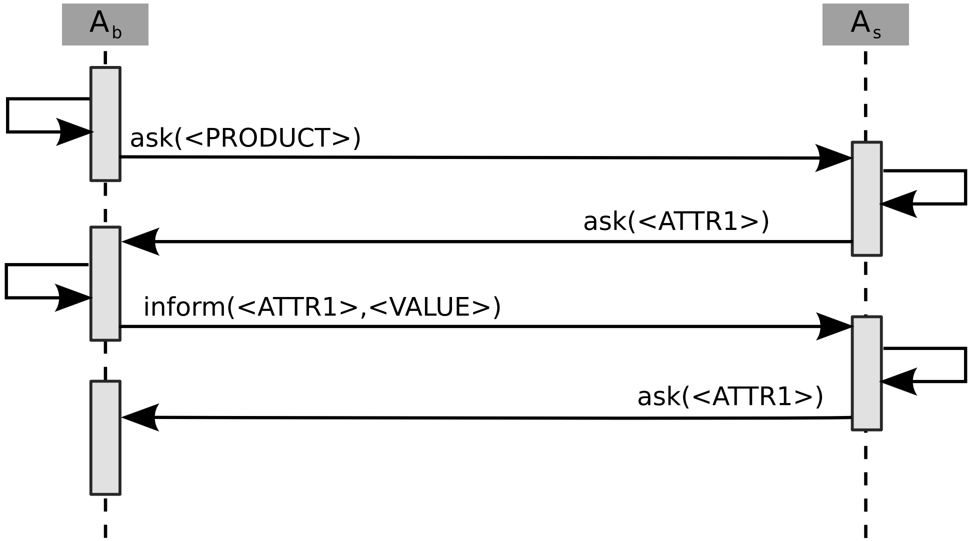
Assertions

- The possible values for a slot are modelled by M assertions
- Each assertion assigns a probability to the hypothesis that the matching entity for the slot belongs to a set Ψ:

Definition

$$A_j^{\langle \mathtt{Ni,A} \rangle_{\mathtt{R}}} \doteq Pr\left(slot\_value \in \Psi\right) \quad (1)$$

Example Scenario



A protocol for the purchase of a product: the buyer ask the product, and the vendor asks for some specs before making an offer.

How to obtain assertions

- Assertions are created and updated every time a protocol is executed.
- The predictor uses the feedback from the mapper to create the assertions
- The frequencies of the mappings are used to compute dynamically the probabilities in the assertions.

How to obtain assertions

- Assertions are created and updated every time a protocol is executed.
- The predictor uses the feedback from the mapper to create the assertions
- The frequencies of the mappings are used to compute dynamically the probabilities in the assertions.

If the purchase protocol is used by the buyer a number of time with different vendors, the frequencies of the terms appeared in the slot `ask<product>` are:

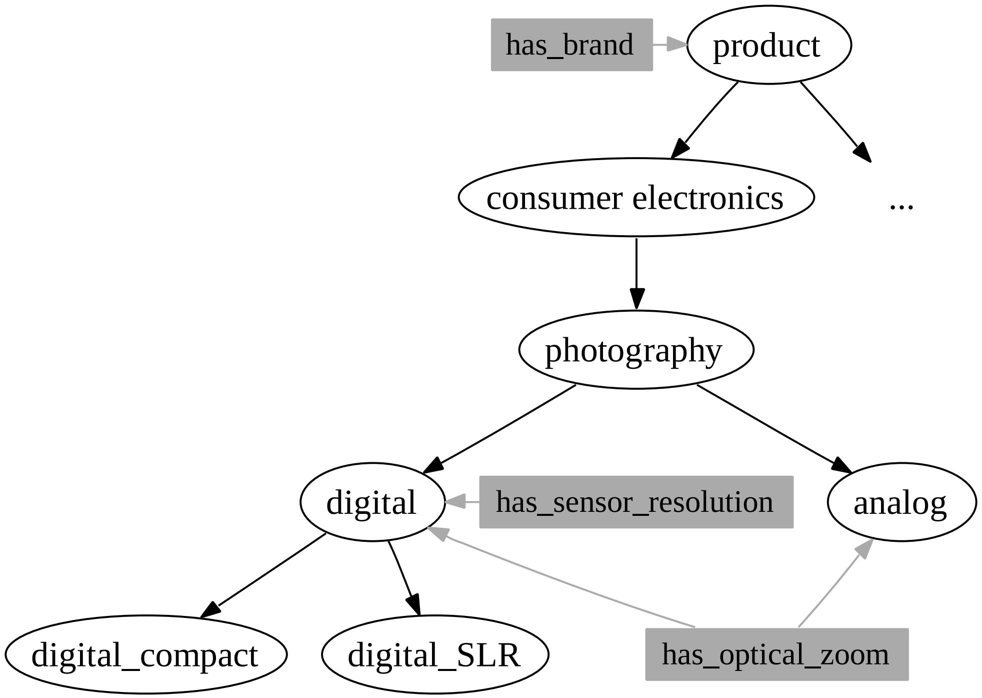
ask<product>	*Laptop*	*digital_SLR*	**Total**
has_brand	4	5	9
has_cpu	6	0	6
has_ram	6	0	6
has_hard_disk	4	0	4
has_weight	3	1	4
has_optical_zoom	0	5	5
has_sensor_resolution	**0**	**6**	6
Total	**23**	**17**	40

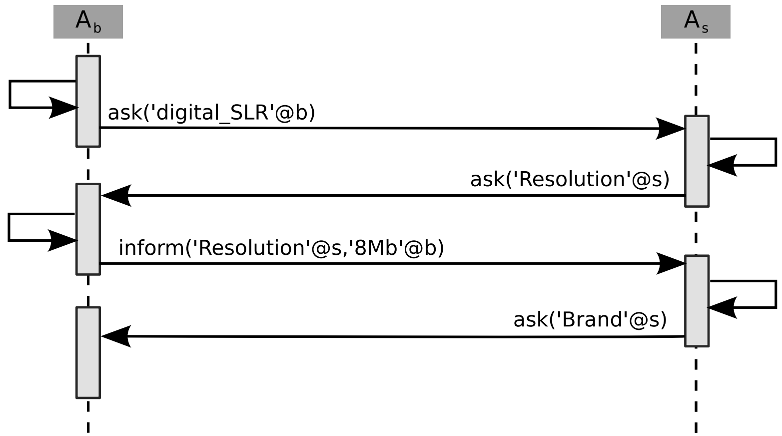
Ontology Relations

It is possible to use the ontological relations between terms in the dialogue. During the interaction, the system:

- makes hypotheses about the possible relations between the terms.
- try to prove them using the ontology,
- generalise them,
- update the frequencies of the successful hypotheses.

Ontology Relations





Example

For example, if the protocol is used for buying a Digital_SLR, when the message `ask('Resolution')`arrives, the *predictor* receives as feedback the mapped value '`has_sensor_resolution`' for the foreign term and generates a set of relations between the terms that have appeared:

`isSubClass(has_sensor_resolution,digital_SLR), ...,isInstance(has_sensor_resolution, digital_SLR),...,hasDomain(has_sensor_resolution,Digital_SLR)`

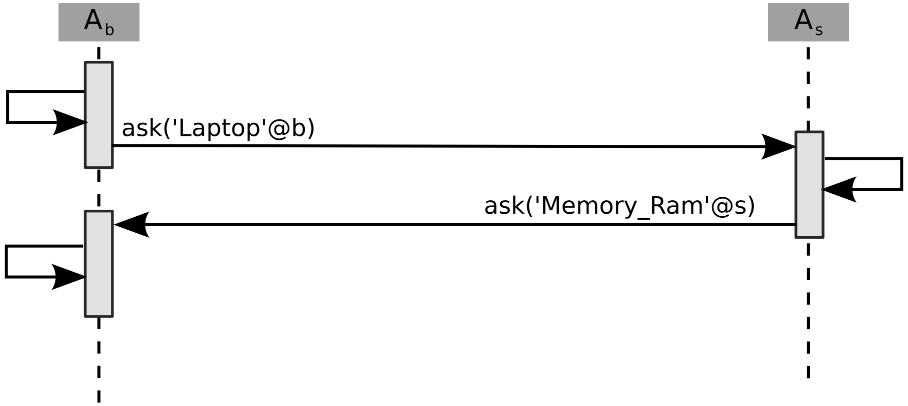
Stored relation

- Then the system tries to prove all the generated relations
- In proved relations, the values are replaced with the dialogue slot that contained the entity.
- The stored relation keeps track of the frequency with which they have been proved in the dialogues.

Example

`hasDomain(has_sensor_resolution, digital_SLR)` can be proved: the stored `hasDomain` relation is between the slot in the sent `ask(<Product>)` message and the `ask(<ATTR*>)` received from the seller.

Example of use



The purchase protocol is used for buying a laptop. We want to predict the content of the `ask(<ATTR>)` message: the mapper will compare the foreign term `Memory_Ram` only with these suggestions, and find the correct mapping with a probability higher thatn 90%.

Simple frequencies

$\langle \texttt{ask,1} \rangle_{\texttt{nb}}$	*Laptop*	*digital_SLR*	**Total**
has_brand	4	5	9
has_cpu	**6**	**0**	**6**
has_ram	6	0	6
has_hard_disk	4	0	4
has_weight	3	1	4
has_optical_zoom	0	5	5
has_sensor_resolution	0	6	6
Total	**23**	**17**	**40**

The probability of a term in a slot can be computed simply by its frequency.

Simple frequencies

$\langle \texttt{ask,1} \rangle_{\texttt{nb}}$	*Laptop*	*digital_SLR*	**Total**
has_brand	4	5	9
has_cpu	**6**	**0**	**6**
has_ram	6	0	6
has_hard_disk	4	0	4
has_weight	3	1	4
has_optical_zoom	0	5	5
has_sensor_resolution	0	6	6
Total	**23**	**17**	**40**

The probability of a term in a slot can be computed simply by its frequency.

Example

$A_1)\ Pr\left(slot\_value \in \{\text{"}has\_cpu\text{"}\}\right) = \frac{6}{40} = 0.15$

Simple frequencies

$\langle \mathtt{ask}, 1 \rangle_{\mathtt{nb}}$	*Laptop*	*digital_SLR*	**Total**
has_brand	4	5	9
has_cpu	6	0	6
has_ram	6	0	6
has_hard_disk	4	0	4
has_weight	3	1	4
has_optical_zoom	0	5	5
has_sensor_resolution	**0**	**6**	**6**
Total	**23**	**17**	**40**

The information provided by these simple assertions is rather poor, as it is not possible to exploit any contextual information: the probability of terms not related with the context can be the same.

Simple frequencies

$\langle \texttt{ask,1} \rangle_{\texttt{nb}}$	*Laptop*	*digital_SLR*	**Total**
has_brand	4	5	9
has_cpu	6	0	6
has_ram	6	0	6
has_hard_disk	4	0	4
has_weight	3	1	4
has_optical_zoom	0	5	5
has_sensor_resolution	**0**	**6**	**6**
Total	**23**	**17**	**40**

The information provided by these simple assertions is rather poor, as it is not possible to exploit any contextual information: the probability of terms not related with the context can be the same.

Example

$A_2)\ Pr\left(slot\_value \in \{\text{"}has\_sensor\_resolution\text{"}\}\right) = \frac{6}{40} = 0.15$

Conditional frequencies

$\langle \texttt{ask,1} \rangle_{\texttt{nb}}$	*Laptop*	*digital_SLR*	**Total**
has_brand	4	5	9
has_cpu	6	0	6
has_ram	6	0	6
has_hard_disk	4	0	4
has_weight	3	1	4
has_optical_zoom	0	5	5
has_sensor_resolution	0	6	6
Total	**23**	**17**	**40**

The probability computed using the frequency of a term conditional to the value of another term provides more information:

Conditional frequencies

$\langle \texttt{ask,1} \rangle_{\texttt{nb}}$	*Laptop*	*digital_SLR*	**Total**
has_brand	4	5	9
has_cpu	6	0	6
has_ram	6	0	6
has_hard_disk	4	0	4
has_weight	3	1	4
has_optical_zoom	0	5	5
has_sensor_resolution	0	6	6
Total	**23**	**17**	**40**

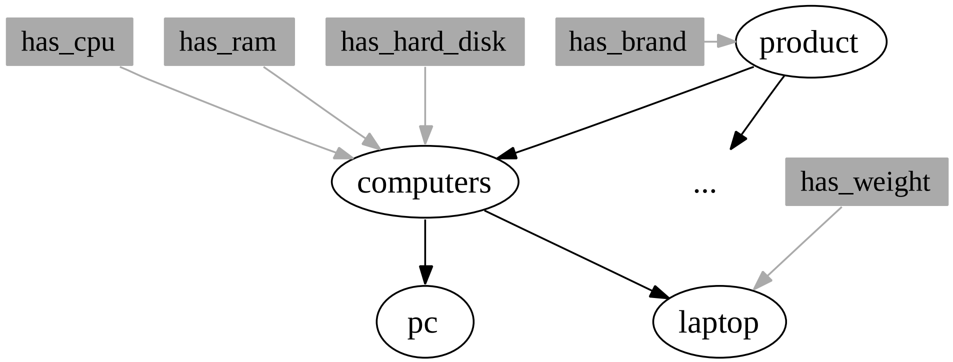
The probability computed using the frequency of a term conditional to the value of another term provides more information:

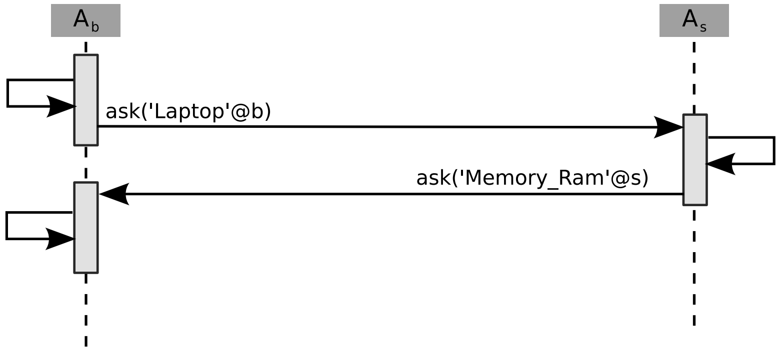
Example

A_3) $Pr\left(slot\_value \in \{\text{"}has\_cpu\text{"}\} \mid \text{"}\langle PRODUCT\rangle = Laptop\text{"}\right) = \frac{6}{23} = 0.260$

A_4) $Pr\left(slot\_value \in \{\text{"}has\_sensor\_resolution\text{"}\} \mid \langle PRODUCT\rangle = \text{"}Laptop\text{"}\right) = \frac{0}{23} = 0.0$

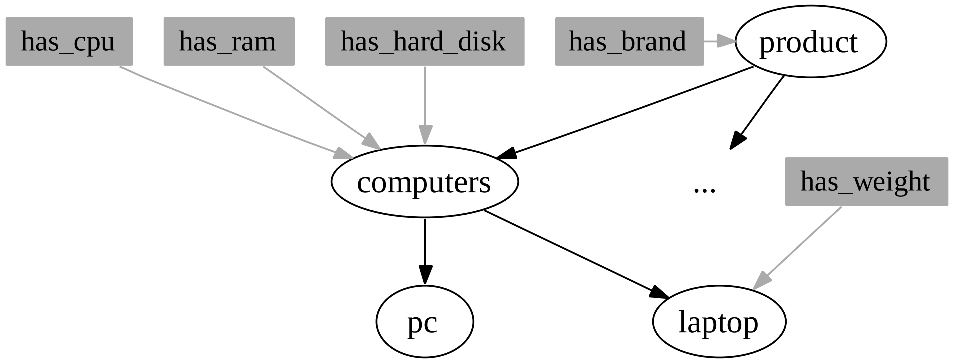
Relation assertions

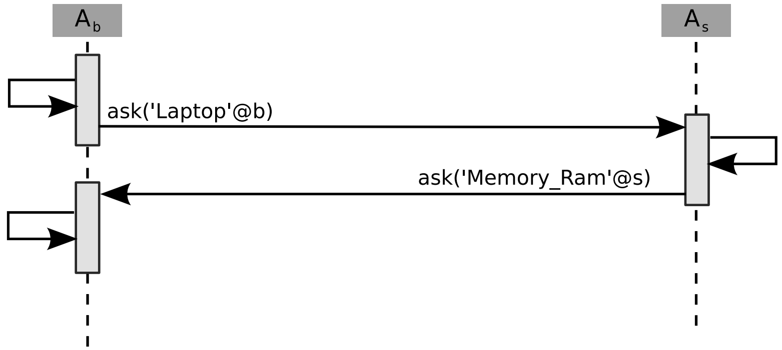




- The relations stored for the slot are instantiated, replacing the references with the values of the already filled slots
- The remaining variables are unified with the entities in the ontology that satisfy the predicate

Relation assertions





- The relations stored for the slot are instantiated, replacing the references with the values of the already filled slots
- The remaining variables are unified with the entities in the ontology that satisfy the predicate

Example

The relation `hasDomain(X, <PRODUCT>)` becomes: `hasDomain(X, Laptop)`.
The system searches the entities in the ontology that satisfy the relation:
`{has_cpu, has_ram, has_hard_disk, has_brand, has_weight}`
The assertion becomes:
$A_5)\ \ Pr(slot\_value \in \{has\_cpu, has\_ram, has\_hard\_disk, has\_brand, has\_weight\}) = 1$

Computing term probabilities

1. Probabilities given to sets are uniformly distributed among the members: according to the *principle of indifference*, the probability of mutually exclusive elements in a set should be evenly distributed.
2. Hence, the probability of an entity t_i is computed by summing over all its probabilities, and dividing the sum by the sum of all the probabilities given by assertions regarding the slot:

Definition

$$p\left(t_i\right) = \frac{\sum A_j\left(slot\_value \in \{t_i\}\right)}{\sum A_k} \quad (2)$$

Computing term probabilities

Example

To compute the probability of `has_cpu`.

Group all the assertions about the term

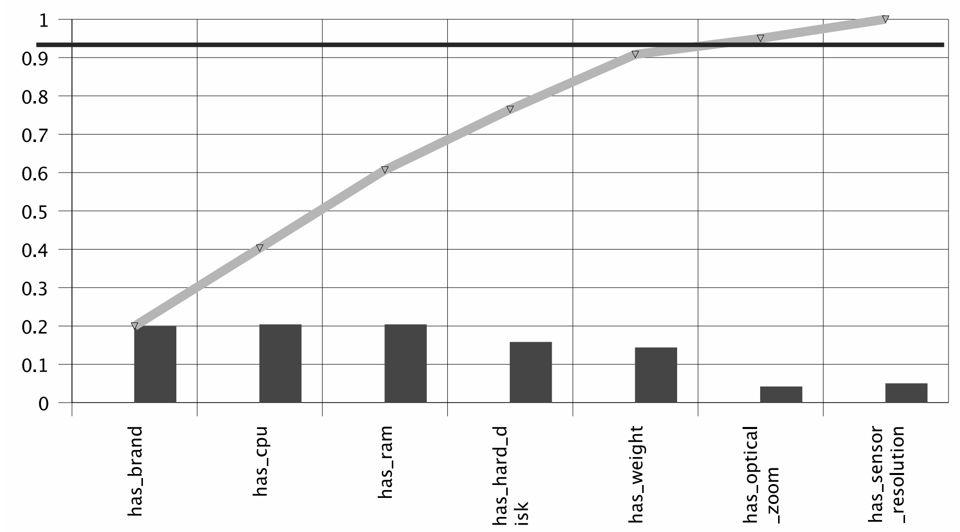
$A_1)$ $Pr\left(\langle \texttt{ask}, 1\rangle_{\texttt{nb}} \in \{''has\_cpu''\}\right) = \frac{6}{40} = 0.15$

$A_3)$ $Pr\left(\langle \texttt{ask}, 1\rangle_{\texttt{nb}} \in \{''has\_cpu''\} \mid sent\_ask = ''Laptop''\right) = \frac{6}{23} = 0.260$

$A_5)$ $Pr(slot\_value \in \{has\_cpu, has\_ram, has\_hard\_disk, has\_brand, has\_weight\}) = 1$

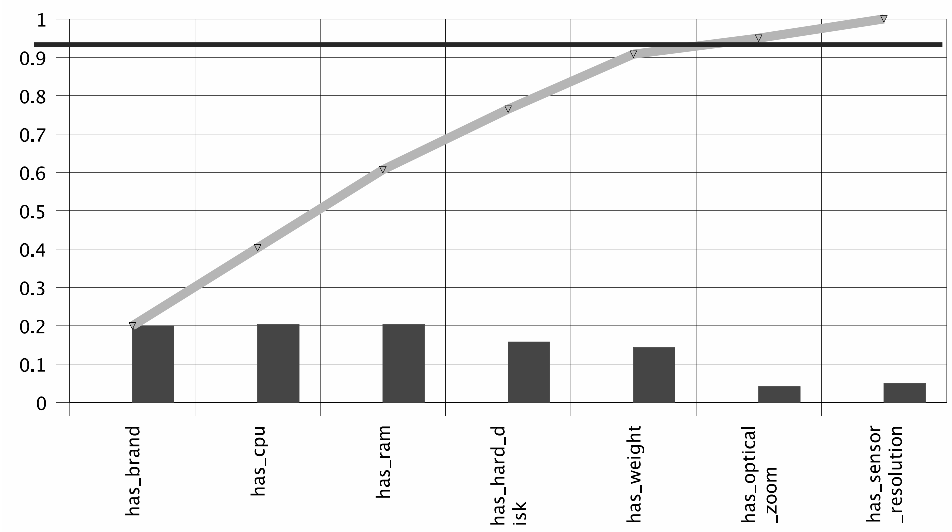
$$P(has\_cpu) = \frac{A_1 + A_3 + \frac{A_5}{5}}{A_1 + \ldots + A_N} = \frac{0.15_1 + 0.26 + 0.2}{3} = 0.203$$

Choosing the terms



- The probabilities are computed for all the terms
- The terms are ordered in descending order of probability
- The first N terms whose cumulative probability reaches the chosen threshold are used as suggestions

Choosing the terms



- The probabilities are computed for all the terms
- The terms are ordered in descending order of probability
- The first N terms whose cumulative probability reaches the chosen threshold are used as suggestions

Example

In this case, with a threshold of 95%, the terms `has_brand`, `has_cpu`, `has_ram`, `has_hard_disk`, `has_weight` are kept and passed to the mapper as suggestions. The terms `has_optical_zoom` and `has_sensor_resolution` are discarded.

Conclusions

- We showed an approach for improving dynamic ontology mapping that exploits knowledge about interactions to reduce the waste of resources employed to verify unlikely similarities between unrelated terms
- Traditional approaches aim at finding all the possible mappings between the ontologies, so that any possible interaction can occur.
- Our goal is pragmatic: only the mappings required for the interactions that take place need to be found.
- In the standard approach, an ontology mapper oracle compares the "foreign" terms with all the terms in the agent's ontology, although most of the compared terms are not related.
- However, the terms that appear in messages are not all equally probable: given the context of the interaction, some will be more likely than others.
- The use of protocols allows us to collect consistent information about the mappings used during an interaction

Thanks

Thanks for your attention

Any question?