



OntoVPA -

**Building an ontology-based rule engine for
dialogue management on SPARQL + OWL**



**Michael Wessel,
Girish Acharya, James Carpenter and Min Yin**

Background – VPAs @ SRI International (Think “SIRI”)

The screenshot shows a web browser window titled "Siri | SRI International". The main content is a timeline titled "Timeline of Innovation" with a horizontal axis from 2000 to 2010. A green dot marks the year 2007. Below the timeline, under the heading "2007 Siri", there is a photograph of an iPhone screen displaying the Siri interface with the text "What can I help you with?". To the right of the timeline, a yellow callout box contains the following text:

- **Siri, Inc. SRI Spinoff 2007**
- **Acquired by Apple in 2010**
- **Since October 2011 integrated feature of iPhone 4S**

Below the timeline, there is descriptive text about Siri's development and a section about its acquisition by Apple. At the bottom of the page, there are links for "Learn More" and "Related Timeline Items".

Siri is great and was / is revolutionary, but has some shortcomings when it comes to dialogue management (state tracking, ...) – SRI hasn't stopped working on VPA's...



What is OntoVPA?

- A dialogue management system (DMS) for
 - Conversational VPAs
 - Mixed initiative, usually task-oriented / goal driven, but not necessarily
 - Sub-dialogues
 - Deep domain knowledge and complex workflows
 - Virtual Personal **Specialists**
 - Ontologies for domain- and dialogue-representation
 - Frame / intent -based (OWL2)
 - Rule-based – **ontology-aware rules** (SPARQL custom rule engine)
 - Declarative / knowledge-based
 - Customizable by swapping in and out ontologies
 - Little to no conventional programming
 - Reuse of standard ontologies and generic VPA Upper Level



Benefits of Formal Ontologies for DMS / VPA

- **More faithful domain representation**
 - Captures more semantics of the domain
 - Enables (sound and complete) reasoning over the domain
 - Closer to human conceptualization of the domain and hence „more useful“
- **Natural Language Understanding**
 - Understanding requires knowledge
 - Polysemy of natural language
 - Syonyms, hyponyms, hypernyms, ...
 - Disambiguation
 - Co-reference / anaphora resolution
 - Context-dependent interpretation of utterances in the context of the dialogue
- **Formal Knowledge Representation**
 - Explainability, transparency, understandability, reliability (Banking VPA) ...



SRI Virtual Personal Assistant

OntoVPA Demo Video -

Chat Event

VPA How can I help you today?

Chat

Talk

Universals

Semantic Search

Semantic Similarity

Anaphora Resolution

Synonyms, Hypernyms, ...

Reflexiveness (*knows what it knows and what not*)

Spatial Search

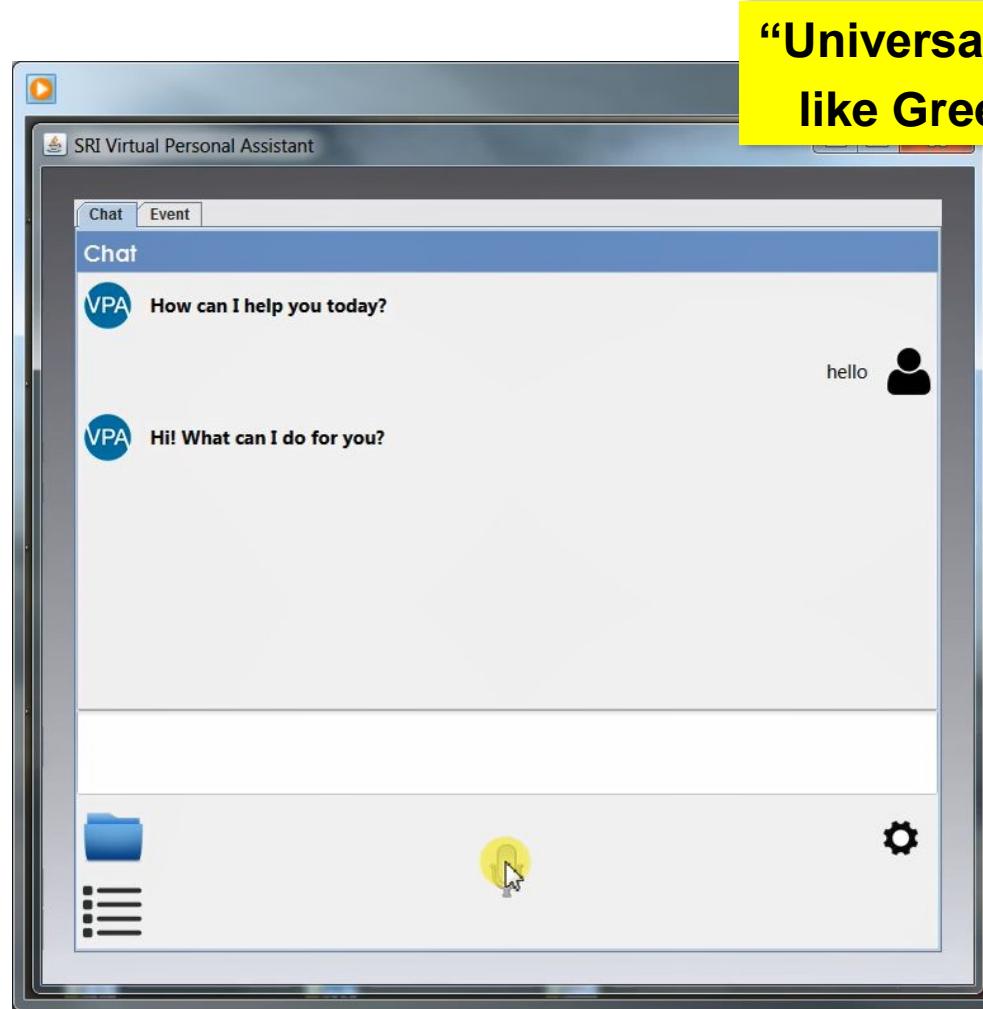
Ability to refer to previous dialogue entities

Interpretation of arbitrary input (“Palo Alto”) in dialogue context

Disambiguation



OntoVPA Demo 1 / 23



“Universals”
like GreetingIntent etc.



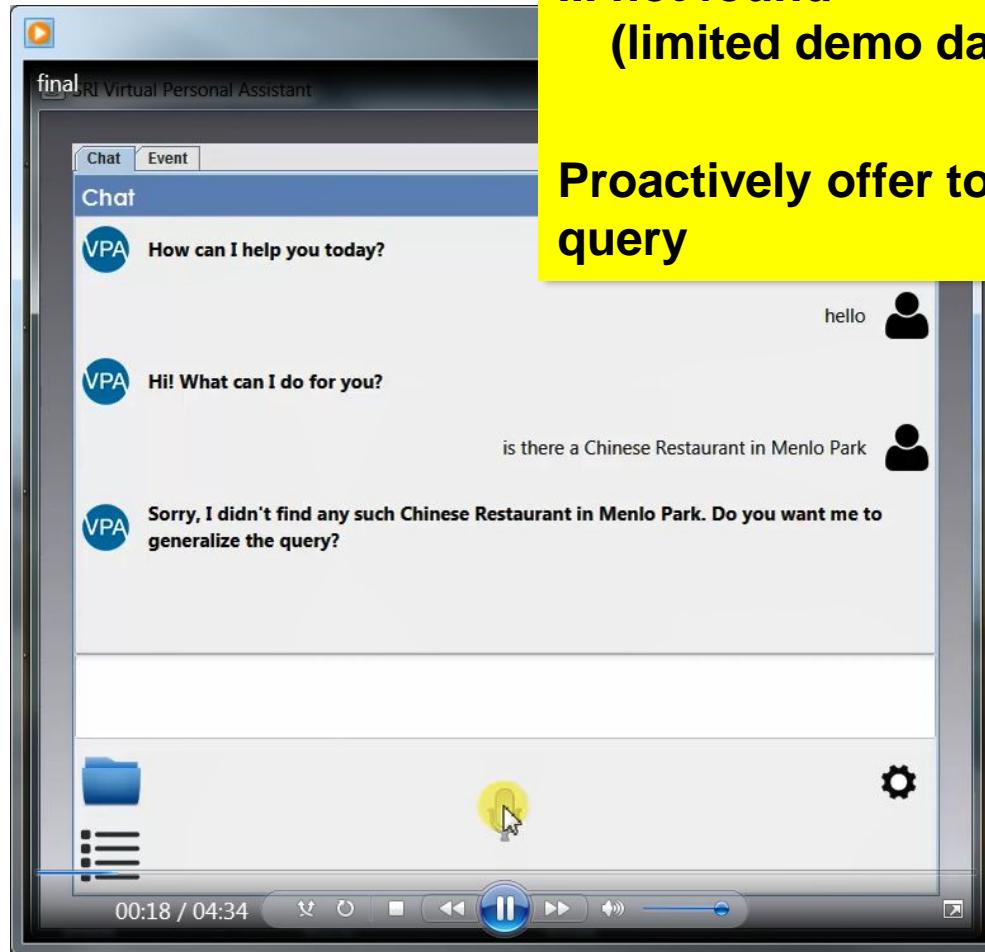
OntoVPA Demo 2 / 23



```
Intent1 : {  
    type = FindPoiIntent  
    dialogEntity = anon1 : {  
        type = ChineseRestaurant  
        directlyInside = MenloPark }  
}
```



OntoVPA Demo 3 / 23

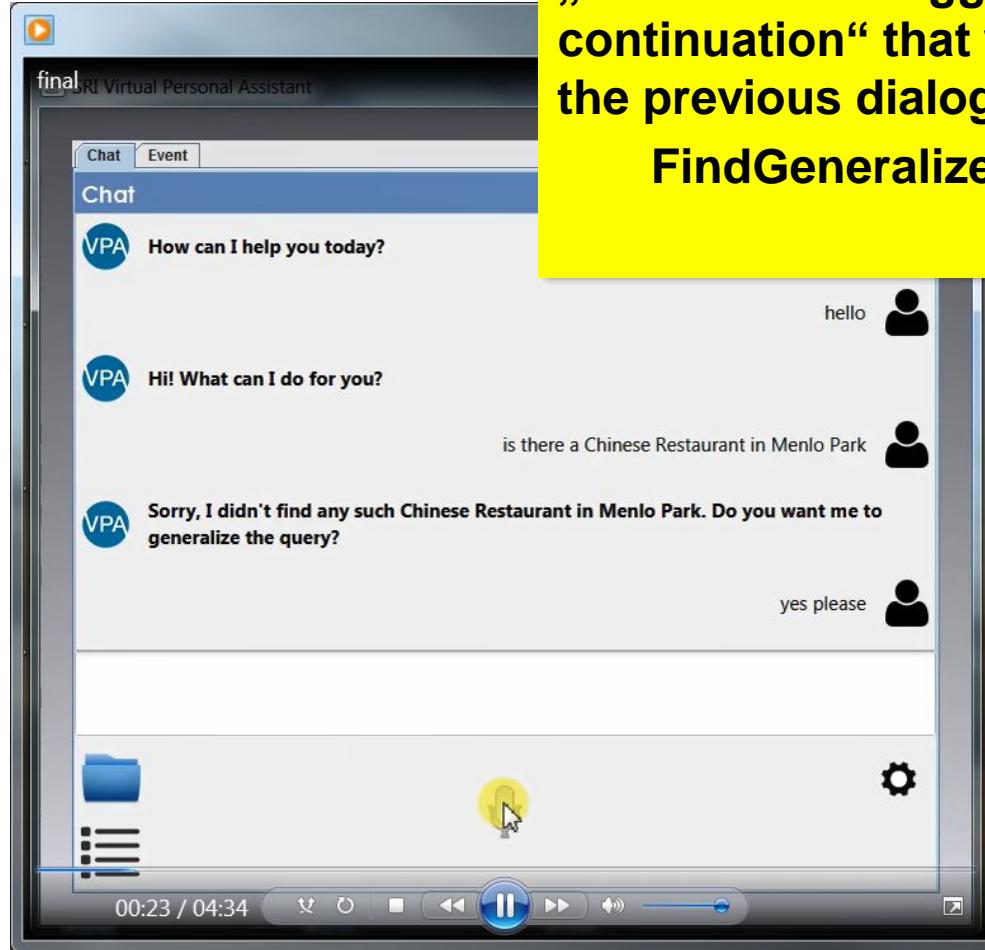


... not found
(limited demo data set)

Proactively offer to generalize
query



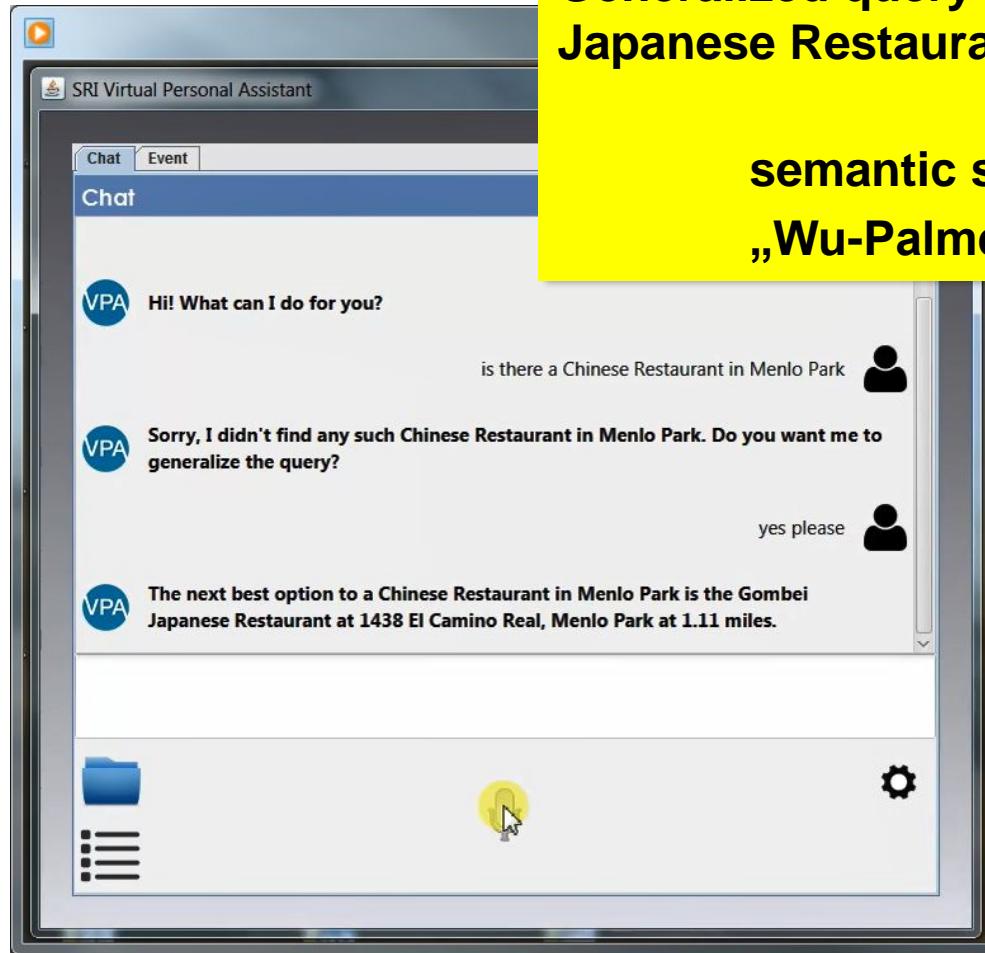
OntoVPA Demo 4 / 23



„Yes“ answer triggers the „yes continuation“ that was set up by the previous dialogue step ->
FindGeneralizedPOIntent



OntoVPA Demo 5 / 23

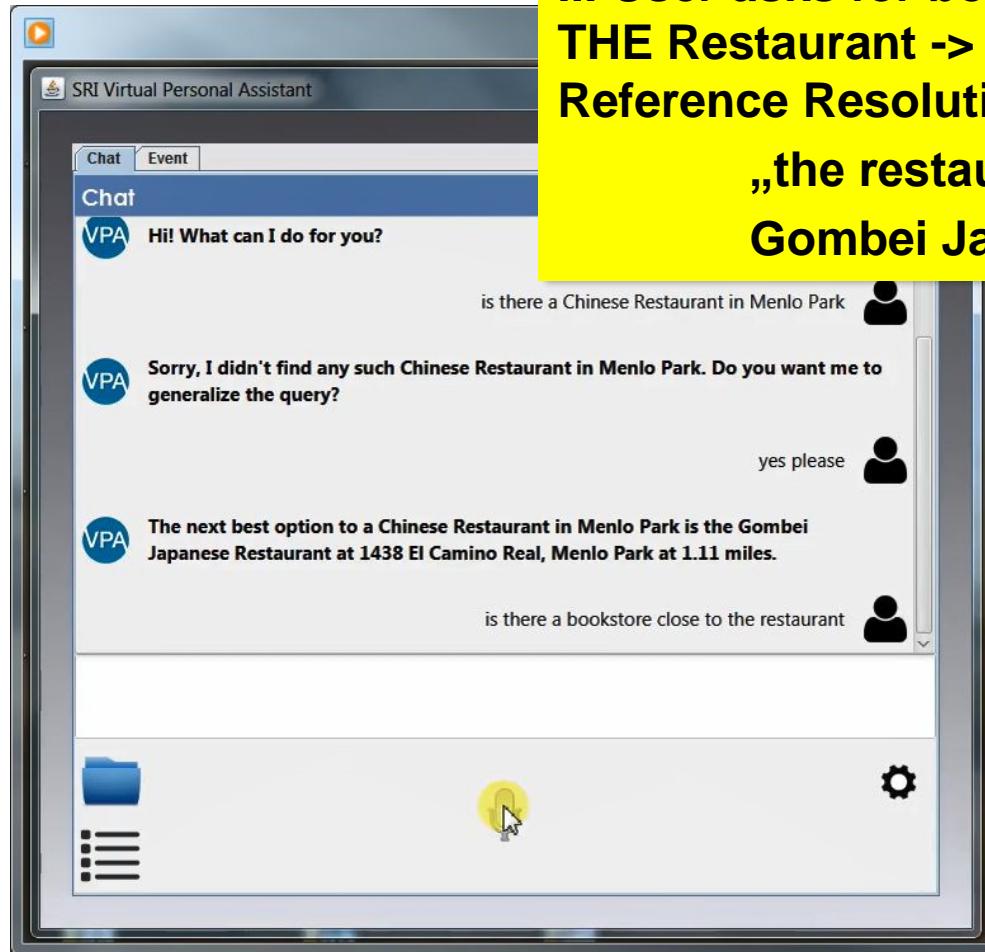


Generalized query returned
Japanese Restaurant

semantic similarity
„Wu-Palmer“



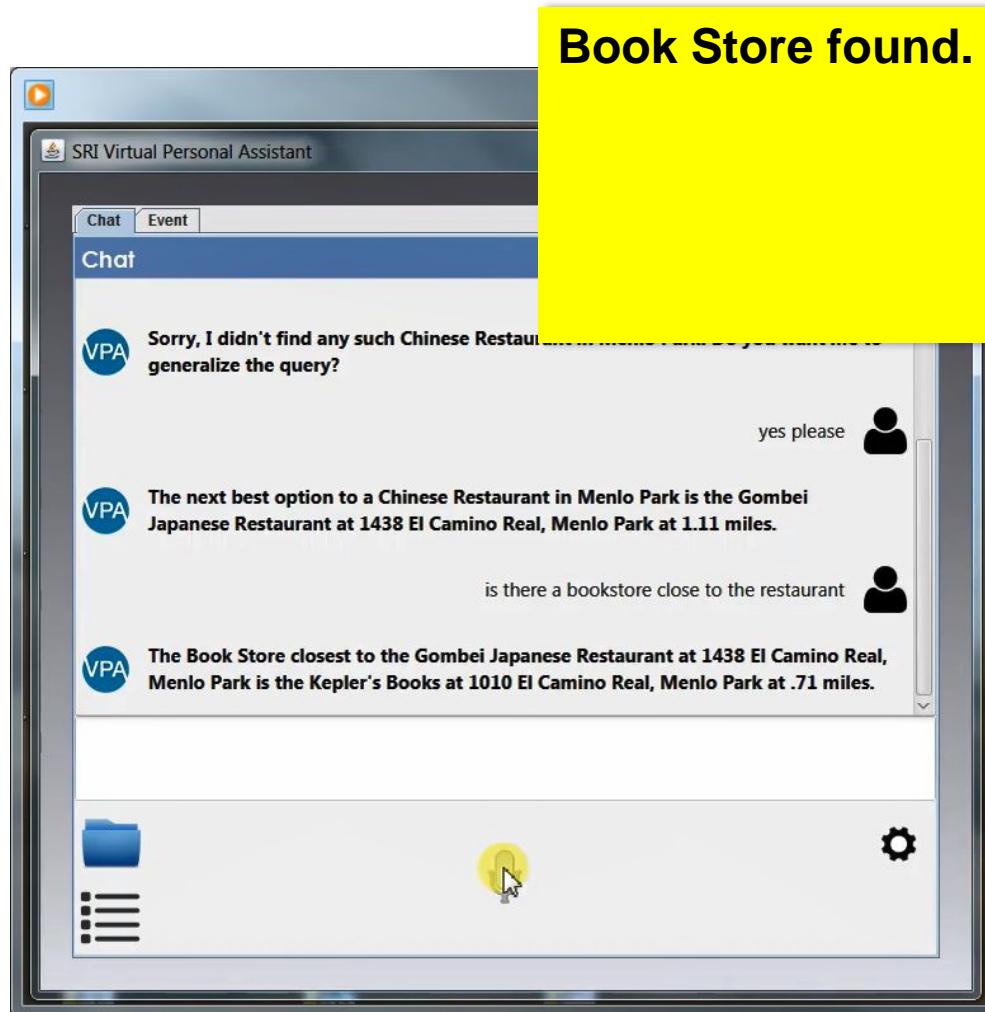
OntoVPA Demo 6 / 23



... User asks for bookstore close to THE Restaurant -> Anaphora / Co-Reference Resolution

„the restaurant“ matches
Gombei Jap. Restaurant

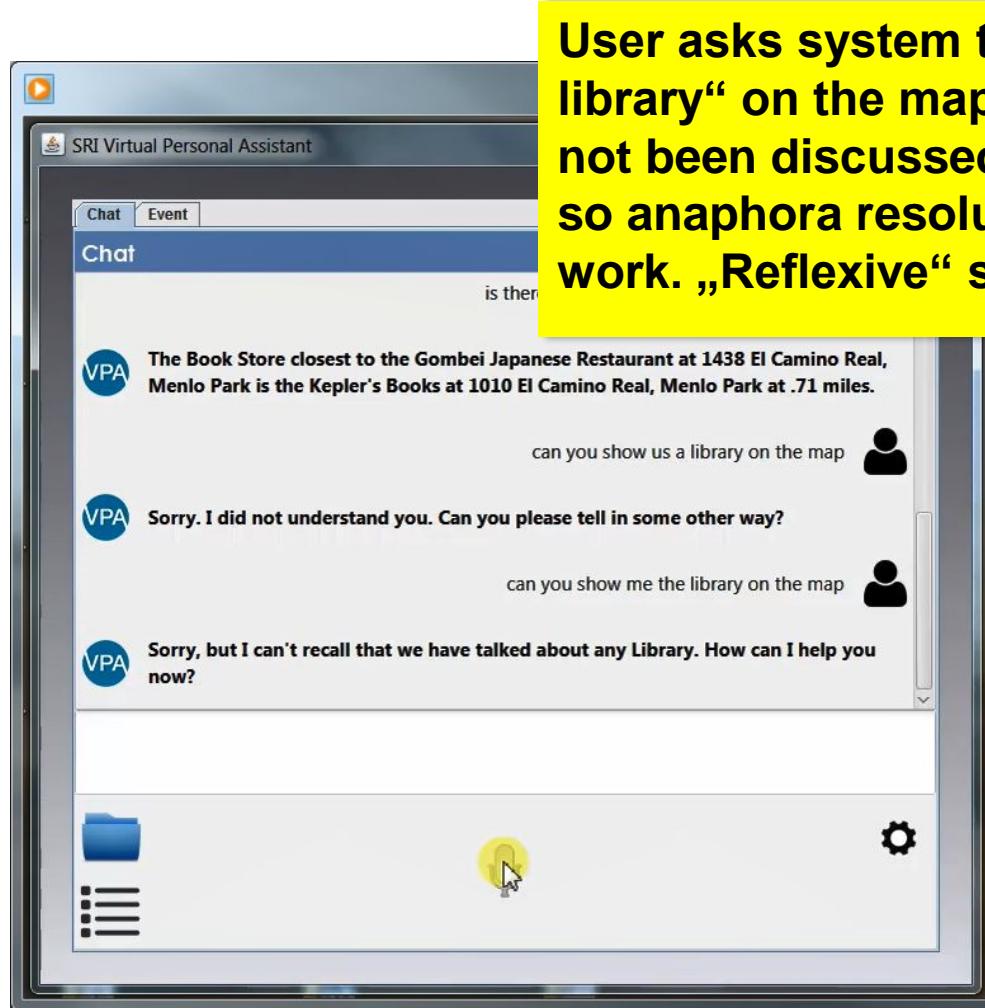
OntoVPA Demo 7 / 23



Book Store found.

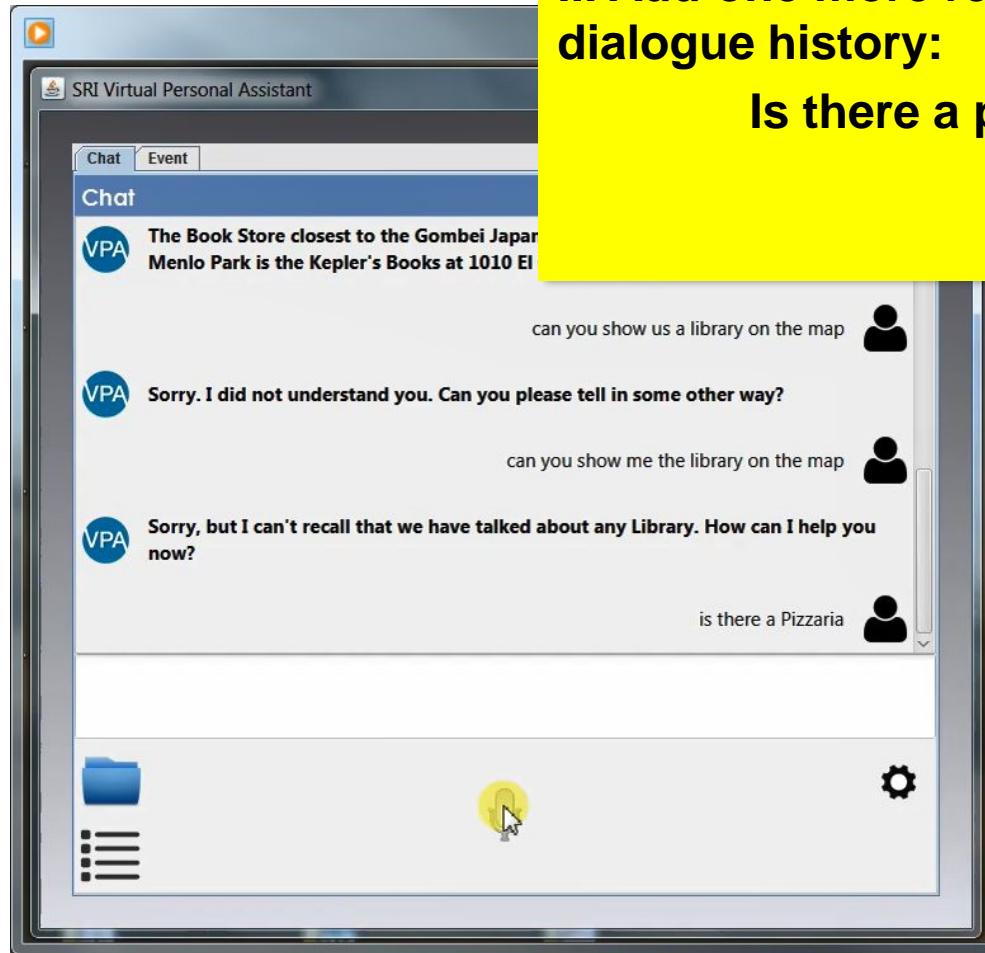


OntoVPA Demo 8 / 23



User asks system to show „the library“ on the map – library has not been discussed in the dialogue, so anaphora resolution does not work. „Reflexive“ system!

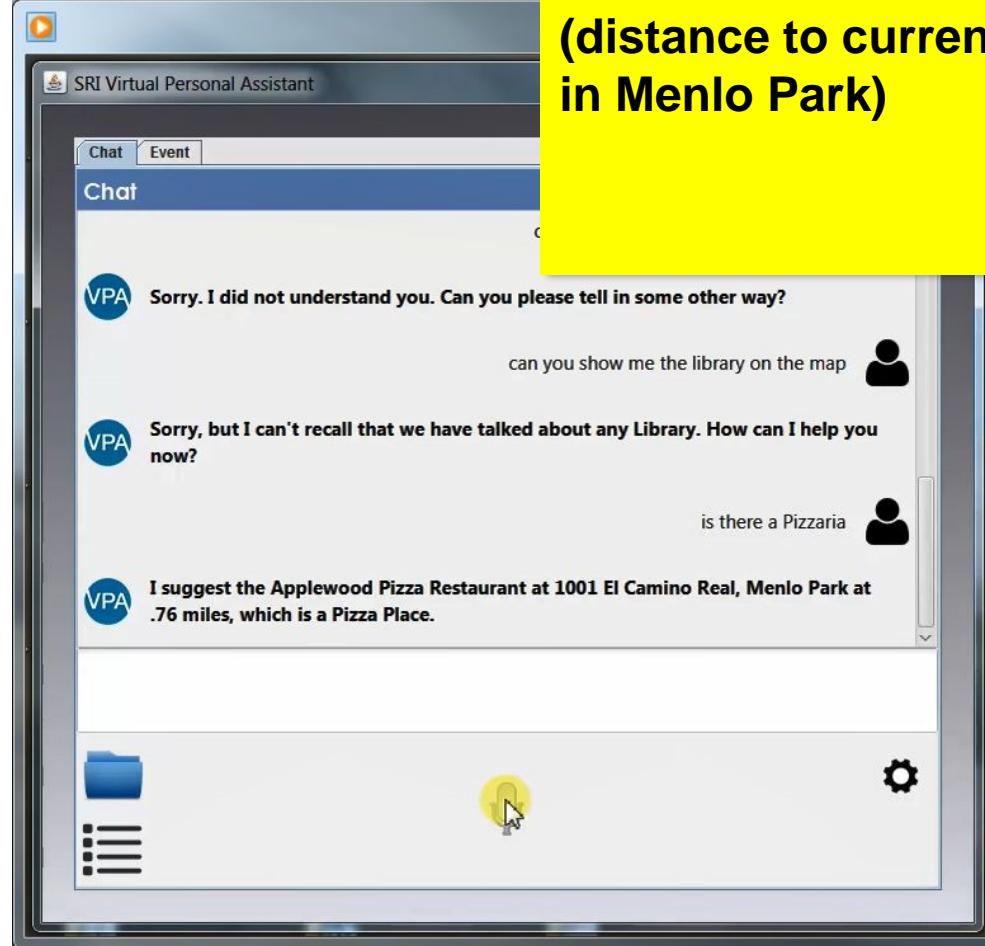
OntoVPA Demo 9 / 23



... Add one more restaurant to dialogue history:
Is there a pizzeria?

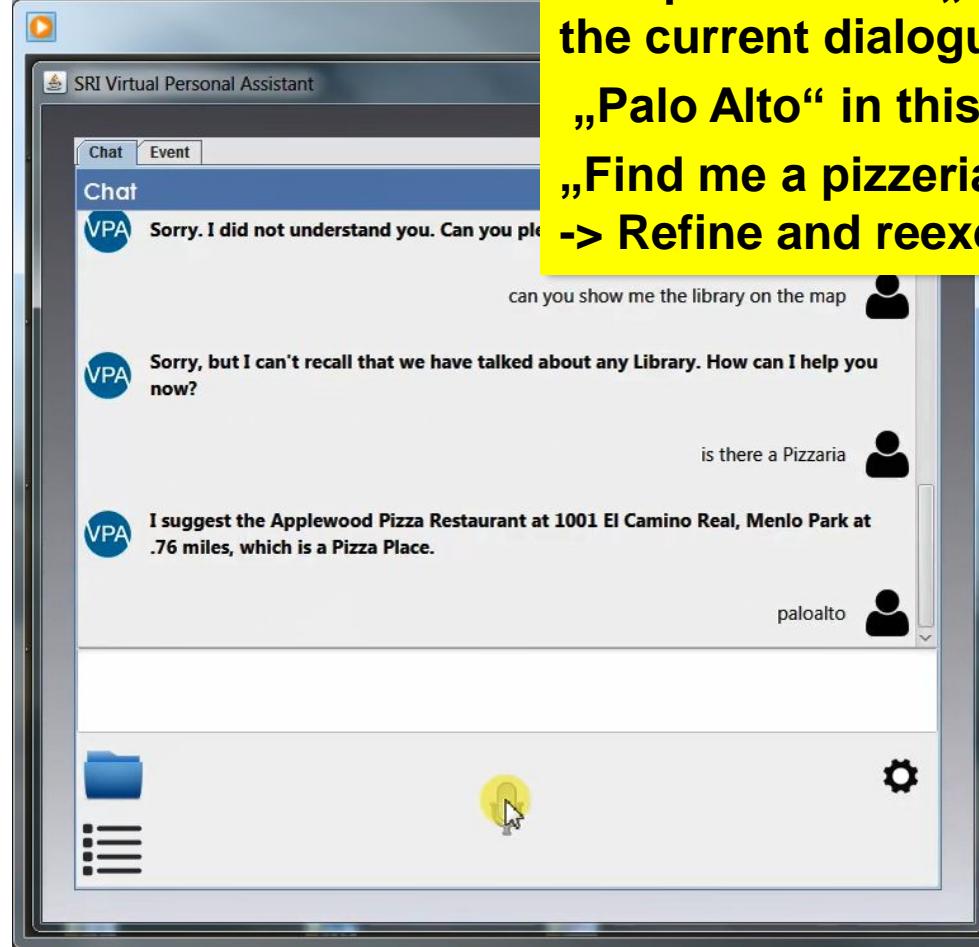


OntoVPA Demo 10 / 23



Pizzeria in Menlo Park found
(distance to current location at SRI
in Menlo Park)

OntoVPA Demo 11 / 23



Interpretation of „arbitrary input“ in the current dialogue context

„Palo Alto“ in this context means:

„Find me a pizzeria in Palo Alto!“

-> Refine and reexecute prev. intent

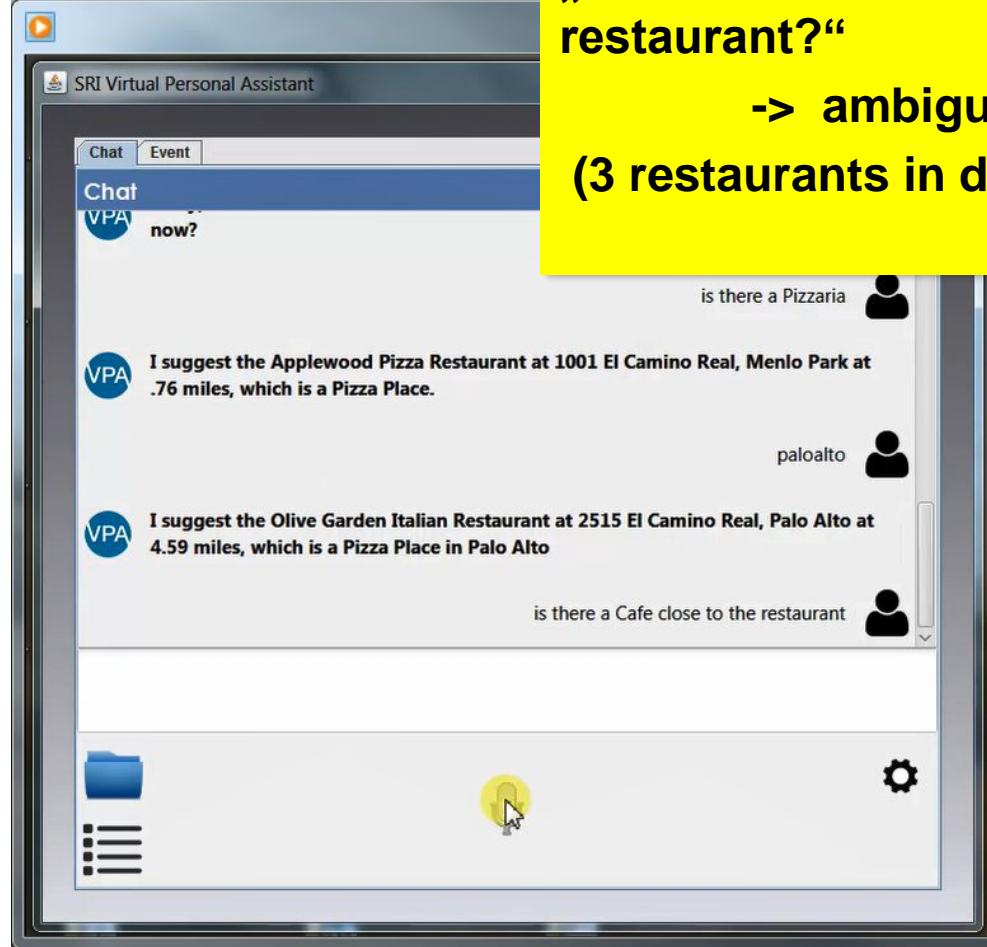


OntoVPA Demo 12 / 23

... Pizza-serving restaurant found in Palo Alto



OntoVPA Demo 13 / 23



„Is there a cafe close to THE
restaurant?“

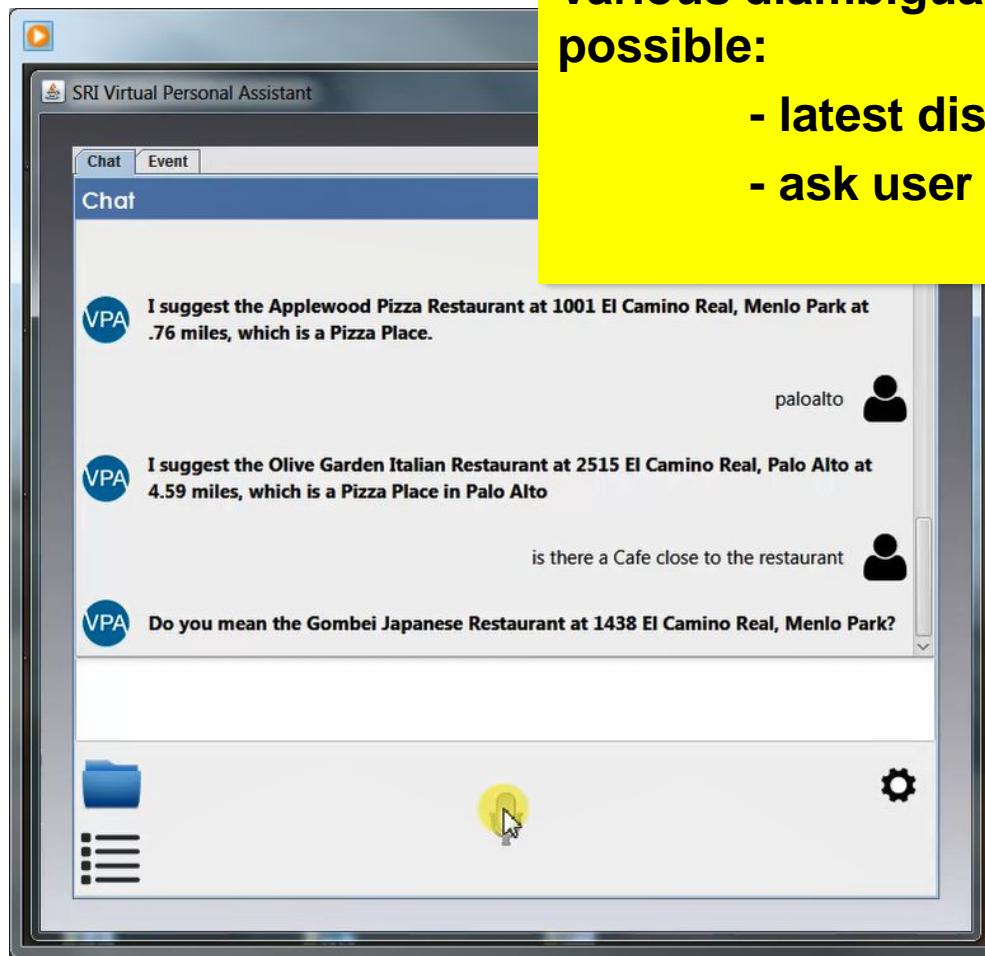
-> ambiguous by now...
(3 restaurants in dialogue history)



OntoVPA Demo 14 / 23

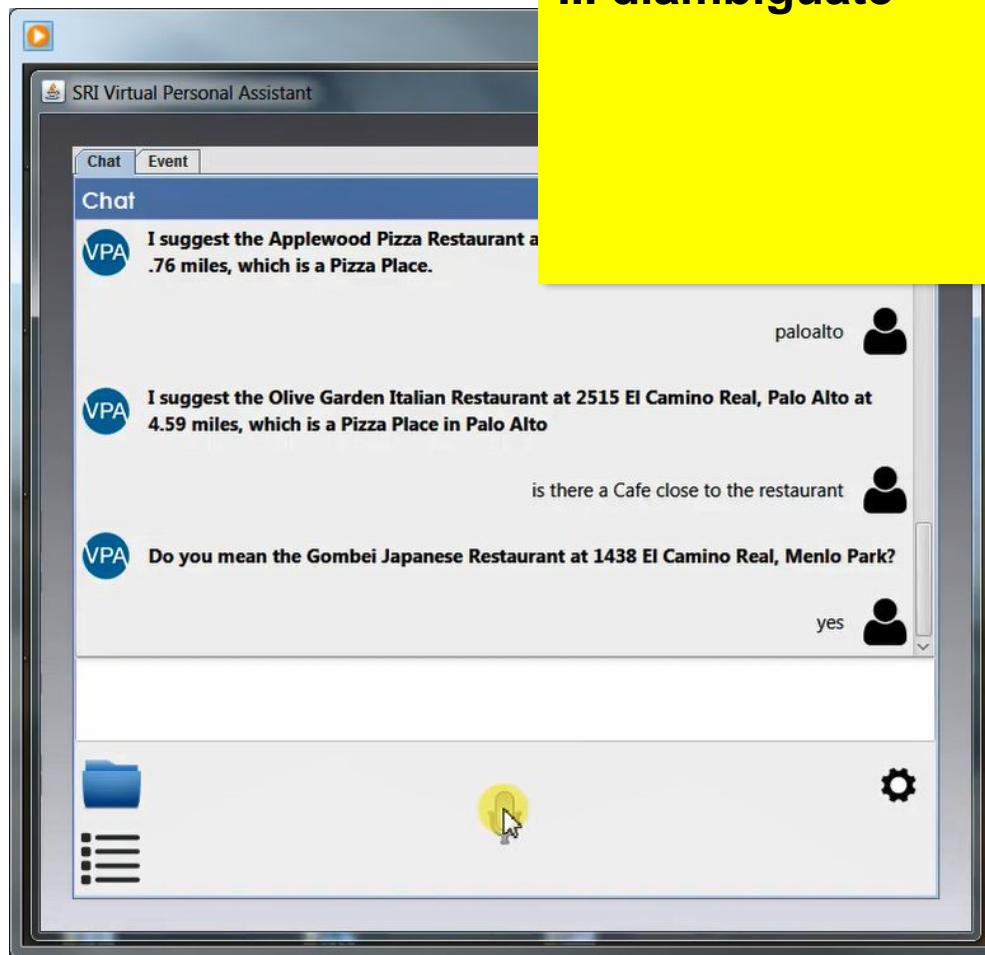
Various diambiguation strategies possible:

- latest discussed
- ask user (here!)

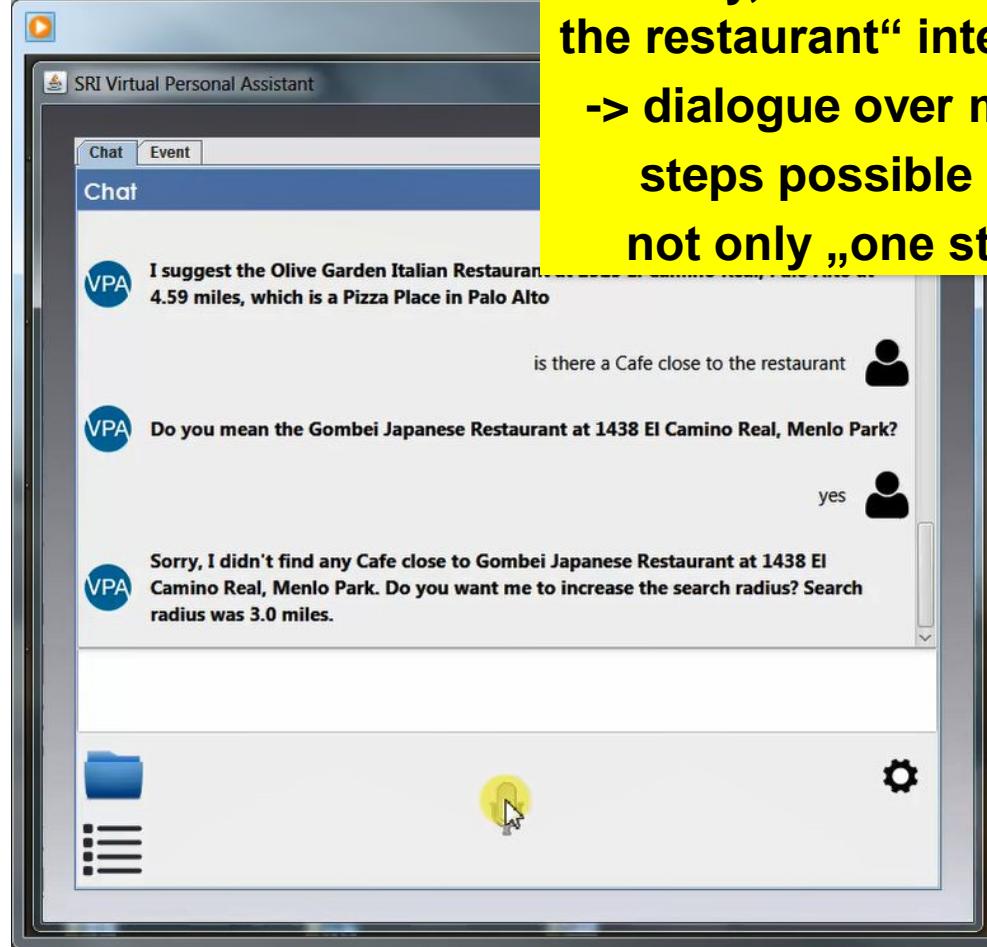


OntoVPA Demo 15 / 23

... diambiguare



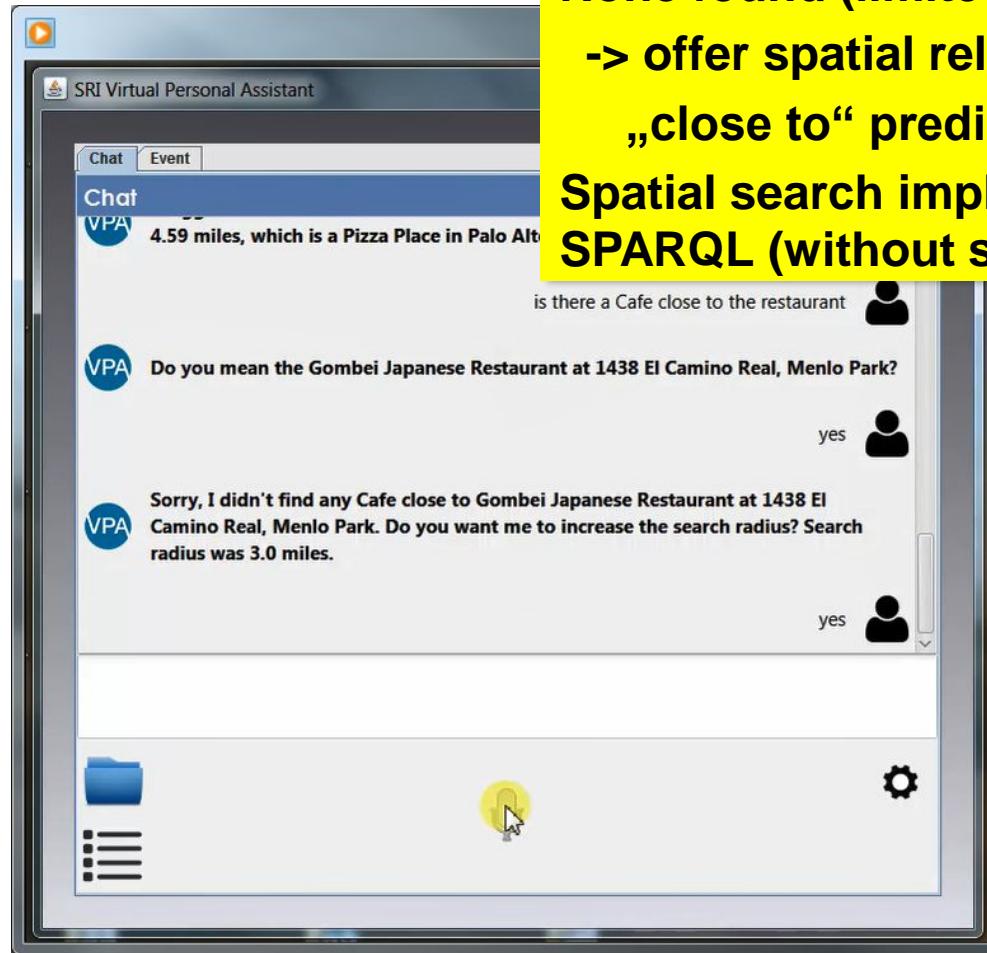
OntoVPA Demo 16 / 23



... finally, execute the „cafe close to the restaurant“ intent
-> dialogue over multiple steps possible in OntoVPA,
not only „one step“ Q / A



OntoVPA Demo 17 / 23

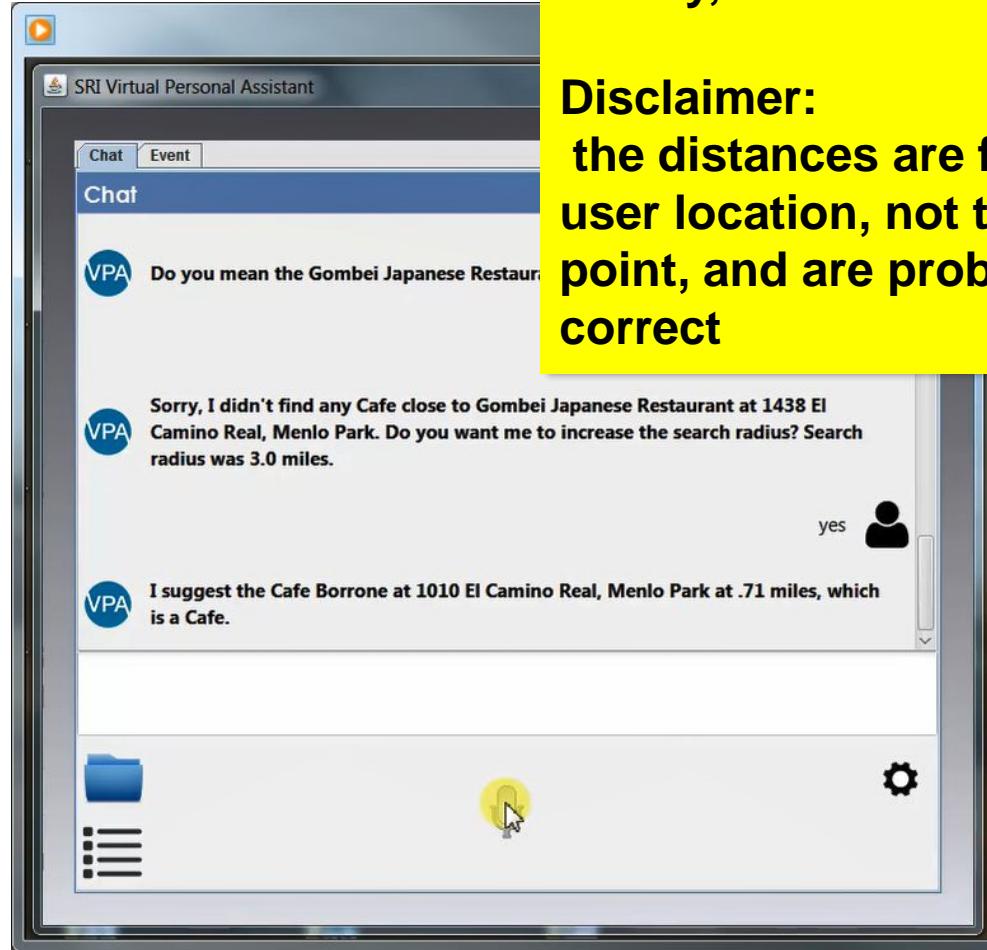


None found (limited dataset)

**-> offer spatial relaxation of
„close to“ predicate**

**Spatial search implemented in
SPARQL (without spatial index)**

OntoVPA Demo 18 / 23

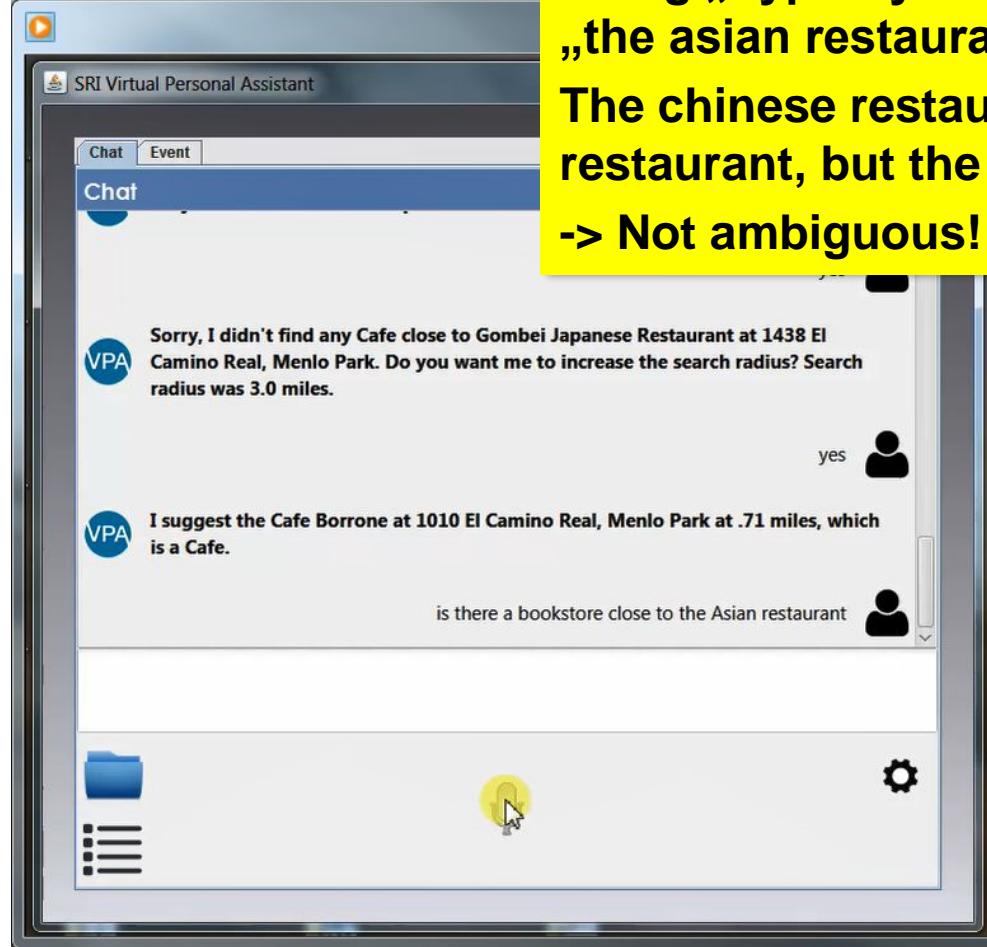


Finally, a Café is found...

Disclaimer:
the distances are from current user location, not to reference point, and are probably not 100% correct



OntoVPA Demo 19 / 23

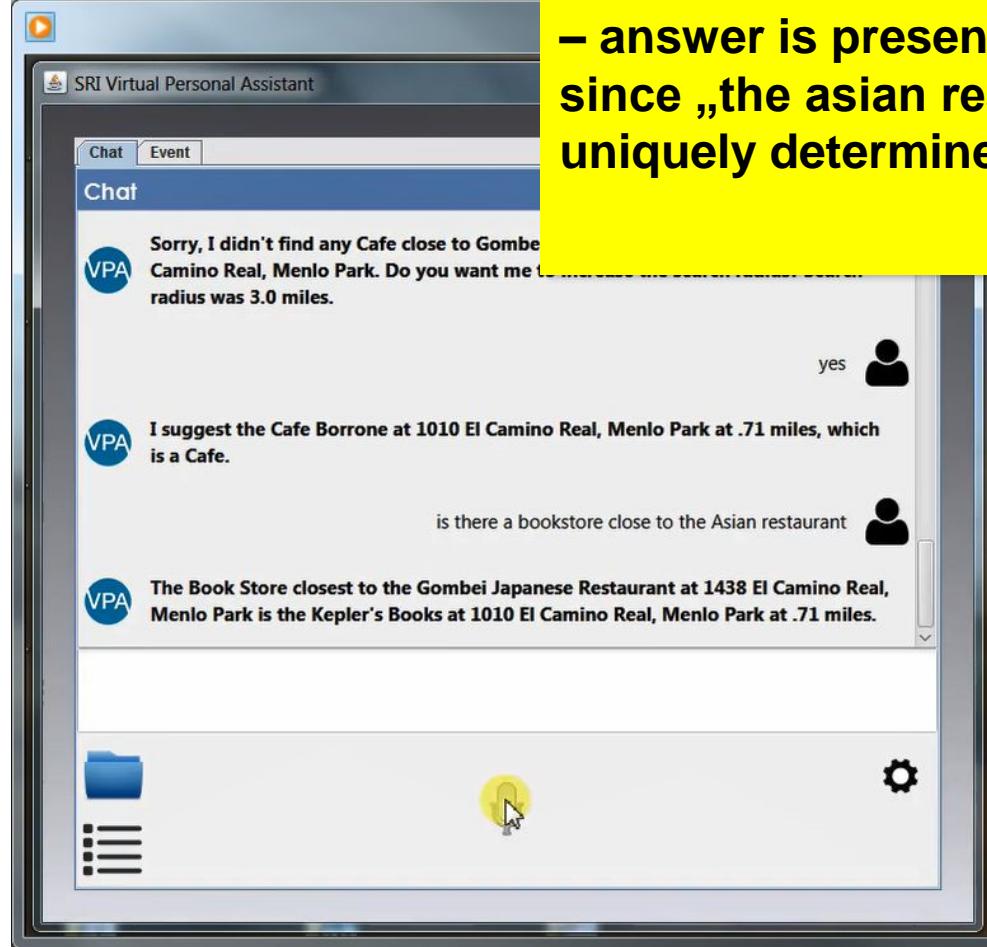


Using „hyperonyms“ in anaphora –
„the asian restaurant“

The chinese restaurant is an asian
restaurant, but the pizzerias are not
-> Not ambiguous!



OntoVPA Demo 20 / 23

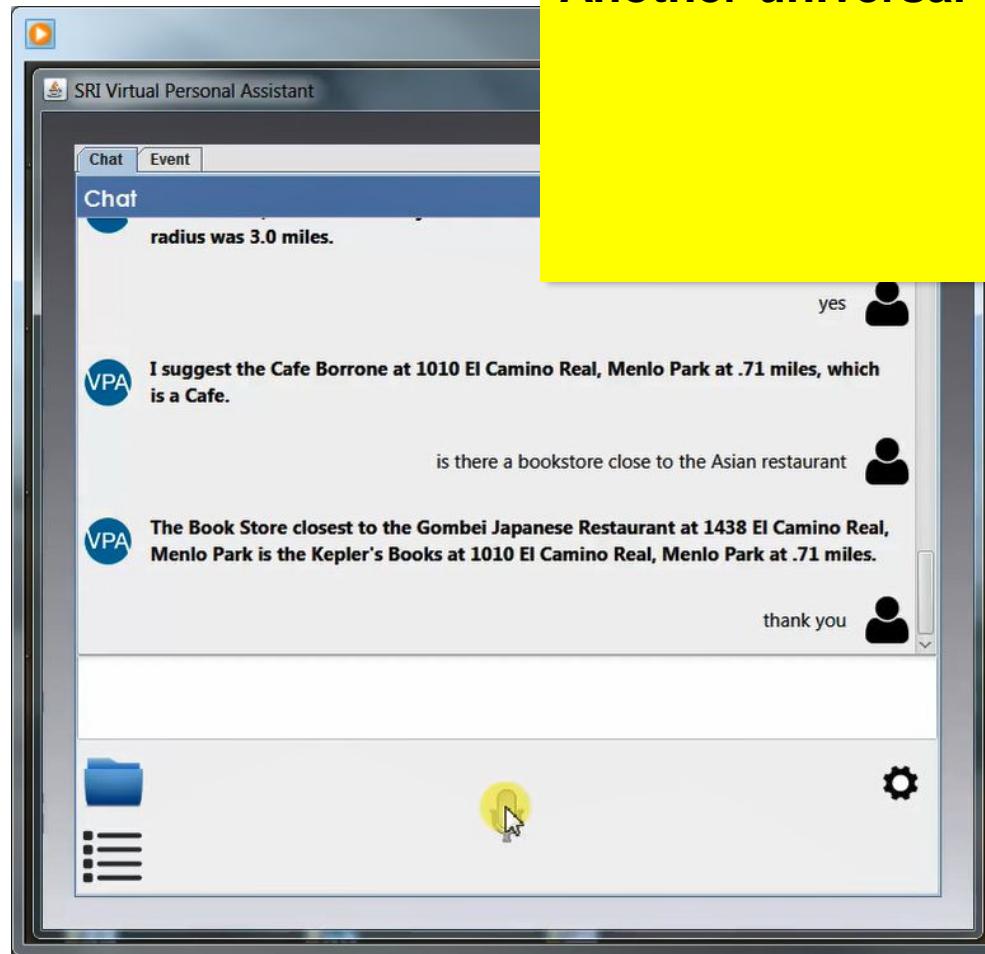


**Does not ask user to disambiguate
– answer is presented right away,
since „the asian restaurant“ is
uniquely determined.**



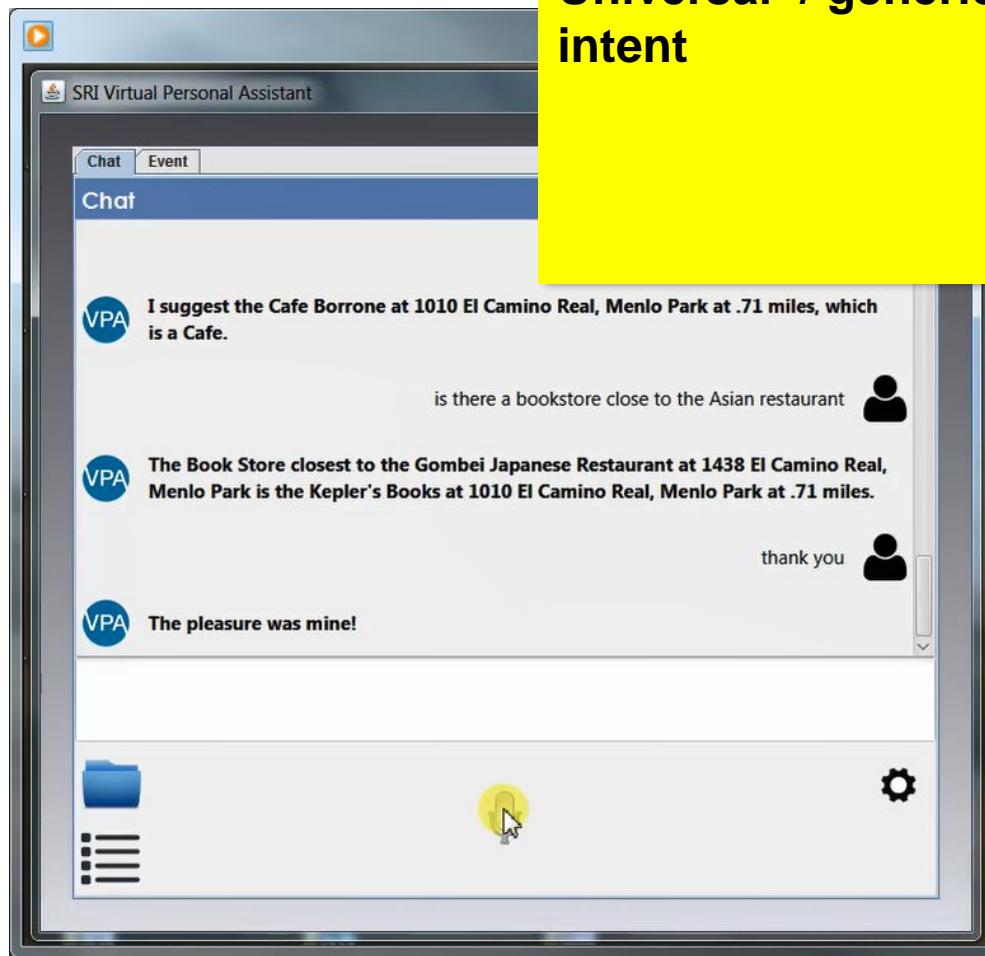
OntoVPA Demo 21 / 23

Another universal - „thank you“



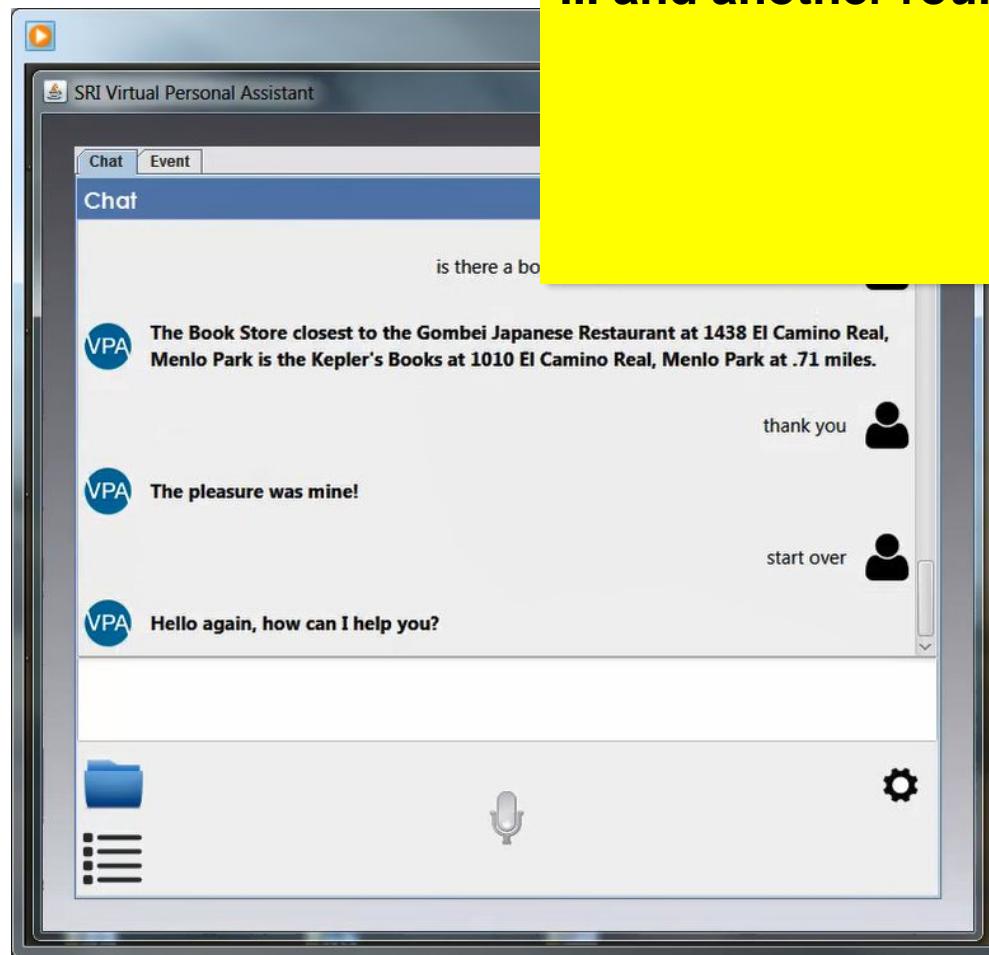
OntoVPA Demo 22 / 23

Universal / generic „Start over,, intent

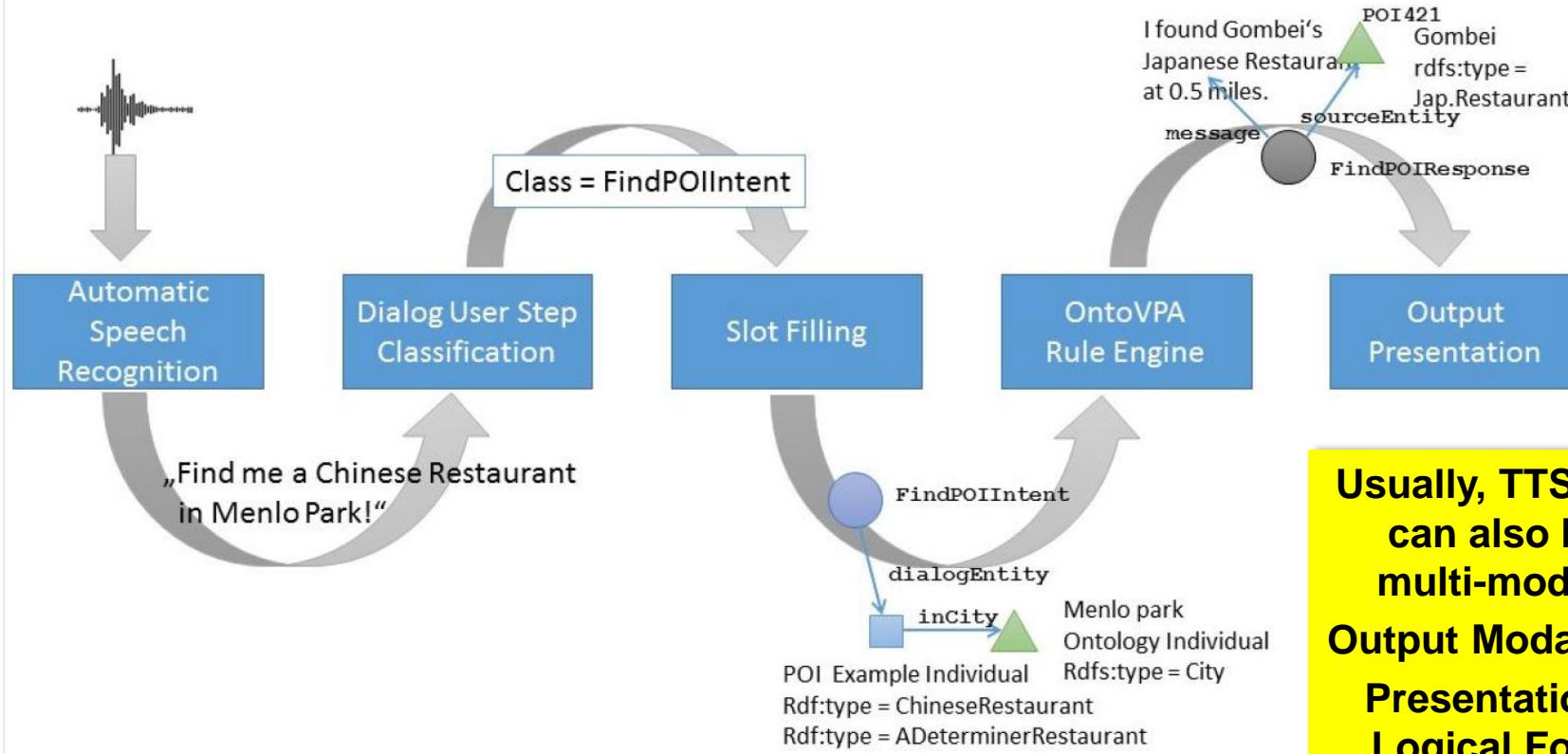


OntoVPA Demo 23 / 23

... and another round 😊



OntoVPA Processing Pipeline



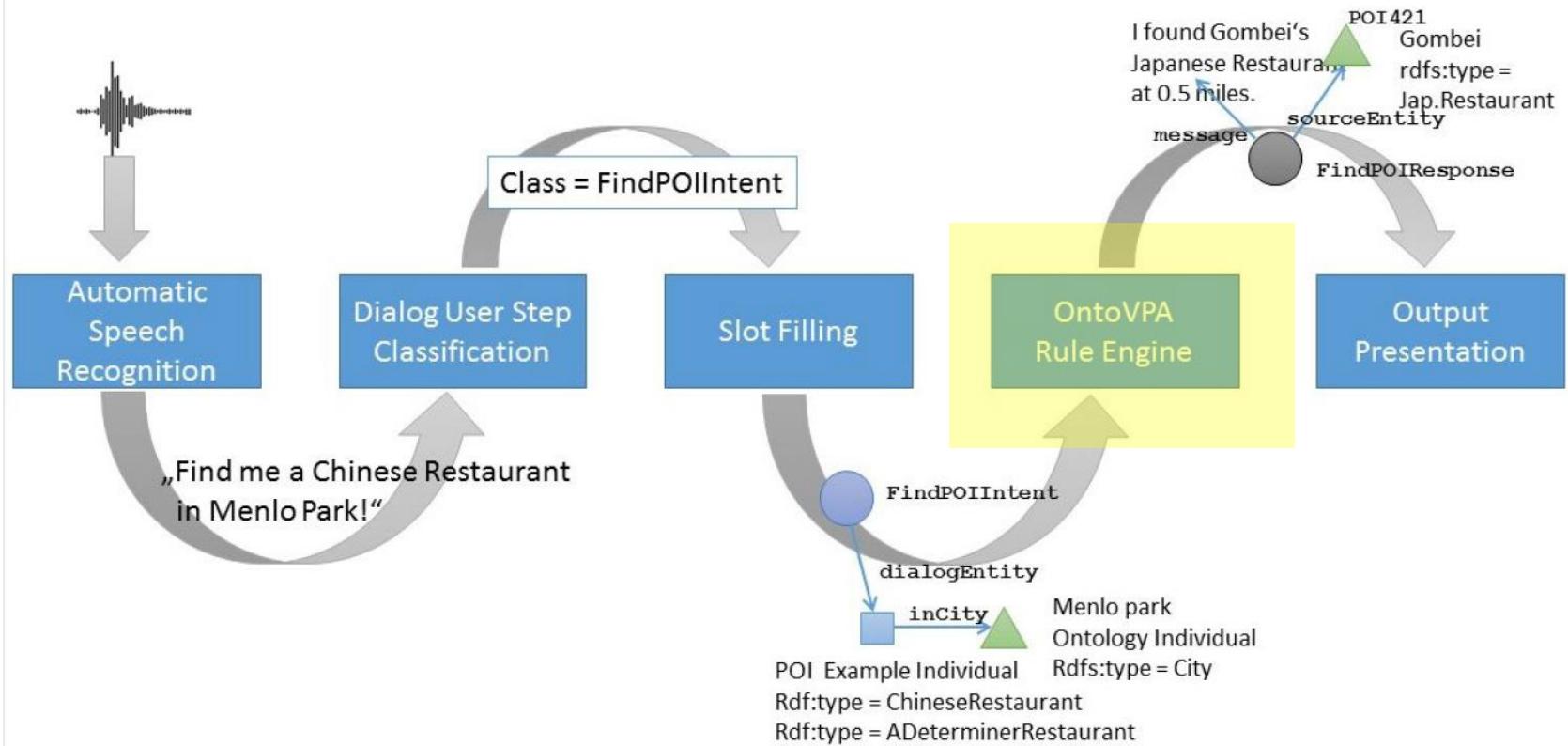
Usually, TTS, but
can also be
multi-modal!

Output Modalities
Presentations
Logical Form

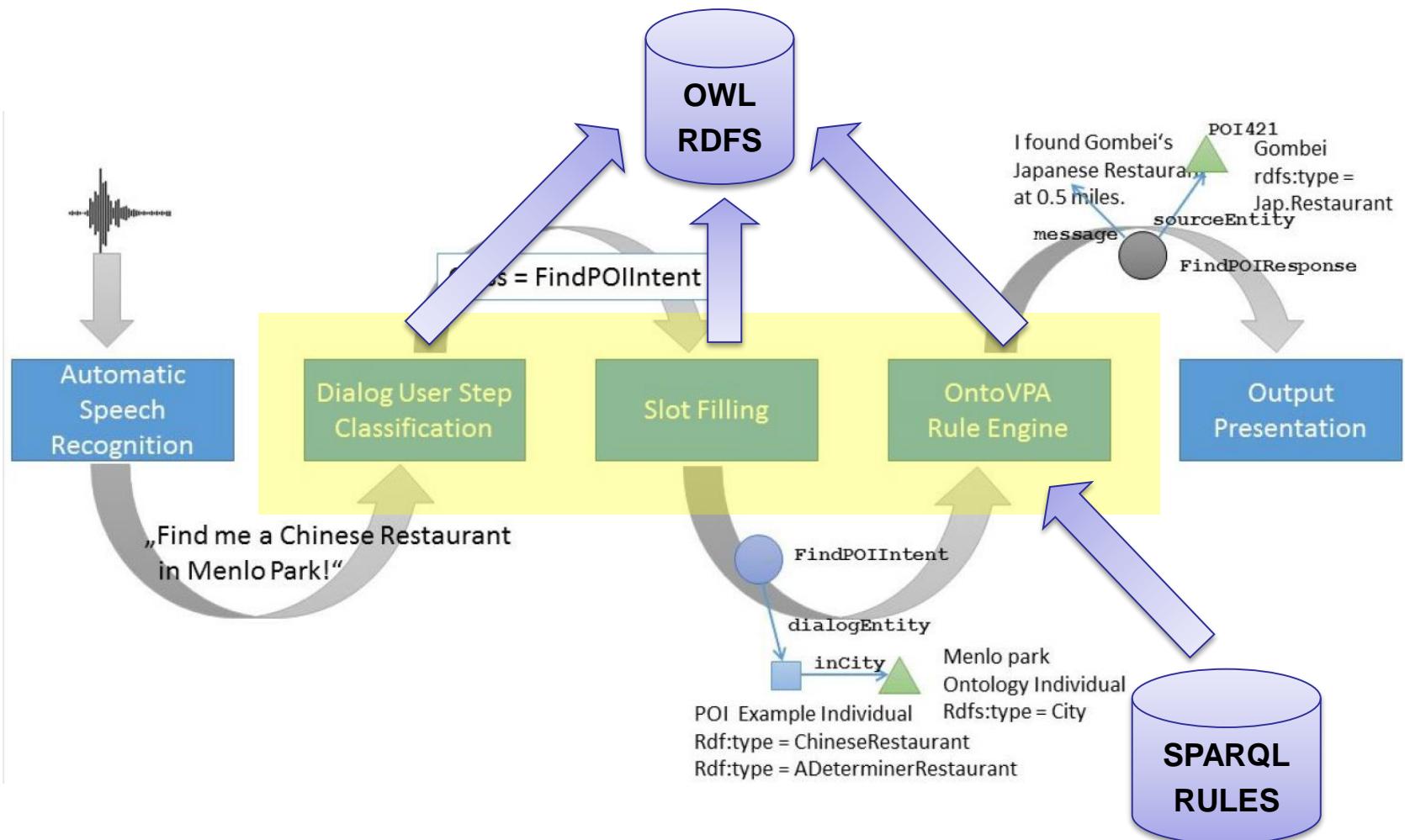
“Plug & Play” Component Architecture –
ASR, TTS, Classifier exchangeable.



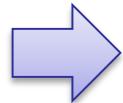
OntoVPA Processing Pipeline – DMS



OntoVPA Processing Pipeline – Ontology Usage



Structure of OntoVPA Ontology



- **Domain Ontology (Static)**

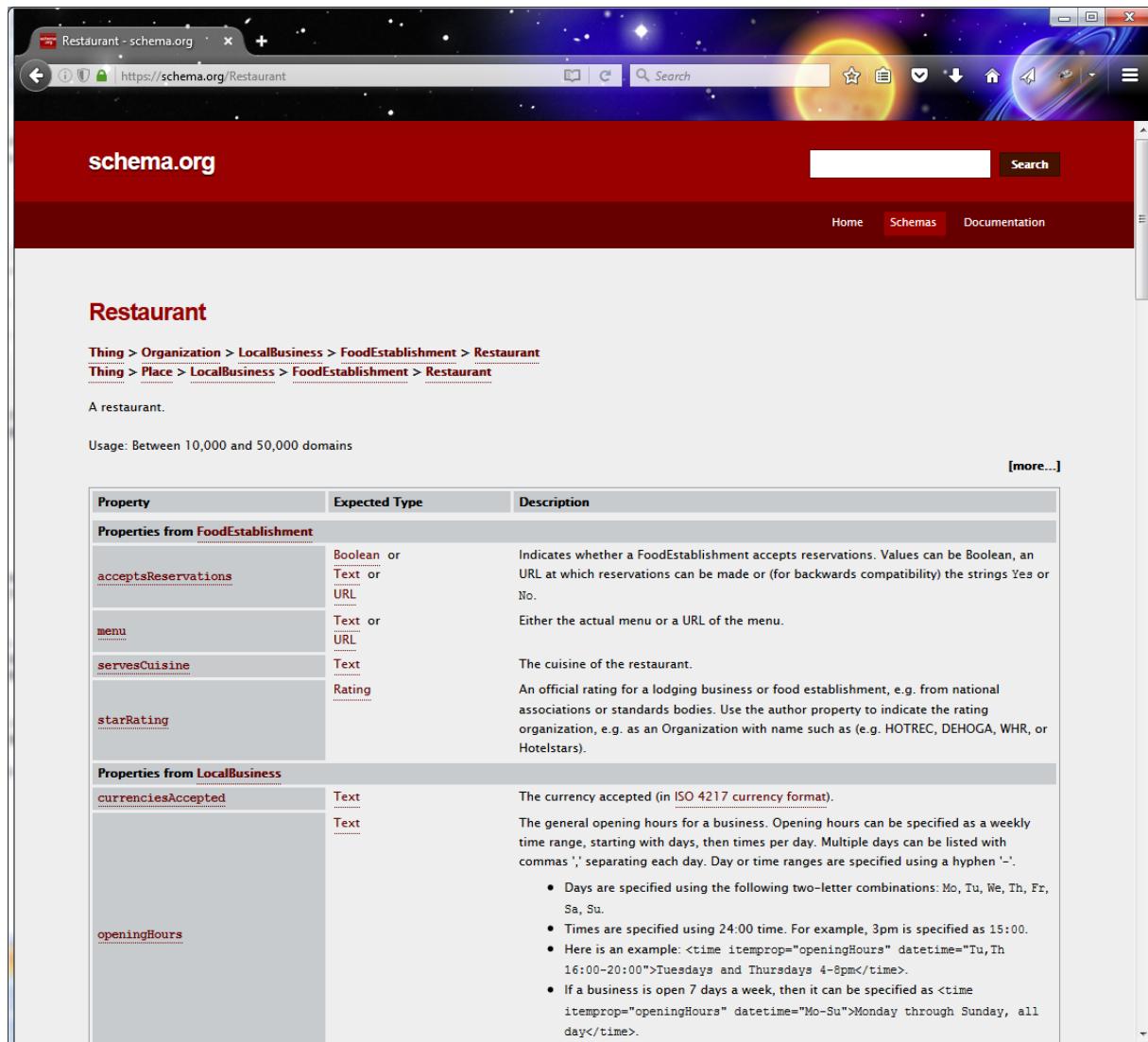
- Domain Vocabulary („TBox“)
 - Classes: POI, Restaurant, ChineseRestaurant, City, ...
 - Relations: inCity, nearBy, name, hasAddress, ...
- Reuse existing ontologies where possible

OWL2 offers

- Multiple inheritance for classes & properties (relations)
- Multiple types per individual
- Semi-structured “data” representation
- Optional & required properties / parameters



RDFS Version of Schema.org „Upper Level Ontology“



The screenshot shows a web browser window with the URL <https://schema.org/Restaurant>. The page title is "Restaurant". Below the title, there are two breadcrumb paths:
Thing > Organization > LocalBusiness > FoodEstablishment > Restaurant
Thing > Place > LocalBusiness > FoodEstablishment > Restaurant

The main content area starts with the definition: "A restaurant." followed by "Usage: Between 10,000 and 50,000 domains". A "[more...]" link is present. Below this, there is a table with properties:

Property	Expected Type	Description
Properties from FoodEstablishment		
acceptsReservations	Boolean or Text or URL	Indicates whether a FoodEstablishment accepts reservations. Values can be Boolean, an URL at which reservations can be made or (for backwards compatibility) the strings 'Yes' or 'No'.
menu	Text or URL	Either the actual menu or a URL of the menu.
servesCuisine	Text	The cuisine of the restaurant.
starRating	Rating	An official rating for a lodging business or food establishment, e.g. from national associations or standards bodies. Use the author property to indicate the rating organization, e.g. as an Organization with name such as (e.g. HOTREC, DEHOGA, WHR, or Hotelstars).
Properties from LocalBusiness		
currenciesAccepted	Text	The currency accepted (in ISO 4217 currency format).
openingHours	Text	The general opening hours for a business. Opening hours can be specified as a weekly time range, starting with days, then times per day. Multiple days can be listed with commas ',' separating each day. Day or time ranges are specified using a hyphen '-'.

- Person
- Place
 - Accommodation
 - Apartment
 - CampingPitch
 - House
 - SingleFamilyResidence
 - Room
 - HotelRoom
 - MeetingRoom
 - Suite
- AdministrativeArea
 - City
 - Country
 - State



POI Domain Ontology – Subclasses of schema:Restaurant



Definition of “Point of Interest (POI)” Class

Description: POI

Equivalent To + schema:Place

SubClass Of +

- Location and Point and (directlyInside **only** City) and (hasFeature **only** SearchFeature) and (locatedAt **only** Street) and (streetNumber **only** xsd:string) and (nearTo **max 1** POI) and (openingAt **only** rdfs:Literal)
- Location
- Point
- schema:Thing
- SchemaOrgThing

General class axioms +

SubClass Of (Anonymous Ancestor)

- POI
- Entity and SpatialNotion
- schema:Thing
- SpatialEntity and (directlyInside **max 1** Region)

Class hierarchy (inferred): POI

- venezuelanRestaurant
- schema:Bakery ≡ Bakery
- schema:BarOrPub ≡ Bar
- Casino ≡ schema:Casino
- schema:Casino ≡ Casino
- schema:Brewery
- schema:Restaurant ≡ Restaurant
 - AmericanRestaurant
 - AsianRestaurant
 - DinerOrCoffeeShop
- FastFoodPlace ≡ FastFoodRestaurant ≡ schema:FastFoodRestaurant
 - BurgerPlace
 - MexicanRestaurant ≡ Taqueria
 - SandwichPlace ≡ SandwichRestaurant
 - SandwichRestaurant ≡ SandwichPlace
 - Taqueria ≡ MexicanRestaurant
- FastFoodRestaurant ≡ schema:FastFoodRestaurant ≡ FastFoodPlace
 - FishRestaurant ≡ SeaFoodRestaurant
 - FrenchRestaurant
 - GermanRestaurant
 - GreekRestaurant
 - Grill
 - IceCreamPlace ≡ IceCreamRestaurant ≡ schema:IceCreamShop
 - IceCreamRestaurant ≡ IceCreamPlace ≡ schema:IceCreamShop
 - IceCreamShop ≡ IceCreamPlace ≡ IceCreamRestaurant ≡ schema:IceCreamShop
 - IndianRestaurant
 - ItalianRestaurant
- schema:FastFoodRestaurant ≡ FastFoodRestaurant ≡ FastFoodPlace
- schema:IceCreamShop ≡ IceCreamPlace ≡ IceCreamRestaurant
- SeaFoodRestaurant ≡ FishRestaurant
- SpanishRestaurant

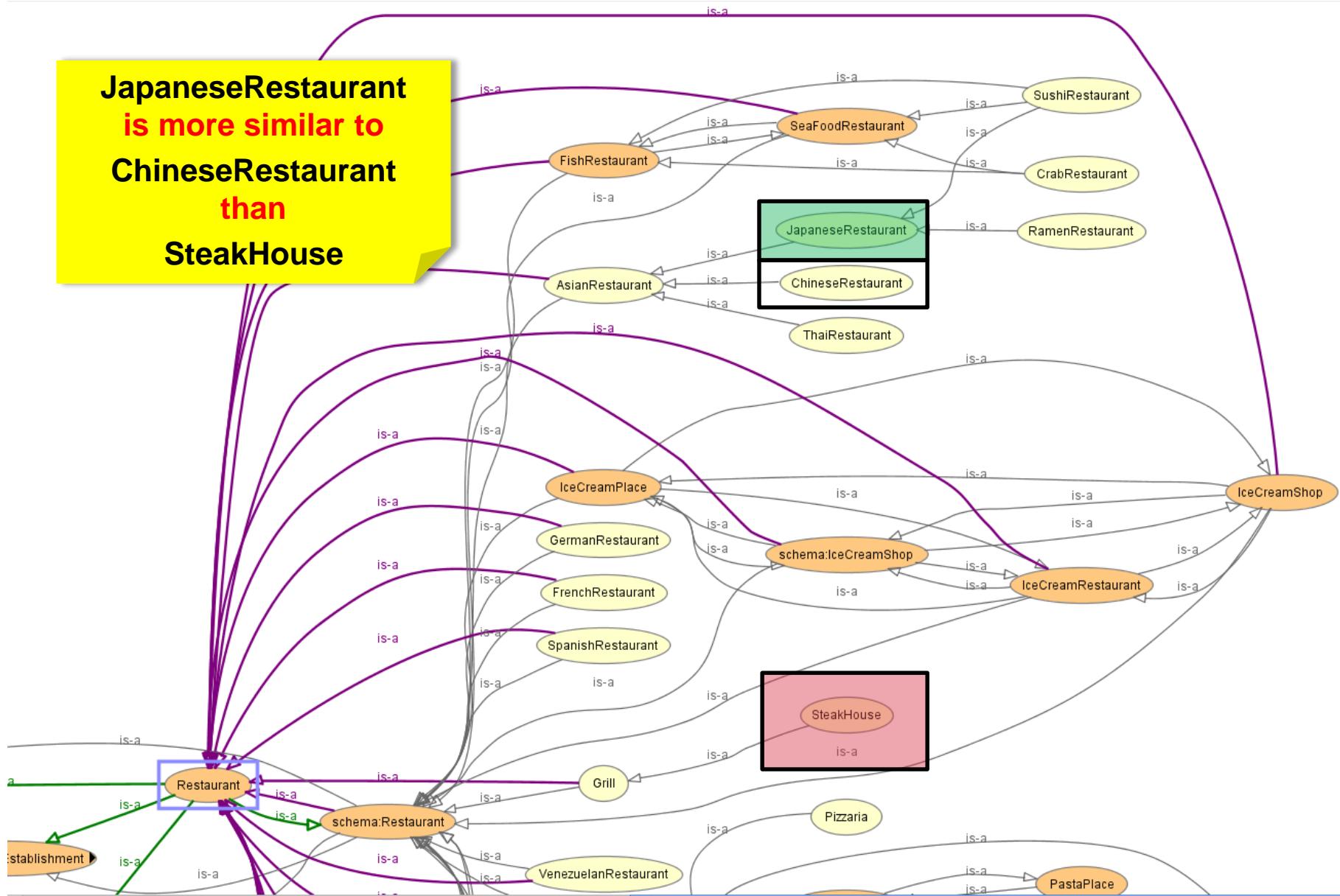
Equivalent to
schema:Place
Bridge Axiom

Parameters / Slots & Ranges



Ontology-Based Similarity

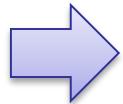
JapaneseRestaurant
is more similar to
ChineseRestaurant
than
SteakHouse



Description of OntoVPA Ontology

- **Domain Ontology (Static)**

- Domain Vocabulary („TBox“)
 - Classes: POI, Restaurant, ChineseRestaurant, City, ...
 - Relations: inCity, nearBy, name, hasAddress, ...



- Domain „Database“ („ABox“)
 - Instances & relationships: restaurant123, inCity(restaurant123, menloPark), ...

OWL2 offers

- Multiple inheritance for classes & properties (relations)
- Multiple types per individual
- Semi-structured “data” representation
- Optional & required properties / parameters



Ontology Domain Instances – “POI Database” ABox

The screenshot displays the OWLviz interface for an ontology's ABox. On the left, the 'Class hierarchy' tab is active, showing a detailed tree of restaurant categories such as GreekRestaurant, ItalianRestaurant, JapaneseRestaurant, and various subtypes like RamenRestaurant and SushiRestaurant. In the center, the 'Individuals by type (inferred)' tab is selected, focusing on the individual `poi225`, which is identified as a `JapaneseRestaurant`. On the right, the 'Annotations' tab is open, listing properties like `rdfs:label` with the value "Gombei Japanese Restaurant at 1438 El Camino Real", and `assertedType` with values "Food", "JapaneseRestaurant", and "SourcePOI". Below these, the `hasFeature` annotation is shown with values "AcceptsReservations" and "Beef". A yellow callout box on the right side of the interface contains the following text:

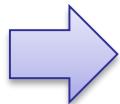
- However, “sources” can also be materialized dynamically, on the fly as requested
- API call (Web Service)
- Federated Sparql / Sparql endpoint



Structure of OntoVPA Ontology

- **Domain Ontology (Static)**

- Domain Vocabulary („TBox“)
 - Classes: POI, Restaurant, ChineseRestaurant, City, ...
 - Relations: inCity, nearBy, name, hasAddress, ...
- Domain „Database“ („ABox“)
 - Instances & relationships: restaurant123, inCity(restaurant123, menloPark), ...



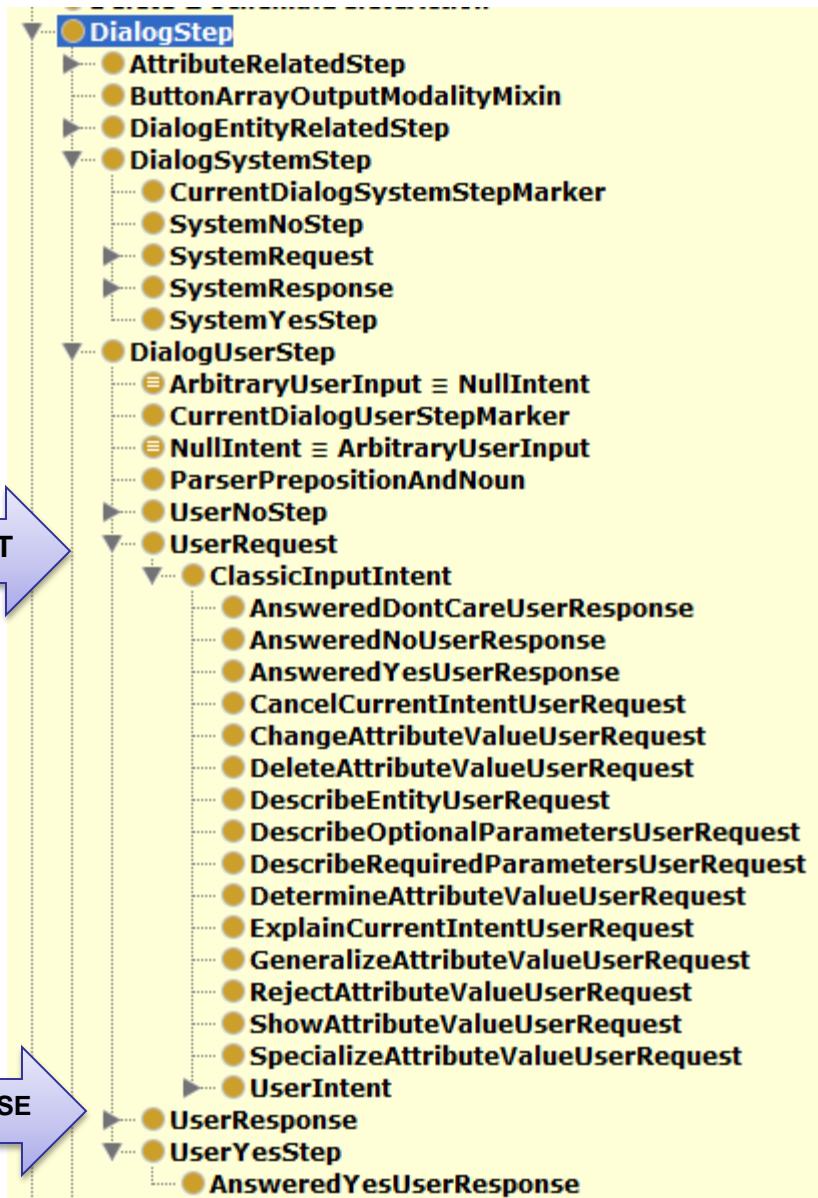
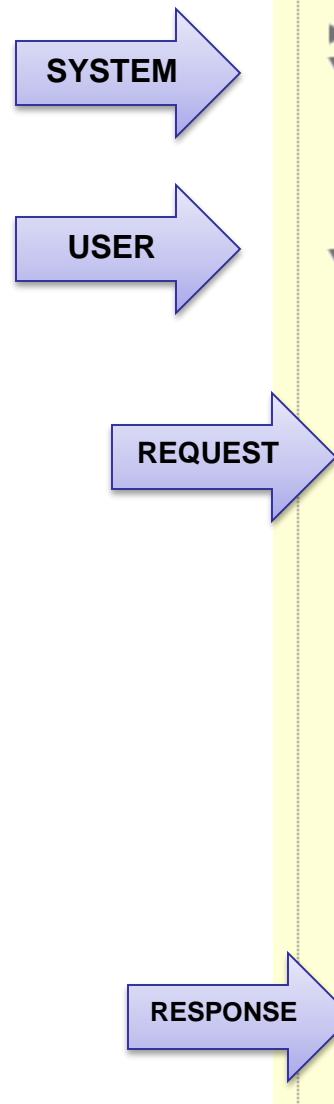
- **Dialogue Ontology (Static)**

- Speech Act Theory Inspired
 - Request, Response, UserRequest, UserResponse, SystemRequest, SystemResponse, Yes/No, Greeting, ...
 - UserIntents are special UserRequests: FindPOIntent

OWL2 offers

- Multiple inheritance for classes & properties (relations)
- Multiple types per individual
- Semi-structured “data” representation
- Optional & required properties / parameters

OntoVPA Dialogue Ontology – Dialogue Steps & Subclasses



Structure of “Find Point of Interest” Intent

String-to-Class Mapping

Description: SearchPoiSchemaOrgIntent

Equivalent To +

SubClass Of +

- (parserKeyword value "FindPoi"^^xsd:string) and (parserKeyword value "SearchPoi"^^xsd:string)
- SchemaOrgSearchIntent
- and (maxDist some xsd:real)
and (dialogEntity only POI)
and (finalExpectedSystemResponse only ShowPoisSy)
and (aboutPrice only xsd:real)
and (maxPrice only xsd:real)
and (minPrice only xsd:real)
and (minSimilarity value "1.0"^^xsd:real)
and (maxDist value "3.0"^^xsd:real)
- SchemaOrgSearchIntent

General class axioms +

SubClass Of (Anonymous Ancestor)

- DialogStep
and (dialogEntity some Entity)
- UserIntent
and (performedAction some Action)
- yesAnswerFollowUpRequest only ConceptMarker
- schema:targetProduct only schema:SoftwareApplication

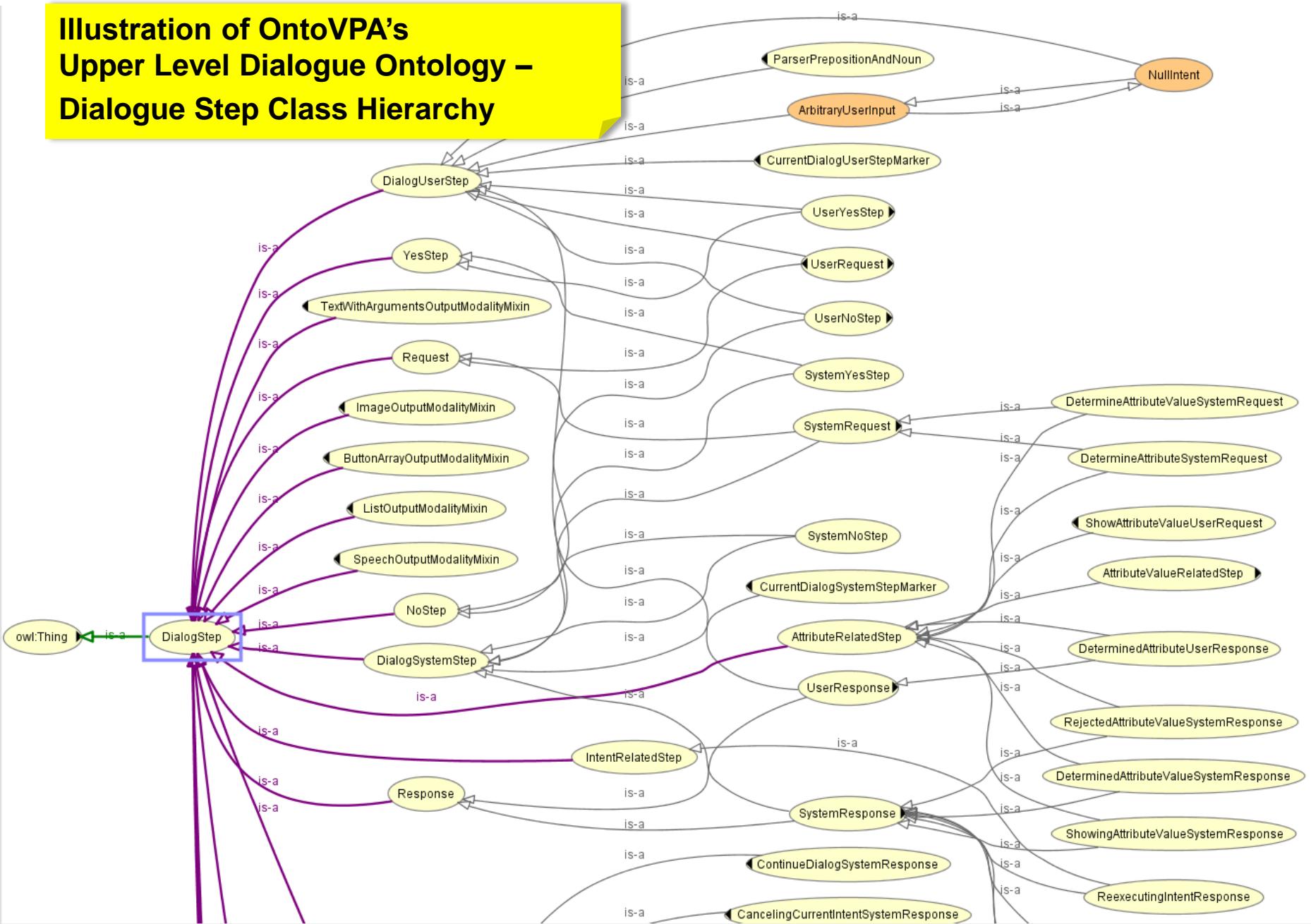
Class hierarchy (inferred): SearchPoiSchemaOrgIntent

- schema:CookAction
- schema:DrawAction
- schema:FilmAction
- schema:PaintAction
- schema:PhotographAction
- schema:WriteAction
- DialogEntityRelatedStep
 - DescribeEntityUserRequest
 - QueryDetailsIntent
 - DescribingDialogEntitySystemResponse
 - ReportDetailsSystemResponse
 - SearchSourceEntityIntent
 - SchemaOrgSearchIntent
 - SearchPoiSchemaOrgIntent
 - FindSimilarRestaurantIntent
 - ShowDialogEntityIntent
 - SchemaOrgShowDialogEntityIntent
 - ShowPoiSchemaOrgIntent
 - ShowingDialogEntitySystemResponse
 - FillUpGas
 - Move ≡ schema:MoveAction
 - schema:ArriveAction
 - schema:DepartAction
 - schema:TravelAction
 - QueryDistanceIntent
 - schema:AchieveAction
 - schema:LoseAction
 - schema:TieAction
 - schema:WinAction
 - schema:AssessAction
 - schema:ChooseAction

Intent Class Hierarchy

Required and optional Parameters, Ranges, ...

Illustration of OntoVPA's Upper Level Dialogue Ontology – Dialogue Step Class Hierarchy



Structure of OntoVPA Ontology

- **Domain Ontology (Static)**

- Domain Vocabulary („TBox“)
 - Classes: POI, Restaurant, ChineseRestaurant, City, ...
 - Relations: inCity, nearBy, name, hasAddress, ...
- Domain „Database“ („ABox“)
 - Instances & relationships: restaurant123, inCity(restaurant123, menloPark), ...

OWL2 offers

- Multiple inheritance for classes & properties (relations)
- Multiple types per individual
- Semi-structured “data” representation
- Optional & required properties / parameters

- **Dialogue Ontology (Static)**

- Speech Act Theory Inspired
 - Request, Response, UserRequest, UserResponse, SystemRequest, SystemResponse, Yes/No, Greeting, ...
 - UserIntents are special UserRequests: FindPOIntent

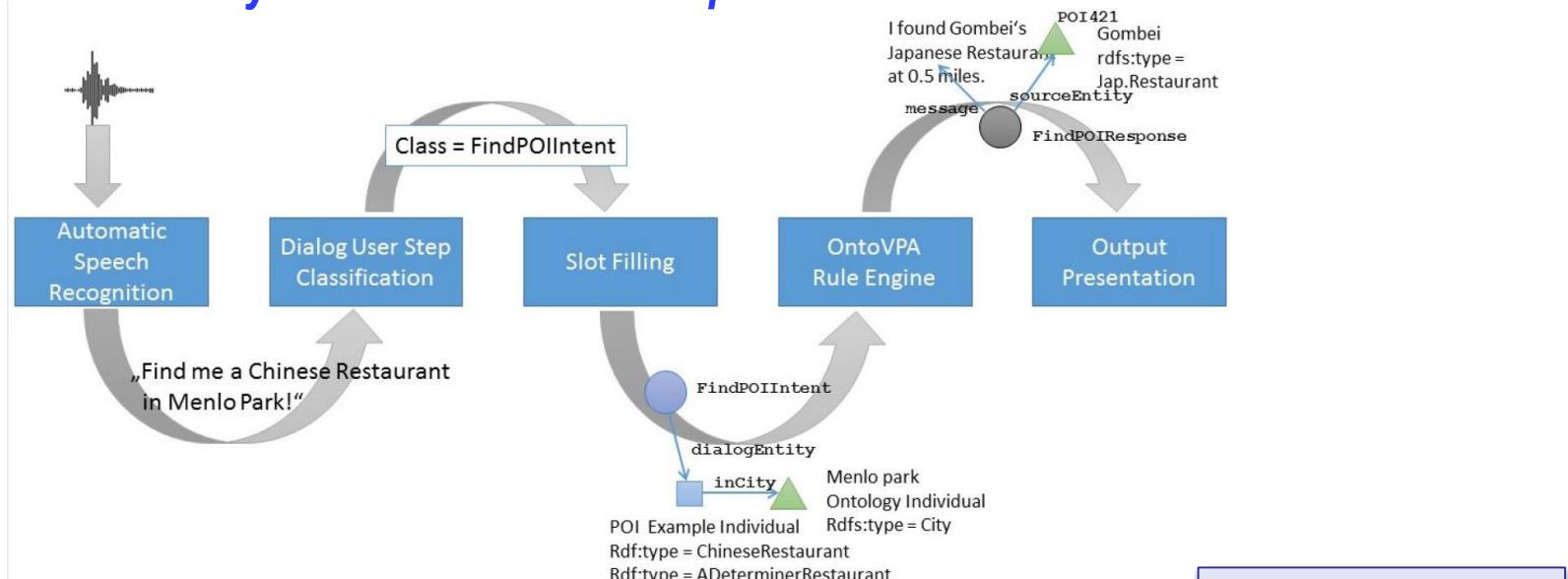
- **Actual Ontology-Based Runtime Dialogue Representation (Dynamic)**

- Dialogue ABox as a „dialogue history graph“
- Dynamically updated by SPARQL rules



Dialogue Dynamics – Runtime Dialogue Representation

*Is there a Japanese Restaurant in Menlo Park?
Can you show IT on the map?*



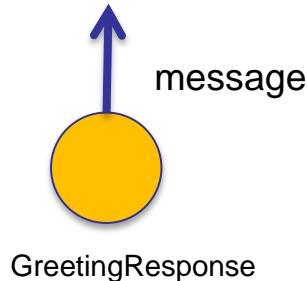
1. Statistical intent classification -> FindPoiIntent
2. Parameter extraction & „slot filling“
$$\text{dialogueEntity} = \{ \text{type : ChineseRestaurant}, \\ \text{inCity : menloPark} \}$$
3. Intent is asserted into dialogue engine
4. Dialogue engine: more interpretation, reasoning, output gen.



Example Dialogue – Dialogue Representation - 1

VPA

Hello, what can I do for you?

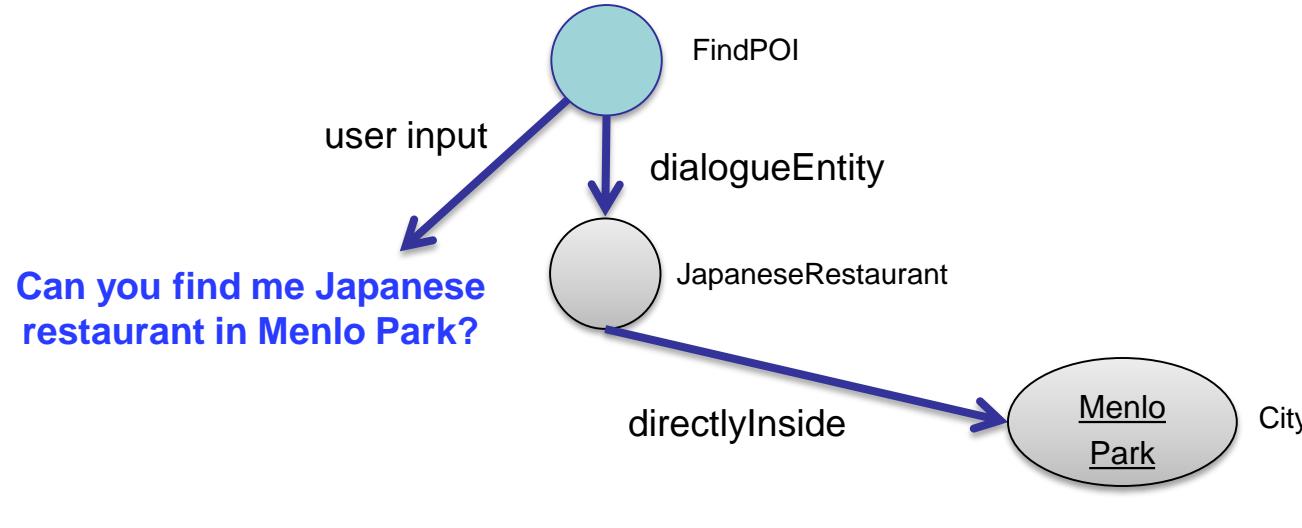
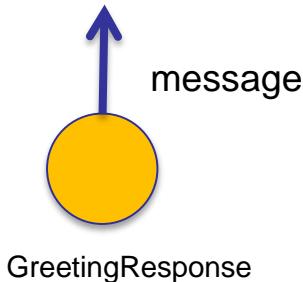


USER

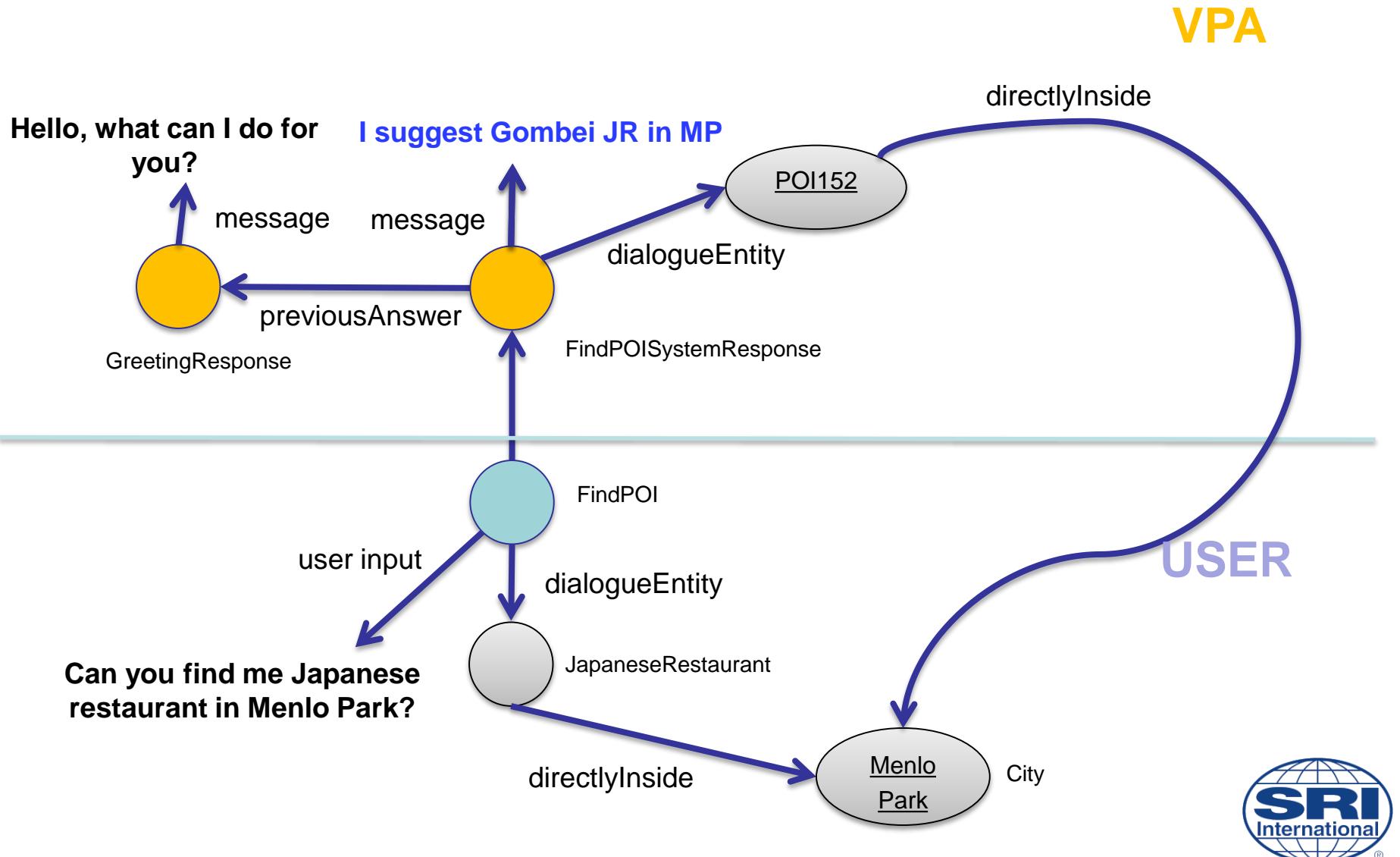


Example Dialogue – Dialogue Representation - 2

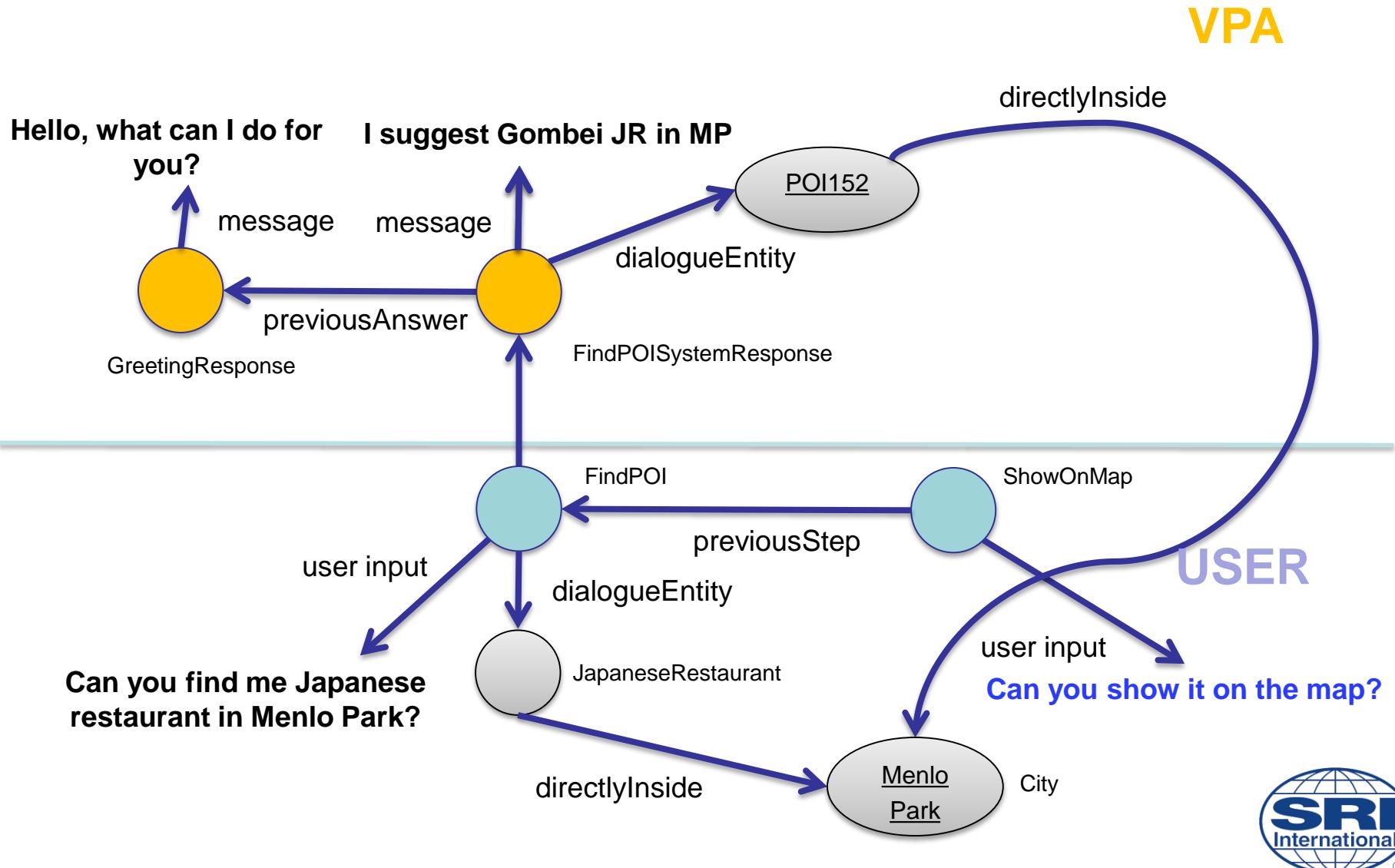
Hello, what can I do for you?



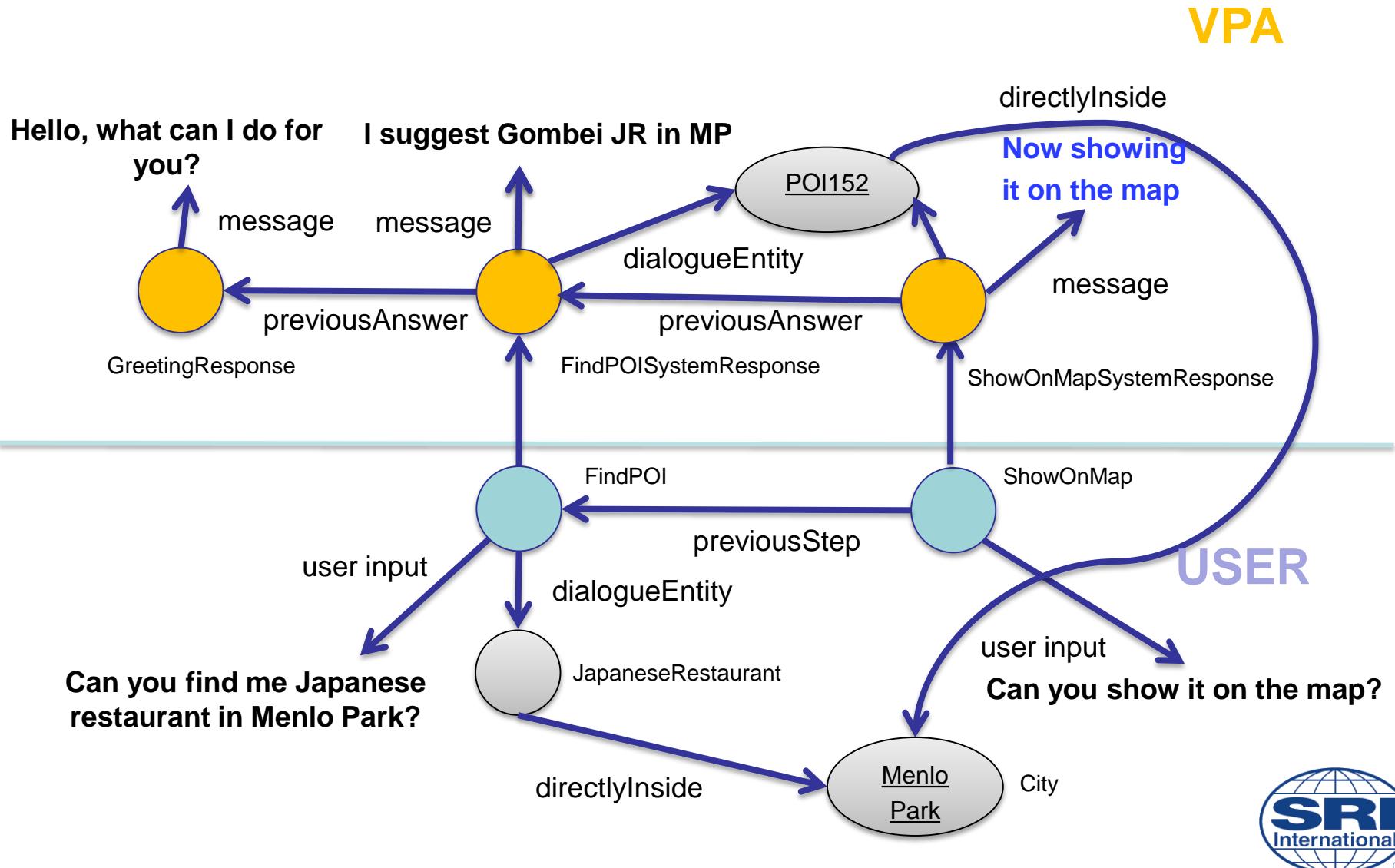
Example Dialogue – Dialogue Representation - 3



Example Dialogue – Dialogue Representation - 4



Example Dialogue – Dialogue Representation - 5



Structure of OntoVPA Ontology

- **Domain Ontology (Static)**

- Domain Vocabulary („TBox“)
 - Classes: POI, Restaurant, ChineseRestaurant, City, ...
 - Relations: inCity, nearBy, name, hasAddress, ...
- Domain „Database“ („ABox“)
 - Instances & relationships: restaurant123, inCity(restaurant123, menloPark), ...

OWL2 offers

- Multiple inheritance for classes & properties (relations)
- Multiple types per individual
- Semi-structured “data” representation
- Optional & required properties / parameters

- **Dialogue Ontology (Static)**

- Speech Act Theory Inspired
 - Request, Response, UserRequest, UserResponse, SystemRequest, SystemResponse, Yes/No, Greeting, ...
 - UserIntents are special UserRequests: FindPOIntent

- **Actual Runtime Dialogue Representation (Dynamic)**

- Dialogue ABox as a „dialogue history graph“
- Dynamically updated by SPARQL rules

- **Upper Level Ontologies & Generic Rule Layer**



OntoVPA Rule Engine (based on Apache Jena)

- **Why SPARQL? SPARQL is a RDF(s) query language...**
 - However, due to CONSTRUCT, it is also a “one rule” rule language
 - Can construct consequence triples
 - Add rule application strategy => rule engine
- **DMS-specific, but generic rule engine in OntoVPA**
 - Defeasibility / conflict resolution
 - Hierarchical layers of rule (augmentation, dialogue, recover)
 - Rules can fire or disable other rules, or loop, ...
- **Useful features of SPARQL**
 - CONSTRUCT
 - User-defined SPARQL functions (interface to Java world)
 - “Second-order like” expressivity
 - Fast, W3C standard, mature implementations
 - Aware of ontology-consequences, customizable (Entailment Regimes)



Query By Example Search – SPARQL Expressivity

- Candidate POI from data source has to fulfil all requirements stated by the exemplar POI
- An source POI $?x$ is a match if it has all the properties P that the exemplar dialogueEntity has (but may have more)

$$\forall P \forall val : P(dialogEntity, val) \rightarrow P(x, val)$$

- Hence, there is no property P that dialogueEntity has, that $?x$ does not have

```
FILTER NOT EXISTS {  
    dialogueEntity ?P ?val  
    FILTER NOT EXISTS {  
        ?x ?P ?val  
    }  
}
```

Much of OntoVPA's generic behavior is specified using concise, expressive and generic "second-order like" rules in its upper level rule layer.



Summary

- **Explicit, symbolic, declarative representations with formal semantics**
 - Full dialogue history, fully “reflexive”
 - Transparent, understandable, explainable, reliable
 - Data / knowledge driven
- **Multi-lingual, multi-domain, multi-modal**
- **Based on W3C® Standards (OWL, RDFS, SPARQL, ...)**
 - Rich tool stack for authoring
 - Compatibility
 - No “vendor lock in” for customers
 - Less steep learning curve for customers and better acceptance (books)
- **Reusability & Core Functionalities**
 - Dialogue management specific upper-level ontologies & upper level rules
 - Ontologies (“off the shelf”) and services on the Web exploitable
 - SPARQL endpoints, Semantic Web Services, ...

=> ***Flexible, Powerful, and Cost-Efficient VPA Development***



Thank you!

Questions?



Generic, Reusable OntoVPA Behavior

- **Generic Rules and Behavior For**
 - Greeting, Cancel Intent, Exit
 - Identify and inquire for missing required parameters
 - Merging and interpretation of „arbitrary user input“
 - What does Palo Alto mean in the context of the current dialogue?
 - Identify previous user request from history that has a parameter for which the input makes sense, and re-execute that intent
 - Anaphora / reference / pronoun resolution
(it, the <type>, a <type>, my, her, his, ..)
 - Uses dialogue history and every object / dialogue entity „that was discussed“
 - Uses „ontological similarity“ of between referring and referred to entity
 - Disambiguation in case reference is ambiguous (Stanford – Uni or city?)
 - More than one open intent – „LIFO“ stack of open intents
 - „interrupt“ and resume („... But now, let us continue:“)
 - But full control over the stack by means of rules if not desirable
 - Rules can cancel, reopen, reexecute intents



Generic, Reusable OntoVPA Behavior

- **Complex Workflows**

- Complex Intents / complex goals can require more than one sub-dialogue step
 - Multiple steps required – „Book my business trip!“
 - Sub-dialogues possible
 - An intent is not complete until „the final expected system response“ is reached
 - Arbitrary state information for managing that information state can be put on the agenda by rules for
 - State must not be passed along by rules
 - In addition to the dialogue history, the dialogue state can contain arbitrary complex structured objects that represent the information state
 - Important for „book keeping“ of the state



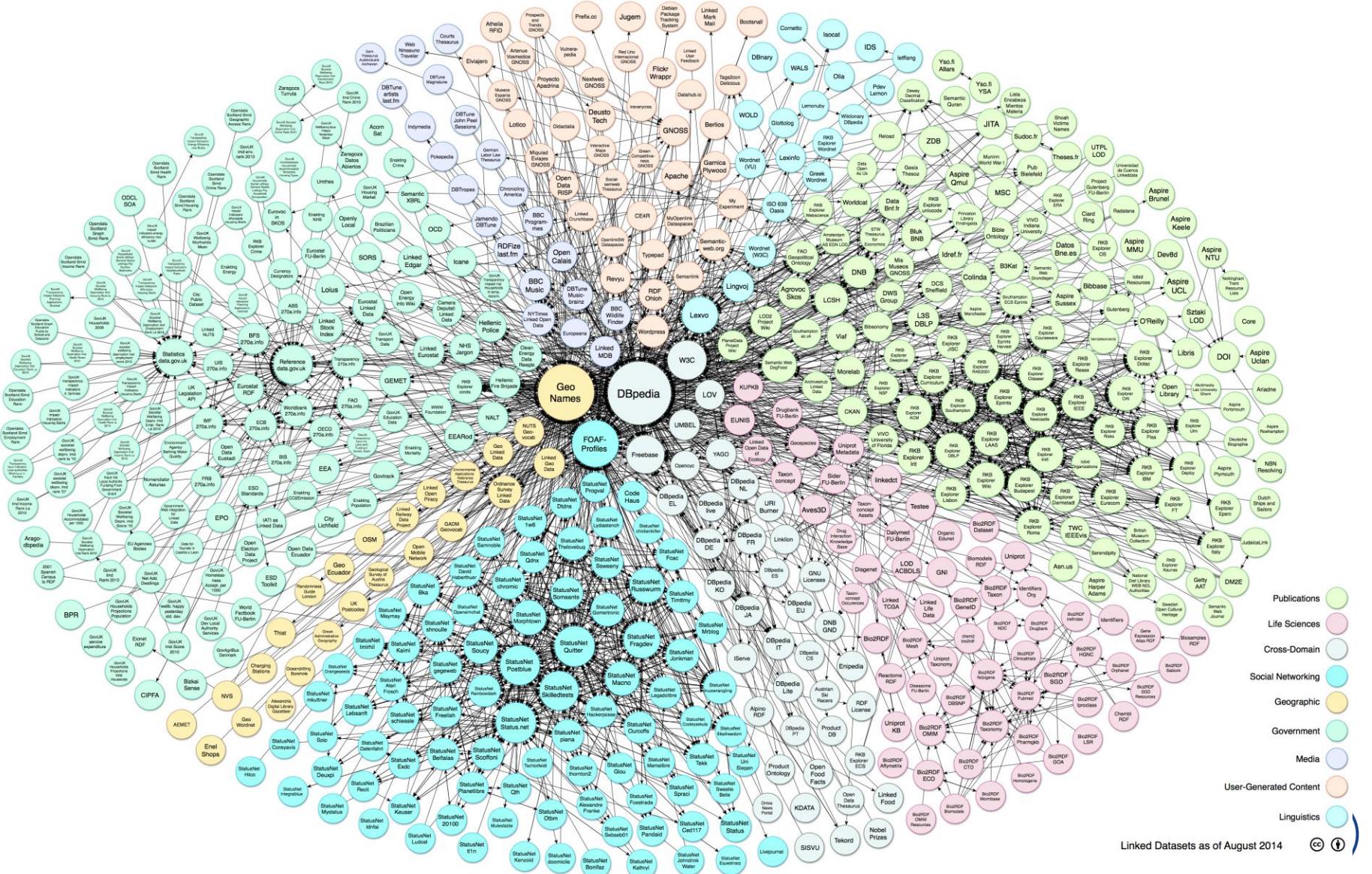
Example SPARQL Rule – GreetingIntent & Response

```
g HelloToplevel
Toplevel - VPAGreeting received

BASE    <http://vpa.sri.com/EVVPA/1/>
PREFIX afn:  <http://jena.hpl.hp.com/ARQ/function#>
PREFIX f:   <http://vpa.sri.com/EVVPA/1/functions>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX res:  <http://vpa.sri.com/EVVPA/1/res/>
PREFIX ns:   <http://vpa.sri.com/EVVPA/1/>
PREFIX xfn:  <http://www.w3.org/2005>xpath-functions#>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX java: <http://vpa.sri.com/EVVPA/1/java/>

CONSTRUCT
{ ?o rdf:type java:OutputMessage .
  ?o java:message "Hello, good to see you again! How can I help you today?" .}
WHERE
{ ?i rdf:type ns:current_intent .
  ?i rdf:type ns:input_intent .
  ?i rdf:type ns:VpaGreeting .
  ?i ns:inferred ?o .
  ?o rdf:type ns:current_intent .
  ?o rdf:type ns:output_intent
}
LIMIT 1
```

Public Sparql Endpoints via Federated SPARQL



WikiData

- Similar to DBPedia, but crowd sourced like Wikipedia
- Web-based knowledge base /content editor

The screenshot shows the main page of Wikidata. At the top, there's a navigation bar with links for English, Not logged in, Talk, Contributions, Create account, and Log in. Below the navigation bar is a search bar. The main content area features a large, semi-transparent central box containing the text "Welcome to Wikidata" and "the free knowledge base with 23,907,705 data items that anyone can edit". Above this box is a network diagram with red lines connecting nodes labeled "open", "multilingual", and "free" on the left, and green lines connecting nodes labeled "collaborative", "linked", and "structured" on the right. Below the central box are two main sections: "Welcome!" and "Learn about data". The "Welcome!" section contains text about Wikidata being a free and open knowledge base, its role as central storage for structured data, and its support for other Wikimedia projects. It also mentions that Wikidata content is available under a free license and can be interlinked to other open data sets. The "Learn about data" section provides an introduction to data literacy with a call to action: "Develop and improve your data literacy through content designed to get you up to speed and feeling comfortable with the fundamentals in no time." Below this section are three images: a globe, a stack of books, and a mountain peak. Below each image is a caption: "item: Earth (Q2)", "property: highest point (P610)", and "custom value: Mount Everest (Q813)". On the left side of the page, there's a sidebar with links for Main page, Community portal, Project chat, Create a new item, Item by title, Recent changes, Random item, Query Service, Nearby, Help, Donate, Print/export, Create a book, Download as PDF, Printable version, In other projects, and Tools. The "Welcome!" section also has a "Get involved" button at the bottom.