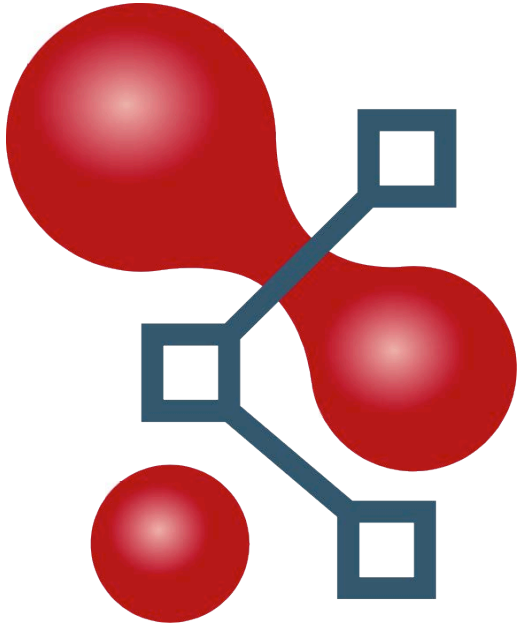
A Framework for the Automatic Extraction of Rules from Online Text

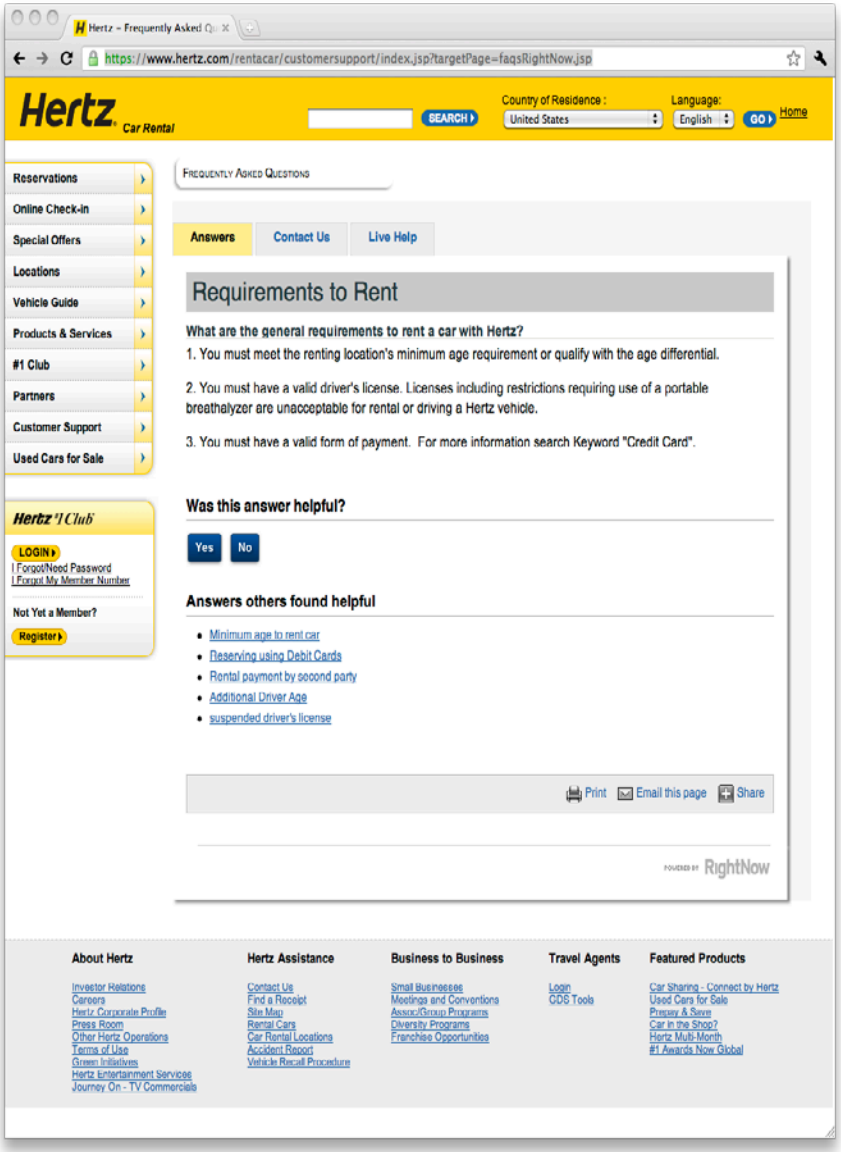


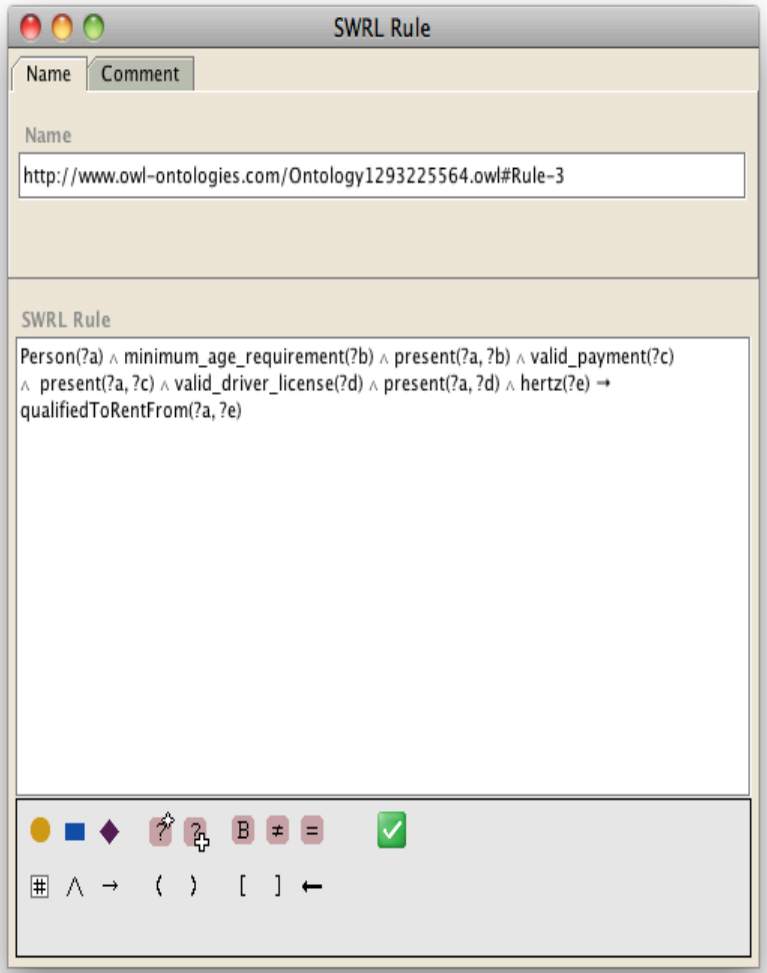
Saeed Hassanpour, Martin J. O'Connor, Amar Das

Stanford Center for Biomedical Informatics Research
Stanford, CA, U.S.A.

RuleML, Barcelona, Spain, July 21st, 2011

Automatic Rule Extraction from Text





General Requirements in Extracting Rules from Text

- Ability to recognize domain concepts in text
- Recognize *relationships* between concepts
- Assemble these relationships into chains
- Requires understanding grammatical structure of sentence to detect relationships

Method Overview

- Method requires:
  - User-selected text containing rule-like information
  - Existing OWL domain ontology
  - Existing SWRL rules
- Using these in combination with NLP techniques, automatically generate SWRL rule(s) from specified text

What is SWRL?

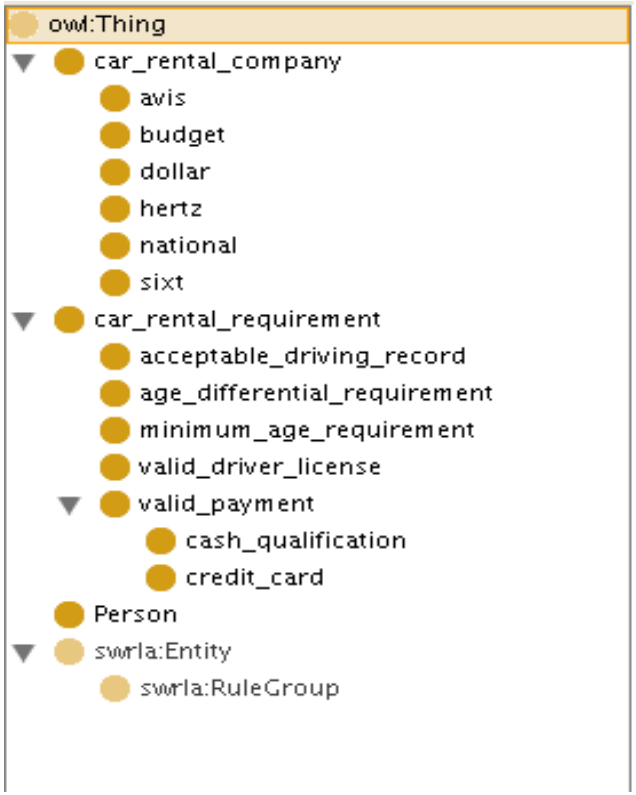
- SWRL is an acronym for Semantic Web Rule Language
- SWRL includes a high-level abstract syntax for Horn-like rules
- SWRL is an OWL-based rule language
- All rules are expressed in terms of OWL concepts (classes, properties, individuals)
  - e.g., Person(?a) ^ minimum_age_requirement(?b) ^ meet(?a, ?b) -> EligibleToRent(?a)

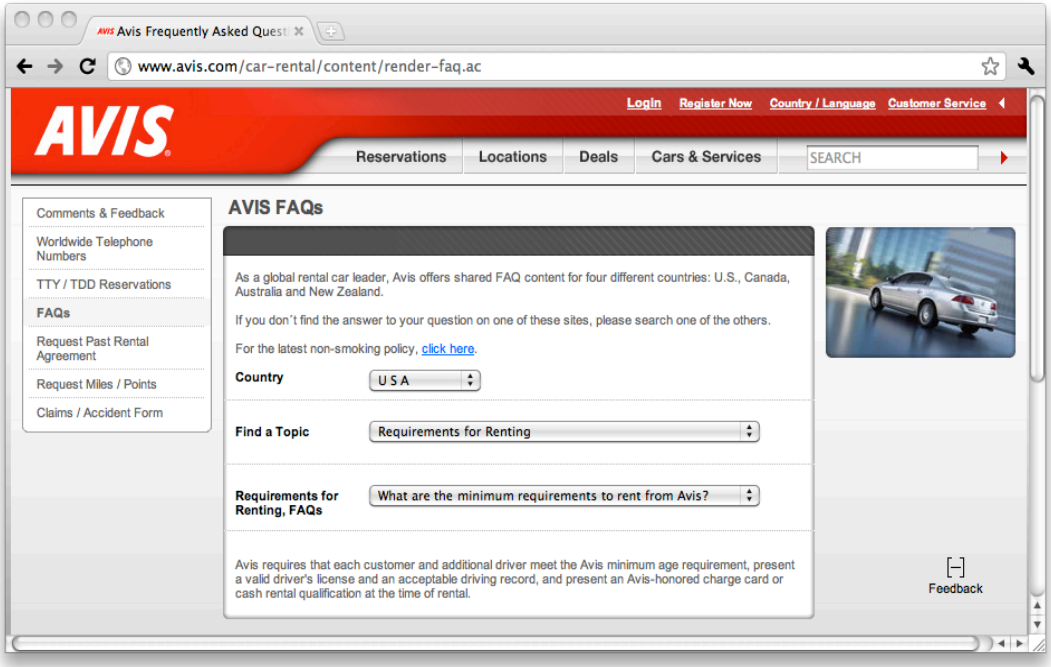
Tight Interaction between SWRL and OWL

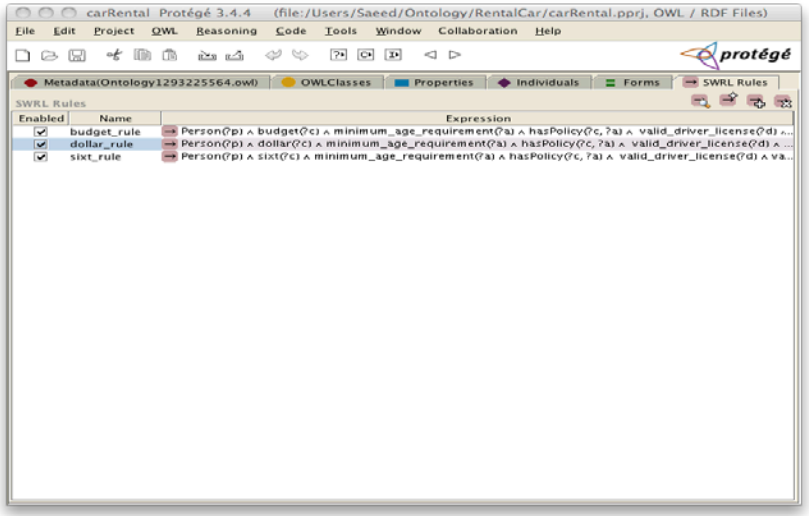
SWRL Atom Type	Example Atom
Class atom	`Person(?x), Car(?y)`
Object property atom	`hasLicense(?x, ?y)` `hasLocation(?x, ?y)`
SameAs/DifferentFrom atom	`sameAs(?x, ?y)` `differentFrom(?x, ?y)`
Data property atom	`hasName(?x, “Joe”)` `hasAge(?x, ?g)`
Built-in atom	`swrlb:notEqual(?state, “CA”)` `swrlb:lessThan(?g, 18)`
Data range atom	`xsd:double(?x), [3-5](?x)`

Method Inputs

- User-selected text
- OWL domain ontology
- SWRL domain rules





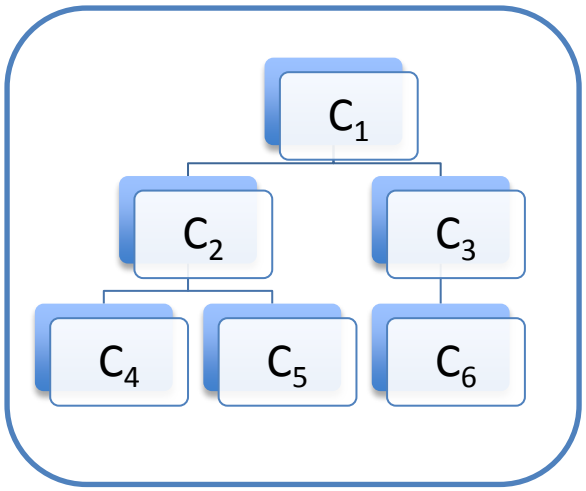


Task Decomposition

- Expansion of domain ontology terms
- Grammatically analyze text
- Finding (property) relationships between entities in text
- Finding (class) concepts for those entities
- Assemble rule bodies
- Restricting relationships' domain and range
- Assembling rule head

Step I: Expansion of Domain Ontology Terms

- Using WordNet, we expand each ontology term (classes and properties) to synonym terms and their morphological variations
  - e.g., ***Present:*** *present, show, demonstrate*



Extracting related terms

Concept	Related terms
C_1	t_1, t_2, t_3
C_2	t_4, t_5
C_3	t_6, t_7, t_8
C_4	t_9, t_{10}
C_5	t_{11}, t_{12}
C_6	t_{13}, t_{14}, t_{15}

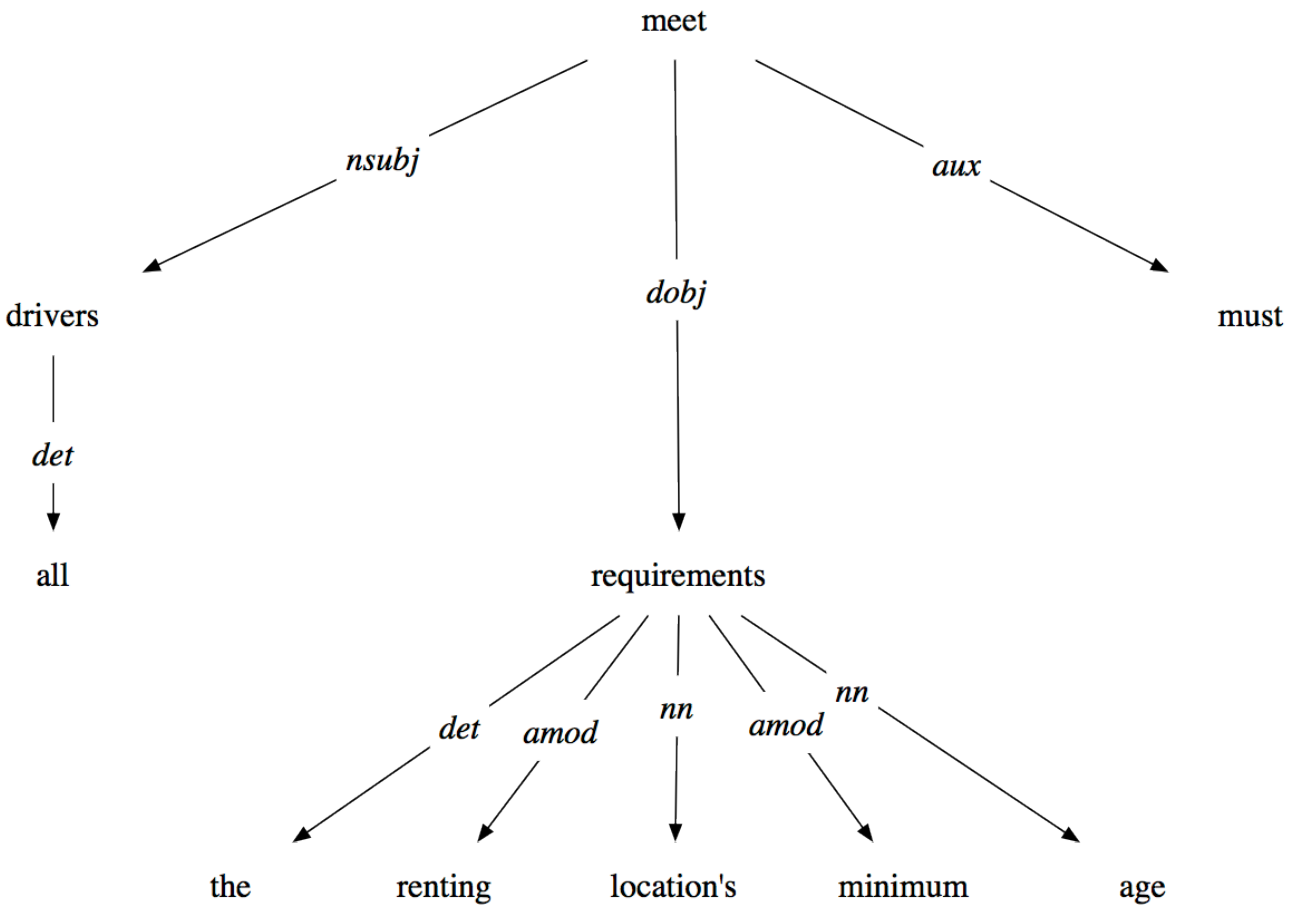
Step II: Finding Grammatical Relationships in Text

- Using Stanford Parser, find all grammatical relationships in text
- Tree-based statistical parser
- Six main relevant types of relationships

Abbreviation	**Explanation**
Det	determiner
nsubject	nominal subject
Aux	auxiliary
Amod	adjectival modifier
nn	noun compound modifier
dobj	direct object

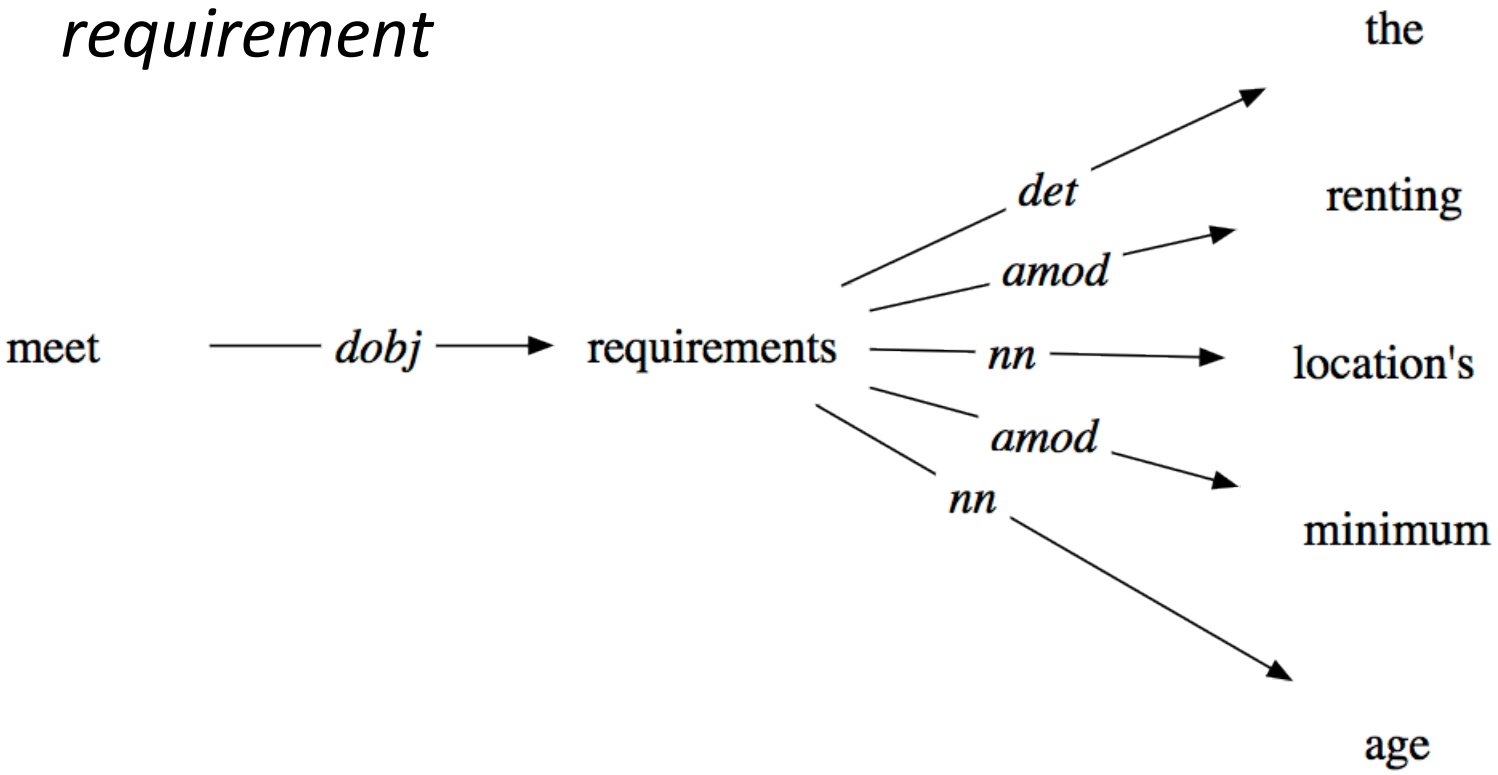
Step II: Finding Grammatical Relationships in Text

e.g., *"All drivers must meet the renting location's minimum age requirements."*



Step III: Finding Property Concepts in the Text

- Expand dependency graph for each OWL property and reorder terms
  - e.g., meet: *the renting location's minimum age requirement*



Step IV: Finding Class Concepts in Text

- After we find dependent phrases, we try to identify OWL classes in the domain ontology corresponding to each phrase
- Use Needleman-Wunsch global sequence alignment algorithm
- Uses dynamic-programming approach to find the best sequence alignment based on the scores of all possible alignments for two sequences
  - *e.g., "the renting location's minimum age requirement":* minimum_age_requirement

Step V: Assembling Rule Bodies

- Assemble each rule body using the relationships that were identified in the last step
- First assemble chains of object property atoms, where each atom property contains an OWL object property corresponding to a relationship identified in the source text
- Add class atoms
- If no disjunctions are identified by the parser, we generate conjunctions of relevant object property atoms
- Since SWRL does not support disjunctions of atoms, rules containing disjunctions expand to multiple rules

Step VI: Restricting relationships' domain and range

- Restrict relationships in generated rules to domain and range specified in domain ontology
- For example, in
  - e.g., minimum_age_requirement(?b) ^ meet(?a, ?b)
  - ?a could refer to any matched domain concept
- Use ontology to restrict it:
  - Person(?a) ^ minimum_age_requirement(?b) ^ meet(?a, ?b)

Step VII: Generating Rule Heads

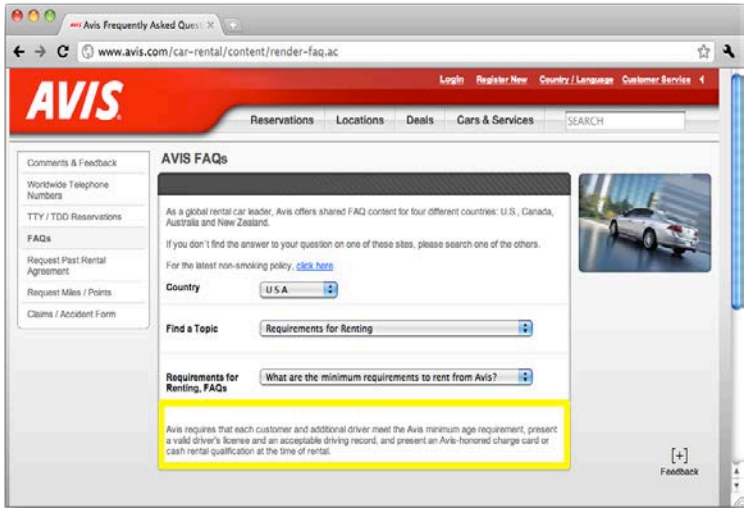
- Use existing rule head for this type of rule
- Align variables in head with variables from corresponding type in body using ontology
- Limitation: currently only supports simple rule heads from existing rule set

Evaluation

- Used rental requirements for 17 California car rental companies
  - 6 training
  - 11 testing
- User-selected relevant text
- Applied method

No.	Company	Location(s)	Usage
1	Ace	SAN	Testing
2	Advantage	LAX, SAN, SJC	Training
3	Alamo	LAX, SAN, SFO, SJC	Testing
4	Avis	LAX, SAN, SFO, SJC	Testing
5	Budget	LAX, SAN, SFO, SJC	Training
6	Dollar	LAX, SAN, SFO, SJC	Training
7	Enterprise	LAX, SAN, SFO, SJC	Testing
8	Fox	LAX, SAN, SFO, SJC	Testing
9	Hertz	LAX, SAN, SFO, SJC	Testing
10	Midway	SAN	Testing
11	National	LAX, SAN, SFO, SJC	Testing
12	Pacific	SAN	Testing
13	Payless	LAX, SAN	Testing
14	Renty	SAN	Testing
15	Thrifty	LAX, SAN, SFO, SJC	Training
16	TravCar	SAN	Training
17	West Coast	SAN	Training

Example Result: Avis



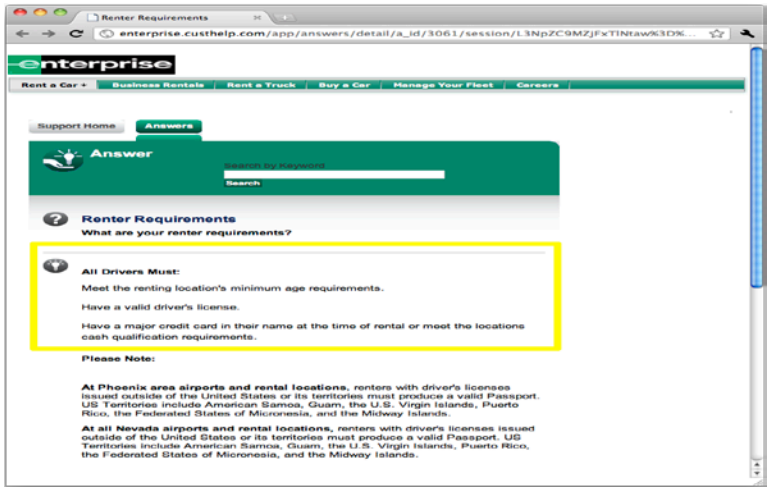
- Input text:

  "Avis requires that each customer and additional driver meet the Avis minimum age requirement, present a valid driver's license and an acceptable driving record, and present an Avis-honored charge card or cash rental qualification at the time of rental."

- Output rules:
  - Person(?a) ^ credit_card(?b) ^ present(?a , ?b) ^ minimum_age_requirement(?c) ^ present(?a , ?c) ^ valid_driver_license(?d) ^ present(?a , ?d) ^ acceptable_driving_record(?e) ^ present(?a , ?e) ^ avis(?f) -> qualifiedToRentFrom(?a , ?f)

  - Person(?a) ^ cash_qualification(?b) ^ present(?a , ?b) ^ minimum_age_requirement(?c) ^ present(?a , ?c) ^ valid_driver_license(?d) ^ present(?a , ?d) ^ acceptable_driving_record(?e) ^ present(?a , ?e) ^ avis(?f) -> qualifiedToRentFrom(?a , ?f)

Example Result: Enterprise



- Input text:

  "All Drivers Must: Meet the renting location's minimum age requirements. Have a valid driver's license. Have a major credit card in their name at the time of rental or meet the location's cash qualification requirements."

- Output rules:
  - Person(?a) ^ credit_card(?b) ^ has(?a, ?b) ^ valid_driver_license(?c) ^ has(?a, ?c) ^ minimum_age_requirement(?d) ^ meet(?a, ?d) ^ enterprise(?e) -> qualifiedToRentFrom(?a, ?e)

  - Person(?a) ^ cash_qualification(?b) ^ meet(?a, ?b) ^ valid_driver_license(?c) ^ has(?a, ?c) ^ minimum_age_requirement(?d) ^ meet(?a, ?d) ^ enterprise(?e) -> qualifiedToRentFrom(?a, ?e)

Results

- For 9 out of 11 cases we generated one or more rules that accurately reflected the rental company's requirements.
- Precision: 100%
- Recall: 96%
  - The 4% missing was related to missing concepts in the ontology, e.g., home address, phone number

Conclusion

- **Structured knowledge bases** in combination with **NLP** techniques can be used to automatically **formalize** knowledge in **free text**
- Can dramatically expand the amount of knowledge that can be automatically extracted

Future Work

- Improve the precision and recall of the information retrieval and text mining methods
- Evaluation the rule extraction method to the biomedical domain
- Assemble (semi-) automated pipeline
  - Auto-generate domain ontology
  - Auto-select text fragments with rules
  - Generate rules (with more elaborate heads)