Processing complex questions in the commercial domain

Presented by: Amine Hallili
Advisors: Fabien Gandon
            Catherine Faron Zucker
Headlines

• Introduction & motivations
• SynchroBot overview
• Question analysis and modeling
• Learning regex for property value identification
• Evaluation
• Future work
Introduction

• Huge evolution of the e-Commerce
• Huge amount of data generated every second
• User needs are getting more complex and specific
• Several systems try to satisfy these needs
  • Search engines, comparative shopping systems, question answering systems

• **Research question**: how can a system understand and interpret complex natural language (NL) questions (also known as n-relation questions) in a commercial context?
SynchroBot

- Natural Language Question Answering system for commercial domain
- From QAKiS (open domain) => domain specific (e-Commerce)
SynchroBot

Question Interpretation
- Expected Answer Type Recognition
- Property Identification
- Named Entity Recognition

Answer Visualization
- Media Picker
- Natural Language Generation

Ontology

Graph Construction
- Relational Graph
- Graph Instantiation

Knowledge Base
- Triple Store
- SPARQL Query

API Feeding

Property Linking

NLP
Question Analysis and modeling

Expected Answer Type (EAT) Recognition
Named Entity Recognition (NER)
Property identification

Example: Give me the price of Nexus 5 phone!
EAT Recognition

- Detecting types in NL questions
  - Specifying the type of Named Entities
    Ex: Give me the price of Nexus 5 phone # Give me the price of Nexus 5
  - Specifying the type of resources
    Ex: Give me the price of available phones

- Why?
  - To improve precision
  - To limit the number of retrieved Named Entities
EAT Recognition

Give me the price of **phones** cheaper than 200$.

Give me the address of Nexus 5 **seller**.
Named Entity Recognition

- Classic definition
  - (persons, organizations, locations, times, dates)

- Commercial domain?
  - More types (Phones, Cases, …)
### Named Entity Recognition

<table>
<thead>
<tr>
<th>mso:legalName</th>
<th>Samsung Galaxy S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>mso:name</td>
<td>AT&amp;T GoPhone - <strong>Samsung Galaxy S5</strong> 4G LTE No-Contract Cell Phone - Dark Gray</td>
</tr>
<tr>
<td>mso:description</td>
<td>The 4.5&quot; WVGA Super AMOLED Plus touch screen on this AT&amp;T GoPhone <strong>Samsung Galaxy S5</strong> SGH-i437 cell phone makes it easy to navigate features. The 5.0MP rear-facing camera features a 4x digital zoom and an LED flash for clear image capture.</td>
</tr>
</tbody>
</table>

Give me the price of Samsung Galaxy S5 ?
Give me the price of Samsung S5 ?
Give me the price of Samsung 5 ?
```javascript
var score = 0
var match // contains the matched string (occurrence in the NL question)
List namedEntities
var stringToMatch // we put the first word. Our goal is to find the largest match

for( word in question )
begin
    if (findMatch(stringToMatch)) then
        update(match)
        update(score)
        addNamedEntities(namedEntities)
        stringToMatch = concat(stringToMatch, word)
    else
        if (findMatch(word)) then
            update(match)
            update(score)
            addNamedEntities(namedEntities)
            stringToMatch = word
        endif
    endif
end

cleanNamedEntities()
sortNamedEntities()
computeScore(namedEntites) // computing score for each named entity according to the general number of retrieved named entities
```
Named Entity Recognition : Algorithm

• **Example** : ”What is the battery life time of Nokia - Lumia Icon 4G LTE Cell Phone - White (Verizon Wireless)”

• **Cleaned sentence** : What Nokia Lumia Icon 4G LTE Cell Phone White Verizon Wireless


• **Cleaned sentence** : What Nexus 5 Nokia Lumia

Property Identification

Label based method
Value based method
Label based property identification

```xml
<rdf:Property rdf:ID="price">
  <rdfs:label xml:lang="en">price</rdfs:label>
  <rdfs:label xml:lang="en">cost</rdfs:label>
  <rdfs:label xml:lang="en">value</rdfs:label>
  <rdfs:label xml:lang="en">worth</rdfs:label>
  <rdfs:label xml:lang="en">tariff</rdfs:label>
  <rdfs:label xml:lang="en">amount</rdfs:label>
  <rdfs:label xml:lang="fr">prix</rdfs:label>
  <rdfs:label xml:lang="fr">couter</rdfs:label>
  <rdfs:label xml:lang="fr">cout</rdfs:label>
  <rdfs:label xml:lang="fr">tarif</rdfs:label>
  <rdfs:label xml:lang="fr">valeur</rdfs:label>
  <rdfs:comment xml:lang="en">The price of a product.</rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="http://schema.org/price" />
  <rdfs:domain rdf:resource="http://i3s.unice.fr/MerchantSiteOntology#Product" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double" />

  <rdfs:comment xml:lang="fr">Le prix d’un produit.</rdfs:comment>
  <rdfs:comment xml:lang="en">The price of a product.</rdfs:comment>

  <sbmo:responsePattern xml:lang="fr">Le prix de _resource_ est de : _value_ .</sbmo:responsePattern>
  <sbmo:responsePattern xml:lang="en">The price of _resource_ is _value_ .</sbmo:responsePattern>

  <sbmo:regexExtractionPattern>&lt;![CDATA[([-]?([0-9]+([,\.|][0-9]+)?(€|\$|\£)?)+</sbmo:regexExtractionPattern>
  <sbmo:mediaType>text</sbmo:mediaType>
  <sbmo:valueType>unit</sbmo:valueType>
</rdf:Property>
```

Give me the price of Nexus 5!
Value based property identification

Give me details of the products cheaper than 200$
Value based property identification

- **Constraints:**
  - A value can correspond to multiple properties
    - 200$ \rightarrow \text{[price, cost]}
  - A property can have multiple values
    - Storage [4GB, 8GB]

  Must be handled during the graph construction
Graph construction

Relational graph creation
Graph instantiation
Graph construction

Goal: creating one connected graph to generate SPARQL query

give me the dimensions and the seller address of available black Nexus 5 that costs 449.99$
Relational graph creation

Give me details about the products cheaper than $200$
Relational graph creation

Give me the address of the products cheaper than 200$
Give me **details** about the **products** cheaper than **200$**
SPARQL query

Give me details about the products cheaper than 200$.

Select distinct *
where {
?ne a <http://i3s.unice.fr/MerchantSiteOntology#Product> 
?ne <http://i3s.unice.fr/MerchantSiteOntology#name> ?n
optional {
?ne <http://i3s.unice.fr/MerchantSiteOntology#description> ?var1
}
optional {
?ne <http://i3s.unice.fr/MerchantSiteOntology#price> ?v
?v rdf:value ?var2
 filter (contains (?var2, lcase(str("200"))))
}
bind( IF(bound(?var1),1,0)+ IF(bound(?var2),1,0) as ?c)
}
order by desc (?c) limit 20
Learning regex Automatically

Why?
Anticipating most forms of property values
   In case new properties are introduced
   In case the domain is changed
Learning regex Automatically

- Genetic Programming (GP) approach:
- “In artificial intelligence, genetic programming (GP) is an evolutionary algorithm-based methodology inspired by biological evolution to find computer programs that perform a user-defined task” - Wikipedia
Genetic Programming: Goal

[Petrovski et al. 2014][Bartoli et al. 2012]

<table>
<thead>
<tr>
<th>Text</th>
<th>Value to extract</th>
<th>regex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patriot Memory - FUEL+ 5200 mAh Rechargeable Lithium-Ion Battery and Signature Series <strong>8GB</strong> microSDHC Memory Card &amp; <strong>8GB</strong></td>
<td>8GB</td>
<td>?</td>
</tr>
<tr>
<td>Apple - iPhone 4s <strong>8GB 499.99$</strong> Cell Phone - Black (Verizon Wireless)</td>
<td>499.99$</td>
<td>?</td>
</tr>
<tr>
<td>Nokia - <strong>Lumia 1520</strong> 4G Cell Phone - Black (AT&amp;T)</td>
<td>Lumia 1520</td>
<td>?</td>
</tr>
<tr>
<td>HTC - One (<strong>M7</strong>) 4G LTE with <strong>32GB</strong> Memory Cell Phone - <strong>Black</strong> (Sprint) &amp; <strong>32GB</strong></td>
<td>Black</td>
<td>?</td>
</tr>
</tbody>
</table>
Flowchart for Genetic Programming

1. **Gen = 0**
   - Create Initial Random Population
   - Termination Criterion Satisfied?
     - Yes → Designate Result
     - No → Evaluate Fitness of Each Individual in Population
   - individuals = 0
     - Yes → Gen = Gen + 1
     - No → individuals = M?

   **reproduction**
   - Select One Individual Based on Fitness
     - Perform Reproduction
       - Copy into New Population
       - individuals = individuals + 1
   - Select Two Individuals Based on Fitness
     - Perform Crossover
       - Insert Two Offspring into New Population
       - individuals = individuals + 2

   **mutation**
   - Select One Individual Based on Fitness
     - Perform Mutation
     - Insert Mutant into New Population
     - individuals = individuals + 1
Genetic programming: algorithm

- Create population (500 individuals)
- Repeat 150 or precision = 1
  - For each individual
    - For each example
      - Compute individual fitness
  - While new population < 500
    - Select 2 individuals
    - crossover
Genetic programming

- **Individuals**: valid regex represented by a tree

Operators:
- concatenate node: a binary node to concatenate two leaves.
- possessive quantifiers: \{
  \text{”
  }^\ast\text{”}, \text{”}\+\text{”}, \text{”}\?\text{”}, \text{”}m,n\text{”}\}
- Group operator: \text{”}(\text{”)}\text{”}
- Class operator: \text{”}[\text{”}]

Terminals:
- constants: a single character, a number or a string.
- Ranges: \text{”a-z”}, \text{”0-9”}, \text{”a-z0-9”}, \text{”A-Z”}
- Character class: \{
  \text{”}\w\text{”}, \text{”}\d\text{”}\}
- White space: \text{”}\s\text{”}
- Wildcard character: \text{”}.\text{”}

\begin{equation}
(foo)\mid(ba++r)
\end{equation}
Genetic programming

- Population:
  - Half of the population derived from the examples by replacing:
    (characters, \w) and (numbers, \d)
      - (‘200$’ -> ‘\d\d\d\w’) (32GB -> \d\d\w\w)
  - The other Half is generated randomly using the ramped half-and-half method
    - Generate random trees with different depth
Genetic programming

• Fitness function:
  • Precision

\[
\text{Precision} = \frac{tp}{tp + fp}
\]

• Matthews Correlation Coefficient (MCC)

\[
\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}
\]
Genetic Operation

Crossover | Mutation | Reproduction

P.S: Before performing genetic operation, node compatibility must be checked
Selection

- Fitness proportionate selection also known as the roulette wheel selection

\[ p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \]

where \( N \) is the number of individuals

- The selection token \((r)\) is randomly generated

\[ 0 < r < \sum_{j=1}^{N} f_j \]
Evaluation

Genetic programming result
SynchroBot performances
## GP : result

<table>
<thead>
<tr>
<th>Property</th>
<th>Precision</th>
<th>Automatic Regex</th>
<th>Manually regex</th>
</tr>
</thead>
<tbody>
<tr>
<td>storage</td>
<td>100%</td>
<td>[0-9]++G[a-zA-Z]</td>
<td>\d++[Gg][Bb]</td>
</tr>
<tr>
<td>price</td>
<td>97.33%</td>
<td>\d+.\d++\D</td>
<td>(?i)[0-9]+([,</td>
</tr>
<tr>
<td>Release date</td>
<td>~60%</td>
<td>\d\d\W[0-9]++\D\d?+</td>
<td>((19</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>~20%</td>
<td>(?:[^\d]+\s[a-zA-Z0-9]+)+</td>
<td>([A-Z]\w++)*([A-Z]\d)</td>
</tr>
<tr>
<td>color</td>
<td>~11%</td>
<td>\w\w\w\w</td>
<td>(?i)aliceblue</td>
</tr>
</tbody>
</table>
SynchroBot

QALM [Hallili et al 2014]: Question Answering Linked Merchant data
Benchmark for evaluating question/answering systems that use commercial data

<table>
<thead>
<tr>
<th>Questions / Number</th>
<th>Training</th>
<th>Goldstandard</th>
<th>Handled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>v1</td>
</tr>
<tr>
<td>1-relation questions</td>
<td>15</td>
<td>12</td>
<td>yes</td>
</tr>
<tr>
<td>2-relations questions</td>
<td>9</td>
<td>8</td>
<td>no</td>
</tr>
<tr>
<td>N-relations questions</td>
<td>2</td>
<td>5</td>
<td>no</td>
</tr>
<tr>
<td>Named-Entityless questions</td>
<td>19</td>
<td>11</td>
<td>no</td>
</tr>
<tr>
<td>Boolean questions</td>
<td>7</td>
<td>4</td>
<td>partially</td>
</tr>
<tr>
<td>Aggregations questions</td>
<td>8</td>
<td>4</td>
<td>no</td>
</tr>
</tbody>
</table>

**Table 3. Question analysis**

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th></th>
<th>Goldstandard</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>v1</td>
<td>v2</td>
<td>v3</td>
<td>v1</td>
</tr>
<tr>
<td>Answered questions</td>
<td>21/40</td>
<td>25/40</td>
<td>25/40</td>
<td>13/30</td>
</tr>
<tr>
<td>Partially right answers</td>
<td>10/21</td>
<td>16/25</td>
<td>10/17</td>
<td>16/25</td>
</tr>
</tbody>
</table>

**Table 4. General analysis**
## SynchroBot

<table>
<thead>
<tr>
<th></th>
<th>Version 1</th>
<th>Version 2</th>
<th>Version 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited set</td>
<td>19%</td>
<td>25.44%</td>
<td>38%</td>
</tr>
<tr>
<td>Whole set</td>
<td>10.23%</td>
<td>21.01%</td>
<td>35.56%</td>
</tr>
</tbody>
</table>
Conclusion & future work

• Proposing generic NE classification for domain specific systems
• Optimizing the learning of regular expression (LRE)
• Applying the LRE to other topical domains