Extracting STRIPS representations of Actions and Events

Avirup Sil and Alexander Yates
Center for Data Analytics and Biomedical Informatics
Temple University
Philadelphia, PA
Outline

• Novel Technique of Information Extraction of Commonsense Knowledge

• STRIPS Representation

• Our Extraction Method

• Experiments and Results
Introduction

Understanding language requires both linguistic knowledge and knowledge about how the world works, also known as common-sense knowledge.

We attempt to extract common-sense knowledge about action and event semantics.

In particular, if a system attempts to learn the “meaning” of an action like heal, it must understand the states of the world before and after this action.
A Novel Technique of Information Extraction of Commonsense Knowledge

- We extend our previous approach, S10 (Sil et al, 2010) to extract:
  - Fully formed STRIPS representation (preconditions and effects of actions)
  - Identify additional kinds of effects = delete effects
  - Identify argument variables for each predicate in the representations

- Importance of this system:
  A system can use this information to understand the meaning of the action which requires knowing the states of the world true immediately before or after it.
Related Work

- **Script Knowledge Extraction (Chambers and Jurafsky, 2009)**
  - Our work is different from theirs as script relates one event $e$ to a subsequent event $e'$. But we deal with event $e$ and the states of the world $s$ before or after $e$.

- **Causal Relationships (Girju, 2003)**
  - They don’t differentiate between event-state relationships e.g. $\text{is\_wet(\text{grass})}$ follows a $\text{raining}$ event $e$.
  - They don’t consider precondition relationships which are central to AI representations of actions and events.
Outline

• Novel Technique of Information Extraction of Commonsense Knowledge

• STRIPS Representation

• Our Extraction Method

• Experiments and Results
Sample STRIPS representations

- Action “awaken”

<table>
<thead>
<tr>
<th>STRIPS</th>
<th>Arguments:</th>
<th>Precondition:</th>
<th>Add effect:</th>
<th>Delete Effect:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>asleep(x)</td>
<td>awake(x)</td>
<td>asleep(x)</td>
</tr>
</tbody>
</table>

I was asleep and was awakened from neck pain. ... afterward, when I was awake it was gone.
Sample STRIPS representations

- Action “heal”

<table>
<thead>
<tr>
<th>STRIPS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments:</td>
<td>(x, y, p)</td>
</tr>
<tr>
<td>Precondition:</td>
<td>(medical_practitioner(x), patient(y), pain(p), in(y, p))</td>
</tr>
<tr>
<td>Add effect:</td>
<td>(\neg in(y, p))</td>
</tr>
<tr>
<td>Delete Effect:</td>
<td>(in(y, p))</td>
</tr>
</tbody>
</table>
Challenges

- Lack of explicitly stated knowledge
  - As this is knowledge is almost never stated explicitly
- Temporality
  - We must distinguish implications true before vs. after an event
- Generalization
  - We must extract generalized predicates. E.g. Instead of extracting doctor as a precondition for action heal, a system should choose a larger class of person e.g. a medical_practitioner.
- Rule extraction
  - We should be able to discover implications like:

\[
\forall_{x,t_1,t_2} \text{awaken}(x,t_2) \land \text{after}(t_2,t_1) \Rightarrow \text{asleep}(x,t_1)
\]
Problem Formulation

**INPUT**
- Corpus C with action/event

**STRIPS Extraction System**
- Action E.g. heal

**OUTPUT: STRIPS**

<table>
<thead>
<tr>
<th>STRIPS</th>
<th>pre:</th>
<th>add:</th>
<th>del:</th>
</tr>
</thead>
<tbody>
<tr>
<td>heal</td>
<td>medical_practitioner(x), patient(y), pain(p), in(y,p)</td>
<td>~in(y,p)</td>
<td>in(y,p)</td>
</tr>
</tbody>
</table>

**Limitation:** Currently, we don’t have the scope to deal with durative or repetitive actions like escalate or accelerate.
Outline

- Novel Technique of Information Extraction of Commonsense Knowledge
- STRIPS Representation
- Our Extraction Method
- Experiments and Results
Our Extraction Method

• Extracting Preconditions and Add Effects

• Connecting Extractions to an ontology

• Detecting Delete Effects

• Determining Arguments
Extracting Preconditions and Add Effects

1. Select Candidate Precondition and add effects

2. Compute Features from counts over texts

3. Train an SVM to classify between true preconditions/add effects and others

4. Rank Candidate words using the SVM
Extracting Preconditions and Add Effects

1. Select Candidate Preconditions and add effects

2. Compute Features from counts over texts

3. Train an SVM to classify between true preconditions/add effects and others

4. Rank Candidate words using the SVM
Select Candidate Precondition and add effects

**INPUT**

WEB DOCUMENT COLLECTION (HAVING THE ACTIONS)

Choose top 500 words with high PMI

**OUTPUT**

Words occurring frequently with corresponding action

- **Boil:** heat, liquid,…
- **Cut:** fruits, blood,…
- **Heal:** patient, pain,…
- …
- …

\[
\text{PMI}(A, w) = \log \frac{|\{d \in D_A | w \text{ appears in } d\}|}{|D_A| \times |\{d \in D | w \text{ appears in } d\}|}
\]

Here, \( D_A \) = document set containing \( A \), \( w \) = words in \( D_A \), \( d \) = document under consideration
Extracting Preconditions and Add Effects

1. Select Candidate Precondition and add effects

2. Compute Features from counts over texts

3. Train an SVM to classify between true preconditions/add effects and others

4. Rank Candidate words using the SVM
Compute Features from Counts over Texts

- Discriminator Features
- Features from Extracted Relationships
Compute Features from Counts over Texts

- Discriminator Features

- Features from Extracted Relationships
Features from Extracted Relationships

• Annotate texts with structural information with a Semantic Role Labeling (SRL) system.

• Count how often candidate words are associated through predicate-argument structure with certain actions.

  • **Example of one such feature**: Count how often candidate word C appears as an argument to the action word A.

    “The nurse was able to heal my chronic pain …”

    \[\Rightarrow \text{heal}(\text{nurse}, \text{pain})\]

    Action = heal and Candidate = pain

    **Feature**: Count how many times ‘pain’ occur as an argument to action ‘heal’.

We call our system with the features : S10.
Extracting Preconditions and Add Effects

1. Select Candidate Pre/Postcondition words

2. Compute Features from counts over texts

3. Train an SVM to classify between true preconditions/add effects and others

4. Rank Candidate words using the SVM
TRAINING OF S10

Action | Candidate | Label
---|---|---
boil | heat | +1
boil | eggs | -1
heal | pain | +1
...

WEB CORPUS WITH THE ACTIONS

INPUT

COMPUTE FEATURES FROM COUNTS OVER TEXT

Action | Candidate | Label | Discriminator Features | PMI | Features from Ext. Rel.
---|---|---|---|---|---
boil | heat | +1 | 4 | 3 | 2 | 0 | ... | 1.2 | 2 | 4 | ...
boil | eggs | -1 | 0 | 0 | 1 | 0 | ... | 0.1 | 1 | 0 | ...
...

TRAIN AN SVM

LEARNED MODEL
### TESTING OF S10

<table>
<thead>
<tr>
<th>Action</th>
<th>Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>cut</td>
<td>scissors</td>
</tr>
<tr>
<td>cut</td>
<td>tomatoes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>defrost</td>
<td>microwave</td>
</tr>
</tbody>
</table>

**WEB CORPUS WITH THE ACTIONS**

**Discriminator Features**

**PMI**

**Features from Ext. Rel.**

**Input**

**Features from Counts Over Text**

<table>
<thead>
<tr>
<th>Action</th>
<th>Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>cut</td>
<td>scissors</td>
</tr>
<tr>
<td>cut</td>
<td>tomatoes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Learned Model**

<table>
<thead>
<tr>
<th>Action</th>
<th>Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>cut</td>
<td>scissors</td>
</tr>
<tr>
<td>cut</td>
<td>tomatoes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Shortcomings of S10

One of the shortcomings of S10 is that its extractions are too specific.
E.g. It extracts hammer as a precondition for action crush.
Hence, $crush(x, y) \Rightarrow hammer(x)$

Using this incorrect knowledge, a system from the sentence:
“Jane crushed the soda can with her hands!”
infers: $crush(hands, soda\_can) \Rightarrow hammer(hands)$
i.e. hands are a type of hammer.

Objective should be to select a general class of objects as candidates.
Our Extraction Method

- Extracting Preconditions and Add Effects

- Connecting Extractions to an ontology

- Detecting Delete Effects

- Determining Arguments
Connection to an External Ontology to generalize the Predicates

- We connect the S10 candidates to Wordnet to extract their hypernyms (super-classes).

- We keep all the direct and indirect hypernyms of every candidate and create a new candidate list. We call this new system S10’.

<table>
<thead>
<tr>
<th>Wordnet Superclasses (ws) (new candidates)</th>
<th>Words in corpus which have ws as a hypernym(C_\text{ws})</th>
</tr>
</thead>
<tbody>
<tr>
<td>nurse#1</td>
<td>{nurse}</td>
</tr>
<tr>
<td>doctor#1</td>
<td>{doctor, allergist}</td>
</tr>
<tr>
<td>health_professional#1</td>
<td>{doctor, nurse, allergist}</td>
</tr>
<tr>
<td>person#1</td>
<td>{doctor, nurse, poet,…}</td>
</tr>
</tbody>
</table>

Table 1. Sample candidate preconditions for action ‘heal’ with the set of words in corpus for heal that have candidate synset as a hypernym
Shortcomings of S10’

- Ranking becomes difficult as overly general hypernyms are chosen too often.
  - E.g. Synsets like `physical_entity#1` tend to rank highly as preconditions and add effects.
  - This is because the candidate list has a lot of hyponyms of `physical_entity#1`. 
Include compensating features to create **HYPER**

- Include features to let the SVM balance between candidate synsets that score highly in S10’ and those which are less general.

Example:
Compute “root” distance between a candidate and any root node in Wordnet hierarchy.

```
physical_entity#1 -> physical_object#1 -> tool#1 -> hammer#1
```

Here, \( \text{Distance} (\text{physical\_entity}\#1, \text{hammer}\#1) > \text{Distance} (\text{physical\_entity}\#1, \text{tool}\#1). \)

Hence, tool#1 is a better and general precondition than hammer#1.

- We call this system **HYPER**.
Our Extraction Method

- Extracting Preconditions and Add Effects
- Connecting Extractions to an ontology
- Detecting Delete Effects
- Determining Arguments
Detecting Delete Effects

Objective: To determine which of the preconditions become false after the action takes place.

Method:
- Input: Action with their preconditions (as candidates)
- Compute Features for the candidates
  - Example Feature: Determine $PMI(\text{"no" + candidate+}\text{\"after"}, \text{action})$.
    So, for action heal, we compute $PMI(\text{\"no pain after"}, \text{\"heal\")}$
- Train a Binary SVM classifier
- Test the SVM on unseen actions
Our Extraction Method

- Extracting Preconditions and Add Effects
- Connecting Extractions to an ontology
- Detecting Delete Effects
- Determining Arguments
Determining Arguments

**Input:** STRIPS representation (without arguments)

<table>
<thead>
<tr>
<th>action</th>
<th>pre</th>
<th>add</th>
</tr>
</thead>
<tbody>
<tr>
<td>maim</td>
<td>person#1</td>
<td>hurt</td>
</tr>
<tr>
<td></td>
<td>unhurt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>object#1</td>
<td></td>
</tr>
</tbody>
</table>

Use Semantic Role labels with which the preconditions/add effects occur most as argument to the action.

**Output:** Complete STRIPS representation (with arguments)

<table>
<thead>
<tr>
<th>action</th>
<th>pre</th>
<th>add</th>
</tr>
</thead>
<tbody>
<tr>
<td>maim</td>
<td>person#1(A1)</td>
<td>hurt(A1)</td>
</tr>
<tr>
<td></td>
<td>unhurt(A1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>object#1(A2)</td>
<td></td>
</tr>
</tbody>
</table>
Outline

• Novel Technique of Information Extraction of Commonsense Knowledge

• STRIPS Representation

• Our Extraction Method

• Experiments and Results
Experiment 1: Extracting Preconditions and Add Effects

- Compare S10' and HYPER

- Perform 5-Fold Cross Validation
  - Train on 32 Actions
  - Test on remaining 8 Actions

- SVM produces real-valued predictions for candidate-action pairs

- We rank these pairs as per the SVM output
Results

- AUC for the S10' Model:
  - For preconditions is 0.48 and add effects is 0.52
- AUC for the HYPER Model:
  - For preconditions is 0.82 and add effects is 0.72
- The AUCs for S10' and HYPER were statistically significantly different.
Experiment 2: Extracting Delete Effects

- We perform 5-Fold Cross Validation
  - First, train on 32 Actions
  - Then, test on remaining 8 Actions
- Results:

<table>
<thead>
<tr>
<th>Technique</th>
<th>Prec.</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>All pre. are deleted</td>
<td>26</td>
<td>90</td>
<td>40.3</td>
</tr>
<tr>
<td>No pre. are deleted</td>
<td>100</td>
<td>10</td>
<td>18.2</td>
</tr>
<tr>
<td><strong>SVM trained model</strong></td>
<td><strong>72.2</strong></td>
<td><strong>52.6</strong></td>
<td><strong>60.8</strong></td>
</tr>
</tbody>
</table>

Precision and Recall are macro-averaged across actions.
Experiment 3: Argument Addition

We match arguments using our technique as mentioned before.

- We measure **precision** by calculating how many of the predicted predicates match our predicate in gold-standard.

- We measure **recall** by calculating how many of gold standard predicates were found in auto-generated representation.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Prec.</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>All preds. have same var.</td>
<td>32</td>
<td>33</td>
<td>32</td>
</tr>
<tr>
<td>Each pred. has distinct var.</td>
<td>56</td>
<td>58</td>
<td>57</td>
</tr>
<tr>
<td><strong>Semantic role heuristic</strong></td>
<td><strong>73</strong></td>
<td><strong>72</strong></td>
<td><strong>72</strong></td>
</tr>
</tbody>
</table>
Conclusion

- We have shown a system which extracts STRIPS representation with an overall F1 of 0.72.

- It significantly outperforms our previous system and extracts richer representation.
Future Work

- Future work includes extracting more sophisticated representations, multi-argument predicates etc.

- Extract STRIPS representation for durative/repetitive actions.
Thank you! Questions?