Rapid Prototyping of Robust Language Understanding Modules for Spoken Dialogue Systems

†Yuichiro Fukubayashi, †Kazunori Komatani,
‡Mikio Nakano,
‡Kotaro Funakoshi, ‡Hiroshi Tsujino,
‡Tetsuya Ogata, ‡Hiroshi G. Okuno

1Graduate School of Informatics, Kyoto University
1Honda Research Institute Japan Co., Ltd.

Background

- In an early phase of the development of spoken dialogue systems...
- Large amounts of data are required for robust language understanding (LU).
- However, large amounts of data are not available.
- To construct robust LU modules needs a lot of efforts and is time-consuming.

Goal

Rapid prototyping of LU modules
1. Robust against various expressions.
2. Easy to construct (requires less training data).

More robust LU modules with less training data.

Related Work

Rule- or grammar-based approach
- keyword spotting (e.g. VoiceXML)
- heuristic rules (Seneff, 1992)

Less robust against various expressions.
- Cannot reject automatic speech recognition (ASR) errors.
- Keyword spotting does not consider grammatical rules.

Easy to construct (requires less data)
- Preparing grammars takes less efforts.

Stochastic approach
- corpus-based (Sudoh, 2005; He, 2005)
- Weighted Finite State Transducer (WFST)-based (Potamianos, 2004; Wuttiwatchai, 2004)

Robust against various expressions.
- Reject ASR errors with trained LU modules.
- WFST is considering grammatical rules.

Not easy to construct (requires much data)
- Large amount of data for training is required for robust LU.
- Collecting a large amount of data takes much effort.

Our Approach

WFST-based LU with simpler weightings
- Weighting should be simpler than conventional methods.
- Optimal parameters are obtained with small amount of data.

Robust against various expressions.
- Reject ASR errors with trained WFST.
- WFST is considering grammatical rules.

Easy to construct (requires less data)
- Preparing grammars takes less efforts.
- Required data for training is small.
Position of Our Method

- A modest and realistic approach.
- 1. Takes more robust than rule-based or grammar-based approaches.
- 2. Takes less efforts than stochastic approaches.
- More robust against ASR errors.
- Takes less robust against ASR errors.
- More efforts for collecting data.
- More efforts for stochastic approaches.

WFST-based Language Understanding

- WFST accepts ASR outputs as its input.

FILLER Transition

- FILLER transition accepts any words.
- FILLER transition enables to ignore unnecessary words for LU and suppress insertion errors.

Input: $ twenty two, please
Output: $ twenty two value=22 please
Cumulative weight: +1.0
LU result: value=22

Issue: Design of Weighting Schemes

- The path with the highest cumulative weight is selected from various output sequences.

<table>
<thead>
<tr>
<th>LU output</th>
<th>LU result</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>It  is February twenty second please</td>
<td>month=2, day=22</td>
<td>2.0</td>
</tr>
<tr>
<td>It  is FILLER twenty second please</td>
<td>day=22</td>
<td>1.0</td>
</tr>
<tr>
<td>It  is FILLER twenty second FILLER</td>
<td>day=22</td>
<td>1.0</td>
</tr>
<tr>
<td>FILLER FILLER FILLER FILLER FILLER FILLER FILLER</td>
<td>-</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Requirements for weighting schemes

1. Robust against various expressions (ASR errors).
2. Simple features for weighting.
3. Reduce the amount of data for training.
**Outline of Our Method**

Training data → Minimize concept error rate (CER) by changing parameters

- ASR N-best output
- Optimal parameters

**Weighting on Two Levels**

1. Weighting for ASR outputs
2. Weighting for concepts

\[ w^j = w^j_s + \alpha_w \sum w^j_w + \alpha_c \sum w^j_c \]

\[ w^j = w^j_s + \alpha_w \sum w^j_w + \alpha_c \sum w^j_c \]

\[ \text{score}_i = \sum_j e^{\beta \cdot \text{score}_i} \]

\[ \text{value} = \frac{22}{W} \]

**Parameters for Training**

- Five kinds of parameters to determine.

  - ASR N-best (N=1 or 10)
  - Accepted words
  - Concept

  \[ w^i = w^i_s + \alpha_w \sum w^i_w + \alpha_c \sum w^i_c \]

  \[ \text{score}_i = \sum_j e^{\beta \cdot \text{score}_i} \]

  \[ \text{value} = \frac{22}{W} \]

- Coefficient \( \alpha_w = 0 \) or 1.0?
- Coefficient \( \alpha_c = 0 \) or 1.0?

**Weighting on Two Levels**

1. Weighting for ASR outputs
2. Weighting for concepts

\[ w^j = w^j_s + \alpha_w \sum w^j_w + \alpha_c \sum w^j_c \]

\[ w^j = w^j_s + \alpha_w \sum w^j_w + \alpha_c \sum w^j_c \]

\[ \text{score}_i = \sum_j e^{\beta \cdot \text{score}_i} \]

\[ \text{value} = \frac{22}{W} \]

**Weighting on Two Levels**

1. Weighting for ASR outputs
2. Weighting for concepts

\[ w^j = w^j_s + \alpha_w \sum w^j_w + \alpha_c \sum w^j_c \]

\[ w^j = w^j_s + \alpha_w \sum w^j_w + \alpha_c \sum w^j_c \]

\[ \text{score}_i = \sum_j e^{\beta \cdot \text{score}_i} \]

\[ \text{value} = \frac{22}{W} \]
Candidates for accepted words
const.: \( w_w = 1.0 \)
#phone: \( w_w = 0.5 \)
CM: \( w_w = 0.9 - \theta_w \)

length of word
I("second")
CM of word
CM("second")

Example:
Weighting Scheme for Accepted Words

Candidates for concepts
const.: \( w_c = 1.0 \)
avg: \( w_c = 0.95 - \theta_c \)

#pCM(avg): \( w_c = 0.525 - \theta_c \)

Example:
Weighting Scheme for Concepts

Training: Determine Parameters

Determine optimal parameter sets

Minimize concept error rate (CER) by changing parameters

Optimal parameters

Optimal N-best

ASR N-best

No. of concepts

Concept error rate

Coefficient

Cumulative Weight

WSNT output

FILLER: it is February twenty second

WFST output

FILLER: it is February twenty second

Concept

month: 2

day: 22

Coefficient

Coefficient

Experimental Conditions

Two different domains

<table>
<thead>
<tr>
<th></th>
<th>Video</th>
<th>Rent-a-car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vocabulary size</td>
<td>209</td>
<td>891</td>
</tr>
<tr>
<td>Example sentences</td>
<td>10000</td>
<td>40000</td>
</tr>
<tr>
<td># utterance</td>
<td>4186</td>
<td>3364</td>
</tr>
<tr>
<td></td>
<td>(25 x 8sessions)</td>
<td>(23 x 8sessions)</td>
</tr>
<tr>
<td>ASR Acc.</td>
<td>83.9%</td>
<td>85.7%</td>
</tr>
</tbody>
</table>

- Rent-a-car is more complicated domain.
- Larger vocabulary size
- Lower ASR accuracy
**Experimental Conditions**

- We evaluated the results with 4-fold cross validation.
- Compared concept error rate (CER).
- Two baseline methods: simple keyword spotting
  1. **Grammar & spotting**: Grammar-based ASR + keyword spotting
  2. **SLM & spotting**: Statistical language model-based ASR + keyword spotting
- Takes as many concepts as possible without considering grammatical rules.
- Assuming a condition that a large amount of data is not available.

**Result 1: Obtained Optimal Parameters**

- The optimal parameters depend on the domain.
- Complexity of domains reflects the parameters.

<table>
<thead>
<tr>
<th>Domain</th>
<th>$\alpha$</th>
<th>$\omega$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>1.0</td>
<td>const.</td>
<td>0</td>
</tr>
<tr>
<td>Rent-a-car</td>
<td>1.0</td>
<td>CM-0.8</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**Example of keyword spotting in rent-a-car domain**

- **ASR Output**: From June third uhmm FIT please
- **Month** = 6, **Day** = 3, **Car** = FIT  (**FIT** is the name of a car)

**Result 2: Performance of WFST-based LU**

- **Lower CER with our method**
  - Better performance with “SLM & spotting” than “Grammar & spotting” because of robust ASR.
  - Further improvement with optimal weightings for WFST.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Grammar &amp; spotting</th>
<th>SLM &amp; spotting</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>22.1</td>
<td>16.9</td>
<td>13.5</td>
</tr>
<tr>
<td>Rent-a-car</td>
<td>51.1</td>
<td>28.9</td>
<td>22.0</td>
</tr>
</tbody>
</table>

- Due to SLM-based ASR
- Due to optimal weightings for WFST

- Our method outperformed two kinds of baseline.
- More robust than keyword spotting

**Result 3: Performance and Training Data**

- Our method outperformed baseline methods with **about 100 utterances**.
- Easier to construct than stochastic methods.

**Conclusion**

- Rapidly prototyping robust LU modules.
  - WFST-based LU with simpler weighting.
  - More robust than rule- or grammar-based methods.
  - Easier to construct than stochastic methods.

**Experiments and Evaluation**

- Our method outperformed baseline methods with optimal weightings for WFST.
- Our method outperformed baseline methods with less utterances.
- Conventional methods required several thousands of utterances.

**Future Work**

- **When to switch to stochastic approaches?**
  - Stochastic approaches are more robust than our method if using large amounts of data.
  - How many data are needed for stochastic approach?