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

[†]Graduate School of Informatics, Kyoto University
Yoshida-Hommachi, Sakyo, Kyoto
606-8501, Japan
{fukubaya,komatani}@kuis.kyoto-u.ac.jp
{ogata,okuno}@kuis.kyoto-u.ac.jp

[‡]Honda Research Institute Japan Co., Ltd.
8-1 Honcho, Wako, Saitama
351-0188, Japan
nakano@jp.honda-ri.com
{funakoshi,tsujino}@jp.honda-ri.com

Abstract

Language understanding (LU) modules for spoken dialogue systems in the early phases of their development need to be (i) easy to construct and (ii) robust against various expressions. Conventional methods of LU are not suitable for new domains, because they take a great deal of effort to make rules or transcribe and annotate a sufficient corpus for training. In our method, the weightings of the Weighted Finite State Transducer (WFST) are designed on two levels and simpler than those for conventional WFST-based methods. Therefore, our method needs much fewer training data, which enables rapid prototyping of LU modules. We evaluated our method in two different domains. The results revealed that our method outperformed baseline methods with less than one hundred utterances as training data, which can be reasonably prepared for new domains. This shows that our method is appropriate for rapid prototyping of LU modules.

1 Introduction

The language understanding (LU) of spoken dialogue systems in the early phases of their development should be trained with a small amount of data in their construction. This is because large amounts of annotated data are not available in the early phases. It takes a great deal of effort and time to transcribe and provide correct LU results to a large amount of data. The LU should also be robust, i.e., it should be accurate even if some automatic speech recognition (ASR) errors are contained in its input. A robust LU module is also helpful when collecting dialogue data for the system because it suppresses incorrect LU and unwanted behaviors. We developed a method of rapidly prototyping LU modules that is easy to construct and robust against various expressions. It makes LU modules in the early phases easier to develop.

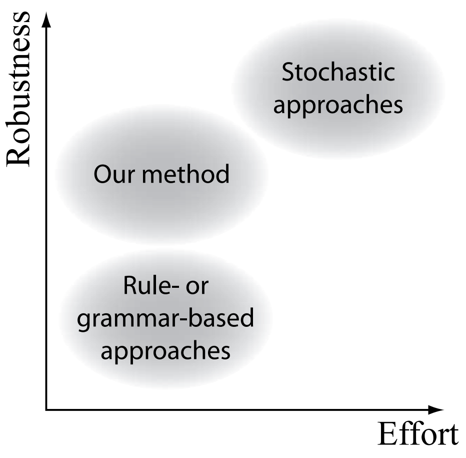

Figure 1: Relationship between our method and conventional methods

Several methods of implementing an LU module in spoken dialogue systems have been proposed. Using grammar-based ASR is one of the simplest. Although its ASR output can easily be transformed into concepts based on grammar rules, complicated grammars are required to understand the user's utterances in various expressions. It takes a great deal of effort to the system developer. Extracting con-

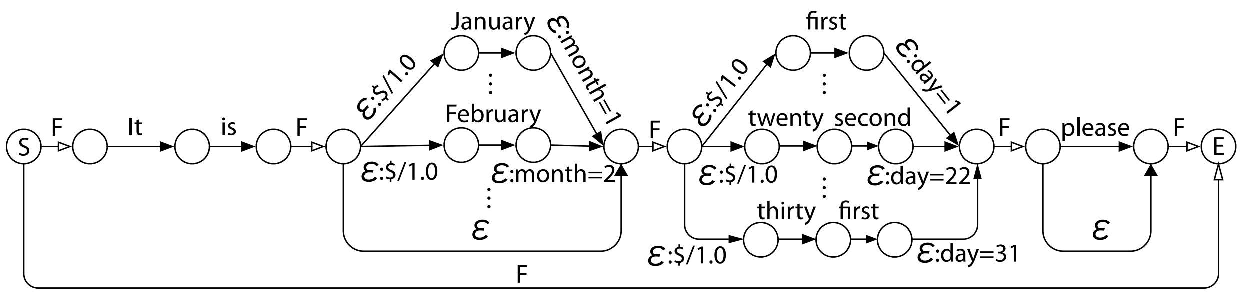

Figure 2: Example of WFST for LU

cepts from user utterances by keyword spotting or heuristic rules has also been proposed (Seneff, 1992) where utterances can be transformed into concepts without major modifications to the rules. However, numerous complicated rules similarly need to be manually prepared. Unfortunately, neither method is robust against ASR errors.

To cope with these problems, corpus-based (Sudoh and Tsukada, 2005; He and Young, 2005) and Weighted Finite State Transducer (WFST)-based methods (Potamianos and Kuo, 2000; Wutiwiwatchai and Furui, 2004) have been proposed as LU modules for spoken dialogue systems. Since these methods extract concepts using stochastic analysis, they do not need numerous complicated rules. These, however, require a great deal of training data to implement the module and are not suitable for constructing new domains.

Here, we present a new WFST-based LU module that has two main features.

1. A statistical language model (SLM) for ASR and a WFST for parsing that are automatically generated from the domain grammar description.

2. Since the weighting for the WFST is simpler than that in conventional methods, it requires fewer training data than conventional weighting schemes.

Our method accomplishes robust LU with less effort using SLM-based ASR and WFST parsing. Figure 1 outlines the relationships between our method and conventional schemes. Since rule- or grammar-based approaches do not require a large amount of data, they take less effort than stochastic techniques. However, they are not robust against ASR errors. Stochastic approaches, on the contrary, take a great deal of effort to collect data but are robust against ASR errors. Our method is an intermediate approach that lies between these. That is, it is more robust than rule- or grammar-based approaches and takes less effort than stochastic techniques. This characteristic makes it easier to rapidly prototype LU modules for a new domain and helps development in the early phases.

2 Related Work and WFST-based Approach

A Finite State Transducer (FST)-based LU is explained here, which accepts ASR output as its input. Figure 2 shows an example of the FST for a video recording reservation domain. The input, ε, means that a transition with no input is permitted at the state transition. In this example, the LU module returns the concept [month=2, day=22] for the utterance "It is February twenty second please". Here, a FILLER transition in which any word is accepted is appropriately allowed between phrases. In Figure 2, 'F' represents 0 or more FILLER transitions. A FILLER transition from the start to the end is inserted to reject unreliable utterances. This FILLER transition enables us to ignore unnecessary words listed in the example utterances in Table 1. The FILLER transition helps to suppress the insertion of incorrect concepts into LU results.

However, many output sequences are obtained for one utterance due to the FILLER transitions, because the utterance can be parsed with several paths. We used a WFST to select the most appropriate path from several output sequences. The path with the highest cumulative weight, w, is selected in a

Table 2: Many LU results for input “It is February twenty second please”

LU output					LU result	w
It	is	February	twenty second	please	month=2, day=22	2.0
It	is	FILLER	twenty second	please	day=22	1.0
It	is	FILLER	twenty second	FILLER	day=22	1.0
FILLER	FILLER	FILLER	FILLER FILLER	FILLER	n/a	0

Table 1: Examples of utterances with FILLERs

ASR output
Well, it is February twenty second please
It is **uhm,** February twenty second please
It is February, **twe-**, twenty second please
It is February twenty second please, **OK?**
(LU result = [month=2, day=22])

WFST-based LU. In the example in Table 2, the concept [month=2, day=22] has been selected, because its cumulative weight, w, is 2.0, which is the highest.

The weightings of conventional WFST-based approaches used an n-gram of concepts (Potamianos and Kuo, 2000) and that of word-concept pairs (Wutiwiwatchai and Furui, 2004). They obtained the n-grams from several thousands of annotated utterances. However, it takes a great deal of effort to transcribe and annotate a large corpus. Our method enables prototype LU modules to be rapidly constructed that are robust against various expressions with SLM-based ASR and WFST-based parsing. The SLM and WFST are generated automatically from a domain grammar description in our toolkit. We need fewer data to train WFST, because its weightings are simpler than those in conventional methods. Therefore, it is easy to develop an LU module for a new domain with our method.

3 Domain Grammar Description

A developer defines grammars, slots, and concepts in a domain in an XML file. This description enables an SLM for ASR and parsing WFST to be automatically generated. Therefore, a developer can construct an LU module rapidly with our method.

Figure 3 shows an example of a description. A definition of a slot is described in `keyphrase-class` tags and its keyphrases and

```
...
<keyphrase-class name="month">
...
  <keyphrase>
    <orth>February</orth>
    <sem>2</sem>
  </keyphrase>
...
</keyphrase-class>
...
<action type="specify-attribute">
  <sentence> {It is} [*month] *day [please]
  </sentence>
</action>
```

Figure 3: Example of a grammar description

the values are in `keyphrase` tags. The `month` is defined as a slot in this figure. `February` and `2` are defined as one of the phrases and values for the slot `month`. A grammar is described in a sequence of terminal and non-terminal symbols. A non-terminal symbol represents a class of keyphrases, which is defined in `keyphrase-class`. It begins with an asterisk “`*`” in a grammar description in `sentence` tags. Symbols that can be skipped are enclosed by brackets `[ ]`. The FILLER transition described in Section 2 is inserted between the symbols unless they are enclosed in brackets `[ ]` or braces `{ }`. Braces are used to avoid FILLER transitions from being inserted. For example, the grammar in Figure 3 accepts “It is February twenty second please.” and “It is twenty second, OK?”, but rejects “It is February.” and “It, uhm, is February twenty second.”.

A WFST for parsing can be automatically generated from this XML file. The WFST in Figure 2 is generated from the definition in Figure 3. Moreover, we can generate example sentences from the grammar description. The SLM for the speech recognizer is generated with our method by using many example sentences generated from the defined grammar.

4 Weighting for ASR Outputs on Two Levels

We define weights on two levels for a WFST. The first is *a weighting for ASR outputs*, which is set to select paths that are reliable at a surface word level. The second is *a weighting for concepts*, which is used to select paths that are reliable on a concept level. The weighting for concepts reflects correctness at a more abstract level than the surface word level. The weighting for ASR outputs consists of two categories: a weighting for ASR N-best outputs and one for accepted words. We will describe the definitions of these weightings in the following subsections.

4.1 Weighting for ASR N-Best Outputs

The N-best outputs of ASR are used for an input of a WFST. Weights are assigned to each sentence in ASR N-best outputs. Larger weights are given to more reliable sentences, whose ranks in ASR N-best are higher. We define this preference as

$$w_s^i = \frac{e^{\beta \cdot score_i}}{\sum_j^N e^{\beta \cdot score_j}},$$

where w_s^i is a weight for the i-th sentence in ASR N-best outputs, β is a coefficient for smoothing, and $score_i$ is the log-scaled score of the i-th ASR output. This weighting reflects the reliability of the ASR output. We set β to 0.025 in this study after a preliminary experiment.

4.2 Weighting for Accepted Words

Weights are assigned to word sequences that have been accepted by the WFST. Larger weights are given to more reliable sequences of ASR outputs at the surface word level. Generally, longer sequences having more words that are not fillers and more reliable ASR outputs are preferred. We define these preferences as the weights:

1. **word(const.)**: $w_w = 1.0$,
2. **word(#phone)**: $w_w = l(W)$, and
3. **word(CM)**: $w_w = CM(W) - \theta_w$.

The **word(const.)** gives a constant weight to all accepted words. This means that sequences with more words are simply preferred. The **word(#phone)** takes the length of each accepted word into consideration. This length is measured by its number of phonemes, which are normalized by that of the longest word in the vocabulary. The normalized values are denoted as $l(W)$ ($0 < l(W) \leq 1$). By adopting **word(#phone)**, the length of sequences is represented more accurately. We also take the reliability of the accepted words into account as **word(CM)**. This uses confidence measures (Lee et al., 2004) for a word, W, in ASR outputs, which are denoted as $CM(W)$. The θ_w is the threshold for determining whether word W is accepted or not. The w_w obtains a negative value for an unreliable word W when $CM(W)$ is lower than θ_w. This represents a preference for longer and more reliable sequences.

4.3 Weighting for Concepts

In addition to the ASR level, weights on a concept level are also assigned. The concepts are obtained from the parsing results by the WFST, and contain several words. Weights for concepts are defined by using the measures of all words contained in a concept.

We prepared three kinds of weights for the concepts:

1. **cpt(const.)**: $w_c = 1.0$,
2. **cpt(avg)**:
$$w_c = \frac{\sum_{\boldsymbol{W}}(CM(W) - \theta_c)}{\#\boldsymbol{W}}, \text{and}$$
3. **cpt(#pCM(avg))**:
$$w_c = \frac{\sum_{\boldsymbol{W}}(CM(W) \cdot l(W) - \theta_c)}{\#\boldsymbol{W}},$$

where $\boldsymbol{W}$ is a set of accepted words, W, in the corresponding concept, and $\#\boldsymbol{W}$ is the number of words in $\boldsymbol{W}$.

The **cpt(const.)** represents a preference for sequences with more concepts. The **cpt(avg)** is defined as the weight by using the $CM(W)$ of each word contained in the concept. The **cpt(#pCM(avg))** represents a preference for longer and reliable sequences with more concepts. The θ_c is the threshold for the acceptance of a concept.

Table 3: Examples of weightings when parameter set is: **word(CM)** and **cpt(#pCM(avg))**

ASR onput	No,	it	is	February	twenty	second
LU output	FILLER	it	is	February	twenty	second
$CM(W)$	0.3	0.7	0.6	0.9	1.0	0.9
$l(W)$	0.3	0.2	0.2	0.9	0.6	0.5
Concept	-	-	-	month=2	day=22	
word	-	$0.7-\theta_w$	$0.6-\theta_w$	$0.9-\theta_w$	$1.0-\theta_w$	$0.9-\theta_w$
cpt	-	-	-	$(0.9\cdot 0.9-\theta_c)/1$	$(1.0\cdot 0.6-\theta_c+0.9\cdot 0.5-\theta_c)/2$	

Reference	From	June	third			please	LU result
ASR output	From	June	third	uhm	FIT	please	
$CM(W)$	0.771	0.978	0.757	0.152	0.525	0.741	
LU reference	From	June	third	FILLER	FILLER	FILLER	month:6, day:3
Our method	From	June	third	FILLER	FILLER	FILLER	month:6, day:3
Keyword spotting	From	June	third	FILLER	FIT	please	month:6, day:3, car:FIT

('FIT' is the name of a car.)

Figure 4: Example of LU with WFST

4.4 Calculating Cumulative Weight and Training

The LU results are selected based on the weighted sum of the three weights in Subsection 4.3 as

$$w^i = w_s^i + \alpha_w \sum w_w + \alpha_c \sum w_c$$

The LU module selects an output sequence with the highest cumulative weight, w^i, for $1 \leq i \leq N$.

Let us explain how to calculate cumulative weight w^i by using the example specified in Table 3. Here, **word(CM)** and **cpt(#pCM(avg))** are selected as parameters. The sum of weights in this table for accepted words is $\alpha_w(4.1 - 5\theta_w)$, when the input sequence is "No, it is February twenty second.". The sum of weights for concepts is $\alpha_c(1.335 - 2\theta_c)$ because the weight for "month=2" is $\alpha_c(0.81 - \theta_c)$ and the weight for "day=22" is $\alpha_c(0.525 - \theta_c)$. Therefore, cumulative weight w^i for this input sequence is $w_s^i + \alpha_w(4.1 - 5\theta_w) + \alpha_c(1.335 - 2\theta_c)$.

In the training phase, various combinations of parameters are tested, i.e., which weightings are used for each of ASR output and concept level, such as $N = 1$ or 10, coefficient $\alpha_{w,c} = 1.0$ or 0, and threshold $\theta_{w,c} = 0$ to 0.9 at intervals of 0.1, on the training data. The coefficient $\alpha_{w,c} = 0$ means that a corresponding weight is not added. The optimal parameter settings are obtained after testing the various combinations of parameters. They make the concept error rate (CER) minimum for a training data set. We calculated the CER in the following equation: $CER = (S + D + I)/N$, where N is the number of concepts in a reference, and S, D, and I correspond to the number of substitution, deletion, and insertion errors.

Figure 4 shows an example of LU with our method, where it rejects misrecognized concept [car:FIT], which cannot be rejected by keyword spotting.

5 Experiments and Evaluation

5.1 Experimental Conditions

We discussed our experimental investigation into the effects of weightings in Section 4. The user utterance in our experiment was first recognized by ASR. Then, the i-th sentence of ASR output was input to WFST for $1 \leq i \leq N$, and the LU result for the highest cumulative weight, w^i, was obtained.

We used 4186 utterances in the video recording reservation domain (video domain), which consisted of eight different dialogues with a total of 25 different speakers. We also used 3364 utterances in the rent-a-car reservation domain (rent-a-car domain) of eight different dialogues with 23 different speakers. We used Julius [1] as a speech recognizer with an SLM. The language model was prepared by using example sentences generated from the grammars of both domains. We used 10000 example sentences in the video and 40000 in the rent-a-car domain. The number of the generated sentences was determined empirically. The vocabulary size was 209 in the video and 891 in the rent-a-car domain. The average ASR accuracy was 83.9% in the video and 65.7% in the rent-a-car domain. The grammar in the video domain included phrases for dates, times, channels, commands. That of the rent-a-car domain included phrases for dates, times, locations, car classes, options, and commands. The WFST parsing module was implemented by using the MIT FST toolkit (Hetherington, 2004).

5.2 Performance of WFST-based LU

We evaluated our method in the two domains: video and rent-a-car. We compared the CER on test data, which was calculated by using the optimal settings for both domains. We evaluated the results with 4-fold cross validation. The number of utterances for training was 3139 (=4186*(3/4)) in the video and 2523 (=3364*(3/4)) in the rent-a-car domain.

The baseline method was simple keyword spotting because we assumed a condition where a large amount of training data was not available. This method extracts as many keyphrases as possible from ASR output without taking speech recognition errors and grammatical rules into consideration. Both grammar-based and SLM-based ASR outputs are used for input in keyword spotting (denoted as "Grammar & spotting" and "SLM & spotting" in Table 4). The grammar for grammar-based ASR was automatically generated by the domain description file. The accuracy of grammar-based ASR was 66.3% in the video and 43.2% in the rent-a-car domain.

Table 4 lists the CERs for both methods. In keyword spotting with SLM-based ASR, the CERs were improved by 5.2 points in the video and by 22.2 points in the rent-a-car domain compared with those with grammar-based ASR. This is because SLM-based ASR is more robust against fillers and unknown words than grammar-based ASR. The CER was improved by 3.4 and 6.9 points by optimal weightings for WFST. Table 5 lists the optimal parameters in both domains. The $\alpha_c = 0$ in the video domain means that weights for concepts were not used. This result shows that optimal parameters depend on the domain for the system, and these need to be adapted for each domain.

Table 4: Concept error rates (CERs) in each domain

Domain	Grammar & spotting	SLM & spotting	Our method
Video	22.1	16.9	13.5
Rent-a-car	51.1	28.9	22.0

5.3 Performance According to Training Data

We also investigated the relationship between the size of the training data for our method and the CER. In this experiment, we calculated the CER in the test data by increasing the number of utterances for training. We also evaluated the results by 4-fold cross validation.

Figures 5 and 6 show that our method outperformed the baseline methods by about 80 utterances in the video domain and about 30 utterances in the rent-a-car domain. These results mean that our method can effectively be used to rapidly prototype LU modules. This is because it can achieve robust LU with fewer training data compared with conventional WFST-based methods, which need over several thousand sentences for training.

6 Conclusion

We developed a method of rapidly prototyping robust LU modules for spoken language understanding. An SLM for a speech recognizer and a WFST for parsing were automatically generated from a domain grammar description. We defined two kinds of weightings for the WFST at the word and concept levels. These two kinds of weightings were calculated by ASR outputs. This made it possible to create an LU module for a new domain with less effort because the weighting scheme was relatively simpler than those of conventional methods. The optimal parameters could be selected with fewer training data in both domains. Our experiment re-

[1] http://julius.sourceforge.jp/

Table 5: Optimal parameters in each domain

Domain	N	α_w	w_w	α_c	w_c
Video	1	1.0	**word(const.)**	0	-
Rent-a-car	10	1.0	**word(CM)**-0.0	1.0	**cpt(#pCM(avg))**-0.8

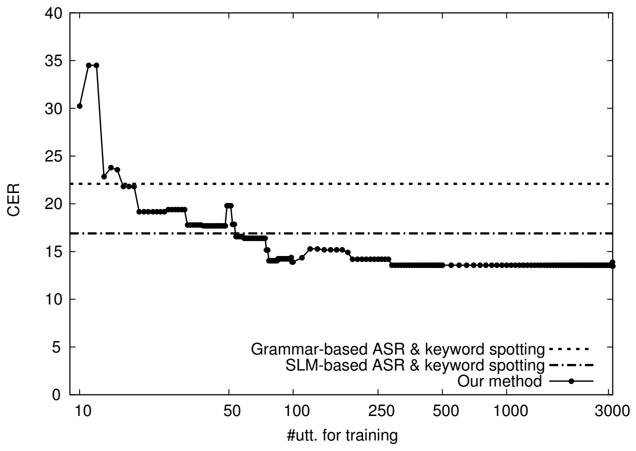

Figure 5: CER in video domain

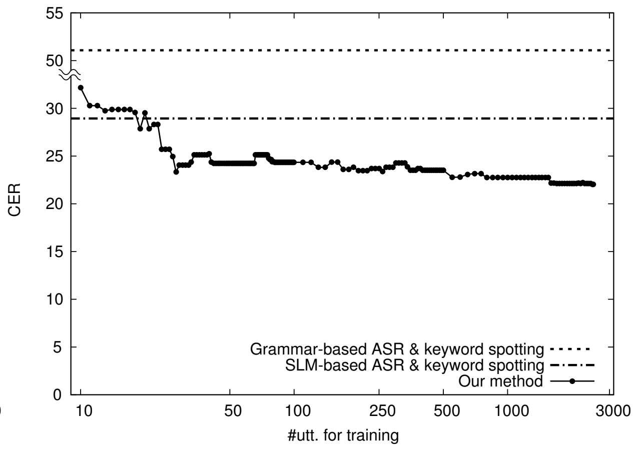

Figure 6: CER in rent-a-car domain

vealed that the CER could be improved compared to the baseline by training optimal parameters with a small amount of training data, which could be reasonably prepared for new domains. This means that our method is appropriate for rapidly prototyping LU modules. Our method should help developers of spoken dialogue systems in the early phases of development. We intend to evaluate our method on other domains, such as database searches and question answering in future work.

Acknowledgments

We are grateful to Dr. Toshihiko Ito and Ms. Yuka Nagano of Hokkaido University for constructing the rent-a-car domain system.

References

Yulan He and Steve Young. 2005. Spoken Language Understanding using the Hidden Vector State Model. *Speech Communication*, 48(3-4):262–275.

Lee Hetherington. 2004. The MIT finite-state transducer toolkit for speech and language processing. In *Proc. 6th International Conference on Spoken Language Processing (INTERSPEECH-2004 ICSLP)*.

Akinobu Lee, Kiyohiro Shikano, and Tatsuya Kawahara. 2004. Real-time word confidence scoring using local posterior probabilities on tree trellis search. In *Proc. 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, volume 1, pages 793–796.

Alexandors Potamianos and Hong-Kwang J. Kuo. 2000. Statistical recursive finite state machine parsing for speech understanding. In *Proc. 6th International Conference on Spoken Language Processing (INTERSPEECH-2000 ICSLP)*, pages 510–513.

Stephanie Seneff. 1992. TINA: A natural language system for spoken language applications. *Computational Linguistics*, 18(1):61–86.

Katsuhito Sudoh and Hajime Tsukada. 2005. Tightly integrated spoken language understanding using word-to-concept translation. In *Proc. 9th European Conference on Speech Communication and Technology (INTERSPEECH-2005 Eurospeech)*, pages 429–432.

Chai Wutiwiwatchai and Sadaoki Furui. 2004. Hybrid statistical and structural semantic modeling for Thai multi-stage spoken language understanding. In *Proc. HLT-NAACL Workshop on Spoken Language Understanding for Conversational Systems and Higher Level Linguistic Information for Speech Processing*, pages 2–9.