IEICE TRANS. INF. & SYST., VOL.E95–D, NO.9 SEPTEMBER 2012

PAPER

# Automatic Allocation of Training Data for Speech Understanding Based on Multiple Model Combinations*

**Kazunori KOMATANI**[†a)], **Mikio NAKANO**[††], ***Members***, **Masaki KATSUMARU**[†††], **Kotaro FUNAKOSHI**[††], **Tetsuya OGATA**[†††], ***and*** **Hiroshi G. OKUNO**[†††], ***Nonmembers***

**SUMMARY** The optimal way to build speech understanding modules depends on the amount of training data available. When only a small amount of training data is available, effective allocation of the data is crucial to preventing overfitting of statistical methods. We have developed a method for allocating a limited amount of training data in accordance with the amount available. Our method exploits rule-based methods for when the amount of data is small, which are included in our speech understanding framework based on multiple model combinations, i.e., multiple automatic speech recognition (ASR) modules and multiple language understanding (LU) modules, and then allocates training data preferentially to the modules that dominate the overall performance of speech understanding. Experimental evaluation showed that our allocation method consistently outperforms baseline methods that use a single ASR module and a single LU module while the amount of training data increases.

***key words:*** *spoken dialogue system, language understanding, rapid prototyping, limited amount of training data*

## 1. Introduction

The objective of speech understanding in spoken dialogue systems is to extract a semantic representation of a user's utterance. This process is composed of automatic speech recognition (ASR) and language understanding (LU); ASR transcribes the speech signals of a spoken utterance into text, and LU fills in a slot structure in accordance with the obtained text. The result represents the semantics of the utterance. In this process, the main issue is how to construct the speech understanding module for the target domain of the spoken dialogue system because the vocabularies and language expressions depend on its domain. This means that training data are required for each system. In general, the performance of speech understanding depends on the amount of training data when ASR and LU are based on statistical methods.

An important issue when collecting training data is how to collect "real" data. In general, training data are better when they are more similar to user utterances that are observed at run time. A Wizard-of-Oz (WoZ) method is often used for collecting user utterances, but users tend to speak to a WoZ system differently than to real systems using ASR. That is, the utterances collected by a WoZ system are often different from those collected by a system using ASR. Hence, a *rapid prototyping* technique, which enables the construction of a prototype system even when only a small amount of training data is available, that collects more real data is required. More real data and their reference tags lead to higher performance of statistical LU methods.

One key idea to realize such rapid prototyping is to use multiple model combinations. We have been developing a framework called "Multiple Language models for ASR and Multiple language Understanding models (MLMU)" [2]. In the MLMU framework, different kinds of ASR and LU methods based on hand-crafted grammar and statistical models are used, and the most reliable speech understanding result is selected from candidates produced by the various model combinations. The framework can include a hand-crafted grammar-based method, which is effective at an early stage of system development because it does not require training data. Since the different kinds of ASR and LU methods work complementarily, this framework has better speech understanding performance than ones that use a single combination of ASR and LU methods.

The other key idea is to allocate limited available data. When there is a small amount of training data, the data need to be allocated appropriately to the modules based on statistical models to prevent overfitting. There are three kinds of modules in the MLMU framework:

1. ASR modules with language models (LMs),
2. LU modules with LU models (LUMs), and
3. a selection module.

If these modules are trained without data allocation, their performances would degrade due to overfitting. This is the case when there is a small amount of training data available because training of these modules on the same data set would be considered as training under a closed-set condition. More specifically, the data used for training the selection module would include too many correct understanding results. Such overfitting can be avoided by dividing the data into sub-sets. If the amount of data available for training is large, such overfitting does not occur because a variety of data has already been obtained in the training set. Hence, all available data should be used to train each statistical module because using more training data generally improves perfor-

Manuscript received December 28, 2011.
Manuscript revised April 6, 2012.

[†]The author is with the Graduate School of Engineering, Nagoya University, Nagoya-shi, 464–8603 Japan.
[††]The authors are with Honda Research Institute Japan, Co., Ltd., Wako-shi, 351–0188 Japan.
[†††]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.

*This paper is a modified and extended version of our earlier report [1].

a) E-mail: komatani@nuee.nagoya-u.ac.jp

DOI: 10.1587/transinf.E95.D.2298

mance.

We have developed a method for switching data allocation policies as the amount of training data increases. More specifically, two points are automatically determined at which statistical modules having more parameters start to be trained. As a result, our method consistently outperforms baseline methods that use a single ASR module and a single LU module while the amount of training data increases, especially when a small amount of data is available.

The rest of the paper is organized as follows: Sect. 2 explains the concept behind our MLMU framework and describes its current implementation. Section 3 compares related work with our work. Section 4 presents our automatic data allocation method, and Sect. 5 describes our experimental evaluation. We conclude in Sect. 6 with a summary of the key points and a mention of future work.

## 2. MLMU Framework

### 2.1 Concept

Many methods for improving robustness in speech understanding have been proposed. Most of them aim at accurately extracting semantic representations from noisy ASR results using unstructured statistical language models such as N-gram models. When such models are used, the results may be ungrammatical. Robust language understanding methods have been developed for obtaining semantic representations by also using statistical methods. In addition, they improve understanding accuracy by ignoring recognized words with low confidence scores.

Most deployed spoken dialogue systems, however, still use finite-state-grammar-based language models for speech recognition. There seems to be two reasons for this. One is that it is not easy to collect enough training data for statistical language models during system development, resulting in poor ASR accuracy. The other is that, if a user utterance is covered by the finite-state-grammar, grammar-based ASR tends to perform better than statistical-language-model-based ASR. In most cases, users make in-grammar utterances when using dialogue systems in a limited domain.

Nevertheless, robust speech understanding based on statistical language models is desirable because users sometimes make out-of-grammar utterances, which are always misrecognized by grammar-based ASR. We therefore need a way to achieve both robustness using statistical-language model-based methods and accuracy using grammar-based methods.

One possible approach is to obtain the N-best ASR results with a statistical-model-based speech recognizer and rescore them on the basis of whether they are covered by grammar or whether parts of the utterances can be recognized as grammatical phrases [3]. The problem with this approach is that an ASR result that is covered by grammar may not appear in the N-best results. This problem could be solved by setting $N$ to a larger value, but this would increase computation cost substantially. Another approach is to use probabilistic finite-state grammars for the ASR language models [4]. Although this would improve ASR accuracy, some utterances might be misrecognized due to their structural constraints.

What we need is not high ASR accuracy but high speech understanding accuracy. Therefore, the role of speech recognition in speech understanding should be to generate a set of recognition results that is highly likely to include the correct ASR result or recognition results from which the correct understanding result can be obtained. Selecting a correct result is not the role of ASR but the role of speech understanding.

The probability that the speech recognizer generates the correct ASR result can be increased by using a variety of language models, such as finite state grammar and statistical language models. It is also important to use a variety of language understanding methods [5], [6] such as ones using grammatical constraints and ones using less-structured statistical models. This is because no single understanding method achieves both robustness and accuracy.

We developed our MLMU framework [2] on the basis of these insights. As its name implies, it uses multiple language models for ASR, such as finite-state-grammars and N-gram statistical models, and multiple models for language understanding, such as finite-state-transducers [7], weighted finite-state-transducers [8], [9], keyphrase extractors, and conditional-random-field-based models.

### 2.2 Current Implementation

An overview of an MLMU implementation used in this paper is depicted in Fig. 1. As mentioned, MLMU uses multiple LMs for ASR and multiple LUMs for LU and selects the most reliable speech understanding result from candidates produced by the various module combinations.

Figure 2 depicts example speech understanding results produced by multiple ASR modules based on LMs and multiple LU modules based on LUMs. Only two results are shown here out of the four combinations of two ASR modules and two LU modules. This example shows that the correct speech understanding result can be obtained from a particular LM-LUM combination.

Each LM-LUM combination is referred to as a speech understanding (SU) module ($SU_i$, where $i = 1, \ldots, n$). It produces a semantic representation consisting of a set of concepts. A concept is either a semantic slot and its value or the dialogue act type. Note that $n = N \times M$ when $N$ LMs and $M$ LUMs are used. The LMs and LUMs are trained on given training data if they are statistical ones; for example, an N-gram LM is trained by using the available collected utterances. A hand-crafted grammar rule and its ASR result do not change when the amount of training data increases.

The confidence measure per utterance for the result of the $i$-th speech understanding module $SU_i$ is denoted as $CM_i$. The speech understanding result having the highest confidence measure is selected as the final result for the utterance. That is, the result is the output of $SU_m$, where

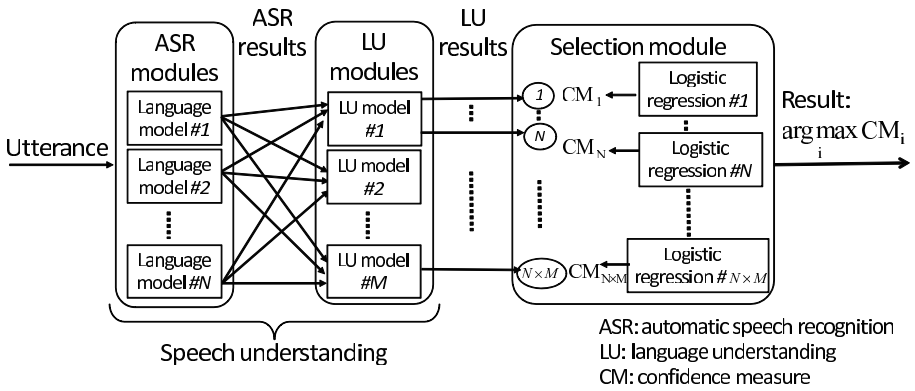



**Fig. 1** Overview of MLMU speech understanding framework.

**U1: It is June ninth.**
ASR result:
- **grammar** "It is June ninth."
- **N-gram** "It is June noon and"

LU result:
- **grammar + FST** "month:6 day:9 type:refer-time"
- **N-gram + WFST** "month:6 type:refer-time"

**U2: I will borrow it on the twentieth.**
(Underlined part is out-of-grammar.)
ASR result:
- **grammar** "Around two pm on the twentieth."
- **N-gram** "Around two at ten on the twentieth."

LU result:
- **grammar + FST** "day:20 hour:14 type:refer-time"
- **N-gram + WFST** "day:20 type:refer-time"

Combination of an LM and an LUM is denoted as "LM+LUM."
(FST: finite state transducer, WFST: weighted FST)

**Fig. 2** Example of speech understanding results in MLMU framework.

$m = \mathrm{argmax}_i CM_i$. The confidence measure is calculated using logistic regression on the basis of the features of each speech understanding result:

$$CM_i = \frac{1}{1 + \exp(-(a_{i0} + a_{i1}F_{i1} + \cdots + a_{i7}F_{i7}))}. \quad (1)$$

We used the data mining software Weka [10] for training the logistic regression function.

The functions are constructed for each speech understanding module $i$. Parameters $a_{i0}, \ldots, a_{i7}$ are determined by using training data†. In the training phase, teacher signal 1 is given when a speech understanding result is completely correct; that is, when no error is contained in the result. Otherwise, 0 is given. We use seven features, $F_{i1}, F_{i2}, \ldots, F_{i7}$, as independent variables. Each feature value is normalized so as to make its mean zero and its variance one.

The features used are listed in Table 1. They were selected by performing backward stepwise feature selection from more features we previously prepared. For example, features such as utterance duration in seconds and the average number of concepts in ten-best ASR candidates, used previously [2], were removed during the selection process. Feature $F_{i1}$ and $F_{i2}$ represents the reliability of the current ASR result. Feature $F_{i1}$ is the acoustic score for the ASR result with the current LM. The score is normalized by the utterance length. $F_{i2}$ is the difference between $F_{i1}$ and the acoustic score of another ASR with a domain-independent large vocabulary LM for utterance verification [11]. $F_{i3}$ and $F_{i4}$ are calculated for each concept in the LU result on the basis of the posterior probability of the ten-best ASR candidates [12]. $F_{i5}$ is the number of concepts in the LU result. This feature is effective because the LU results of lengthy utterances tend to be erroneous in a grammar-based LU. $F_{i6}$ represents the case when an ASR result is not accepted by the subsequent LU module. In such cases, no speech understanding result is obtained, which is regarded as an error. $F_{i7}$ represents that affirmative and negative responses, typically "Yes" and "No," tend to be correctly recognized and understood.

**Table 1** Features of speech understanding result obtained from $SU_i$.

| | |
|---|---|
| $F_{i1}$: | Acoustic score normalized by utterance length |
| $F_{i2}$: | Difference between $F_{i1}$ and normalized acoustic score of verification ASR |
| $F_{i3}$: | Average concept CM in understanding result |
| $F_{i4}$: | Minimum concept CM in understanding result |
| $F_{i5}$: | Number of concepts in understanding result |
| $F_{i6}$: | Whether any understanding result is obtained |
| $F_{i7}$: | Whether understanding result is yes/no |

CM: confidence measure

## 3. Related Work

Many statistical LU methods have been developed, e.g., [13]–[16]. Raymond et al. developed a sequential decision strategy for LU results [5]. Hahn et al. employed six different LU methods with a single ASR result and combined them using a weighted ROVER method [6]. These methods outperform grammar-based LU methods in general when a sufficient amount of training data is available. However, sufficient training data are not always available when developing spoken dialogue systems, i.e., at an early stage of development.

Several LU methods were constructed using a smaller

†No specific initial values were set to $a_{ij}$; the default setting of Weka was used in the experiment.

amount of training data [8], [9], [17]. Fukubayashi et al. [8] constructed an LU method based on the WFST model, in which filler transitions accepting arbitrary inputs and transition weights are added to a hand-crafted FST model. This method falls between grammar-based and statistical methods because a statistically selected weighting scheme is applied to a hand-crafted grammar model. Therefore, the amount of training data can be smaller than with general statistical LU methods. However, this method does not outperform statistical ones when plenty of training data are available. Potamianos and Kuo [9] also developed a semantic parser based on WFST, which combines handcoded rules and probabilities. Dinarelli et al. [17] used a generative model for which overfitting is less prone to occur than with discriminative models when the amount of training data is small, but they did not use a grammar-based model, which is expected to achieve reasonable performance even when the amount of training data is very small.

Raymond and Riccardi [15] compared the performances of statistical LU methods for various amounts of training data. They used a statistical finite-state transducer (SFST) as a generative model and a support vector machine (SVM) and CRF as discriminative models. The generative model is more effective when the amount of data is small, and the discriminative models are more effective when it is large. This shows that the performance of an LU method depends on the amount of training data available and supports our assertion that LU methods need to be switched automatically.

Wang et al. [18] developed a two-stage speech understanding method by applying statistical methods first and then grammatical rules. They also examined the performance of the statistical methods at their first stage for various amounts of training data and showed that the performance is not very good when a small amount of data is used.

Schapire et al. [19] showed that the accuracy of call classification in spoken dialogue systems is improved by incorporating hand-crafted prior knowledge into their boosting algorithm. Their idea is the same as ours in that they improve system performance by using hand-crafted human knowledge when a small amount of training data is available. We furthermore try to solve the data allocation problem because there are multiple statistical models to be trained in speech understanding, while their call classification has only one statistical model.

## 4. Automatic Allocation of Training Data

Here we describe how a limited amount of training data is allocated to the modules in the MLMU framework in accordance with the amount of data. The data need to be allocated to the SU modules (i.e., statistical LM and statistical LUM) and the selection module. If more data are allocated to the ASR and LU modules, the performances of these modules improve, but the overall performance degrades because the performance of the selection module is degraded. On the other hand, even if a lot of training data is allocated to the selection module, the performance of each ASR and LU module remains low. That is, we need to allocate limited available data by considering the total amount of data and the characteristics of each module.

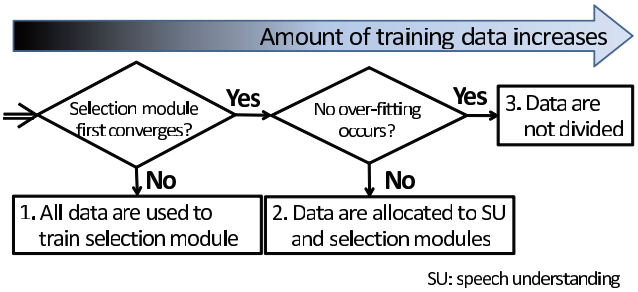



**Fig. 3** Flowchart of data allocation.

### 4.1 Allocation Policy

The performance of the selection module is important because poor selection results in incorrect selection of the final LU result even if correct LU results were obtained as candidates. Furthermore, if only a small amount of training data is available, grammar-based ASR and LU methods can output a correct result in the MLMU framework. We therefore adopt an approach in which available training data are

1. allocated to the selection module first and
2. then allocated to models for ASR and LU after a minimal amount of training data is secured for the selection module.

There are three phases of allocation, as depicted in Fig. 3 and explained below.

In the first phase, the first priority is given to the selection module, so all available data are used to train the selection module. When a very small amount of training data is available, the output from an SU module that uses a grammar-based LM and LUM would be the most reliable because its performance is better than that of other statistical modules. In such a case, we give up training the statistical LM and LUM and concentrate on training the selection module.

In the second phase, training data are also allocated to the SU modules. The aim is to improve the performance of these modules by allocating as much training data to them as possible after the performance of the selection module converges. The amount of training data is fixed in this phase to the amount allocated to the selection module in the first phase. The remaining data are used to train the SU modules.

When the performances of all the SU modules stabilize, allocation proceeds to the third phase. We assume that overfitting no longer occurs in this phase because sufficient training data are available. All available data are used to train all modules without dividing the data.

### 4.2 Determining When to Switch Allocation Policies

We next explain how to determine the two points that divide

the three phases described above. We focus on the convergence of the logistic regression functions as the amount of training data increases. The convergence is defined as the change in their coefficients, which will be shown later as Eq. (2).

Automatic switching from one phase to the next requires determination of two points in the number of training utterances $k$: when the selection module first converges ($k_{onlysel}$) and when the SU modules all become stable ($k_{nodiv}$). These points are determined by the changes in the coefficients of the logistic regression functions as the number of utterances used as training data increases. If the sum of the changes in the coefficients becomes small enough, we consider the training process to have converged. The points are determined individually using the following algorithm.

**Step 1** Construct two logistic regression functions for speech understanding module $SU_i$ by using $k$ and $(k + \delta k)$ utterances out of $k_{max}$ utterances, where $k_{max}$ is the amount of training data available at each point.

**Step 2** Calculate the change in coefficients from the two logistic regression functions using

$$\Delta_i(k) = \sum_j \left| a_{ij}(k + \delta k) - a_{ij}(k) \right|, \quad (2)$$

where $a_{ij}(k)$ denotes the parameters of the logistic regression functions, shown in Eq. (1), for $SU_i$, when $k$ utterances are used to train the function.

**Step 3** If $\Delta_i(k)$ becomes smaller than threshold $\theta$, consider that the training of the function has converged and record this $k$ as the point of convergence. If not, return to Step 1 after $k \leftarrow k + \delta k$.

The $\delta k$ is the minimum unit of training data containing various utterances. We set it as the number of utterances in one dialogue session, with 17 utterances on average.

The first point, $k_{onlysel}$, is determined using an SU module that needs no training data. Specifically, we used "grammar+FST" as $SU_i$. Note again that "*LM+LUM*" denotes a combination of an LM for ASR and an LUM for LU. If the function has converged after $\overline{k}$ utterances, we set $\overline{k}$ to $k_{onlysel}$ and fix the $\overline{k}$ utterances as training data used by the selection module. The remaining $(k_{max} - \overline{k})$ utterances are allocated to the SU modules, that is, the LMs and LUMs. Note that, if $k$ becomes equal to $k_{max}$ before $\Delta_i(k)$ becomes small enough, all training data are allocated to the selection module; that is, no data are allocated to the LMs and LUMs. In this case, no output is obtained from the statistical SU modules, and only outputs from the grammar-based modules are used.

The second point, $k_{nodiv}$, is determined using the SU module that needs the largest amount of training data. The amount of data needed depends on the number of parameters. In the current implementation, this module is "N-gram+CRF", and it is used as $SU_i$ in Eq. (2). $SU_i$ (i.e., "N-gram+CRF") is trained with the same $k$ and $(k + \delta k)$ utterances as used for the logistic regression functions before they are trained in Step 1. This is because the process to determine $k_{nodiv}$ is to determine whether overfitting occurs or not. The training data are not divided in this case: $k_{nodiv}$ is determined independently of $k_{onlysel}$. If the function has converged, i.e., $\Delta_i(k)$ has become small enough, we assume that the performances of all the SU modules have stabilized, so overfitting does not occur. We then stop the allocation of training data and use all available data to train the statistical modules.

The point $k_{nodiv}$ is determined by using information from the selection module instead of from the LUM itself although it is used to determine whether the LUM converges or not. This is because our MLMU framework does not assume specific LUMs. We therefore take into consideration the number of LUM parameters only to select the LUM that needs the largest amount of training data.

## 5. Experimental Evaluation

### 5.1 Target Data and Implementation

We used a data set collected by using a Japanese rent-a-car reservation system [20] with 39 participants. Each participant performed 8 dialogue sessions, and 5,900 utterances were collected in total. Out of these utterances, we used 5,240 after eliminating those for which the automatic voice activity detection (VAD) results agreed with manual annotation. We divided the utterances into two sets: 2,121 by 16 participants as training data and 3,119 by 23 participants as the test data [20]. The accuracies given in the following sections were calculated using the test data.

We also constructed an enhanced version of the reservation system to evaluate our allocation method. The system had two language models (LMs) and four language understanding models (LUMs), so eight speech understanding results in total were obtained. The two LMs were a grammar-based ("grammar," hereafter) one and a domain-specific statistical ("N-gram") one. The grammar model was described by hand to be equivalent to the finite state transducer (FST) model used for LU. The N-gram model was a class 3-gram model and was trained on a transcription of the available training data. The vocabulary size was 281 for the grammar model and 420 for the N-gram model when all the training data were used. The ASR accuracies of the grammar and N-gram models were 67.8% and 90.5% for the training data and 66.3% and 85.0% for the test data when all the training data were used. We used Julius (ver. 4.1.2) as the speech recognizer and a gender-independent phonetic-tied mixture model as the acoustic model [21]. We also used a domain-independent statistical LM with a vocabulary size of 60,250, which was trained on Web documents [21], for utterance verification. This score was used in the selection module as one of the features.

The four LUMs were an FST model, a weighted FST (WFST) model, a keyphrase-extractor (Extractor) model, and a conditional random fields (CRF) model. In the FST-based LUM, the FST was constructed by hand. Its vocabulary size was 278, and its coverage was 81.3% for the training data. It uses ten-best ASR results as its input and out-

puts the first parsing result for the ASR results acceptable by the FST. The WFST-based LUM is based on the method developed by Fukubayashi et al. [8]. The WFSTs were constructed by using the MIT FST Toolkit [22]. The weighting scheme used for the test data was selected by using training data. In the extractor-based LUM, the system tries to simply extract as many concepts as possible from an ASR result. We used open-source software, CRF++[†], to construct the LUM. As its features, we use a word in the ASR result, its first character, its last character, and the ASR confidence of the word [12]. Its parameters were estimated by using training data.

The metric used for speech understanding performance was concept understanding accuracy, defined as

$$1 - \frac{\text{SUB} + \text{INS} + \text{DEL}}{\text{\# of concepts in correct results}},$$

where SUB, INS, and DEL denote the number of substitution, insertion, and deletion errors for concepts.

### 5.2 Effectiveness of Using Multiple LMs and LUMs

The basic idea of the MLMU framework rests on our assumption that ASR and LU modules work complementarily. To verify whether this assumption is correct, especially when the amount of data changes, we used oracle selection when multiple ASR and LU modules were used; i.e., the most appropriate result was selected by hand. Then we calculated how much the performance of our framework degraded when one ASR or LU module was removed. The result revealed the contribution of each ASR and LU module to the overall performance of the framework. A module is regarded as more important when the accuracy degrades more when it is removed than when another one is removed. Two cases, A and B, were defined: in case A, a small amount of training data was available; in case B, a large amount of training data was available. We used 141 utterances with 1 participant for case A[††] and 2,121 utterances with 16 participants for case B. The results are shown in Table 2.

For case A, the accuracy degraded by 12.0 points when the grammar-based ASR module was removed and by 6.1 points when the N-gram-based ASR module was removed. The accuracy thus degrades substantially when either ASR module is removed. This indicates that the two ASR modules work complementarily when only a small amount of training data is available.

For case B, the accuracy degraded by 1.1 points when the grammar-based ASR was removed. This means that this module is less important when there are plenty of training data because the coverage of the N-gram-based ASR becomes wider. In summary, an ASR module based on a handcrafted grammar is more important because of the low performance of a statistical one when the amount of training data is smaller.

In case A, the accuracy was degraded when any of the LUM modules was removed. In case B, the CRF-based module was particularly important. This is because CRF can handle various utterance patterns when a sufficient amount of data is available.

**Table 2** Absolute degradation in oracle accuracy when a module was removed.

| Case | A | B |
|---|---|---|
| With all modules (%) | 86.6 | 90.1 |
| w/o grammar ASR | −12.0 | −1.1 |
| w/o N-gram ASR | −6.1 | −7.7 |
| w/o FST LUM | −0.4 | 0.0 |
| w/o WFST LUM | −1.2 | −0.5 |
| w/o Extractor LUM | −0.1 | 0.0 |
| w/o CRF LUM | −0.6 | −3.7 |
| (w/o FST & Extractor LUMs) | −1.0 | −0.1 |

A: 141 utterances, 1 participant
B: 2,121 utterances, 16 participants

### 5.3 Evaluation of Automatic Allocation

Figure 4 shows the change in the sum of the coefficients, $\Delta_i$, with an increase in the amount of training data. The number of training utterances (plotted on the $x$-axis) increases by $\delta k$, that is, by the number in one dialogue session (17 on average). As shown in Fig. 4 (a), the change was large when the amount of training data was small. It decreased dramatically and converged when around 100 utterances were available. Then, we set threshold $\theta$ to 8[†††]. By applying the threshold to $\Delta_i$, we set the first point, $k_{onlysel}$, to 111 utterances. That is, up to that point, all the training data were allocated to the selection module, as described in Sect. 4.1. Similarly, from the results shown in Fig. 4 (b), we set the second point, $k_{nodiv}$, to 207 utterances. That is, from that point, the training data were not allocated.

To evaluate our method for allocating training data, we compared it with two baseline methods:

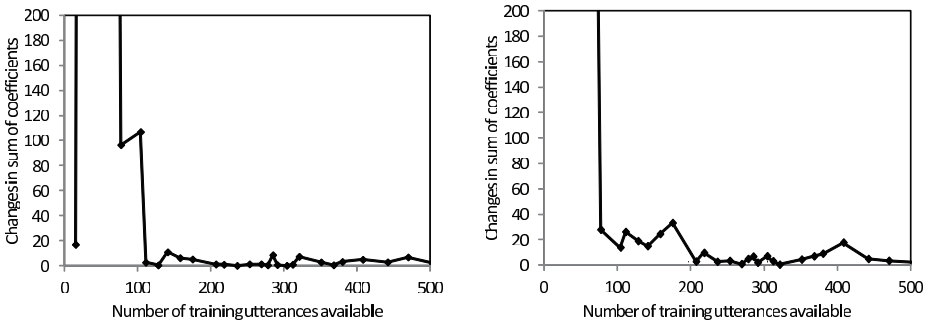
- No-division method: All data available at each point are used to train both the SU modules and the selection module. That is, the same data set is used to train them.
- Naive-allocation method: Training data available at each point are allocated equally (i.e., half to each) to the SU modules and the selection module.

The results are shown in Fig. 5. The $x$-axis represents the number of training utterances available on a log scale. The points were 56, 104, 141, 278, 521, 1115, and 2121 utterances, which correspond to the number of utterances by 3/8, 5/8, 1, 2, 4, 8, and 16 participants, respectively. One participant made 133 utterances on average over 8 dialogue sessions. We calculated the concept understanding accuracy at

[†]http://crfpp.sourceforge.net/

[††]This participant was selected simply because his ID was #1. We aimed to select a small amount of utterances randomly.

[†††]The value of $\theta$ needs to be determined empirically. However, we think it is possible for a system developer to determine it manually. He/she can judge whether the training process converges or not after seeing $\Delta_i(k)$ while training data are collected and their amount increases. Since the value does not seem very critical after seeing the results shown in Fig. 4, we did not conduct experiments for other values of $\theta$.




(a) grammar+FST (b) N-gram+CRF

**Fig. 4** Change in sum of coefficients, $\Delta_i$, when amount of training data increased ("LM+LUM" denotes combination of an LM and an LUM.).

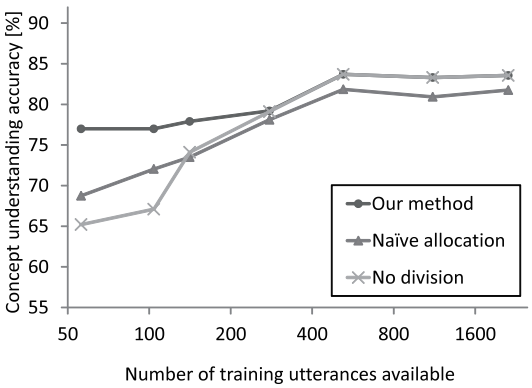



**Fig. 5** Results of allocation methods.

each point for each method.

Figure 5 shows that our method achieved the best concept understanding accuracy when the amount of training data was small, that is, up to 278 utterances. When the amount of training data is small, our method mainly uses the results of "grammar+FST", which needs no training data. As a result, its performance is equivalent or better to that of "grammar+FST" if the selection module works correctly, even when the performances of the other statistical SUs are low. This is why we allocate the training data to the selection module until the first point, $k_{onlysel}$. In the no-division method, the statistical SUs and selection module are trained on the same data. Thus, overfitting occurs when sufficient training data is not available. The overfitting of the selection module especially hurts the overall concept understanding accuracy.

When more utterances were available, the performances of our method and the two baseline methods were almost the same. This indicates that our method for allocating the available training data is effective especially when the amount of training data is small.

Let us examine this result in more detail by using the case in which 141 utterances were used as the training data (participant 1): 111 (= $k_{onlysel}$) were used to train the selection module and 30 were used to train the SU modules. The results are shown in Table 3. We can see that the accuracy with our method was 3.8 points higher than with the no-division baseline method. This was achieved by avoiding overfitting of the logistic regression functions; i.e., the data input to the functions was distributed similarly to that of the test data, so the concept understanding accuracy for the test set was improved. The accuracy with our method was 4.4 points higher than with the naive-allocation baseline method. This was because the amount of training data allocated to the selection module was less than with our method, so the selection module was not trained sufficiently.

**Table 3** Concept understanding accuracy when small amounts of utterances (those by one participant) were available.

| participant ID | Our method | Naive allocation | No division |
|---|---|---|---|
| #1 (141 utterances) | 77.9 | 73.5 | 74.1 |
| #3 (99 utterances) | 77.0 | 67.7 | 69.5 |
| #5 (204 utterances) | 73.5 | 72.4 | 68.2 |
| #7 (151 utterances) | 77.8 | 75.0 | 75.0 |

We also conducted the same experiment for utterances made by three other participants. The results are also shown in Table 3. We can also see that our method outperformed the two baseline methods for their utterances as well. This indicates that the results do not depend on a specific set of utterances.

### 5.4 Comparison with Methods Using a Single ASR and a Single LU

Finally, we compared the concept understanding accuracy of our method with those of eight baseline methods using a single ASR module and a single LU module for various amounts of training data. These baseline methods correspond to conventional speech understanding methods without the MLMU framework. Each module for comparison was constructed by using all available training data at each point as the amount of training data increased because these methods have no selection module. Our method switched the allocation phase at 111 and 207 utterances, as described in Sect. 5.3.

The results are shown in Fig. 6. The accuracies of only three speech understanding modules are shown, out of the eight obtained by combining two LMs for ASR and four LUMs. These three are the ones with the highest accuracies

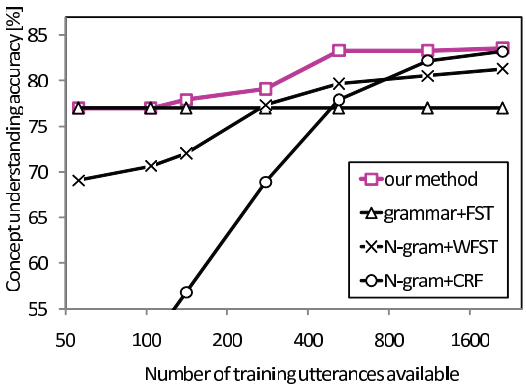



**Fig. 6** Comparison with baseline methods using single ASR module and single LU module.

1. Divide training data into $m$ data sets.
2. Use one of $m$ sets as development data.
3. Use $j$ sets for training LUMs and $k$ sets for training selection module ($j + k + 1 = m$).
4. Perform speech understanding using trained LUMs and selection module.
5. Change development sets and calculate concept understanding accuracies for each case by repeating steps 2 to 4 $m$ times. Average accuracy is the result for the current $j$ and $k$.
6. Repeat steps 2 to 5 with $k = 1, \ldots, m-2$ and find $j$ and $k$ with which the highest concept understanding accuracy is obtained.
7. Allocate available training data: $j/(j + k)$ to selection module and $k/(j + k)$ to LUMs.

**Fig. 7** Procedure for allocating training data with cross validation-like method.

at least at one point, when the amount of training data was increasing.

Our method performed equivalently or better than all the baseline methods at every point while the number of training utterances increased. This is because the speech understanding results of multiple modules are complementary, and our method selects the more reliable results. This is due to allocating training data to the selection module when the amount available is small.

### 5.5 Comparison with Cross Validation-Like Method

We also tested a cross validation (CV)-like method to allocate available training data when the amount of training data is small. CV is a commonly used technique to avoid overfitting of statistical methods in general when the amount of training data is limited [23]. In this method, the optimal allocation ratio is determined experimentally for the available training data. The training data are then allocated to the selection module and SU modules on the basis of the ratio. The procedure is shown in Fig. 7. We used 141 utterances by participant ID #1 consisting of 8 dialogues, which is the same data as the latter half of Sect. 5.3. They were divided into 8 sets per dialogue; that is, $m = 8$ at Step 1 in Fig. 7. Next, $j = 4$ and $k = 3$ were found to give the highest concept understanding accuracy for the training data: 77.0%. The available data were allocated accordingly and used to train the modules.

The calculated concept understanding accuracy for the test set was 75.8%, 2.1 points lower than with our method when the same training data were used, as shown in Table 3. The reason the CV-like method underperformed our method was that the allocation ratio was overfitted for the training data set. This means that the performances of statistical methods are not stable even with CV when the amount of training data is small. Our use of grammar-based LM and LUM is effective for such cases.

## 6. Conclusion

We have developed a method for automatically allocating training data to the statistical modules in our "Multiple Language models for ASR and Multiple language Understanding models (MLMU)" framework [2]. The method prevents performance degradation caused by overfitting. Experimental evaluation showed that concept understanding accuracies achieved with our method were equivalent or better than those of baseline methods based on all combinations of a single automatic speech recognition (ASR) module and a single language understanding (LU) module at every point as the amount of training data increased. This includes a case in which only a small amount of training data was available, meaning that our method is also effective for rapid prototyping.

The two major contributions of this work are summarized as follows:

1. We showed that ASR and LU modules based on multiple language models (LMs) and language understanding models (LUMs) work complementarily, especially when a small amount of training data is available. This is a key advantage of our MLMU framework, which uses both statistical and grammar-based methods.
2. We alleviated the problem of overfitting when the amount of training data is small by setting a data allocation policy that considers the data amount and module characteristics.

When plenty of training data are available, there is little difference between our method and the speech understanding method that requires the most training data, i.e., one based on an N-gram LM and a CRF-based LUM, in the current implementation. It is possible that our method combining multiple speech understanding modules would outperform it, as Schapire et al. [19] reported. There are some examples in their data that only hand-crafted rules can parse. Our method needs to be evaluated for other tasks, including ones requiring more sophisticated structural analysis. The parameters we set to determine the convergence of the statistical methods need to be verified for other tasks, too. This remains as future work.

We have focused on speech understanding, especially on how to construct language understanding modules, and used limited amounts of domain-dependent data to train

them. This is because language understanding is generally domain-dependent. A well-known technique for language models for automatic speech recognition is to mix a large amount of domain-independent data and a small amount of domain-dependent data. How to combine domain-independent resources with our MLMU framework is an important issue that also must be addressed.

## Acknowledgments

This work was partially supported by KAKENHI.

### References

[1] K. Komatani, M. Katsumaru, M. Nakano, K. Funakoshi, T. Ogata, and H.G. Okuno, "Automatic allocation of training data for rapid prototyping of speech understanding based on multiple model combination," Coling 2010: Posters, Beijing, China, pp.579–587, Aug. 2010.

[2] M. Katsumaru, M. Nakano, K. Komatani, K. Funakoshi, T. Ogata, and H.G. Okuno, "Improving speech understanding accuracy with limited training data using multiple language models and multiple understanding models," Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), pp.2735–2738, 2009.

[3] K. Zechner and A. Waibel, "Using chunk based partial parsing of spontaneous speech in unrestricted domains for reducing word error rate in speech recognition," Proc. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, vol.2, pp.1453–1459, 1998.

[4] G. Riccardi, R. Pieraccini, and E. Bocchieri, "Stochastic automata for language modeling," Comput. Speech Lang., vol.10, no.4, pp.265–293, 1996.

[5] C. Raymond, F. Béchet, N. Camelin, R.D. Mori, and G. Damnati, "Sequential decision strategies for machine interpretation of speech," IEEE Trans. Speech Audio Process., vol.15, no.1, pp.162–171, 2007.

[6] S. Hahn, M. Dinarelli, C. Raymond, F. Lefévre, P. Lehnen, R.D. Mori, A. Moschitti, H. Ney, and G. Riccardi, "Comparing stochastic approaches to spoken language understanding in multiple languages," IEEE Trans. Speech Audio Process., vol.19, no.6, pp.1569–1583, 2011.

[7] C. Raymond, F. Béchet, R.D. Mori, and G. Damnati, "On the use of finite state transducers for semantic interpretation," Speech Commun., vol.48, no.3-4, pp.288–304, 2006.

[8] Y. Fukubayashi, K. Komatani, M. Nakano, K. Funakoshi, H. Tsujino, T. Ogata, and H.G. Okuno, "Rapid prototyping of robust language understanding modules for spoken dialogue systems," Proc. International Joint Conference on Natural Language Processing (IJCNLP), pp.210–216, 2008.

[9] A. Potamianos and H.K.J. Kuo, "Statistical recursive finite state machine parsing for speech understanding," Proc. Int'l Conf. Spoken Language Processing (ICSLP), pp.510–513, 2000.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten, "The WEKA data mining software: An update," SIGKDD Explor. Newsl., vol.11, pp.10–18, Nov. 2009.

[11] K. Komatani, Y. Fukubayashi, T. Ogata, and H.G. Okuno, "Introducing utterance verification in spoken dialogue system to improve dynamic help generation for novice users," Proc. 8th SIGdial Workshop on Discourse and Dialogue, pp.202–205, 2007.

[12] K. Komatani and T. Kawahara, "Flexible mixed-initiative dialogue management using concept-level confidence measures of speech recognizer output," Proc. Int'l Conf. Computational Linguistics (COLING), pp.467–473, 2000.

[13] Y.Y. Wang and A. Acero, "Discriminative models for spoken language understanding," Proc. Int'l Conf. Spoken Language Processing (INTERSPEECH), pp.2426–2429, 2006.

[14] M. Jeong and G.G. Lee, "Exploiting non-local features for spoken language understanding," Proc. COLING/ACL 2006 Main Conference Poster Sessions, pp.412–419, 2006.

[15] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), pp.1605–1608, 2007.

[16] S. Hahn, P. Lehnen, and H. Ney, "System combination for spoken language understanding," Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), pp.236–239, 2008.

[17] M. Dinarelli, A. Moschitti, and G. Riccardi, "Re-ranking models for spoken language understanding," Proc. European Chapter of the Association for Computational Linguistics (EACL), pp.202–210, 2009.

[18] Y.Y. Wang, A. Acero, C. Chelba, B. Frey, and L. Wong, "Combination of statistical and rule-based approaches for spoken language understanding," Proc. Int'l Conf. Spoken Language Processing (ICSLP), pp.609–612, 2002.

[19] R.E. Schapire, M. Rochery, M. Rahim, and N. Gupta, "Boosting with prior knowledge for call classification," IEEE Trans. Speech Audio Process., vol.13, no.2, pp.174–181, 2005.

[20] M. Nakano, Y. Nagano, K. Funakoshi, T. Ito, K. Araki, Y. Hasegawa, and H. Tsujino, "Analysis of user reactions to turn-taking failures in spoken dialogue systems," Proc. 8th SIGdial Workshop on Discourse and Dialogue, pp.120–123, 2007.

[21] T. Kawahara, A. Lee, K. Takeda, K. Itou, and K. Shikano, "Recent progress of open-source LVCSR engine Julius and Japanese model repository," Proc. Int'l Conf. Spoken Language Processing (ICSLP), pp.3069–3072, 2004.

[22] L. Hetherington, "The MIT finite-state transducer toolkit for speech and language processing," Proc. Int'l Conf. Spoken Language Processing (ICSLP), pp.2609–2612, 2004.

[23] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, Springer New York, 2001.

**Kazunori Komatani** received B.E., M.S., and Ph.D. degrees in Informatics in 1998, 2000, and 2002 from Kyoto University, Japan. From 2002 to 2010, he was an assistant professor in the Graduate School of Informatics, Kyoto University. He is currently an associate professor in the Graduate School of Engineering, Nagoya University. From 2008 to 2009, he was a visiting scientist at Carnegie Mellon University, Pittsburgh, PA, USA. He has received several awards including the 2002 FIT Young Researcher Award and the 2004 IPSJ Yamashita SIG Research Award, both from the Information Processing Society of Japan (IPSJ). He is a member of the IPSJ, NLP, JSAI, ACL, and ISCA.

**Mikio Nakano** is a principal researcher at Honda Research Institute Japan Co., Ltd. (HRI-JP). He received his M.S. degree in Coordinated Sciences and Sc.D. degree in Information Science from the University of Tokyo, respectively in 1990 and 1998. From 1990 to 2004, he worked for Nippon Telegraph and Telephone Corporation. He was a visiting scientist at MIT Laboratory for Computer Science from 2000 to 2002. In 2004, he joined HRI-JP. His research interests include spoken dialogue systems, speech understanding, and conversational robots. He is a member of ACM, ACL, IEEE, ISCA, JSAI, IPSJ, RSJ, and ANLP.

**Masaki Katsumaru** received the B.E. and M.S. degrees in Informatics in 2008 and 2010 from Kyoto University. He currently works at Panasonic Corporation.

**Kotaro Funakoshi** is with Honda Research Institute Co., Ltd. since 2006. He received the B.S. degree in 2000 from Tokyo Institute of Technology, the M.S. and the Dr. Eng. degrees from Tokyo Institute of Technology in 2002 and 2005, respectively. His research interests are natural language understanding/generation, spoken dialogue systems and conversational robots. He is a member of ACM SIGCHI, IPSJ, JSAI, and ANLP.

**Tetsuya Ogata** received the B.S., M.S. and Dr.Eng. degrees in Mechanical Engineering in 1993, 1995, and 2000, respectively, from Waseda University. From 1999 to 2001, he was a research associate at Waseda University. From 2001 to 2003, he was a research scientist in the Brain Science Institute, RIKEN. Since 2003, he has been a faculty member in the Graduate School of Informatics, Kyoto University, where he is currently an associate professor. He received the 2000 JSME Outstanding Paper Medal from the Japan Society of Mechanical Engineers and the Best Paper Award of IEA/AIE-2005. He is a member of the IPSJ, JSAI, RSJ, HIS, SICE, and IEEE.

**Hiroshi G. Okuno** received the B.A. and Ph.D. degrees from the University of Tokyo, Japan, in 1972 and 1996, respectively. He is currently a professor of the Graduate School of Informatics, Kyoto University, Japan. He received various awards including the Best Paper Awards of JSAI. His research interests include computational auditory scene analysis, robot audition and music scene analysis.