Driver prediction to improve interaction with in-vehicle HMI

Bret Harsham[1], **Shinji Watanabe**[1], **Alan Esenther**[2,5], **John Hershey**[1], **Jonathan Le Roux**[1], **Yi Luan**[3,5], **Daniel Nikovski**[1], **Vamsi Potluru**[4,5]

[1]: MERL, USA
E-mail: {harsham,watanabe,hershey,leroux,nikovski}@merl.com
[2]: The MathWorks, USA
E-mail: alan.esenther@mathworks.com
[3]: University of Washington, USA
E-mail: luanyi@u.washington.edu
[4]: Comcast Labs, USA
E-mail: vamsi_potluru@cable.comcast.com
[5]: *Alan Esenther, Yi Luan, and Vamsi Potluru contributed to this work while at MERL.*

Abstract Recently there has been a trend toward increasing the capability of the in-vehicle interface in terms of access to information and complex controls. This has been accompanied by an increase in the complexity of the car Human Machine Interface [HMI]. At the same time, studies have shown that driver distraction can contribute to accidents. This paper provides some possible ways to reduce driver cognitive load by augmenting the interface. We use prediction of the driver's next action or intention in order to provide UI affordances for more quickly selecting actions. Two examples of this are presented: prediction of driver interaction with the car HMI based on the driving history, and prediction of driver intention from the driver speech. In the first example, we used signal processing techniques to extract meaningful features from vehicle CAN and history data, and then we used machine learning techniques to predict the driver's next action. In the second example, we used ASR and natural language processing to extract text features from driver speech, and predict user intention using a neural network and word embedding. The proposed prediction methods for user actions and intentions can be used to improve in-vehicle task performance.

Keywords Car HMI, Prediction of driver interaction, Prediction of driver intention, Machine learning, Spoken language understanding

1. INTRODUCTION

It has been shown in numerous studies that driver distraction is linked with higher accident risk [1]. Driver distraction during performance of a secondary task is a problem that occurs both with handheld devices [2] and built-in car HMI devices [3]. While it can be difficult to predict which cognitive resources will be most likely to be overloaded by a particular task, the National Highway Traffic Safety Administration [NHTSA] has published guidelines for limiting driver distraction during interaction with a car HMI [4]. These include limiting task completion time, and limiting the amount of visual information presented to a driver during task performance. One way of achieving both of these improvements is to reduce the number of actions that a driver needs to perform and/or the number of choices offered to a driver (either visually or aurally) while performing a task. This paper looks at two separate methods for achieving this, both based on using machine learning techniques to make predictions about what the driver will do next. Based on the predictions, the user interface can be modified so that the average number of actions needed to perform a task can be reduced. This will in general reduce the task performance time.

Section 2 introduces a method for prediction of driver interaction with the car HMI based on the driving history. Modern vehicles are equipped with many different kinds of sensors which are capable of providing information about the state of the vehicle, the road network, and the occupants of the vehicle. Traffic patterns and human behavioral patterns are strongly dependent on time and circumstances. We developed a feature set based on the driving history, and unsupervised method of feature selection by ordering the feature set based on which features were most informative.

Section 3 focuses on spoken language understanding (SLU) for in-vehicle dialog systems, which predict user intentions from the output of an automatic speech recognizer (ASR) [5, 6]. In this paper we focus on two key tasks in targeted dialog and understanding applications: user intention understanding and user goal estimation. User intention understanding is the extraction of the intended meaning (called "intention" hereafter) of one user utterance, performed by the SLU module. User goal estimation is a similar concept, but is an estimation of the final system action (called "goal" hereafter) that the user wishes to accomplish during the dialog. A dialog usually consists of a series of user utterances and system actions, so the goal estimation takes place over a longer time scale than user intention understanding. Intention understanding and goal estimation are performed by employing a (recurrent) neural network with word embedding based feature augmentation and/or network pre-training [7, 8, 9, 10].

2. PREDICTION OF DRIVER INTERACTION WITH THE CAR HMI BASED ON THE DRIVING HISTORY

2.1. Data Collection

The data set used for this work was collected in Japan in several vehicles equipped with a logging system which captured driving state data from the CAN bus, and HMI event and state data from a modified car navigation system. CAN bus data is high frequency, but low level, information about the state of the car including vehicle speed, position of brake and accelerator pedal, window, shift lever, lighting and gasoline states, etc. The car nav system reported information about the current destination (such as distance, location, genre and category), map data about the road network (distance from home, distance from destination, road type, data about points of interest and roads near the vehicle, etc.), and information about the HMI state and user actions. Each of the instrumented vehicles was driven by one user at a time for a period ranging from a few days to a month, but typically approximately two weeks. Users were asked to use the car to perform scripted scenarios, such as going to a particular kind of store during a set time of the day. The scripted tasks were designed to include regular trips such as going to/from work, and also more spontaneous trips such as shopping or going to a restaurant. This is a scripted data collection, but because the users interacted with the car HMI while driving, it also contains naturalistic user actions, e.g. re-routing or changing the map view in response to driving conditions. The full data set consists of data from 26 users across 186 driving days. Our data processing and analysis treats all actions the same - we made no attempt to separate scripted actions and naturalistic actions.

2.2. Feature Processing

Many machine learning algorithms work best with indicator variables that indicate the presence or absence of a particular condition, for example, “it is rush hour”. However, using hard range boundaries to determine indicator values is problematic with many of the kinds of data variables present in the data set, and with driving data in general, in the sense that users tend to take similar actions during a time range rather than always at the same time. For instance, even if morning rush hour is typically 7 am - 9 am in a particular area, if it is 9:30 in the morning, there may still be rush hour traffic, and the user may still be doing his or her typical morning actions.

Because of this, we use sampling functions to generate a set of probabilistic (“soft”) indicator variables from a single data variable over a continuous range: we overlay a series of N evenly spaced Gaussian random variables G_1 - G_N on the range of the input variable. Given a particular input value x, we use the value of the probability density function $f_{G_i}(x)$ of each Gaussian at x as a sampling function, with the values normalized so that they sum to 1. This gives us a group of N soft bin indicator values I_1 - I_N representing the input value x.

$$I_m = \frac{f_{G_m}(x)}{\sum_{i \in \{1..N\}} f_{G_i}(x)} \quad (1)$$

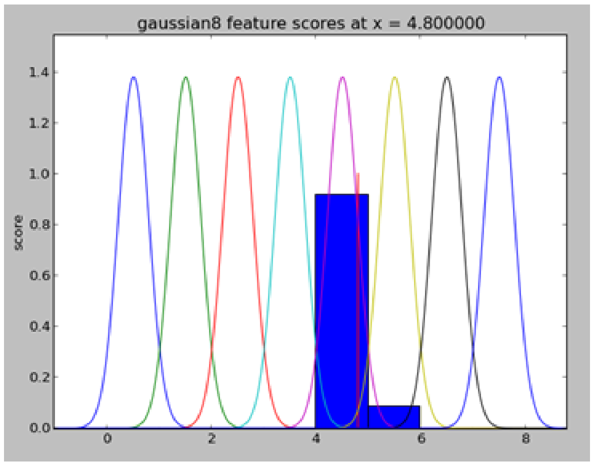

Fig. 1. *Soft bin Indicators, evenly spaced*

Figure 1 shows an example group of 8 soft indicator bins. The indicator variables change smoothly as the input value x varies over its range. This type of indicator is suitable for many kinds of data, including time data. For other kinds of data, such as the vehicle speed, it can be more suitable to use logarithmically spaced bins, as shown in Figure 2.

2.3. Prediction Models and Feature Selection

We used the Scikit-learn [11] machine learning Python framework to build several kinds of prediction models based on our features. In order to decide which features to include, we implemented an automatic method of feature selection based on repeatedly evaluating prediction performance with different sets of features groups. In general, the feature selection method is as follows:

By repeatedly removing the feature group that reduces performance the least, we first remove features that are not actually informative, thus reducing overtraining, and improving performance. As we continue to remove features, we remove the least helpful features first, generating an ordered list of how informative the features are. Figure 3 shows an example of prediction accuracy during feature selection. In this

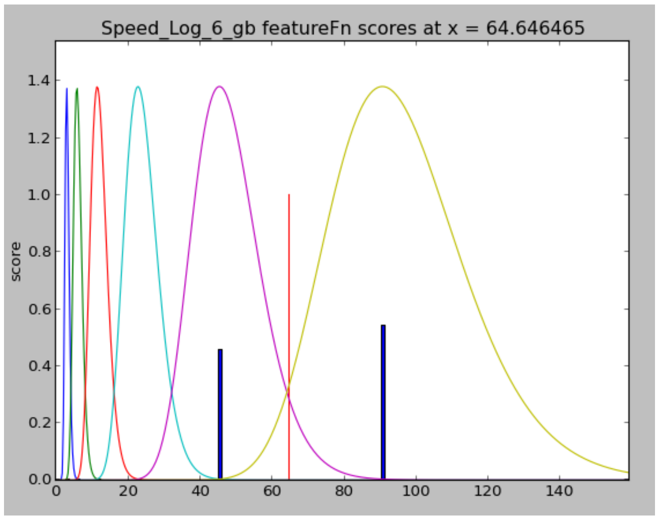

Fig. 2. *Soft bin Indicators, log spacing*

Algorithm 1 Feature Selection for a set S of n feature groups

```
procedure SELECTFEATURES($S$)
    $L \leftarrow []$
    $best\_acc \leftarrow 0$
    $least\_useful \leftarrow None$
    $R \leftarrow S$        ▷ $R$ is the set of remaining feature groups
    while $\|R\| \geq 0$ do
        for g in R do        ▷ $g$ is a group of related features
            $R' \leftarrow R - g$
            $m \leftarrow train\_model(R')$
            $acc \leftarrow eval\_model(m)$
            if $acc \geq best\_acc$ then $least\_useful \leftarrow g$
            end if
        end for
        $L \leftarrow L.append(least\_useful)$
        $R \leftarrow R - least\_useful$
    end while
    return $L$        ▷ $L$ is the list of feature groups
end procedure
```

case, the model was trained to predict the general type of action that the driver would perform next. We started with 40 feature groups. Performance improves until there are only a few feature groups remaining. In this case the remaining feature groups were, in decreasing order of information content: Whether the car had arrived at the destination, The elapsed time from the start of the trip to setting the destination, The current distance to the destination, The total distance of traffic jams on the current route, Whether the current day was a weekday, The motion status of the car (driving, paused, or stopped), and The user's current preference for the type of route to be selected by the navigation system. Our proposed method of feature selection improved prediction performance from 93.0% to 94.1% when predicting the general type of the next action performed. Although this is a preliminary result based on scripted data, the prediction accuracy is promising and the most relevant features after feature selection are in general intuitively related to driver behavior.

2.4. Interface Augmentation using predictions

Several different kinds of dynamic modifications to the car HMI are possible based on prediction of what the user's next actions will be. Active modifications include removing or reordering choices in the standard HMI menus (graphic or dialog state). We have some concerns about this kind of modification, as this can result in a mismatch between the user's mental model of the system and the actual system behavior, which can lead to user confusion or stress. It is also possible to have a more passive interface affordance which is a dynamic overlay of suggested actions in addition to the fixed set of HMI menus. This preserves the match between the user's mental model and the system, while still providing a set of shortcuts that the user can choose when they correspond to a desired action. Anytime one of the shortcuts is correct, the number of actions needed to complete the task will be reduced, thus reducing task completion time.

3. PREDICTION OF DRIVER INTENTION FROM THE DRIVER SPEECH

This section describes the prediction of driver intention from the driver speech based on a neural network classifier. Let $g \in \mathcal{G}$ be a classification category with a classification set $\mathcal{G}$, which can be either goal or intention. The prediction is formulated as the following multiple output classification problem:

$$\hat{g} = \arg\max_{g \in \mathcal{G}} p(g|\mathcal{X}). \tag{2}$$

$p(g|\mathcal{X})$ is a posterior distribution given word sequences (ASR outputs) $\mathcal{X} = \{X(1), X(2), \dots, X(i), \dots, X(T)\}$ in a speech utterance/dialog, where $X(i) \in \{0,1\}^{|\mathcal{W}|}$ is a one-hot word vector at word position i. T is the number of words in a sequence. The number of dimensions $|\mathcal{W}|$ corresponds to the vocabulary size (the number of distinct words) in vocabulary $\mathcal{W}$. One of the issues for this classification is that $\mathcal{X}$ is a variable-length vector sequence in an utterance or a dialog, and we need to deal with this property in the classification framework. Conventional intention and goal estimation approaches use the bag of word (BOW) feature, $X_{BOW} \in \mathbb{R}^{|\mathcal{W}|}$ (or bag of intention features in goal estimation) as fixed-dimension vector inputs, i.e.,

$$\hat{g} = \arg\max_{g \in \mathcal{G}} p(g|X_{BOW}). \tag{3}$$

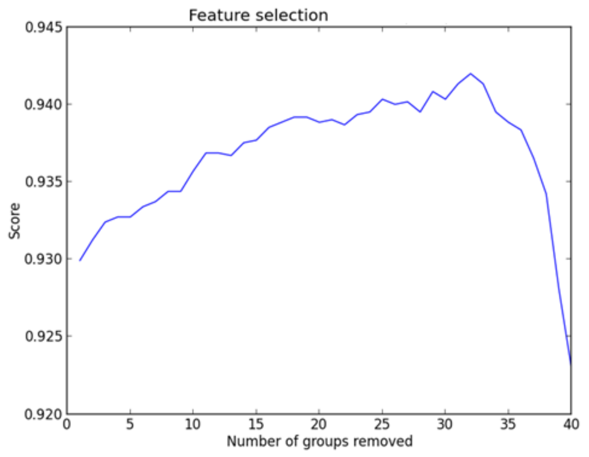

Fig. 3. *Feature selection - accuracy as feature groups are removed*

We use multivariate logistic regression to compute $P(g|X_{BOW})$ for classification target g and feature vector X_{BOW} as

$$P(g|X_{BOW}) = \mathrm{softmax}([W_{BOW}X_{BOW}]_g), \qquad (4)$$

where $[X]_g$ means a g-th raw element of vector X. $W_{BOW} \in \mathbb{R}^{|\mathcal{G}|\times|\mathcal{W}|}$ is a weight matrix to be estimated at the training step. The softmax function $\mathrm{softmax}(\cdot)$ is defined as follows

$$\mathrm{softmax}(z_m) \triangleq \frac{e^{z_m}}{\sum_n e^{z_n}} \qquad (5)$$

Thus, once we obtain the fixed-length vector representation (X_{BOW} in the BOW feature case), the method provides a posterior distribution of the classification category, which can predict intention or goal.

However, one of the problems of applying BOW features to SLU tasks is that the feature vector tends to be very sparse. Each utterance only includes a dozen words at most unlike document analysis. Therefore, the feature vector sometimes lacks enough semantic information to accurately estimate user intentions or goals. This section first investigates the use of additional semantic information by using word embedding [12] and latent topic model based on Latent Dirichlet Allocation (LDA) [13] techniques for intention understanding and goal estimation.

3.1. Incorporating Semantic information

In this section, we import semantic text embeddings to our SLU tasks. Two kinds of semantic features are trained in unsupervised fashion using large-scale web data. One is a word-level embedding which learns contextual information about word sequences by a feed-forward neural network. The other is document-level topic embedding, which models the latent topic information of a given sentence or article. The two kinds of semantic features have respective advantages: word-level embedding captures local contextual information, while topic embedding capture the statistics of the corpus, and thus provides more global information. Then, we extracted an augmented feature $X_{LDA} \in \mathbb{R}^K$ from LDA and $X_w \in \mathbb{R}^J$ from word embedding, where K is the number of latent topics in LDA and J is the number of dimensions in an embedded space. Note that $|\mathcal{W}| > K$ or J. Thus, the classification problem in Eq. (3) can be extended as:

$$\hat{g} = \arg\max_{g\in\mathcal{G}} p(g|[X_{BOW}^\top, X_{LDA}^\top, X_w^\top]^\top). \qquad (6)$$

Augmented vectors X_{LDA} and X_w provide the dense features in addition to the BOW based sparse features, which contributes to improve the classification performance.

3.1.1. Word embedding

Many current NLP systems use a bag-of-words or one-hot word vector as an input, which leads to feature vectors of extremely large dimension. An alternative is a word embedding, which projects the large sparse ($|\mathcal{W}|$) word feature vector into a low-dimensional ($J < |\mathcal{W}|$), dense vector representation.

There are two main model families for learning word vectors: 1) matrix factorization methods, such as latent semantic analysis (LSA) [14] and 2) neural network language model (NNLM) based methods, which model on local context window, such as Continuous Bag of Words (CBOW), Skip-gram [12]. The Skip-gram predicts surrounding words given the current word $X(i)$, as follows:

$$p(X(i-L), \ldots X(i-1), X(i+1), \ldots, X(i+L)|X(i)). \quad (7)$$

$X(i-L), \ldots X(i-1), X(i+1), \ldots, X(i+L)$ are $2L$ surrounding word contexts, which will be predicted in the skip gram framework. This can be estimated by one-hidden layer neural network, and the first liner transformation matrix $\phi \in K \times |\mathcal{W}|$ corresponds to embed high-dimensional ($|\mathcal{W}|$) one-hot vector to the low dimensional (J) dense vector, i.e.,

$$X_w(i) = \phi X(i). \qquad (8)$$

Then for each turn or sentence, X_w is obtained by summing over normalized embedded word vectors for each word in the turn or sentence:

$$\bar{X}_w = \sum_{i\in\{1..T\}} \frac{X_w(i)}{||X_w(i)||} \qquad (9)$$

$\bar{X}_w$ is used in Eq. (6) as an augmented feature vector, instead of unnormalized feature X_w.

Mikolov's toolkit 'word2vec' which implement Skip-gram and CBOW can train on large-scale corpora very efficiently. Therefore, in this paper, we used word2vec to train

Skip-gram and CBOW on a large scale web corpus collected from the internet and chose the embedding that gave the best SLU performance. The typical word usage in our route guidance task differs from web texts, so we expected that fine-tuning of the word embedding would yield additional performance improvements by adapting the word embedding model to our target task.

3.1.2. Latent topic models

Latent topic models are algorithms that can discover semantic information from a collection of documents. Topic embedding, widely used in information retrieval, treats a document as a mixture of topics and uses a vector to represent the topic distribution. Classic latent topic models have been used for SLU including Probabilistic Latent Semantic Analysis (PLSA) [15] and LDA [13]. Most latent variable models are generative models, which therefore can be used in unsupervised fashion. The latent topic model provides the posterior probability $p(k|X_{BOW})$ for each topic given (BOW) feature X_{BOW}, which is used as the kth element of an augmented LDA feature vector X_{LDA}, i.e.,

$$[X_{LDA}]_k = p(k|X_{BOW}), \tag{10}$$

The augmented feature X_{LDA} is also used in Eq. (6). However, since LDA embedding is obtained an iterative inference algorithm (variational EM) or sampling method, it is hard to fine-tune an LDA embedding within a neural network framework. Therefore, in this paper, we do not fine-tune LDA features. It could be possible to use a deep unfolding method to fine-tune the LDA inference, such as that presented in [16].

3.2. Fine-tuning of linear input networks

The previous section applies the semantic information as input features. This section uses them as an additional input layer in neural network architectures, by following the great success of neural network approaches in various SLU tasks with pre-training [17]. In applying these techniques to SLU, one major difficulty is that we often have insufficient training data for a task, since the annotation of collected data can be expensive. The performance of a neural network trained in low resource conditions is usually inferior because of overtraining. Our approach uses word embedding as an additional input layer of a neural network, which mitigates the overtraining problem. To accomplish this, we initialize the affine transformation of the first layer by using a word embedding matrix estimated from a large-scale general corpus with unsupervised training methods (pre-training). Then, the entire SLU network is trained with the annotated training data, with word embeddings fine-tuned to the SLU task. This concept has been studied in [8, 18], and the following section also describes the pre-training idea by employing word embedding framework.

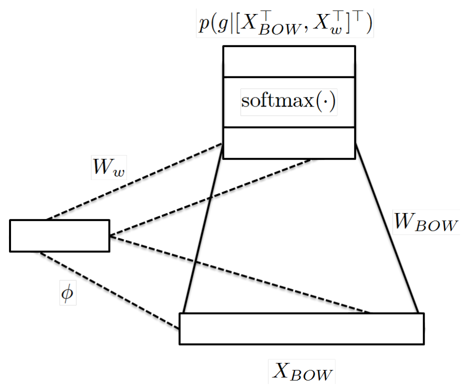

Fig. 4. *Two-layer feed-forward architecture of goal/intention prediction network with pre-training based on ϕ.*

3.2.1. Feed-forward architecture

We start from Eq. (6) without the LDA feature X_{LDA}. By using the logistic regression equation (Eq. (4)), the posterior distribution $p(g|[X_{BOW}^\top, X_w^\top]^\top)$ can be rewritten as the two matrix multiplication for input features X_{BOW} and X_w as follows:

$$\begin{aligned} &p(g|[X_{BOW}^\top, X_w^\top]^\top) \\ &\quad = \text{softmax}([W_{BOW}X_{BOW} + W_w X_w]_g), \end{aligned} \tag{11}$$

where $W_w \in \mathbb{R}^{|\mathcal{G}| \times J}$ is a weight matrix. Let X_w be obtained as an unnormalized vector of Eq. (9), i.e., $X_w = \sum_{i \in \{1..T\}} X_w(i)$, Eq. (11) is rewritten with Eq. (8), as follows:

$$\begin{aligned} &p(g|[X_{BOW}^\top, X_w^\top]^\top) \\ &\quad = \text{softmax}([W_{BOW}X_{BOW} + W_w \phi X_{BOW}]_g) \\ &\quad = \text{softmax}([(W_{BOW} + W_w \phi)\, X_{BOW}]_g) \end{aligned} \tag{12}$$

Thus, we can regard Eq. (12) as a two-layer neural network given input feature X_{BOW}. That is, the feed-forward architecture changes the baseline structure by adding a linear hidden layer between the Bag-of-Words input layer and the output layer (as Figure 4). The posterior probability of the goal/intention given the input features is calculated through the final softmax layer. Although the proposed architecture has three weight matrices ϕ, W_{BOW}, and W_w, ϕ can be initialized from the word embedding matrix (pre-training), and we can efficiently estimate these matrices. Note that the goal estimation task uses several utterances as an input feature, and our experiments use utterance-wise feature X_I. We use the N-best intention confidence score obtained by the other intention estimation module than that described in this paper as X_I.

3.2.2. *Multi-scale recurrent neural network (MSRNN)*

The limitation of the above methods is that we do not explicitly deal with a sequential property. This can be performed by using RNN. By considering the dialog structure based on multiple scale property based on word and utterance unit, we also propose a novel Multi-Scale RNN (MSRNN) architecture pre-trained with word embedding in order to capture long-term characteristics over the entire dialog for goal estimation. The proposed MSRNN uses two sub-networks to model the different time scales represented by word and turn sequences.

The goal estimation task has two input sequences for each sample: a word sequence $\mathcal{X} = \{X(1), \dots, X(i), \dots, X(T)\}$ and an utterance sequence $\mathcal{I} = \{I(1), \dots, I(j), \dots, I(M)\}$. The two sequences have different time scales, and $T > M$. However, the baseline architecture treats word input as bag-of-words, which ignores the contextual information of the input. Both input sequences, word and intention, contain contextual information, and intuitively a system that captures this information may perform better than one which does not. Therefore, the proposed MSRNN architecture can model the different time scales represented by word and intention sequences, shown in Figure 5.

The lower half of this figure represents the short time scale RNN, which accepts feature vectors for words $X(i)$ in all history utterances, as an entire sequence. The recurrent connection is represented as follows:

$$h_x(i) = \text{sigmoid}(\phi_x X(i) + W_x h_w(i-1)), \qquad (13)$$

where we use the element-wise sigmoid function at the hidden layer defined as:

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}. \qquad (14)$$

W_x and ϕ_x are weight matrices of recurrent connection and word embedding, respectively. ϕ_x is initialized by a word embedding matrix, similar to the discussion in the previous section, and the back propagation through time is used to fine-tune ϕ_x.

The upper half of the figure represents the long time scale RNN, which accepts a single intention feature vector $I(j)$ for each utterance, and use all intentions in history as a sequence. The recurrent connection is represented as follows:

$$h_I(j) = \text{sigmoid}(\phi_I I(j) + W_I h_I(j-1)). \qquad (15)$$

W_I and ϕ_I are also weight matrices of recurrent connection and word embedding, respectively. The upper RNN has longer sequence than lower RNN. The two RNN structures have different parameters and are jointly trained. The goal is predicted at the end of each turn, and the last hidden vector of word sequence $h_x(T)$ and last hidden vector of intention sequence $h_I(M)$ are concatenated to predict the output layer

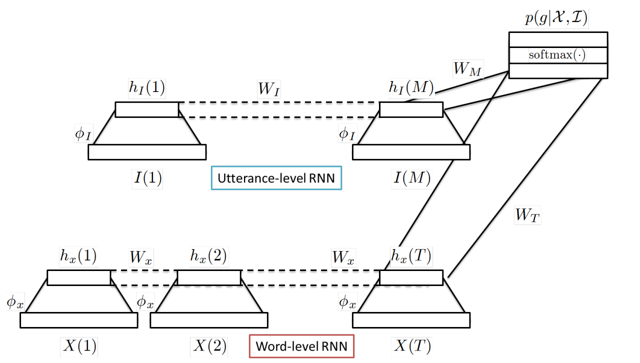

Fig. 5. *Multi-scale recurrent neural network (MSRNN) composed of utterance and word level RNNs*

(goal of the current turn), i.e.,

$$p(g|\mathcal{X},\mathcal{I}) = \text{softmax}(W_T h_x(T) + W_M h_I(M) + b) \qquad (16)$$

where W_T, W_M, and b are weight matrices and bias vector, respectively, which combine the long and short time scale RNNs. ϕ_I, W_x, W_I, W_M, W_T, and b are randomly initialized, unlike ϕ_x.

3.3. Experiments

We built and tested models using data from a Japanese route guidance spoken dialog system. We used a 1.8G Japanese web text corpus from the Internet to train word2vec, and the full Japanese Wikipedia (900k documents) to train LDA. Combination of 50, 100, 150, and 300 word2vec dimensions, and 50, 100 LDA dimensions were tested. The dimensionality with the best result was selected. All text was tokenized and morpholized first by a Japanese morpholizer, ChaSen. For each word, we use the word, pronunciation, and part of speech as individual token elements. Each token is in Word+Pronunciation+POS format. For both Japanese web data and Wikipedia data, low frequency words (frequency $<$ 5) are replaced by UNKNOWN token. The total reduced vocabulary is around 150k.

3.3.1. *Intention understanding*

We evaluated the performance of fine-tuning on the intention understanding task. We collected a database of Japanese utterances in an automobile scenario with annotated intentions, which contains 39,328 training samples, 5,000 dev samples and 5,000 test samples. The number of intention categories is 562, and the vocabulary size is 3,602. Each sentence is represented as a Bag-of-Words feature vector. We tested the performance of word2vec features by concatenating the 300-dimensional word2vec features with the Bag-of-Words features, and additionally tested the system both with and without fine-tuning.

Table 1. Average error rate for intention prediction task, (ft) means with fine-tuning

	Dev (%)	Test (%)
Baseline (Bag-of-Words)	15.7	15.9
BOW + word2vec	13.0	13.5
BOW + word2vec (ft)	12.4	12.3

Table 2. Average error rate for goal estimation task, (ft) means with fine-tuning

	Dev (%)	Test (%)
Baseline (intention only)	18.9	19.5
intention (100 dim) + word2vec (50 dim)	17.2	17.5
intention (100 dim) + word2vec (50 dim + ft)	16.0	16.1
intention (100 dim) + word2vec (50 dim + ft) + LDA (50)	16.0	16.2

The experimental results are shown in Table 1. By using the word2vec features, the performance was improved from the baseline system by 2.7% (dev.) and 2.4% (eval) (absolute reduction in error). With the fine-tuning, the performance further improves to 3.3% (dev.) and 3.6% (eval), absolutely. This result confirms the effectiveness of fine-tuning for our intention understanding task.

3.3.2. Goal estimation

We collected a dataset from a Japanese rule-based spoken dialog system, which contain 7059 turns in total. The log data contains the Japanese ASR results, system prompts, estimated N-best intentions, system actions, and an annotated goal for each turn. The intention vector consists of a sparse vector concatenated by N-best intentions (545 dimensions) and N-best system actions (545 dimensions). The dimension for user intention and system action together is 1090, the dimension for goal is 544 in total. Since the data size was very limited, we split the data into 4 folds, $3/4$ used for training, $1/8$ for validation and $1/8$ for test respectively. To evaluate the performance of our proposed approaches, we used the average accuracy across the 4 folds. We compared the result of using embeddings alone versus embeddings with fine-tuning. In the RNN architecture, hidden layer sizes of 100, 200 and 300 for the intention RNN module were trained and the one with best result was selected (100 for both with and without fine tuning). The model was trained by stochastic gradient descent.

We observe from Table 2 that importing semantic text features gives a better performance than using intention features alone. The best performance occurs on smaller semantic embedding dimensions and smaller hidden layer dimensions. The reason for this is due to the limited data size, for which fine-tuning high dimensional embeddings could lead to overtraining problems.

Fine-tuning word embeddings improves upon the feature engineering results by a small but consistent margin. The first four rows of the table are implemented by the feed-forward structure and the last two rows are implemented by the MSRNN structure. Among all the feed-forward structures, fine-tuning word2vec plus LDA features gives the best result with 2.9% (dev.) and 3.3% (eval.) absolute improvement from the baseline. The MSRNN structure itself (without fine-tuning) already gives a significant performance improvement over simply importing semantic text features. When we add fine-tuning to the MSRNN training, we get the best overall result of 3.3% (dev.) and 3.6% (eval.) absolute improvement from the baseline. This proves that modeling time sequential input by using our MSRNN architecture with different time scales gives a gain to the system.

4. CONCLUSIONS

This paper investigates the use of driver action prediction through 1) the driver interaction with the car HMI based on the driving history and 2) in-vehicle dialog systems based on SLU. The first topic focuses on the processing and selection of data features from the driving history, followed by feature selection to improve prediction performance. The second topic improved the prediction performance of user intention and goal estimation tasks based on a novel recurrent neural network.

In both cases we showed the feasibility of predicting the driver's actions, even from training with a small dataset. Future development of this work will involve using these predictions to improve the car HMI, and evaluating the effect on task completion rates and completion times. Also, we expect that further work on issues of sparse data will be needed: both methods rely on machine learning techniques and currently require large amounts of data to obtain robust performance.

5. REFERENCES

[1] S. G. Klauer, T. A. Dingus, V. L. Neale, J. D. Sudweeks, and D. J. Ramsey, "The impact of driver inattention on

near-crash/crash risk: An analysis using the 100-car naturalistic driving study data," tech. rep., 2006.

[2] D. D. Salvucci, D. Markley, M. Zuber, and D. P. Brumby, "ipod distraction: Effects of portable music-player use on driver performance," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 243–250, ACM, 2007.

[3] L. Garay-Vega, A. Pradhan, G. Weinberg, B. Schmidt-Nielsen, B. Harsham, Y. Shen, G. Divekar, M. Romoser, M. Knodler, and D. Fisher, "Evaluation of different speech and touch interfaces to in-vehicle music retrieval systems," *Accident analysis & prevention*, vol. 42, no. 3, pp. 913–920, 2010.

[4] N. H. T. S. Administration *et al.*, "Visual-manual nhtsa driver distraction guidelines for in-vehicle electronic devices," *Washington, DC: National Highway Traffic Safety Administration (NHTSA), Department of Transportation (DOT)*, 2012.

[5] D. Jurafsky and J. H. Martin, *Speech & Language Processing*. Pearson Education, 2000.

[6] R. De Mori, "Spoken language understanding: a survey.," in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 365–376, 2007.

[7] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 22, no. 4, pp. 778–784, 2014.

[8] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *Proceedings of Interspeech*, 2013.

[9] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random field for language understanding," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4077–4081, IEEE, 2014.

[10] Y. Luan, S. Watanabe, and B. Harsham, "Efficient learning for spoken language understanding tasks with word embedding based pre-training," in *Proceedings of the Interspeech*, 2015.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[14] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *the Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[15] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57, ACM, 1999.

[16] J. R. Hershey, J. Le Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," *arXiv preprint arXiv:1409.2574*, 2014.

[17] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[18] X. Song, X. He, J. Gao, and L. Deng, "Unsupervised learning of word semantic embedding using the deep structured semantic model," Tech. Rep. MSR-TR-2014-109, August 2014.