Lower resources of spoken language understanding from voice to semantics

Hao ZHANG\textsuperscript{1*}, Cheng Guo LV\textsuperscript{2*}

\textsuperscript{1,2}College of computer science and technology, HeiLongJiang University, HaErBing, HeiLongJiang Province, 150000, China

\*Corresponding author’s e-mail: 2181446@s.hlju.edu.cn, 2004085@hlju.edu.cn

Abstract. Spoken language understanding is traditionally designed as a pipeline consisting of multiple components. First, the speech signal is mapped into text through the automatic speech recognition module, and then the natural language understanding module converts the recognized text into structured data, such as domain, intention and slot value. Usually these modules are trained separately. End-to-end speech comprehension, on the other hand, derives structured data directly from speech through a single model. However, end-to-end spoken language understanding based on a large amount of training data is difficult to achieve in different fields and different groups of people. For this reason, we introduced end-to-end oral comprehension based on pre-training with low resources and combined it with capsule vector. The experimental results show that the oral comprehension of this model with low resources is robust under different data sets.

1. Introduction

Spoken language comprehension (SLU) module is a key component of a speech dialogue system (SDS) that parses the user's utterances into the corresponding semantic concepts.[1] For example, "show me a flight from Boston to New York" could be interpreted as (departure city = Boston, arrival city = New York). Most spoken language understanding is based on pipeline structure, including separate automatic speech recognition (ASR), natural language understanding (NLU), and so on. These modules are passed as input to the natural language understanding module, the pipeline view between ASR and NLU has been widely used, and the popular spoken language system Toolkit is assumed to be used as


diagram

Figure 1. Pipeline SLU and end-to-end SLU
the CSLU Toolkit [2] and CMU Olympus Toolkit [3]. The perfect transcription of voice user input is beyond the capabilities of current ASR technologies, especially in conversational systems such as virtual human conversational systems. In many virtual human systems, speech recognition errors can be directly associated with system performance degradation [4].

Recently, deep learning has developed rapidly in various fields, making end-to-end learning possible. However, most of them require a large amount of data for training, which is difficult to obtain for a certain field. Therefore, we introduced a pre-training model by reference [5] and used capsule vector. The performance of spoken language comprehension is greatly improved.

2. Related Work

Under the pipeline structure, speech recognition module is inconsistent with natural language understanding, and speech recognition itself has inevitable errors due to noise interference, speaker speed and accent, etc. The lattice [6] and the word confusion network [7] are used to record the possible sequence of speech recognition output, providing a more informative posterior distribution of word recognition. However, due to the influence of natural language processing, the modeling foundation of most natural language understanding researches is still based on a sentence (word sequence) rather than the more complex n-best hypothesis list, word case, and word confusion network. The following natural language processing is regarded as sequence annotation problem, which is generally solved by combining conditional random field with RNN. The independent optimization of spoken language comprehension has not improved significantly in performance.

Recently, some end-to-end methods have been successfully applied to spoken language comprehension. Firstly [8] explored the intention classification directly from speech signals. It adopts encoder and decoder and uses multilayer bidirectional GRUs to encode and decode feature sequences, but the effect is not better than the independently optimized spoken language comprehension, we also use encode - decode structure. Pretraining model [9] was used to initialize ASR and NLU respectively, and then joint training was conducted. [10] compared four encoder - decoder models, but the training data was huge, up to tens of millions of voices. [11] considering the low resource training, the capsule network was used to learn end-to-end spoken language comprehension. [5] the pre-training model is used to make spoken language comprehension do not need to be trained from scratch and adapt to the new field faster.

This article in view of the migration study in natural language processing to achieve good results and based on the study of the spoken language understanding above, [5] and [11] the main method used in spoken language understanding, in the context of low resources namely voice signal processing in the target language and the training model of capsule network combined with the joint training, the accuracy and response speed has a good performance.

Figure 2. The lower layers of the model are pre-trained and intent module
3. Model
The model proposed in this paper is shown in Figure 4. The overall structure is encoder - decoder. Firstly, the demand for training data is reduced. In this paper, the pre-training model [5] of speech recognition is introduced. The pre-training target is phoneme, and the classifier of the pre-training model is removed, while the audio signal is transformed into a high-level sequence, similar to the word vector in natural language processing, which is similar to Word2Vec [12]. Then the intent module converts the high-level sequence into capsule vector by using the attention mechanism and the distributor, and completes the semantic extraction by using capsule network. Finally, fine-tuning the network layer by layer to complete the experiment.

3.1 Pre-training
Take the audio signal sequence $x$ as input, and then output the high-level sequence $H$.

The pre-training module [5] is implemented using the SincNet [16] layer, which processes the original input waveform, followed by multiple convolution and loop layers with pooling. SincNet is designed only for the first layer of the network, with the intention of learning more meaningful filters. Generally speaking, the ability to extract the first layer of the network is considered to be very important for processing the audio timing signals, because the effectiveness of the low-dimensional features extracted from the first layer is the prerequisite for the higher-level network to learn meaningful high-dimensional feature information, and it has the characteristics of fast convergence and fewer network parameters.

3.2 Attention Mechanism and Distributor
Attention mechanisms used to determine the weight of each time step in an audio sequence, since not all timesteps determine the semantics of speech. Attention mechanism can ignore unimportant parts (e.g., "please", "ah"). For example, attention mechanisms presented in figure 3.

![Figure 3. Attention Mechanisms](image)

The dispenser is used to determine the weight assigned to each capsule for each time step.

The attention mechanism and the dispatcher are used to create the context vector $q_t$ for each capsule vector by calculating the weighted sum of all timesteps:

$$ q_t = \sum_t \alpha_t \delta_{t,i} h_t $$

(1)

$h_t$ contains the high level features for time $t$, $\alpha_t$ is the attention weight for time $t$, $\delta_{t,i}$ contains the distribution weights for timestep $t$.

The calculation method of the weight $a$ of each time step

$$ \alpha_t = \text{sigmoid}(\omega_a \cdot h_t + b_a) $$

(2)
\( \omega_\alpha \) and \( b_\alpha \) are the weights and bias of the sigmoid layer. 

Similar to the attention mechanism, the weight of the allocator is calculated as follows:
\[
\delta_t = \text{softmax}(W^d \cdot h_t + b^d)
\]  

\( W^d \) and \( b^d \) are the weights and biases of the softmax layer.

3.3 Capsule Networks with Attention Mechanism

The activation of neurons in Capsule represents the various properties of a particular entity that exists in the image. These properties can include many different parameters, such as posture (position, size, direction), deformation, speed, reflectivity, color, texture, and so on. The length of the input/output vector represents the probability of the occurrence of an entity, so its value must be between 0 and 1.

\( q_i \) is the context vector for capsule \( i \). The squash layer is a linear transformation followed by a squashing function:
\[
s_i = \sigma(W^s \cdot q_i)
\]  

\( s_i \) is the vector representation for capsule \( i \). \( \sigma(\cdot) \) is the squashing function as defined in [13]:
\[
\sigma(x) = \frac{1}{1 + ||x||^2} x
\]  

The nonlinear function ensures that the length of the short vector can be reduced to almost zero, while the length of the long vector can be reduced to near but not more than 1.

Every hidden capsule will predict the output of every output capsule using a linear transformation:
\[
p_{ij} = W^p_{ij} \cdot s_i
\]  

\( p_{ij} \) is the predicted vector representation of output capsule \( j \) from hidden capsule \( i \). The output capsules are computed using the coupling coefficients \( C \). The coupling coefficients are computed using a \( \text{softmax} \) function on the coupling logits \( B \). The coefficients are iteratively updated and determined by dynamic Routing:

![Figure 4. Pre-training model with capsule network](image-url)
Define variable $B$:

\[
\text{for } n = 1: N do \\
\quad \text{For all hidden capsules } i: c_i = \text{softmax}(b_i^{(n)}); \\
\quad \text{For all output capsules } j: o_j = \sigma(\sum_i c_{ij} p_{ij}); \\
\quad \text{For all logits } b_{ij}^{(n)} \text{ in } B^{(n)}: b_{ij}^{(n)} = b_{ij}^{(n)} + p_{ij} \cdot o_j; \\
\text{end}
\]

Algorithm 1: Dynamic routing algorithm

The probabilities of the output labels $l$ are finally computed using the norm of the output capsules:

\[
l_i = ||o_j|| \tag{7}
\]

The network is trained by minimizing the margin loss:

\[
L = \sum_j t_j \max(0, 0.9 - l_j) + (1 - t_j) \max(0, l_j - 0.1) \tag{8}
\]

where $t_j$ is the target for label $j$, which is either 0 or 1.

3.4 Fine-tuning

ULMFIT \cite{14,15} proposes universal language model fine-tuning, which is a dynamic fine-tuning, gradual thawing, and other new technologies to maintain past knowledge and avoid catastrophic forgetting in fine-tuning. We also fine-tuned the pre-training model in this way and found that thawing the pre-trained network layer gradually worked better than thawing all the network layers immediately.

4. Experiments

4.1 Dataset

In this paper using the fluent \cite{5} open command set training set. Audio and label data sets consist of 16kHz mono. Wav audio files. Each audio file contains a recording of a command that could be used in a smart home or virtual assistant, such as "play music" or "turn up the temperature in the kitchen." Each audio is labeled with three slots: action, object, and location. One of several values is used for a slot: for example, a slot for "location" can be "none," "kitchen," "bedroom," or "bathroom." We target three slot values. The dataset has 31 unique intentions in common. Does not distinguish between domain, intent and slot prediction. The dataset is used as a multi-label classification task, with the goal of predicting actions, objects, and positions. Since slots are not actually independent of each other, a more cautious approach would be slots. In addition, these 31 different intentions can be "flattened" and used as 31 different label classification tasks for a single label. For each intention, there are several possible terms: for example, the intention \{action: "activate", object: "light", position: "none"\} can mean "turn on the light", "turn on the light", "light on", etc. A particular intention may be expressed in various ways. There are 248 different phrases. In this paper, the training set is randomly divided into two parts, and 200 voices are randomly selected from these two parts as sub-data set 1 and sub-data set 2.

Table 1. Information about the fluent speech commands dataset

| Split  | # of speaker | # of utterances | # hours |
|--------|--------------|-----------------|---------|
| Train  | 77           | 23,132          | 14.7    |
| Valid  | 10           | 3,118           | 1.9     |
| Test   | 10           | 3,793           | 2.4     |
| Total  | 97           | 30,043          | 19.0    |

4.2 Methodology

In this paper, the pre-trained SLU model was used for feature extraction. The pre-trained model was trained with the LibriSpeech data set aligned with Montreal Forced Aligner, and the SincNet \cite{15,16} was used to process the speech sequence, which has a good ability of word recognition. Combined with the capsule network, the pre-training model outputs as the capsule network input. Capsule $S$ has 32 hidden vectors, 64 dimensions, and one output capsule. The size of each capsule of the output label
is 8. The model was trained 30 epochs in 16 batches. Adam is used as an optimization method with an initial learning rate of 0.001.

4.3 Analysis
First, we define the accuracy of the model to be that all slot values of the statement are predicted correctly, and if one slot value is incorrect, the whole prediction is incorrect. Then, we conducted the following comparative experiments: 1) capsule network, 2) pre-training and RNN, 3) capsule network and pre-training, and 4) fine-tuning.

| Model                        | Accuracy(1) |
|------------------------------|-------------|
| RNN                          | 70.14%      |
| Capsule Networks             | 88.41%      |
| pre-train+RNN                | 95.48%      |
| pre-train+ Capsule Networks  | 96.64%      |
| pre-train+ Capsule(unfreezing word) | 96.34%   |
| pre-train+ Capsule(unfreezing all layers) | 95.68% |

Performance: As can be seen from table 2, RNN needs a large amount of training data, so it performs the worst under the condition of low resources. The capsule network has a strong advantage in feature extraction under the condition of low resources. After adding pre-training, the improvement is obvious, while the role of fine-tuning is not obvious.

| Model                        | Accuracy(1) | Accuracy(2) |
|------------------------------|-------------|-------------|
| RNN                          | 70.14%      | 65.14%      |
| Capsule Networks             | 88.41%      | 85.75%      |
| pre-train+RNN                | 95.48%      | 96.15%      |
| pre-train+ Capsule Networks  | 96.64%      | 96.45%      |
| pre-train+ Capsule(unfreezing word) | 96.34%   | 96.14%      |
| pre-train+ Capsule(unfreezing all layers) | 95.68% | 93.15%      |

Different subsets: As can be seen from table 3, under sub-data set 2 and the same test set, the performance of the pre-training model is similar to that of table 1, and the difference between RNN and capsule network is larger, which proves that the pre-training model has good generalization performance.

| Model                        | Runtime (1) |
|------------------------------|-------------|
| RNN                          | 1750ms      |
| Capsule Networks             | 1250ms      |
| pre-train+RNN                | 1800ms      |
| pre-train+ Capsule Networks  | 1351ms      |
| pre-train+ Capsule(unfreezing word) | 1310ms | 1295ms      |

Runtime: Table 4 shows the response time of a speech input to the output semantics. RNN has the same weight for each time step, while capsule network uses the attention mechanism to weigh each time step differently. It pays little attention to the words like “please” and “in”, so it has the fastest response time. As the pre-training technique deepens the number of layers of the neural network, the response time is increased slightly.

5. Conclusion
In this paper, we introduce pre-training techniques designed to reduce training data and capsule networks designed to make more efficient use of data. The pre-training technology combines the
results in the universal speech recognition neighborhood with the capsule network, and has more
generalized performance in the low-resource, single and professional neighborhood. However, we also
found that end-to-end SLU failed to perform well among homonyms, which is the next difficulty we
need to overcome.

References
[1] Morbini, F., et al. A reranking approach for recognition and classification of speech input in
conversational dialogue systems. 2012.
[2] Rundle, B., et al. "Universal Speech Tools: The Cslu Toolkit." Proc Icslp 7(1998):3221--3224.
[3] Bohus, Dan, et al. "Olympus: an open-source framework for conversational spoken language
interface research." Workshop on Bridging the Gap: Academic & Industrial Research in
Dialog Technologies Association for Computational Linguistics, 2007.
[4] Wang, William Yang, et al. "Improving Spoken Dialogue Understanding Using Phonetic
Mixture Models." Proceedings of the Twenty-Fourth International Florida Artificial
Intelligence Research Society Conference, May 18-20, 2011, Palm Beach, Florida, USA
DBLP, 2011.
[5] Lugosch, Loren, et al. "Speech Model Pre-training for End-to-End Spoken Language
Understanding." (2019).
[6] Xu, Hainan, et al. "A Pruned Rnnlm Lattice-Rescoring Algorithm for Automatic Speech
Recognition." ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech
and Signal Processing (ICASSP) IEEE, 2018.
[7] Jie, Gao, et al. "Spoken Term Detection Using Dynamic Match Subword Confusion Network."
International Conference on Natural Computation 2008.
[8] Dmitriy Serdyuk *, and Y. Wang . "TOWARDS END-TO-END SPOKEN LANGUAGE
UNDERSTANDING." (2018).
[9] Y. Qian, R. Ubale, V. Ramanarayanan, and P. Lange, “Exploring ASR-free end-to-end modeling
to improve spoken language understanding in a cloud-based dialog system,” ASRU, 2017.
[10] Haghani, Parisa, et al. "From Audio to Semantics: Approaches to end-to-end spoken language
understanding." (2018).
[11] V. Renkens and H. Van hamme, “Capsule Networks for Low Resource Spoken Language
Understanding,” Interspeech, 2018.
[12] Sabour, Sara, N. Frosst, and G. E. Hinton. "Dynamic Routing Between Capsules." (2017).
[13] Bojanowski, Piotr, et al. "Enriching Word Vectors with Subword Information." Transactions of
the Association for Computational Linguistics 5(2017):135-146.
[14] Howard, Jeremy, and S. Ruder. "Universal Language Model Fine-tuning for Text
Classification." (2018).
[15] Shleifer, Sam. "Low Resource Text Classification with ULMFit and Backtranslation." (2019).
[16] Ravanelli, Mirco, and Y. Bengio. "Speaker Recognition from Raw Waveform with SincNet."
(2018).
[17] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, “Montreal Forced
Aligner: Trainable text-speech alignment using Kaldi,” Interspeech, 2017.
[18] Ravanelli, Mirco, and Y. Bengio. "Interpretable Convolutional Filters with SincNet." (2018).