Speech Model Pre-training for End-to-End Spoken Language Understanding

Abstract

Whereas conventional spoken language understanding (SLU) systems map speech to text, and then text to intent, end-to-end SLU systems map speech directly to intent through a single trainable model. Achieving high accuracy with these end-to-end models without a large amount of training data is difficult. We propose a method to reduce the data requirements of end-to-end SLU in which the model is first pre-trained to predict words and phonemes, thus learning good features for SLU. We introduce a new SLU dataset, Fluent Speech Commands, and show that our method improves performance both when the full dataset is used for training and when only a small subset is used. We also describe preliminary experiments to gauge the model's ability to generalize to new phrases not heard during training.

Introduction

Spoken language understanding (SLU) systems infer the meaning or intent of a spoken utterance BIBREF0 . This is crucial for voice user interfaces, in which the speaker's utterance needs to be converted into an action or query. For example, for a voice-controlled coffee machine, an utterance like “make me a large coffee with two milks and a sugar, please” might have an intent representation like {drink: "coffee", size: "large", additions: [{type: "milk", count: 2}, {type: "sugar", count: 1}]}.

The conventional SLU pipeline is composed of two modules: an automatic speech recognition (ASR) module that maps the speech to a text transcript, and a natural language understanding (NLU) module that maps the text transcript to the speaker's intent BIBREF1 , BIBREF2 , BIBREF3 . An alternative approach that is beginning to gain popularity is end-to-end SLU BIBREF4 , BIBREF5 , BIBREF6 , BIBREF7 , BIBREF8 . In end-to-end SLU, a single trainable model maps the speech audio directly to the

speaker's intent without explicitly producing a text transcript (Fig. FIGREF4). Unlike the conventional SLU pipeline, end-to-end SLU:

End-to-end models have been made possible by deep learning, which automatically learns hierarchical representations of the input signal BIBREF9 , BIBREF10 , BIBREF11 , BIBREF12 , BIBREF13 . Speech is natural to represent in a hierarchical way: waveform INLINEFORM0 phonemes INLINEFORM1 morphemes INLINEFORM2 words INLINEFORM3 concepts INLINEFORM4 meaning. However, because speech signals are high-dimensional and highly variable even for a single speaker, training deep models and learning these hierarchical representations without a large amount of training data is difficult.

The computer vision BIBREF14 , BIBREF15 , natural language processing BIBREF16 , BIBREF17 , BIBREF18 , BIBREF19 , BIBREF20 , and ASR BIBREF21 , BIBREF22 communities have attacked the problem of limited supervised training data with great success by pre-training deep models on related tasks for which there is more training data. Following their lead, we propose an efficient ASR-based pre-training methodology in this paper and show that it may be used to improve the performance of end-to-end SLU models, especially when the amount of training data is very small.

Our contributions are as follows:

Related work

Three key papers describing end-to-end SLU were written by Qian et al. BIBREF4 , Serdyuk et al. BIBREF5 , and Chen et al. BIBREF6 . Serdyuk et al. in BIBREF5 use no pre-training whatsoever. Qian et al. in BIBREF4 use an auto-encoder to initialize the SLU model. Chen et al. BIBREF6 pre-train the first stage of an SLU model to recognize graphemes; the softmax outputs of the first stage are then fed to a classifier second stage. The model proposed in this paper is similar to theirs, but removes the restriction

of the softmax bottleneck and uses alternative training targets, as we will describe later.

More recently, Haghani et al. in BIBREF7 compare four types of sequence-to-sequence models for SLU, including a direct model (end-to-end with no pre-training) and a multi-task model (uses a shared encoder whose output is ingested by a separate ASR decoder and SLU decoder). The model proposed here is somewhat similar to their multi-task model, although we do not use or require the ASR targets during SLU training.

The work listed above deals with very high resource SLU—in BIBREF7 , for instance, the Google Home BIBREF23 dataset consists of 24 million labeled utterances. In contrast, Renkens et al. in BIBREF8 consider the problem of end-to-end SLU with limited training data, and find that capsule networks BIBREF24 , compared to conventional neural network models, are more easily capable of learning end-to-end SLU from scratch. However, they do not consider the effect of pre-training on other speech data.

This previous work has all been conducted on datasets that are closed-source or too small to test hypotheses about the amount of data required to generalize well. The lack of a good open-source dataset for end-to-end SLU experiments makes it difficult for most people to perform high-quality, reproducible research on this topic. We therefore created a new SLU dataset, the “Fluent Speech Commands” dataset, which Fluent.ai releases along with this paper.

Dataset

This section describes the structure and creation of Fluent Speech Commands.

Audio and labels

The dataset is composed of 16 kHz single-channel .wav audio files. Each audio file contains a recording of a single command that one might use for a smart home or virtual assistant, like “put on the music” or “turn up the heat in the kitchen”.

Each audio is labeled with three slots: action, object, and location. A slot takes on one of multiple values: for instance, the “location” slot can take on the values “none”, “kitchen”, “bedroom”, or “washroom”. We refer to the combination of slot values as the intent of the utterance. The dataset has 31 unique intents in total. We do not distinguish between domain, intent, and slot prediction, as is sometimes done in SLU BIBREF25 .

The dataset can be used as a multi-label classification task, where the goal is to predict the action, object, and location labels. Since the slots are not actually independent of each other, a more careful approach would model the relationship between slots, e.g. using an autoregressive model, as in BIBREF7 . We use the simpler multi-label classification approach in this paper, so as to avoid the issues sometimes encountered training autoregressive models and instead focus on questions related to generalization using a simpler model. Alternately, the 31 distinct intents can be “flattened” and used as 31 distinct labels for a single-label classification task.

For each intent, there are multiple possible wordings: for example, the intent {action: "activate", object: "lights", location: "none"} can be expressed as “turn on the lights”, “switch the lights on”, “lights on”, etc.. These phrases were decided upon before data collection by asking employees at Fluent.ai, including both native and non-native English speakers, for various ways in which they might express a particular intent. There are 248 different phrases in total.

Data collection

The data was collected using crowdsourcing. Each speaker was recorded saying each wording for each intent twice. The phrases to record were presented in a random order. Participants consented to data being released and provided demographic information about themselves. The demographic information about these anonymized speakers (age range, gender, speaking ability, etc.) is included along with the dataset.

The data was validated by a separate set of crowdsourcers. All audios deemed by the crowdsourcers to be unintelligible or contain the wrong phrase were removed. The total number of speakers, utterances, and hours of audio remaining is shown in Table TABREF12 .

Dataset splits

The utterances are randomly divided into train, valid, and test splits in such a way that no speaker appears in more than one split. Each split contains all possible wordings for each intent, though our code has the option to include data for only certain wordings for different sets, to test the model's ability to recognize wordings not heard during training. The dataset has a .csv file for each split that lists the speaker ID, file path, transcription, and slots for all the .wav files in that split.

Related datasets

Here we review some related public datasets and show the gap that Fluent Speech Commands fills.

The Google Speech Commands dataset BIBREF26 (to which the name “Fluent Speech Commands” is an homage) is a free dataset of 30 single-word spoken commands (“yes”, “no”, “stop”, “go”, etc.). This dataset is suitable for keyword spotting experiments, but not for SLU.

ATIS is an SLU dataset consisting of utterances related to travel planning. This dataset can only be obtained expensively from the Linguistic Data Consortium.

The Snips NLU Benchmark BIBREF2 has a rich set of virtual assistant commands, but contains only text, with no audio, and hence is not suitable for end-to-end SLU experiments.

The Grabo, Domotica, and Patcor datasets are three related datasets of spoken commands for robot control and card games developed by KU Leuven and used in BIBREF8 . These datasets are free, but have only a small number of speakers and phrases.

In contrast to these datasets, Fluent Speech Commands is simultaneously audio-based, reasonably large, and free, and contains several multiple-word commands corresponding to each of the intents.

Model and Pre-training Strategy

The model proposed in this paper, shown in Fig. FIGREF17 , is a deep neural network consisting of a stack of modules, where the first modules are pre-trained to predict phonemes and words. The word and phoneme classifiers are discarded, and the entire model is then trained end-to-end on the supervised SLU task. In what follows, we justify these design decisions and give more details about the model hyperparameters.

Which ASR targets to use?

ASR models are trained using a variety of targets, including phonemes, graphemes, wordpieces, or more recently whole words BIBREF27 , BIBREF28 , BIBREF29 . We choose whole words as the pre-training targets, since this is what a typical NLU module would expect as input. A typical ASR dataset contains too

many unique words (LibriSpeech BIBREF30 has more than 200,000) to assign an output to each one; we only assign a label to the 10,000 most common words. This leaves much of the pre-training data without any labels, which wastes data. By using phonemes as intermediate pre-training targets BIBREF31 , BIBREF19 , BIBREF32 , we are able to pre-train on speech segments with no word label. Additionally, we find that using phonemes as intermediate targets speeds up word-level pre-training BIBREF33 , BIBREF34 .

We use the Montreal Forced Aligner BIBREF35 to obtain word- and phoneme-level alignments for LibriSpeech, and we pre-train the model on the entire 960 hours of training data using these alignments INLINEFORM0 . Using force-aligned labels has the additional benefit of enabling pre-training using short, random crops rather than entire utterances, which reduces the computation and memory required to pre-train the model.

Phoneme module

The first module takes as input the audio signal INLINEFORM0 and outputs INLINEFORM1 , a sequence of hidden representations that are pre-trained to predict phonemes. The phoneme-level logits are computed using a linear classifier: DISPLAYFORM0

The phoneme module is implemented using a SincNet layer BIBREF36 , BIBREF37 , which processes the raw input waveform, followed by multiple convolutional layers and recurrent layers with pooling and dropout. More detailed hyperparameters can be found in our code.

Word module

The second module takes as input INLINEFORM0 and outputs INLINEFORM1 . Similar to the phoneme-level module, it uses recurrent layers with dropout and pooling, and is pre-trained to predict words using another linear classifier: DISPLAYFORM0

Notice that the input to this module is INLINEFORM0 , not INLINEFORM1 , and likewise the output to the next stage is INLINEFORM2 , not INLINEFORM3 . There are two good reasons for forwarding INLINEFORM4 instead of INLINEFORM5 . The first is that we don't want to remove a degree of freedom from the model: the size of INLINEFORM6 is fixed by the number of targets, and this would in turn fix the size of the next layer of the model. The second reason is that computing INLINEFORM7 requires multiplying and storing a large (INLINEFORM8 2.5 million parameters) weight matrix, and by discarding this matrix after pre-training, we save on memory and computation.

Intent module

The third module, which is not pre-trained, maps INLINEFORM0 to the predicted intent. Depending on the structure of the intent representation, the intent module might take on various forms. Since in this work we use a fixed three-slot intent representation, we implement this module using a recurrent layer, followed by max-pooling to squash the sequence of outputs from the recurrent layer into a single vector of logits corresponding to the different slot values, similar to BIBREF5 .

Unfreezing schedule

Although the pre-trained model works well as a frozen feature extractor, it may be preferable to “unfreeze” its weights and finetune them for the SLU task with backpropagation. Similar to ULMFiT BIBREF17 , we find that gradually unfreezing the pre-trained layers works better than unfreezing them all at once. We unfreeze one layer each epoch, and stop at a pre-determined layer, which is a hyperparameter.

Experiments

Here we report results for three experiments on Fluent Speech Commands: using the full dataset, using a subset of the dataset, and using a subset of wordings.

Full dataset

We first trained models given the entire SLU training set. The models used one of: 1) no pre-training (randomly initialized), 2) pre-training with no unfreezing, 3) gradually unfreezing only the word layers, or 4) gradually unfreezing both the word layers and phoneme layers. What we report here as “accuracy” refers to the accuracy of all slots for an utterance taken together—that is, if the predicted intent differs from the true intent in even one slot, the prediction is deemed incorrect.

The validation accuracy of these models over time is shown in Fig. . The best results are obtained when only the word layers of the pre-trained model are unfrozen. This may be because the model begins to forget the more general phonetic knowledge acquired during pre-training. For the test set, the frozen model and partially unfrozen model perform roughly equally well (Table TABREF28 , “full” column), possibly because the test set is “easier” than the validation set. In all cases, the pre-trained models outperform the randomly initialized model.

Partial dataset

To simulate a smaller dataset, we randomly selected 10% of the training set, and used this instead of the entire training set. Fig. shows the validation accuracy (on the entire validation set, not a subset) over time. A similar trend is observed as for the entire dataset: unfreezing the word layers works best. The gap in final test accuracy between the randomly initialized model and the pre-trained models increases (Table TABREF28 , “10%” column); the final test accuracy for the pre-trained models drops only slightly, further highlighting the advantage of our proposed method.

Generalizing to new wordings

What happens if new wordings appear in the test data that never appear in the training data? This is an important question, since it is generally impractical to try to imagine every possible wording for a particular intent while gathering training data.

To test this, we trained models on three specific phrases, “turn on the lights”, “turn off the lights”, and “switch on the lights” (273 utterances total), and tested on those same phrases, as well as a new phrase: “switch off the lights”. If the model incorrectly infers that utterances that contain “switch” always correspond to turning on the lights, it will incorrectly guess that “switch off the lights” corresponds to turning on the lights; if the model infers that the presence of the word “off” corresponds to turning off the lights, it will generalize to the new phrase. The randomly initialized model was unable to fit this tiny training set, even with a very low learning rate and no regularization. The pre-trained models were able to generalize to the new wording (with 97% accuracy on the validation set, which contains more examples of the new phrase than of the training phrases).

However, there are many situations in which our model does not correctly generalize. For example, if the

model is trained only with examples containing “bedroom” and “washroom”, but then tested on an example containing “bathroom”, it will guess the intent corresponding to “bedroom” because “bedroom” sounds more similar to “bathroom” than to “washroom”, even though “washroom” is the correct meaning. In text-based NLU, this scenario can be handled using word embeddings, which represent words in such a way that words with similar meanings have similar vector representations BIBREF1 , BIBREF38 . It may be possible to teach the pre-trained part of the model to output “embedding-like” word representations so that the intent module can recognize the meaning of phrases with synonyms.

Conclusion

In this paper, we proposed a pre-training methodology for end-to-end SLU models, introduced the Fluent Speech Commands dataset, and used this dataset to show that our pre-training techniques improve performance both for large and small SLU training sets. In the future, we plan to continue using Fluent Speech Commands to explore the limitations of end-to-end SLU, like new wordings and synonyms not observed in the SLU dataset, to see if these limitations can be overcome.

Acknowledgements

We would like to acknowledge the following for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs, and CIFAR.

Thanks to Dima Serdyuk and Kyle Kastner at Mila, and Farzaneh Fard, Luis Rodriguez Ruiz, Sam Myer, Mohamed Mhiri, and Arash Rad at Fluent.ai for helpful discussions with us about this work.