Deep Learning Assignment Saarthi.ai

Objective:
Given assignment is designed to check your skills of deep learning frameworks compatibility, code quality, basic software engineering, understanding of problems and discipline.
In the given task Goal is that with given text you need to extract all the labels I.e., action that needs to be taken, action to be taken on which object and location where that object is present.

Data:
Following dataset contains a train_data.csv, valid_data.csv, having columns path of audio data, transcription of the audio data, action that needs to be taken, action to be taken on which object and location where that object is present. Below is the link to the dataset on which you have to perform all your experiments. You just need to use corresponding audio and labels for those texts.
Link for data:
https://drive.google.com/file/d/1slGtHKHYTtiuC98yomV0hP3C85Q5V8sg/view?usp=sharing

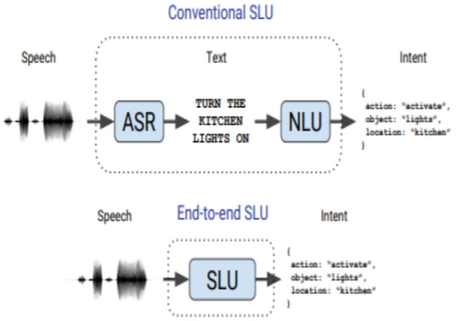
Abstract

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.
Index Terms: speech recognition, spoken language understanding, end-to-end models, transfer learning

Introduction

Spoken language understanding (SLU) systems infer the meaning or intent of a spoken utterance . 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. 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.


Conventional SLU
Speech
Text
Intent
ASR
TURN THE KITCHEN LIGHTS ON
NLU
{ action: "activate", object: "lights", location: "kitchen" }
Speech
End-to-end SLU
Intent
SLU
{ action: "activate", object: "lights", location: "kitchen" }

Unlike the conventional SLU pipeline, end-to end SLU:
• directly optimizes the metric of interest (intent recognition accuracy),
• does not waste modeling effort on estimating the text, thus yielding a more compact model and avoiding an error-prone intermediate step involving search algorithms, language models, finite state transducers, etc.,
• and enables harnessing aspects of the utterance that may be relevant for inferring the intent, but are not present in the text transcript, such as prosody.
End-to-end models have been made possible by deep learning, which automatically learns hierarchical representations of the input signal.

Speech is natural to represent in a hierarchical way: *waveform → phonemes → morphemes → words → concepts → 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

Dataset:

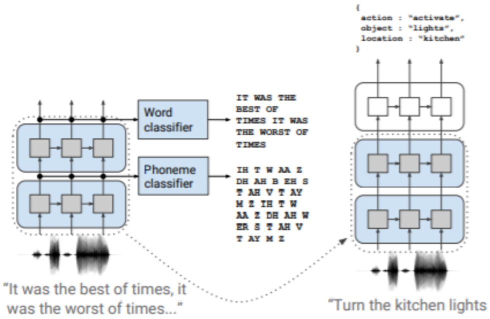
The dataset is composed of single-channel .wav audio files. Each audio file contains a recording of a single spoken English command that one might use for a smart home or virtual assistant, like “put on the music” or “turn up the heat in the kitchen”. Each audio is labeled with three slots: **action**, **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 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.

The simpler multi-label classification approach is best to choose, so as to avoid the issues sometimes encountered training autoregressive models and instead focus on questions related to generalization using a simpler model.

Pretraining and Model:

Using the dataset described in the previous section, test the performance of proposed model and pre-training strategy is done as follows:


{
action : "activate",
object : "lights",
location : "kitchen"
}
Word classifier
Phoneme classifier
IT WAS THE BEST OF TIMES IT WAS THE WORST OF TIMES
IH T W AA Z DH AH B EH S T AH V T AY M Z IH T W AA Z DH AH W ER S T AH V T AY M Z
“It was the best of times, it was the worst of times...”
“Turn the kitchen lights on”

The lower layers of the model are pre-trained using ASR targets (words and phonemes). The word and phoneme classifiers are discarded, and the features from the pre-trained part of the model (blue) are used as the input to the subsequent module (white), which is trained using SLU targets.

The model 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.

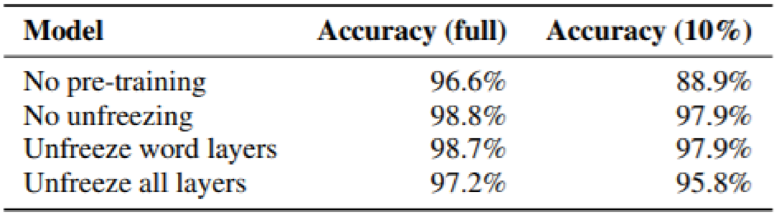
ASR models are trained using a variety of targets, including phonemes, graphemes, wordpieces, or more recently whole words. Three modules, respectively, Phoneme module, Word module and Intent Module are used in layers.

Although the pre-trained model works well as a frozen feature extractor, it may be preferable to “unfreeze” its weights and fine tune them for the SLU task with backpropagation. Similar to

ULMFiT, we find that gradually unfreezing the pretrained layers works better than unfreezing them all at once. We unfreeze one layer each epoch, and stop at a predetermined layer, which is a hyperparameter.

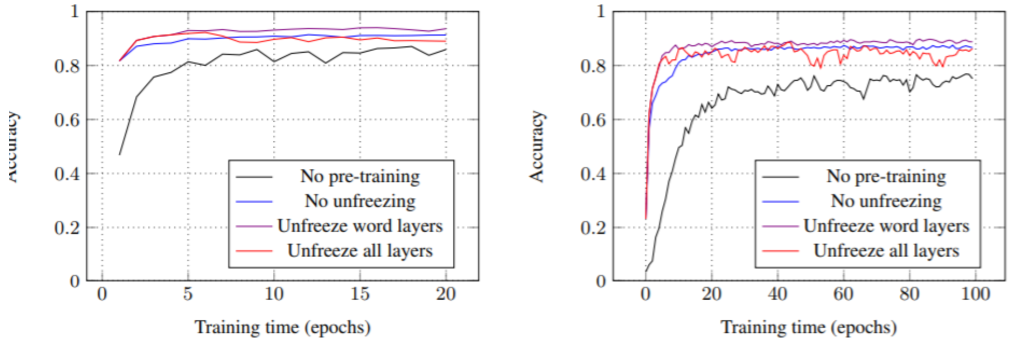
Experiments

According to research paper data from: “Speech Model Pre-training for End-to-End Spoken Language Understanding” by Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar , Yoshua Bengio
Accuracy on the test set for different models, given the full training dataset or a 10% subset of the training data:



Model	Accuracy (full)	Accuracy (10%)
No pre-training	96.6%	88.9%
No unfreezing	98.8%	97.9%
Unfreeze word layers	98.7%	97.9%
Unfreeze all layers	97.2%	95.8%

Accuracy on the validation set over time for models trained on (a) the full SLU dataset or (b) 10% of the dataset.:



(a) *Full dataset.*

(b) *10% of the dataset.*

This clearly shows that this dataset shows that our pre-training techniques improve performance both for large and small SLU training set

Dependencies

PyTorch, torchaudio, numpy, soundfile, pandas, tqdm, textgrid.py

Training

First, change the *asr_path* and *slu_path* in the config file (like experiments/no_unfreezing.cfg, or whichever experiment you want to run) to point to where the LibriSpeech data and/or Fluent Speech Commands data are stored on your computer.

SLU training: To train the model on an SLU dataset, run the following command:

```
python main.py --train --config_path=<path to .cfg>
```

ASR pre-training: `python main.py --pretrain --config_path=<path to .cfg>`

Inference

You can perform inference with a trained SLU model as follows

```
import data
import models
import soundfile as sf
import torch

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
config = data.read_config("experiments/no_unfreezing.cfg");
_,_,_=data.get_SLU_datasets(config)
model = models.Model(config).eval()
model.load_state_dict(torch.load("experiments/no_unfreezing/training/model_state.pth",
map_location=device)) # load trained model

signal, _ = sf.read("test.wav")
signal = torch.tensor(signal, device=device).float().unsqueeze(0)

model.decode_intents(signal)
```

Citation

- Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio, "Speech Model Pre-training for End-to-End Spoken Language Understanding", Interspeech 2019.

- Loren Lugosch, Brett Meyer, Derek Nowrouzezahrai, and Mirco Ravanelli, "Using Speech Synthesis to Train End-to-End Spoken Language Understanding Models", ICASSP 2020.