Using Transfer Learning for Automatic Speech Translation

Tatjana Chernenko, Siting Liang

Institute for Computational Linguistics,
Ruprecht-Karls-Universität Heidelberg
{chernenko/liang}@cl.uni-heidelberg.de

Prof. Dr. Riezler, Sariya Karimova
Speech Recognition Project

Outline

- Introduction
- Transfer Learning
- Approach
- Datasets
- Experimental Setup
- Results
- Conclusion and discussion

Introduction

Automatic Speech Translation is a problem which involves Automatic Speech Recognition in order to transfer the audio input into text, which will be translated into another language by a Machine Translation System.

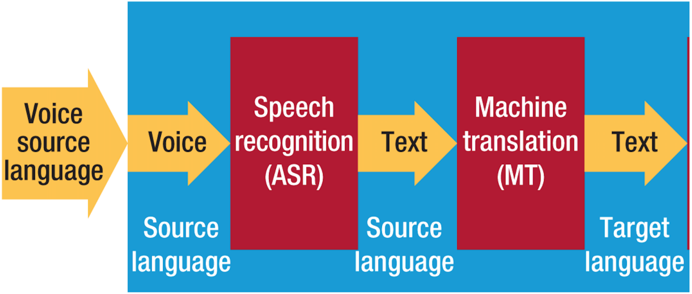
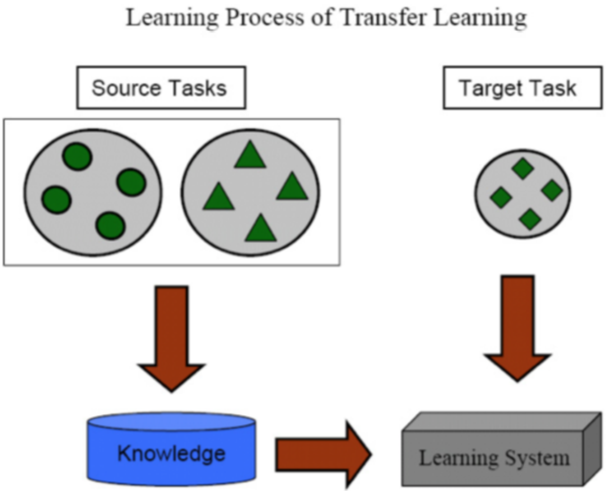


Abbildung: Automatic Speech Translation

Transfer Learning



Approach

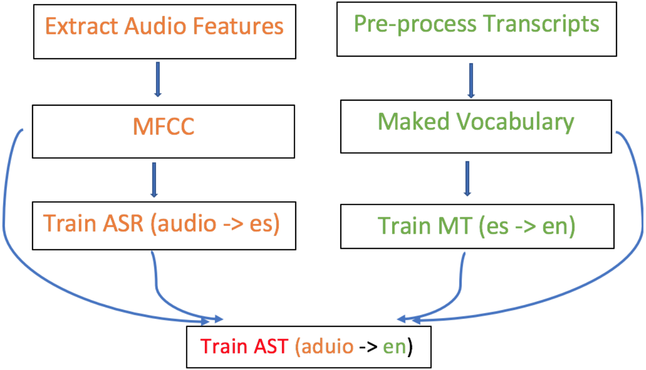


Abbildung: Project Process: ASR + MT -> AST

Datasets

- LDC2010T04
- Audio : Fisher Spanish Speech (819 telephone conversations of 10 to 12 minutes in duration)
- Fisher Spanish - Transcripts (819 transcribed conversations in Spanish)
- Fisher English Translation (obtained by crowd-sourcing using Amazon's Mechanical Turk)

Experiment Setup

Pre-trained Models: ASR, MT

	ASR ("best-276000")	MT ("best-98000")
step	276000	98000
train loss	14.338	20.686
dev wer	40.65	46.17
dev bleu	41.91	35.98
dev ter	46.19	45.90
dev bleu1	64.24	65.87
dev loss	17.36	21.44
dev ratio	1.061	1.015

We used two pre-trained Models (ASR and MT) as checkpoints for our Transfer Learning AST System.

Settings

Pre-trained ASR

- **Hyperparameters *(find other hyperparameters in log files of ASR)***
  - batch_size: 32
  - cell_size: 256
  - attn_size: 256
  - cell_type: LSTM
  - encoder:
    - embedding_size: 41
    - layers: 3
    - input_layers: 256, 128
    - input_layer_activation: tanh
  - decoder:
    - deep_layer_size: 256
    - embedding_size: 128
    - max_len: 285

Settings

Pre-trained MT

- **Hyperparameters *(find other hyperparameters in log files of MT)***
  - batch_size: 64
  - cell_size: 512
  - attn_size: 512
  - cell_type: LSTM
  - encoder:
    - cell_size: 512
    - embedding_size: 256
    - max_len: 75
  - decoder:
    - deep_layer_size: 512
    - embedding_size: 128
    - max_len: 275

New System: AST with Transfer Learning

- Use ASR and MT as checkpoints
- **Hyperparameters of AST *(other hyperparameters are in log and log1 files of AST)***
  - batch_size: 32
  - cell_size: 512
  - attn_size: 512
  - cell_type: LSTM
  - encoder:
    - embedding_size: 41
    - layers: 3
    - input_layers: 256, 128
    - max_len: 1010
  - decoder:
    - deep_layer_size: 256
    - embedding_size: 128
    - max_len: 285

Results

AST with Transfer Learning - BEST MODEL

- Results:
  - Best model: step 364 000 *
  - train loss 16.770
    dev bleu=17.12
    dev ter=72.46
    dev wer=72.71
    dev bleu1=45.72
    dev loss=35.58
    dev penalty=1.000
    rdev ratio=1.017

** Step in log file = 462000 (training starts from step 98000 because of the used checkpoints)*

AST train loss

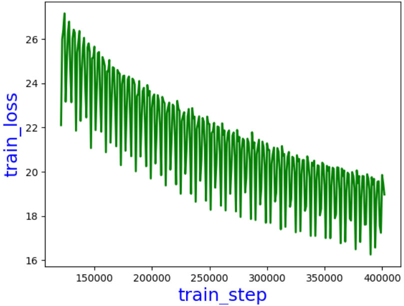


Abbildung: AST with transfer-learning train loss (starting from step 120000. Find steps 1-120000 on pages 17-20)

All steps on all AST transfer-learning Diagrams are real time steps. Add 98000 to get the number of the current step in a log file.

AST dev WER

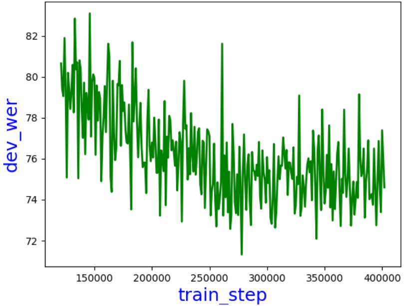


Abbildung: AST with transfer-learning WER (starting from step 120000. Find steps 1-120000 on pages 17-20)

All steps on all AST transfer-learning Diagrams are real time steps. Add 98000 to get the number of the current step in a log file.

AST dev BLEU

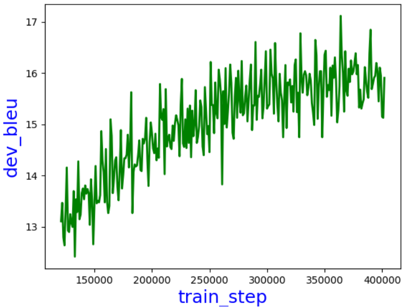


Abbildung: AST with transfer-learning BLEU (starting from step 120000. Find steps 1-120000 on pages 17-20)

All steps on all AST transfer-learning Diagrams are real time steps. Add 98000 to get the number of the current step in a log file.

AST dev BLEU 1

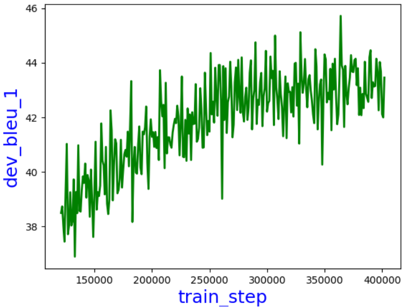


Abbildung: AST with transfer-learning dev BLEU 1 (starting from step 120000. Find steps 1-120000 on pages 17-20)

All steps on all AST transfer-learning Diagrams are real time steps. Add 98000 to get the number of the current step in a log file.

Comparison

AST using Transfer Learning vs. Multi-Task AST from scratch

It would be interesting to compare the model to the AST model with similar hyperparameters trained from scratch without Transfer Learning.

Limited technical/time resources don't allow to train such a baseline model to use it for the direct comparison, that's why we use the available Multi-Task Learning AST system without pre-trained models (Project Group 5, extra model) as our model for comparison.

Multi-Task learning AST without pre-trained models - **later 'Comparison Model'**.
Note that this model uses a bit different set of hyperparameters, that's why the comparison of the two models could not be direct.

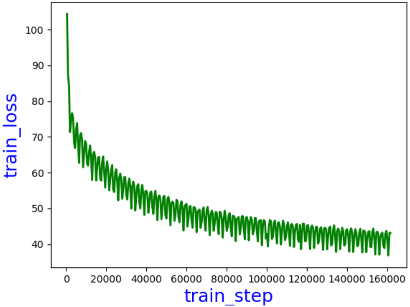
Settings of Comparison Model

Multi-Task AST without pre-trained models - Comparison Model

- **Comparison Model Hyperparameters:**
  - batch_size: 64
  - cell_type: LSTM
  - encoder:
    - embedding_size: 41
    - layers: 3
    - input_layers: 256, 128
    - max_len: 1010
  - decoder:
    - deep_layer_size: 256
    - embedding_size: 128
    - max_len: 285

** Other hyperparameters are in log files of Project Group 5*

Train loss



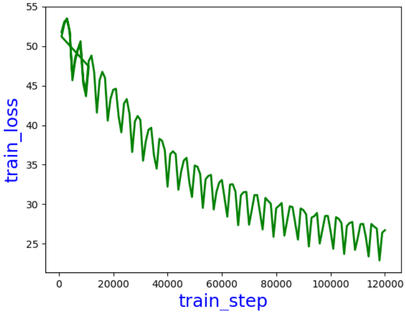


Abbildung: Comparison Model (left) and AST with transfer-learning (right) train loss

Dev WER

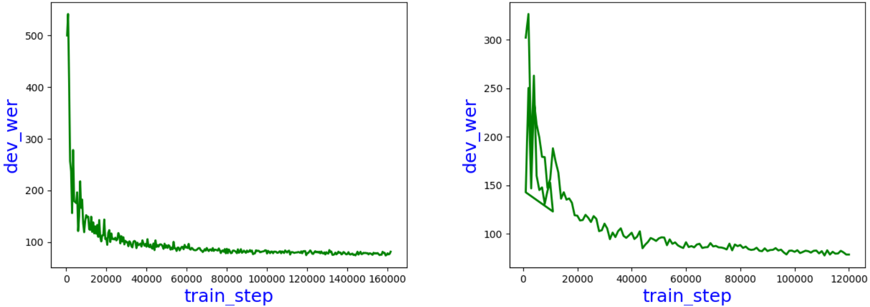
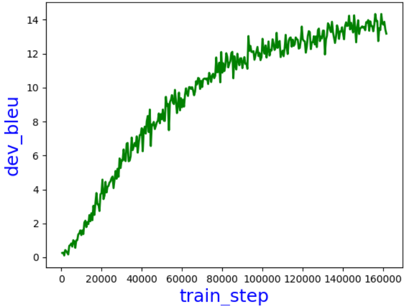


Abbildung: Comparison Model (left) and AST with transfer-learning (right) dev WER

Dev BLEU



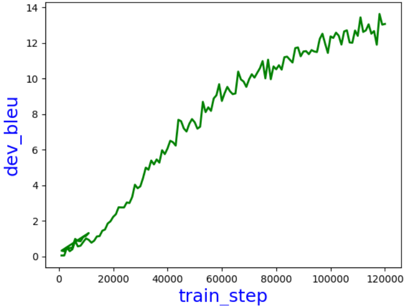
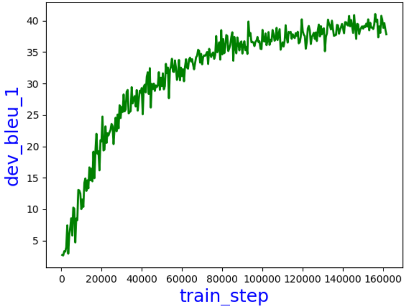


Abbildung: Comparison Model (left) and AST with transfer-learning (right) dev BLEU

Dev BLEU 1



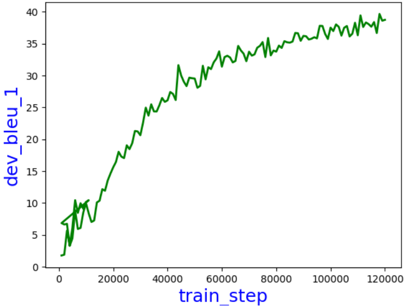


Abbildung: Comparison Model (left) and AST with transfer-learning (right) dev BLEU 1

Possible Improvements, Conclusion and Discussion

- Transfer Learning should allow rapid progress of improved performance. The trained AST model in our case doesn't show significant performance on the development set (bleu=17.12, bleu1=45.72, wer=72.71).
- It could be useful to evaluate the model on unseen data
- To train a suitable comparison model: AST Model with the same hyperparameters and number of steps from scratch (without pre-trained MT and ASR Models)
- To use optimal pre-trained versions of ASR and MT models. Our initialization step was not optimal at that time step (see page 6 - Experiment Setup). Optimal solution **at the time step of our model initialization** would be:
  - ASR step 227000 (best model with train_loss 12.902, dev wer=39.95 bleu=43.03 ter=44.45 bleu1=65.08 loss=16.93 ratio=1.060)
  - MT step 92000 (train_los = 20.643, dev bleu=36.13 ter=45.71 wer=46.19 bleu1=66.01 loss=21.06 penalty=1.000 ratio=1.017)