SpeechJoey: Minimalistic Speech-to-Text Modeling with JoeyNMT

Mayumi Ohta

17th June, 2021

statNLP

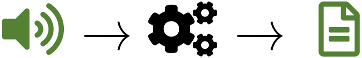
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Today

1. **Speech-to-Text Modeling**
   - Speech-to-Text Tasks
   - Data Preprocessing
   - Architecture
2. **SpeechJoey**
   - Features
   - Performance
   - Code Complexity

assume: basic ML knowledge,
familiarity with neural machine translation with JoeyNMT

Speech-to-Text Tasks



→ `I'm going to talk today about energy and climate.`

Automatic Speech Recognition (ASR)

→ `Heute spreche ich zu Ihnen über Energie und Klima.`

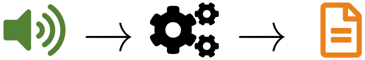
End-to-End Speech Translation (ST)

---

beyond ASR or ST (out of SpeechJoey's scope)

- Speaker identification (audio → speaker id)
- Speech separation (detect who spoke when)
- Spoken language Understanding (audio → sentiment / slots)

Speech-to-Text Tasks



→ I'm going to talk today about energy and climate.

Automatic Speech Recognition (ASR)

→ Heute spreche ich zu Ihnen über Energie und Klima.

End-to-End Speech Translation (ST)

---

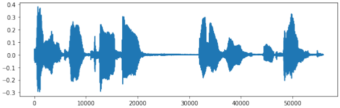
beyond ASR or ST (out of SpeechJoey's scope)

- Speaker identification (audio → speaker id)
- Speech separation (detect who spoke when)
- Spoken language Understanding (audio → sentiment / slots)

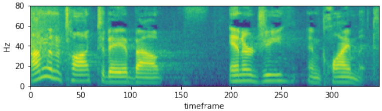
Input Features

🔊 Src: Wave form → Frequency-based 2d-array

ex.) <u>Mel-filterbank</u>, MFCC, etc.



→



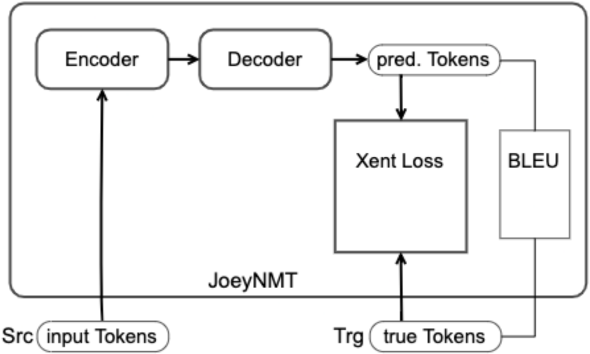
📄 Trg: Text string → Token sequence

ex.) Chars, <u>BPEs</u>, Words, etc.

```
"I’m going to talk today about energy and climate."
↓
["_I", "'", "m", "_going", "_to", "_talk", "_today",
"_", "about", "_energy", "_and", "_climate"]
```

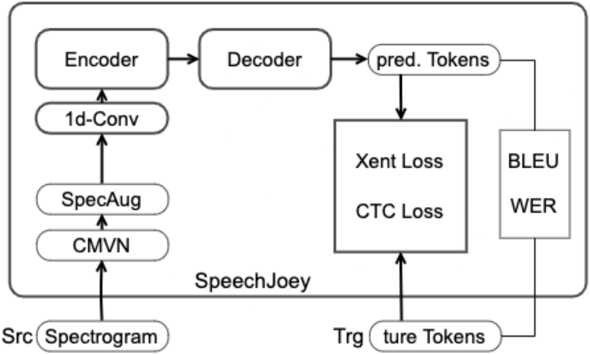
Encoder-Decoder Architecture

JoeyNMT

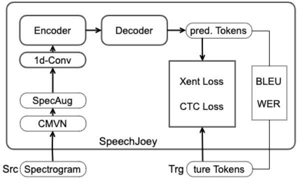


Encoder-Decoder Architecture

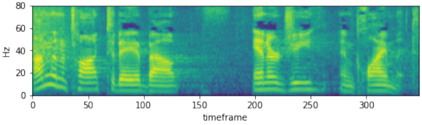
SpeechJoey



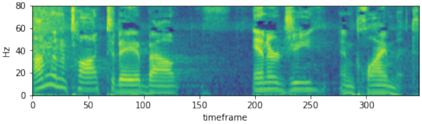
Normalization, Augmentation



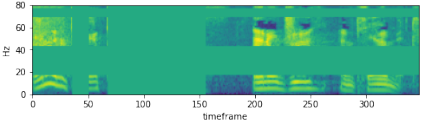
Original:



Cepstral mean and variance normalization (CMVN):

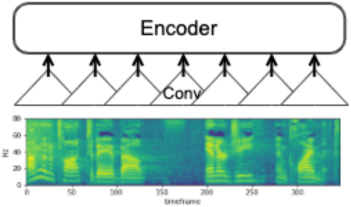


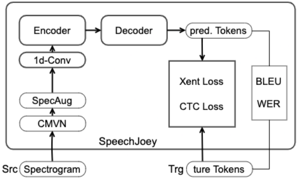
SpecAugment:



On-the-fly

Convolutional Layer





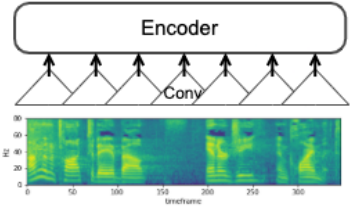
1. downsample the input length to make the computation tractable

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel size} - 1) - 1}{\text{stride}} + 1 \right\rfloor$$

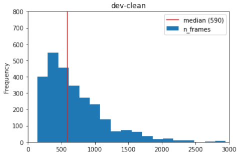
- computational complexity of self-attention is $\mathcal{O}(\ell^2 \cdot d)$
  where ℓ: time-frames, d: frequencies

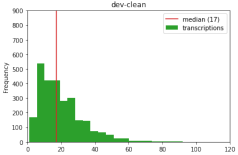
2. audio input is redundant (?)

Convolutional Layer



LibriSpeech dev





audio: 590, transcripts: 17 (median)

1. downsample the input length to make the computation tractable

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel size} - 1) - 1}{\text{stride}} + 1 \right\rfloor$$

- computational complexity of self-attention is $\mathcal{O}(\ell^2 \cdot d)$

  where ℓ: time-frames, d: frequencies

2. audio input is redundant (?)

CTC loss

https://distill.pub/2017/ctc/

We start with an input sequence, like a spectrogram of audio.

The input is fed into an RNN, for example.

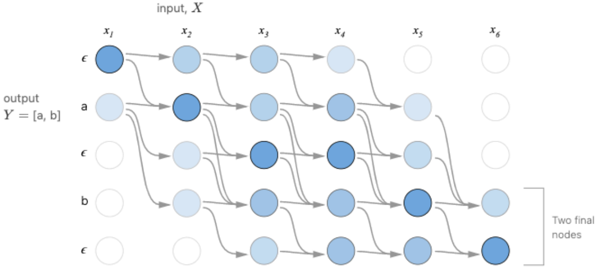
The network gives $p_t(a \mid X)$, a distribution over the outputs $\{h, e, l, o, \epsilon\}$ for each input step.

With the per time-step output distribution, we compute the probability of different sequences

By marginalizing over alignments, we get a distribution over outputs.

CTC loss

https://distill.pub/2017/ctc/



Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

$$\text{ctc prob}(Y|X) = \sum_{a \in \mathcal{A}} \prod_{t=1}^{T} p_t(a_t|X)$$

$$\text{ctc loss}(X; Y) = \sum_{(X,Y) \in \mathcal{D}} -\log \ \text{ctc prob}(Y|X)$$

What is most important to you?

1. state-of-the-art performance
2. clean, open-source code
3. minimalistic system design
4. walk-through tutorial
5. comprehensive API documentation
6. easy to install (no Kaldi!)
7. ready to use
   - pretrained models
   - preprocessing, evaluation included
   - recipe for major benckmarks

Fast prototyping

Education

Performance: ASR on LibriSpeech

	100h (WER ↓)			
	dev-clean	**dev-other**	**test-clean**	**test-other**
[Lüscher et al., 2019]	14.7	38.5	14.7	40.8
[Kahn et al., 2020]	14.0	37.0	14.9	40.0
[Laptev et al., 2020]	10.3	24.0	11.2	24.9
SpeechJoey	9.43	20.29	10.52	21.10
	960h (WER ↓)			
ESPNET	1.9	4.9	2.1	4.9
fairseq	3.0	7.5	3.2	7.5
SpeechBrain	-	-	2.46	5.77
wav2vec-U	1.6	3.0	1.8	3.3
[Zhang et al., 2020]	1.3	2.6	1.4	2.6
SpeechJoey	3.99	9.11	4.54	9.09

*WER=lower is better. *SpeechJoey snapshot as of March 2021.

Performance: ST on MuST-C en-de

	ASR (WER ↓)		**MT** (BLEU ↑)		**ST** (BLEU ↑)	
	CM	**HE**	**CM**	**HE**	**CM**	**HE**
	MuST-C version 1					
[Gangi et al., 2019]	27.0	-	25.30	-	-	-
[Zhang et al., 2020]	-	-	-	-	22.4	-
ESPNET	12.7	-	27.63	-	22.9	-
NeurST	13.6	-	27.9	-	22.8	-
fairseq S2T	18.2	-	-	-	22.7	-
	MuST-C version 2					
SpeechJoey	10.83	8.50	27.77	26.17	22.96	22.53

*$\mathbf{CM}$=tst-COMMON; **HE**=tst-HE

*WER=lower is better; BLEU=higher is better.

*SpeechJoey results on MuST-C v1 will be available soon!

What SpeechJoey can do / can't do

- ✔ ASR / ST
- ✔ colab tutorial (coming soon!)
- ✔ preprocessing scripts
- ✔ CMVN, SpecAugment
- ✔ WER, BLEU
- ✔ baseline benchmark configs
- ✔ easy to install
- ✔ pretrained models
- ✔ module-wise initialization (ASR and MT pre-training for ST)

- 🛇 Audio-to-Audio
  - speech enhancement
  - speech separation
- 🛇 Text-to-Speech
- 🛇 various architectures (RNN, transducers, ...)
- 🛇 self-supervised training (wav2vec)

Code Complexity

Counts	ESPNET [1]	fairseq [2]	SpeechJoey[3]
Files	653	587	50
Code	80710	81763	7079
Comments	17997	13383	2313
Comment/Code Ratio	0.22	0.16	0.33

Table: Code complexity

Note: computed on the whole github repo

Tool: https://github.com/AlDanial/cloc

commit hash: [1]`c36294ea` [2]`3796a80f6` [3]`3ccff729` as of June 2021

Stack Trace in fairseq S2T

for CTC extension, where to change?

- `fairseq_cil/train.py`
- `fairseq/trainer.py`
- `fairseq/tasks/fairseq_task.py`
- `fairseq/models/speech_to_text/s2t_ctc_transformer.py`
- `fairseq/data/audio/speech_to_text_dataset.py`
- `fairseq/criterions/xent_ctc_loss.py`
- `fairseq/sequence_generator.py`

  ⋮

Abstract, hierarchical structure

Stack Trace in SpeechJoey

for CTC extension, where to change?

- `joeynmt/__main__.py`
- `joeynmt/training.py`
- `joeynmt/model.py`
- `joeynmt/loss.py`
- `joeynmt/search.py`

  ⋮

Shallow directory structure

Simple class inheritance

Have I convinced you of the advantages of SpeechJoey?

Q & A

References I

Baevski, A., Hsu, W.-N., Conneau, A., and Auli, M. (2021).
Unsupervised speech recognition.
arXiv:2105.11084.

Gangi, M. A. D., Negri, M., and Turchi, M. (2019).
Adapting Transformer to End-to-End Spoken Language Translation.
Proc. Interspeech 2019.

Hayashi, T., Watanabe, S., Zhang, Y., Toda, T., Hori, T., Astudillo, R., and Takeda, K. (2018).
Back-Translation-Style Data Augmentation for end-to-end ASR.
IEEE Spoken Language Technology Workshop (SLT).

Kahn, J., Lee, A., and Hannun, A. (2020).
Self-training for end-to-end speech recognition.
IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Laptev, A., Korostik, R., Svischev, A., Andrusenko, A., Medennikov, I., and Rybin, S. (2020).
You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation.
13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI).

Liu, A. H., Lee, H.-y., and Lee, L.-s. (2019).
Adversarial training of end-to-end speech recognition using a criticizing language model.
IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

References II

Lüscher, C., Beck, E., Irie, K., Kitza, M., Michel, W., Zeyer, A., Schlüter, R., and Ney, H. (2019).
RWTH ASR Systems for LibriSpeech: Hybrid vs Attention.
Proc. Interspeech 2019.

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., Mori, R. D., and Bengio, Y. (2021).
SpeechBrain: A general-purpose speech toolkit.
arXiv:2106.04624.

Zhang, B., Titov, I., Haddow, B., and Sennrich, R. (2020).
Adaptive feature selection for end-to-end speech translation.
Findings of the Association for Computational Linguistics: EMNLP 2020.

ESPNET https://github.com/espnet/espnet/blob/master/egs/librispeech/asr1/RESULTS.md

fairseq S2T https://github.com/pytorch/fairseq/blob/master/examples/speech_to_text/docs/librispeech_example.md

NeurST https://github.com/bytedance/neurst/tree/master/examples/speech_to_text/must-c

SpeechBrain https://github.com/speechbrain/speechbrain