Karlsruhe Institute of Technology

Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation

Seminar: Speech-to-Speech translation

Dennis Keck | July 15th, 2019

INTERACTIVE SYSTEMS LABS (ISL)

Section 1

Introduction

Abstract

Attention-Passing Models for Robust and Data-Efficient End-to-End Speech Translation

Matthias Sperber, *Graham Neubig*, *Jan Niehues*, *Alex Waibel*

Three main achievements:

- Compares performance and data efficiency of *direct* to *cascaded* models for speech translation
- Application of a *two-stage* model for end to end speech translation
- Introduction of an *attention-passing* enhancement for the two-stage model

Context

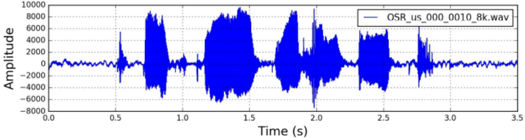
- Speech translation: audio input $\rightarrow$ text translations
- Previously: cascadeding an automatic speech recognition (ASR) and a machine translation (MT) component
- Problem: propagation of error, source text coming from ASR component might be erroneous and lead to folow-up errors

Context

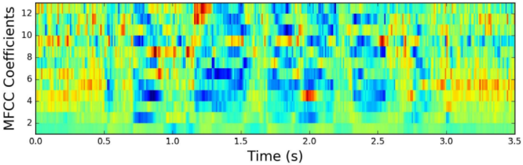
- More recently: Huge interest in direct models for end to end training of speech translation
- But: Reports comparing direct and cascaded models give no clear result yet
- But: usually more training data available for cascaded models as ASR and MT components can be trained seperately

Overview (Task)

Raw audio input



Mel bank speech features



ASR output

<sos>The birch canoe slid on the smooth planks.<eos>

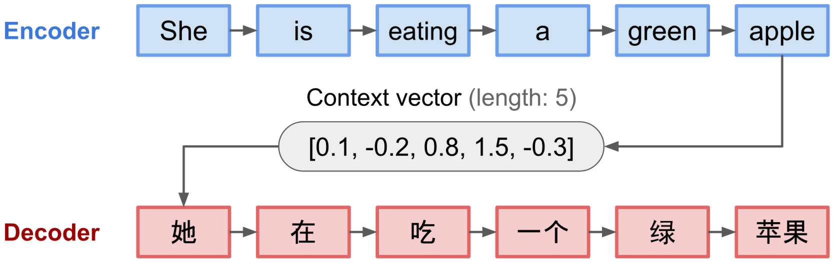
MT output

<sos> 樺樹獨木舟在光滑的木板上滑動。 <eos>

Section 2

Attention mechanism

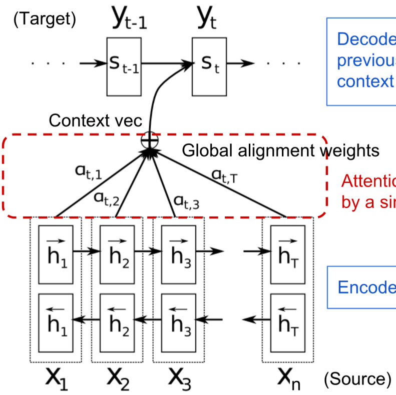
Vanilla sequence to sequence model



- Fixed context vector length from encoder’s last hidden state
- Problem: Can’t remember long sentences. Model has “forgotten” first part when processing whole input.

Attention basic idea

- build “shortcuts” between context vector and source input
- decoder can “attend” to different parts of the input at every output step
- now each decoder output word depends on a weighted combination of **all input states**

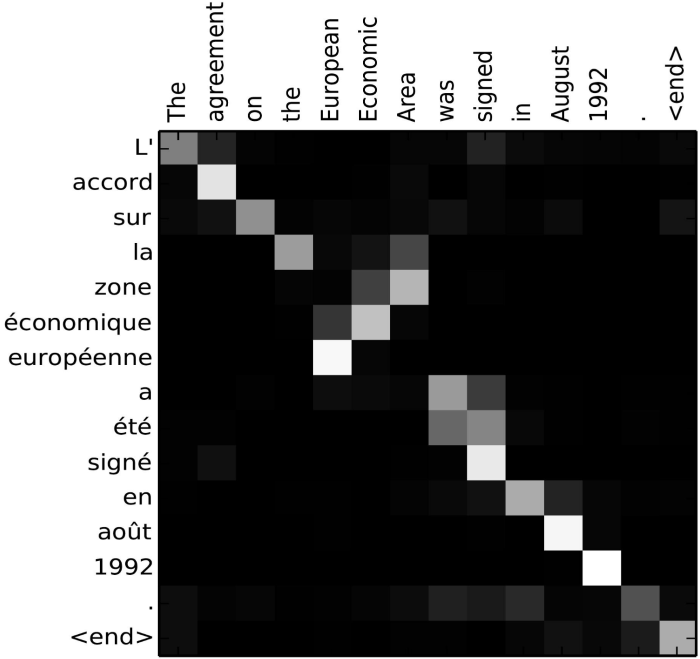


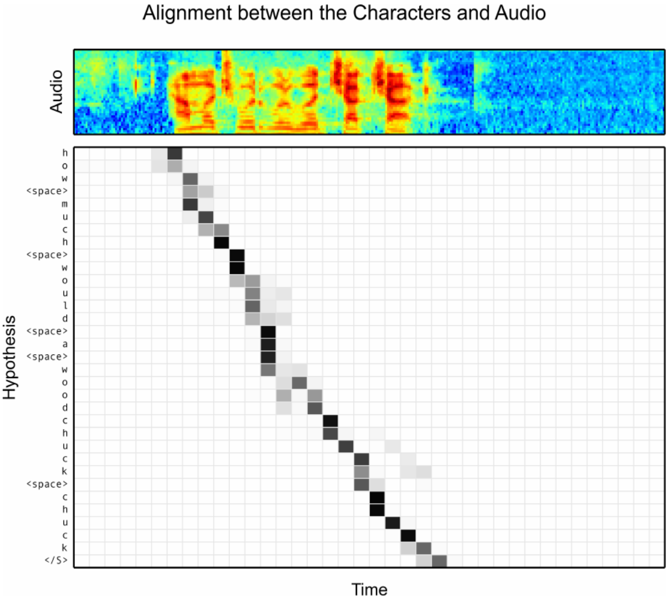
Decoder: RNN with input from previous state + dynamic context vector.

Attention layer: parameterized by a simple feed-forward network

Additive Attention

Encoder: bidirectional RNN


The
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>
L'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>

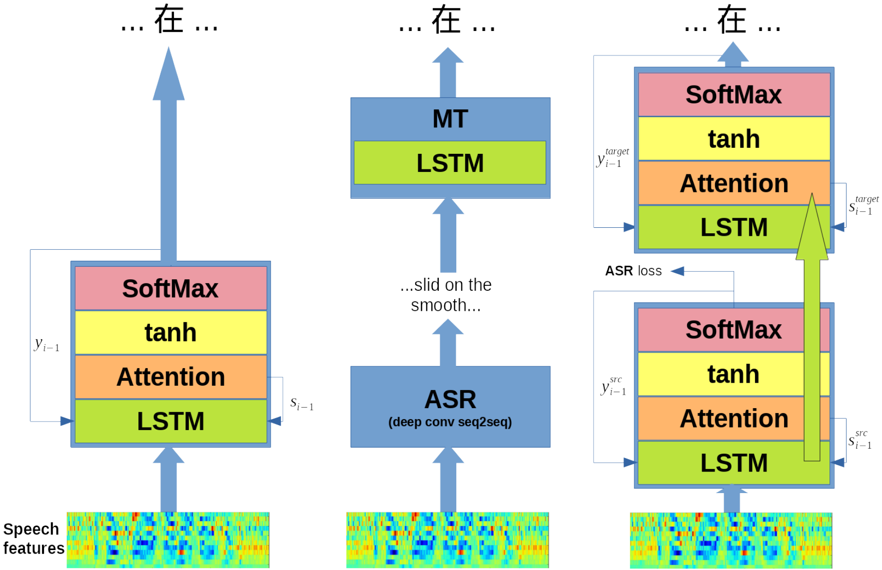


from Chan et al.: **Listen, Attend and Spell** (2017)

Section 3

Models in detail

Overview (Architectures)



a) Direct model b) Cascaded model c) Two stage model

Introduction

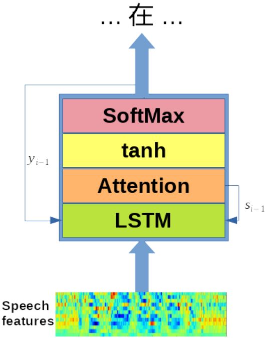
All of the models have in common: - Audio input encoded as Mel-Bank-Features

Cascaded model

TODO

- traditionally used and still state of the art
- easier to learn complex audio to text mapping
- cannot be trained end to end
- but: can make use of more abundant text translation and speech recognition corpi
- propagation of error problem

Direct model

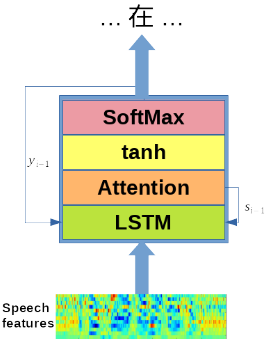


$$
\begin{aligned}
s_i &= \mathrm{LSTM}([W_e y_{i-1}\,;\, c_{i-1}], s_{i-1}\,;\, \theta_{lstm}) \\
c_i &= \mathrm{Attention}([s_i e_{1:L}\,;\, \theta_{att}]) \\
\widetilde{s}_i &= \tanh(W_s[s_i\,;\, c_i] + b_s) \\
p(y_i|y_{<i}, e_{1:L}) &= \mathrm{SoftMaxOut}(\widetilde{s}_i\,;\, \theta_{out})
\end{aligned} \tag{1}
$$

Variables:

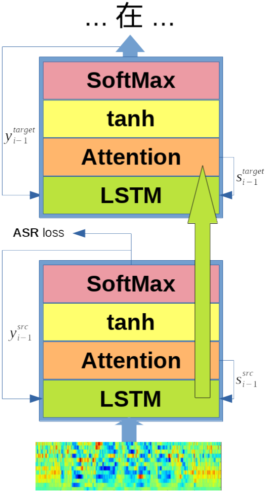
- $e_{1:L}$ L audio encoder states
- W_*, θ_*, b_s trainable parameters
- y_i output characters

Direct model



- more recently shown
- complex mapping from audio to text has to be learned in the model with little guidance
- needs speech to translation datasets for training => a lot less data available

Two stage model



$$
\begin{aligned}
s_i^{src} &= \mathrm{LSTM}([W_e^{src} y_{i-1}^{src} \, ; \, c_{i-1}^{src}], s_{i-1}^{src} \, ; \, \theta_{lstm}^{src}) \\
c_i^{src} &= \mathrm{Attention}([s_i^{src}, e_{1:L} \, ; \, \theta_{att}^{src}]) \\
\tilde{s}_i^{src} &= \tanh(W_s^{src}[s_i^{src} \, ; \, c_i^{src}] + b_s^{src}) \\
p(y_i^{src} | y_{<i}, e_{1:L}) &= \mathrm{SoftMaxOut}(\tilde{s}_i^{src} \, ; \, \theta_{out}^{src})
\end{aligned}
\tag{2}
$$

$$
\begin{aligned}
s_i^{trg} &= \mathrm{LSTM}([W_e^{trg} y_{i-1}^{trg} \, ; \, c_{i-1}^{trg}], s_{i-1}^{trg} \, ; \, \theta_{lstm}^{trg}) \\
c_i^{trg} &= \mathrm{Attention}([s_i^{trg} s_{1:N}^{src} \, ; \, \theta_{att}^{trg}]) \\
\tilde{s}_i^{trg} &= \tanh(W_s^{trg}[s_i^{trg} \, ; \, c_i^{trg}] + b_s^{trg}) \\
p(y_i^{trg} | y_{<i}, e_{1:L}) &= \mathrm{SoftMaxOut}(\tilde{s}_i^{trg} \, ; \, \theta_{out}^{trg})
\end{aligned}
\tag{3}
$$

Two stage model

- two encoder-decoder stages, but decoder of first and encoder of second stage shared:
  - unlike cascaded model the second stage does not use the ASR output
  - calculates attention vectors directly on the first decoder state:
    $c_i^{trg} = \mathrm{Attention}([s_i^{trg} s_{1:N}^{src} ; \theta_{att}^{trg}])$
- keeps end-to-end trainability
- can also be trained with ASR and MT data

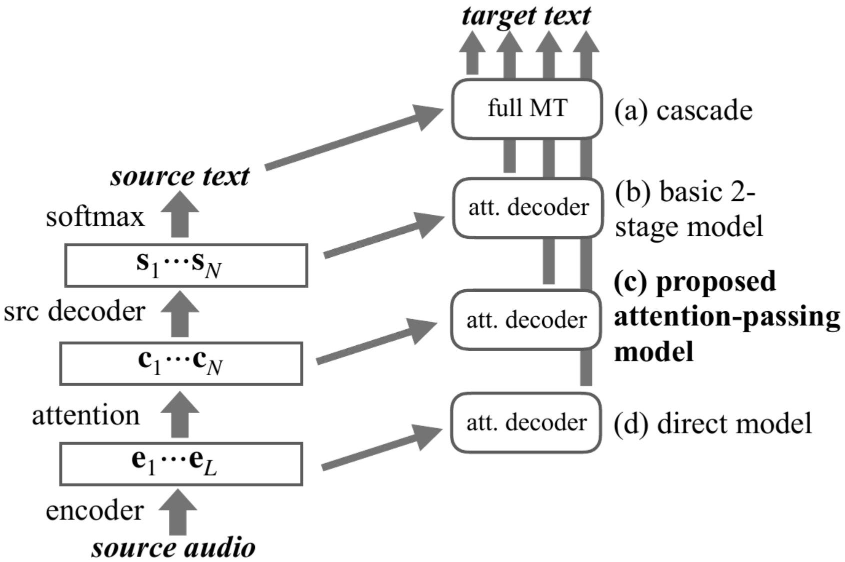
Section 4

Model comparison

Section 5

Attention passing model

Architecture comparision



Section 6

Performance

Bleu score

Datasets

Data efficiency

Section 7

Closing

Comments

- differences in architecture might make them less comparable

Related work

References