Learning to Segment Inputs for NMT Favors Character-Level Processing

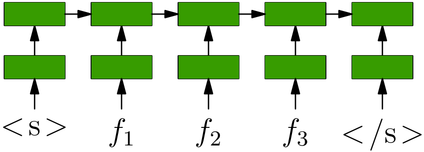
Julia Kreutzer   Artem Sokolov

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

amazon

IWSLT18

Encoder-Decoder Architecture (a sketch)

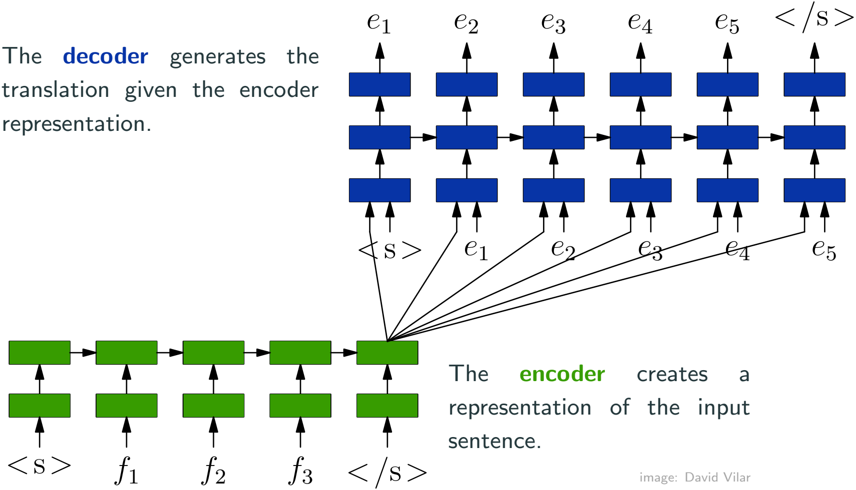


The **encoder** creates a representation of the input sentence.

image: David Vilar

Encoder-Decoder Architecture (a sketch)

The **decoder** generates the translation given the encoder representation.

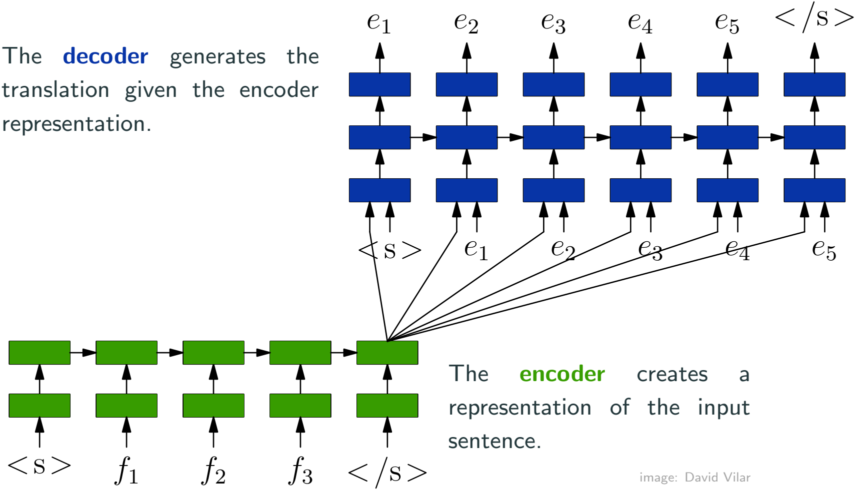


The **encoder** creates a representation of the input sentence.

image: David Vilar

Encoder-Decoder Architecture (a sketch)

The **decoder** generates the translation given the encoder representation.



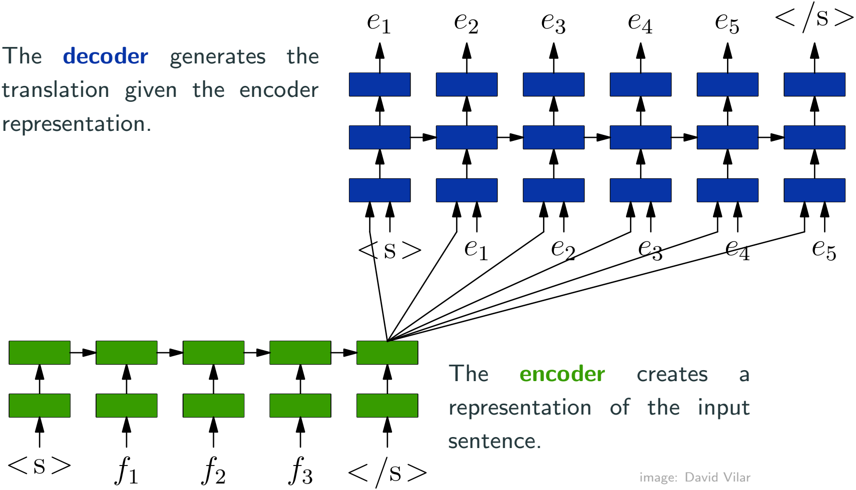
The **encoder** creates a representation of the input sentence.

image: David Vilar

Fixed input/output vocabularies are determined by pre-processing.

Encoder-Decoder Architecture (a sketch)

The **decoder** generates the translation given the encoder representation.



The **encoder** creates a representation of the input sentence.

image: David Vilar

Fixed input/output vocabularies are determined by pre-processing.

Have you ever wondered if is this optimal?
What if we optimize segmentations with the MT objective?

Why optimize segmentation for NMT?

Segmentation is the essential pre-processing step in NMT:

Why optimize segmentation for NMT?

Segmentation is the essential pre-processing step in NMT:

- Modeling: defining elementary units influences
  - sequence length
  - number of parameters
  - sparsity
  - computational costs of the output layer

Why optimize segmentation for NMT?

Segmentation is the essential pre-processing step in NMT:

- Modeling: defining elementary units influences
  - sequence length
  - number of parameters
  - sparsity
  - computational costs of the output layer
- Engineering:
  - Requires segmentation consistency in train/test
  - Aggravates "pipeline jungles" [Sculley et al., 2015]
  - Causes integration overhead

Prior work: sub-word NMT with BPE

State-of-the-art: **Byte-Pair Encoding (BPE)**
[Gage, 1994, Sennrich et al., 2016]

- Idea: merge most frequent sequences of characters
- Hyperparameter: number of merges
- Segmentation: static, variable length

```
This is a sentence split into B@@ PE@@ s.

don@@ au@@ dampf@@ schi@@ f@@ fahrts@@ gesellschaft@@ s@@
kapitä@@ n
```

Prior work: char-level

One can go deeper and work directly on characters:

Pros:

- No out-of-vocabulary words
- Might composes new words $\Rightarrow$ better generalization
- Tiny vocabulary $\Rightarrow$ fast output softmax
- Fewer parameters $\Rightarrow$ deeper models are possible
- No engineering hurdles

Cons:

- Longer sequences (speed, gradients)
- Partially loses attention interpretability
- Might compose nonsense words

Approaches to Character NMT

- **[Luong and Manning, 2016]**: hybrid for UNKs, training for 90d
- **[Chung et al., 2016]**: char-level RNN decoder, BPE RNN encoder
- **[Lee et al., 2017]**: CNN over input characters for speed
- **[Chung et al., 2017]**: hierarchical multi-scale RNNs

Can we do better?

So far: *fixed heuristics vs. going all the way down to character models*

Now, we want to let the model:

- Decide which segmentation is better for the task
- Change segmentation on the fly

Can we do better?

So far: *fixed heuristics vs. going all the way down to character models*

Now, we want to let the model:

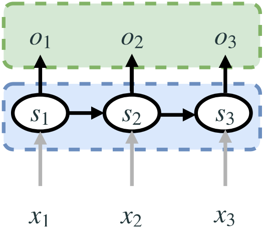
- Decide which segmentation is better for the task
- Change segmentation on the fly

Goals:

- Get a glimpse of what the optimal segmentation could look like
- Avoid manually solving the trade-offs of different segmentation

Adaptive Computation Time

General RNNs



- Fixed processing time per input x_t
- One output o_t for input x_t
- One state s_t per input x_t

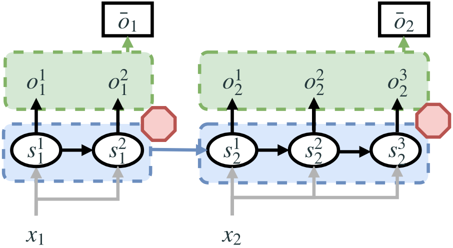
Adaptive Computation Time [Graves, 2016]

Idea: learn how much computation each input x_t needs
Procedure: keep processing the same input until a halt

Adaptive Computation Time [Graves, 2016]

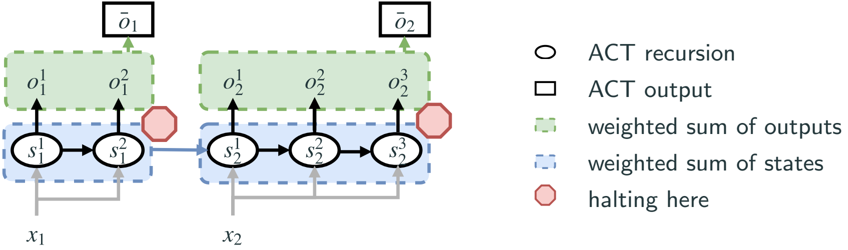
Idea: learn how much computation each input x_t needs

Procedure: keep processing the same input until a halt



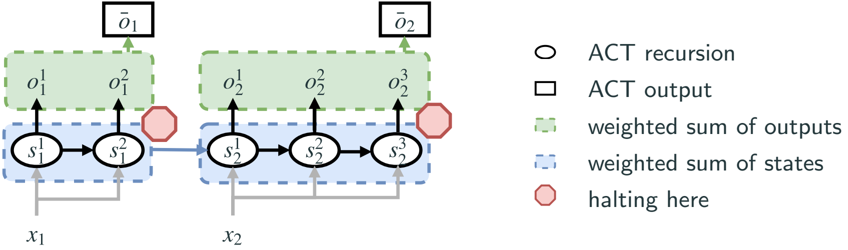
- ACT recursion
- ACT output
- weighted sum of outputs
- weighted sum of states
- halting here

ACT Halting Details



- Halting score in every RNN step: $h_t^n = \sigma(W_h s_t^n + b_h)$

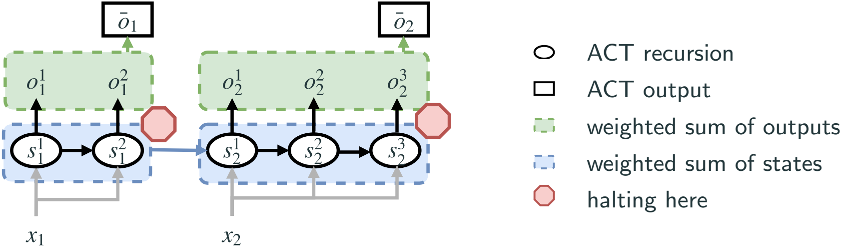
ACT Halting Details



- Halting score in every RNN step: $h_t^n = \sigma(W_h s_t^n + b_h)$
- Halting probability: $p_t^n = \begin{cases} R(t), & \text{if } n = N(t) \\ h_t^n, & \text{otherwise} \end{cases}$

  where $N(t) = \min\{n' = \sum_{n=1}^{n'} h_t^n \geq 1 - \epsilon\}$

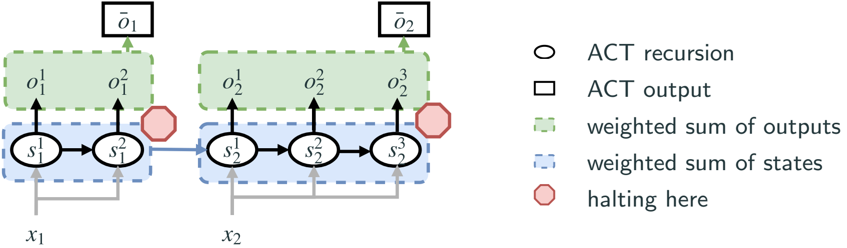
ACT Halting Details



- Halting score in every RNN step: $h_t^n = \sigma(W_h s_t^n + b_h)$
- Halting probability: $p_t^n = \begin{cases} R(t), & \text{if } n = N(t) \\ h_t^n, & \text{otherwise} \end{cases}$

  where $N(t) = \min\{n' = \sum_{n=1}^{n'} h_t^n \geq 1 - \epsilon\}$
- Remainder: $R(t) = 1 - \sum_{n'=1}^{N(t)-1} h_t^{n'}$

ACT Halting Details

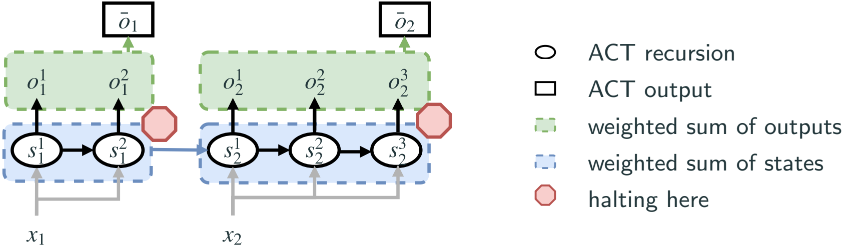


- Halting score in every RNN step: $h_t^n = \sigma(W_h s_t^n + b_h)$
- Halting probability: $p_t^n = \begin{cases} R(t), & \text{if } n = N(t) \\ h_t^n, & \text{otherwise} \end{cases}$

  where $N(t) = \min\{n' = \sum_{n=1}^{n'} h_t^n \geq 1 - \epsilon\}$
- Remainder: $R(t) = 1 - \sum_{n'=1}^{N(t)-1} h_t^{n'}$

Weighted average: weight outputs and states by p_t^n

ACT Halting Details



- Halting score in every RNN step: $h_t^n = \sigma(W_h s_t^n + b_h)$
- Halting probability: $p_t^n = \begin{cases} R(t), & \text{if } n = N(t) \\ h_t^n, & \text{otherwise} \end{cases}$

  where $N(t) = \min\{n' = \sum_{n=1}^{n'} h_t^n \geq 1 - \epsilon\}$
- Remainder: $R(t) = 1 - \sum_{n'=1}^{N(t)-1} h_t^{n'}$

Weighted average: weight outputs and states by p_t^n

Penalty: penalize too much pondering by adding $R(t)$ to loss

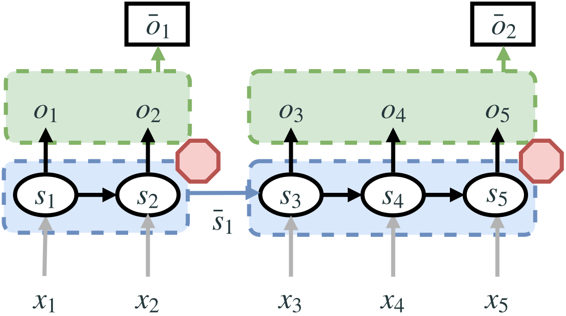
ACT Properties

Properties:

- Differentiable end-to-end
- No sampling and related variance problems
- Two hyperparameters:
  - halting probability threshold, ϵ
  - penalty weight in the loss, τ

ACT for Dynamic Segmentation

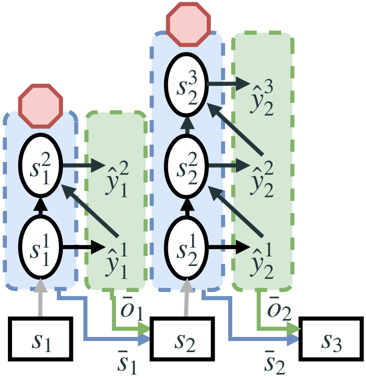
Encoder



ACT Encoding (input segmentation):

- **Inputs**: receive one character at a time
- **Halting**: indicate the end of a segment

Decoder



ACT Decoding (output segmentation):

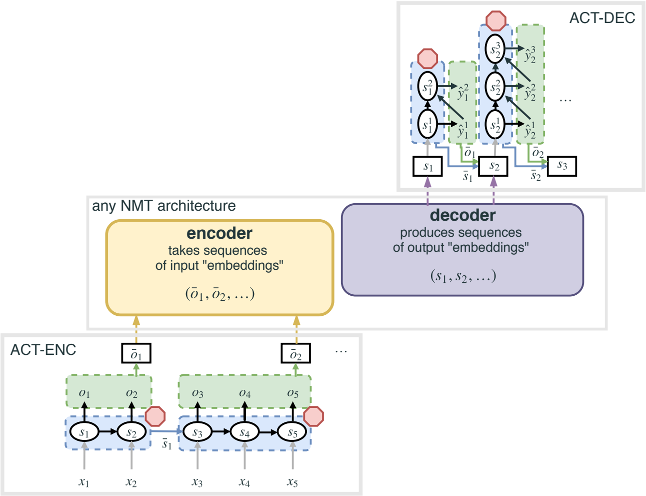
- **Outputs**: produce one character at a time
- **Halting**: indicate the end of a segment

Differences to the Original ACT

- Different purpose:
  - Segmentation vs. alignment of pondering time to input complexity
  - Learns aggregations of inputs vs. how much computation each input requires
- Different halting behavior:
  - Multiple halts per sequence vs. one per character
  - (Our decoder halts once per input, but generates many chars per input)

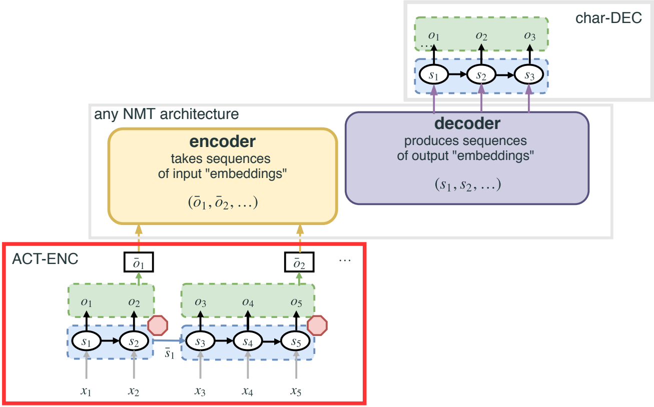
NMT Sandwich

- Segmenting encoder and decoder are “smart” embedding layers
- Any NMT architecture can be sandwiched in between



NMT Sandwich

- Segmenting encoder and decoder are "smart" embedding layers
- Any NMT architecture can be sandwiched in between
- In this work we chose basic RNN Groundhog



* In this work, experiments only with the encoder, decoder was a simple character-based. Similar results for segmenting on both sides.

Experiments

Implementation

Gluonhog: Implementation in MXNet's Gluon

- Dynamic computation graphs
- GRU encoder-decoder architecture
- Beam search from GluonNLP
- Implementation optimizations for mini-batches (see the paper)

Goal

Comparison to three levels:

1. **Word**: 30-32k most frequent words
2. **Sub-word**: 15k-32k most frequent BPEs, SPs
3. **Character**: 100-400 most frequent characters (incl. whitespace)

Questions:

- *What kind of segmentations does ACT learn?*
- *How do these models differ?*

Setup

Data	Domain	Languages	Train
IWSLT	TED talks	de-en	153k
CASIA	crawled web	zh-en	1M
ASPEC	scientific abstracts	ja-en	2M
WMT	news	fr-en	12M

Hyperparameters are mostly constant across models, except for:

- Vocabulary size & granularity
- ACT's τ and cell size (tuned)
- Encoder depth (tuned)

Results: one encoder layer

Data	Model	BLEU
IWSLT de-en	Word	22.11
	BPE	25.38
	Char	22.63
	ACT-ENC	22.67
CASIA zh-en	BPE	10.59
	Char	12.60
	ACT-ENC	9.87
ASPEC ja-en	WP	21.05
	Char	22.75
	ACT-ENC	15.82
WMT fr-en	Word	20.32
	BPE	27.02
	Char	24.25
	ACT-ENC	13.74

Results: one encoder layer

Data	Model	BLEU	Param
IWSLT de-en	Word	22.11	80.5M
	BPE	25.38	46.5M
	Char	22.63	13.4M
	ACT-ENC	22.67	13.5M
CASIA zh-en	BPE	10.59	49.9M
	Char	12.60	21.0M
	ACT-ENC	9.87	21.3M
ASPEC ja-en	WP	21.05	50.0M
	Char	22.75	15.6M
	ACT-ENC	15.82	15.6M
WMT fr-en	Word	20.32	80.5M
	BPE	27.02	86.0M
	Char	24.25	14.1M
	ACT-ENC	13.74	14.2M

Results: one encoder layer

Data	Model	BLEU	Param	SegLen
IWSLT de-en	Word	22.11	80.5M	4.66
	BPE	25.38	46.5M	4.09
	Char	22.63	13.4M	1.00
	ACT-ENC	22.67	13.5M	1.88
CASIA zh-en	BPE	10.59	49.9M	1.72
	Char	12.60	21.0M	1.00
	ACT-ENC	9.87	21.3M	1.006
ASPEC ja-en	WP	21.05	50.0M	2.07
	Char	22.75	15.6M	1.00
	ACT-ENC	15.82	15.6M	1.0007
WMT fr-en	Word	20.32	80.5M	5.19
	BPE	27.02	86.0M	4.05
	Char	24.25	14.1M	1.00
	ACT-ENC	13.74	14.2M	1.82

Results: one encoder layer

Data	Model	BLEU	Param	SegLen	TrainTime
IWSLT de-en	Word	22.11	80.5M	4.66	23h
	BPE	25.38	46.5M	4.09	20h
	Char	22.63	13.4M	1.00	1d22h
	ACT-ENC	22.67	13.5M	1.88	9d21h
CASIA zh-en	BPE	10.59	49.9M	1.72	18h
	Char	12.60	21.0M	1.00	10d6h
	ACT-ENC	9.87	21.3M	1.006	3d13h
ASPEC ja-en	WP	21.05	50.0M	2.07	4d4h
	Char	22.75	15.6M	1.00	24d15h
	ACT-ENC	15.82	15.6M	1.0007	15d4h
WMT fr-en	Word	20.32	80.5M	5.19	4d9h
	BPE	27.02	86.0M	4.05	3d23h
	Char	24.25	14.1M	1.00	9d
	ACT-ENC	13.74	14.2M	1.82	13d8h

Results: tuned number of encoder layers

Data	Model	BLEU
IWSLT de-en	Word, 4L	24.54
	BPE, 1L	25.38
	Char, 5L	**28.19**
	ACT-ENC, 3L	25.10
CASIA zh-en	BPE, 3L	11.01
	Char, 3L	**13.43**
	ACT-ENC, 2L	10.35
ASPEC ja-en	WP, 3L	22.02
	Char, 1L	**22.75**
	ACT-ENC, 1L	15.82
WMT fr-en	Word, 2L	21.04
	BPE, 3L	**27.93**
	Char, 6L	27.23
	ACT-ENC, 2L	14.01

Results: tuned number of encoder layers

Data	Model	BLEU	Param
IWSLT de-en	Word, 4L	24.54	97.0M
	BPE, 1L	25.38	46.5M
	Char, 5L	**28.19**	26.9M
	ACT-ENC, 3L	25.10	25.6M
CASIA zh-en	BPE, 3L	11.01	58.9M
	Char, 3L	**13.43**	30.0M
	ACT-ENC, 2L	10.35	21.3M
ASPEC ja-en	WP, 3L	22.02	61.4M
	Char, 1L	**22.75**	15.6M
	ACT-ENC, 1L	15.82	15.6M
WMT fr-en	Word, 2L	21.04	94.0M
	BPE, 3L	**27.93**	98.0M
	Char, 6L	27.23	27.6M
	ACT-ENC, 2L	14.01	21.7M

Results: tuned number of encoder layers

Data	Model	BLEU	Param	SegLen
IWSLT de-en	Word, 4L	24.54	97.0M	4.66
	BPE, 1L	25.38	46.5M	4.09
	Char, 5L	**28.19**	26.9M	1.00
	ACT-ENC, 3L	25.10	25.6M	1.31
CASIA zh-en	BPE, 3L	11.01	58.9M	1.72
	Char, 3L	**13.43**	30.0M	1.00
	ACT-ENC, 2L	10.35	21.3M	1.00
ASPEC ja-en	WP, 3L	22.02	61.4M	2.07
	Char, 1L	**22.75**	15.6M	1.00
	ACT-ENC, 1L	15.82	15.6M	1.0007
WMT fr-en	Word, 2L	21.04	94.0M	5.19
	BPE, 3L	**27.93**	98.0M	4.05
	Char, 6L	27.23	27.6M	1.00
	ACT-ENC, 2L	14.01	21.7M	1.0001

Results: tuned number of encoder layers

Data	Model	BLEU	Param	SegLen	TrainTime
IWSLT de-en	Word, 4L	24.54	97.0M	4.66	1d8h
	BPE, 1L	25.38	46.5M	4.09	20h
	Char, 5L	**28.19**	26.9M	1.00	3d10h
	ACT-ENC, 3L	25.10	25.6M	1.31	9d7h
CASIA zh-en	BPE, 3L	11.01	58.9M	1.72	24h
	Char, 3L	**13.43**	30.0M	1.00	5d6h
	ACT-ENC, 2L	10.35	21.3M	1.00	10d
ASPEC ja-en	WP, 3L	22.02	61.4M	2.07	4d2h
	Char, 1L	**22.75**	15.6M	1.00	24d15h
	ACT-ENC, 1L	15.82	15.6M	1.0007	15d4h
WMT fr-en	Word, 2L	21.04	94.0M	5.19	4d16h
	BPE, 3L	**27.93**	98.0M	4.05	5d3h
	Char, 6L	27.23	27.6M	1.00	18d13h
	ACT-ENC, 2L	14.01	21.7M	1.0001	9d10h

Analysis

Most Frequent Learned Segments

Data	Length	Segments
IWSLT	2	en; n␣; er; ␣d; ie; e␣; ei; in; ␣s; ␣w
	3	yst; -␣d; xtr; -␣u; 100; xpe; -␣w; xis; -␣e; - ge
	4	--␣d; --␣w; --␣s; --␣i; --␣e; --␣u; --␣g; --␣m; --␣a; --␣k
	5	1965␣; 969␣,; 1987␣; 1938␣; 1621␣; 1994␣; 1985␣; 1979␣; 1991␣; 1990e
CASIA	2	"。; "，; er; "他; --; "的; le; 明，; li; ut; ...
ASPEC	2	きる; きた; きな; きに; りん; きは; き，; きて ...
WMT	2	e␣; s␣; ␣d; t␣; ␣l; es; on; ␣a; de; en ...
	3	übe; Rüc; rüb; öve; ürs; Köp; üsl
	4	ümov; ölln; rüng; Jürg; ülle; Müsl Müni; üric; üdig; ...

- Segments are usually frequent or rare n-grams
- Some should be treated semantically as one unit

Example Translations (IWSLT and WMT)

<table>
<tr><td>Source</td><td>wir leben in einer zivilisation mit jet-lag , weltweiten reisen , nonstop-business und schichtarbeit .</td></tr>
<tr><td>Reference</td><td>we 're living in a culture of jet lag , global travel , 24-hour business , shift work .</td></tr>
<tr><td>Word</td><td>we live in a civilization with <unk> , global travel , <unk> and <unk> .</td></tr>
<tr><td>BPE</td><td>wir leben in einer zivilisation mit jet@@ -@@ lag , weltweiten reisen , non@@ sto@@ p-@@ business und sch@@ icht@@ arbeit .
we live in a civilization with a single , a variety of global travel , presidential labor and checking .</td></tr>
<tr><td>ACT-ENC</td><td>w|ir| l|eb|en| i|n |ei|ne|r |z|iv|il|is|at|io|n |m|it| j|et|-la|g |,| w|el|tw|ei|te|n |re|is|en| ,| n|on|st|op|-bu|si|ne|ss| u|nd| s|ch|ic|ht|ar|be|it| .|
we live in a civilization with jes lag , worldwide rows , nonstop business and failing .</td></tr>
<tr><td>Char</td><td>we live in a civilization with jet walk , global journeys , nonstop-business and layering</td></tr>
</table>

<table>
<tr><td>Source</td><td>Le clou du festival est formé de deux concerts orgnisés le 17 novembre .</td></tr>
<tr><td>Reference</td><td>The main focus of the festival is on two concerts taking place on November 17 .</td></tr>
<tr><td>Word</td><td>The <unk> of the festival is composed of two concerts on 17 November .</td></tr>
<tr><td>BPE</td><td>Le clo@@ u du festival est formé de deux concerts or@@ gn@@ isés le 17 novembre .
The festival 's bell is composed of two concerts , on 17 November .</td></tr>
<tr><td>ACT-ENC</td><td>L|e c|lo|u |du| f|es|ti|v|al| e|st| f|or|mé| d|e |de|ux| c|on|c|er|ts| o|rg|ni|sé|s |le| 1|7 |no|v|em|b|re| .|
The festival club is the form of two concerts organized on 17 November .</td></tr>
<tr><td>Char</td><td>The festival 's cloud is completed with two concerts organized on 17 November .</td></tr>
</table>

Observations

General:

- Char models outperform or are on par with BPE
- Can be trained in reasonable time even with RNNs
  (avg. 4x longer)

Observations

General:

- Char models outperform or are on par with BPE
- Can be trained in reasonable time even with RNNs
  (avg. 4x longer)

ACT-ENC:

- Converges to (almost) character segmentations
- The better the model, the closer segmentations are to characters
- Not surprising, since character models turned out to be the best
- Why couldn't match character models?:

Observations

General:

- Char models outperform or are on par with BPE
- Can be trained in reasonable time even with RNNs (avg. 4x longer)

ACT-ENC:

- Converges to (almost) character segmentations
- The better the model, the closer segmentations are to characters
- Not surprising, since character models turned out to be the best
- Why couldn't match character models?:
  - Uni-directionality of ACT-ENC vs. bi-directionality of other models
  - Char models with lots of non-linearities introduce optimization problems [Ling et al., 2015]
  - Causes premature convergence to a poorer local minima

Observations

General:

- Char models outperform or are on par with BPE
- Can be trained in reasonable time even with RNNs (avg. 4x longer)

ACT-ENC:

- Converges to (almost) character segmentations
- The better the model, the closer segmentations are to characters
- Not surprising, since character models turned out to be the best
- Why couldn't match character models?:
  - Uni-directionality of ACT-ENC vs. bi-directionality of other models
  - Char models with lots of non-linearities introduce optimization problems [Ling et al., 2015]
  - Causes premature convergence to a poorer local minima

Are character models already optimal?

Closer Look at Character Models

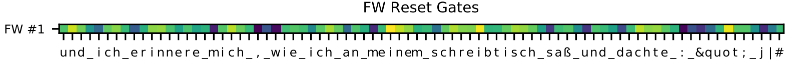
Understanding Character-Level Models

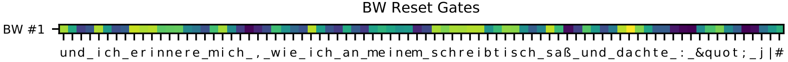
If ACT-ENC mostly prefers segmenting into characters, do character model posses segmenting capacity out of the box?

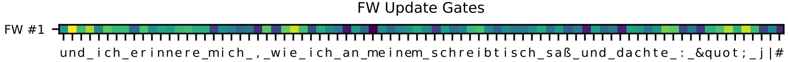
Methods:

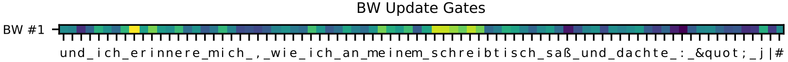
1. Visualizing **gate** state
   When do GRU gates open and close?
2. Visualizing **attention**
   Which inputs does the model attend to and when?

GRU Gates: Shallow



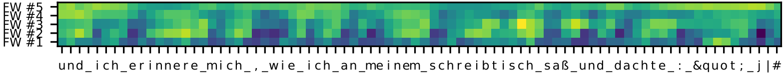




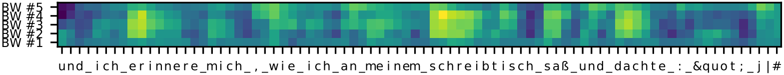


GRU Gates: Deep

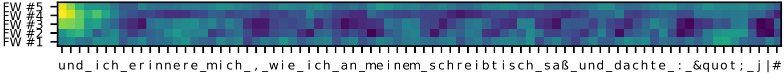
FW Reset Gates



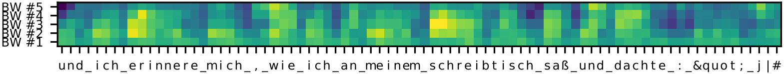
BW Reset Gates



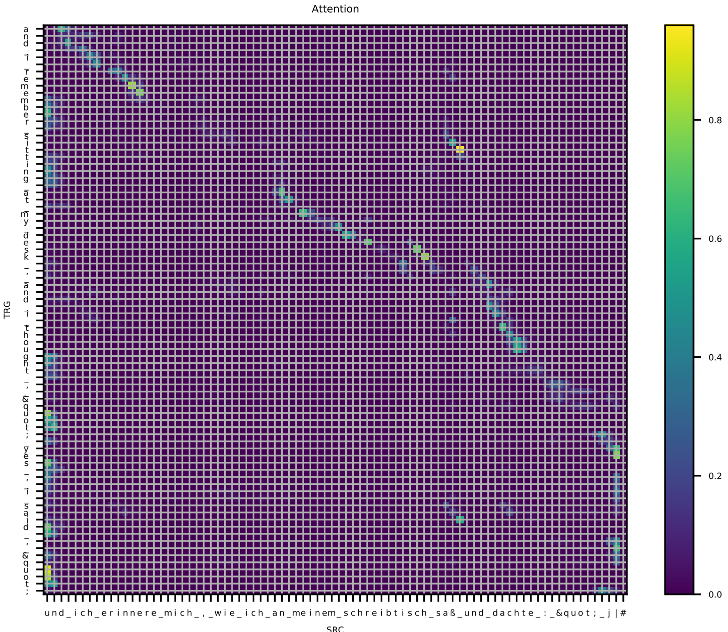
FW Update Gates



BW Update Gates



Character Attention: Deep



Discussion

Summary

1. ACT-ENC' end-to-end segmentation prefers **character segments**.

Summary

1. ACT-ENC' end-to-end segmentation prefers **character segments**.

2. Given the advantages of character models:
    - no pre-processing;
    - no additional hyperparameters;
    - improved robustness;
    - gated RNNs may be already capable of segmentation modeling,

   the explicit dynamic segmentation modeling may not be necessary.

Summary

1. ACT-ENC' end-to-end segmentation prefers **character segments**.

2. Given the advantages of character models:
    - no pre-processing;
    - no additional hyperparameters;
    - improved robustness;
    - gated RNNs may be already capable of segmentation modeling,

   the explicit dynamic segmentation modeling may not be necessary.

3. Rather, **more research should be put into character models**:
    - How to train with longer sequences for character models?
    - Which architectures work best for characters?
    - How to speed up training for character models?

Thanks

Thanks for your attention!

Fixed vs Dynamic Segmentation

Repeat-RNN baseline [Fojo et al., 2018]

- repeat each character a fixed number of times (tuned)
- uniform distribution of additional computation time
- no additional parameters
- no instabilities during training
- for synthetic tasks similar performance to ACT

$\Rightarrow$ Is it just the increased number of nonlinearities?

References I

Chung, J., Ahn, S., and Bengio, Y. (2017).
Hierarchical multiscale recurrent neural networks.
ICLR.

Chung, J., Cho, K., and Bengio, Y. (2016).
A character-level decoder without explicit segmentation for neural machine translation.
In *ACL*.

Fojo, D., Campos, V., and Giró-i Nieto, X. (2018).
Comparing fixed and adaptive computation time for recurrent neural networks.
In *Workshop Track of ICLR*.

Gage, P. (1994).
A new algorithm for data compression.
The C Users Journal, 12(2):23–38.

References II

Graves, A. (2016).
Adaptive computation time for recurrent neural networks.
In *arXiv:1603.08983*.

Lee, J., Cho, K., and Hofmann, T. (2017).
Fully character-level neural machine translation without explicit segmentation.
TACL, 5:365–378.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015).
Character-based neural machine translation.
In *arXiv:1511.04586*.

Luong, M.-T. and Manning, C. D. (2016).
Achieving open vocabulary neural machine translation with hybrid word-character models.
In *ACL*.

References III

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., et al. (2015).
Hidden technical debt in machine learning systems.
In *NIPS*.

Sennrich, R., Haddow, B., and Birch, A. (2016).
Neural machine translation of rare words with subword units.
In *ACL*.