Neural Architectures for Character-level NLP

Jindřich Libovický

February 28, 2022

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Outline

Why characters?

Neural String Edit Distance
- Model
- Cognate Detection
- Transliteration & Grapheme-to-Phoneme

Character-level Machine Translation
- Characters in literature and at WMT
- Curriculum Learning
- Architecture Innovations
- Architecture comparison

Why characters?

Tokenization examples

Different numbers of BPEs, fitted on WMT14 en-de data

plain text	`The cat sleeps on a mat.`
tokenization	`_The _cat _sleeps _on _a _mat .`
32k	`_The _cat _sle eps _on _a _mat .`
8k	`_The _c at _s le eps _on _a _m at .`
500	`_The _c at _s le ep s _on _a _m at .`
0	`_ T h e _ c a t _ s l e e p s _ o n _  a _ m a t .`

How could something like this work?

Motivation

Subwords are sort of ugly

`_The _c at _s le eps _on _a _m at .`

Removing assumptions usually helps in neural models in NLP, but...

subword segmentation seems to be a really good heuristic.

Wishful thinking: what we could get from the character-level

- Simpler processing pipelines
- Learn better segmentation
- Noise robustness
- Generalize towards morphology and domain-specific vocab

Char-level: Harsh reality

- Character sequences are long $\rightarrow$ computationally expensive
- Works well for non-contextual word-level tasks:
  transliteration, morphological inflection, ...
- For semantically heavy tasks (such as MT), worse performance than subwords
- Most of the assumed advantages (domain, morphology) are not real

Neural String Edit Distance

Neural String Edit Distance


arXiv:2104.08388v1 [cs.CL] 16 Apr 2021

Neural String Edit Distance

Jindřich Libovický and Alexander Fraser
Center for Information and Speech Processing
Ludwig Maximilian University of Munich
Munich, Germany
{libovicky, fraser}@cis.lmu.de

Abstract

We propose the neural string edit distance model for string-pair classif cation and sequence generation based on learned string edit distance. We modify the original expectation-maximization learned edit distance algorithm into a differentiable loss function, allowing us to integrate it into a neural network providing a contextual representation of the input. We test the method on cognate detection, transliteration, and grapheme-to-phoneme conversion. We show that we can trade off between performance and interpretability in a single framework. Using contextual representations, which are diff cult to interpret, we can match the performance of state-of-the-art string-pair classif cation models. Using static embeddings and a minor modif cation of the loss function, we can force interpretability, at the expense of an accuracy drop.

Figure 1: An example of applying the dynamic programming algorithm used to compute the edit probability score. It gradually f lls the table of probabilities that pref xes of the word "equal" transcribe into pref xes of phoneme sequence "IY K W AH L". The probability (gray circles) depends on the probabilities of the pref xes and probabilities of plausible edit operations: insert (blue arrows), substitute (green arrows) and delete (red arrows).

1 Introduction

State-of-the-art models for string-pair classif cation and sequence generation employ powerful neural architectures that lack interpretability. E.g., BERT (Devlin et al., 2019) internally compares all input symbols with each other via 96 attention heads, whose functions are diff cult to interpret. Moreover, attention itself can be hard to interpret (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019).

In many cases, such as in transliteration, a relation between two strings can be interpreted more simply as edit operations (Levenshtein, 1966). The edit operations def ne the alignment between the strings and provide an interpretation of how one string is transcribed into another. Learnable edit distance (Ristad and Yianilos, 1998) allows learn-

and Berrada, 2018; Hoover et al., 2020; Lipton, 2018), the restricted set of edit operations allows direct interpretation of the model operation. Unlike hard attention (Mnih et al., 2014; Indurthi et al., 2019) which also provides a discrete alignment between input and output, the edit distance explicitly says how the input symbols are processed. Also, unlike models like Levenshtein Transformer (Gu et al., 2019) which uses edit operation to model intermediate generation steps only within the target string, the learnable edit distance considers both source and target symbols to be a subject of the edit operations.

We reformulate the EM training used to train learnable edit distance as a differentiable loss function that can be used in a neural network. We pro-

Neural String Edit Distance


Jindřich Libovický


Alexander Fraser

Pre-print on arXiv. Rejected from EMNLP 2020, EACL 2021, ACL 2021, EMNLP 2021 and ACL 2022.

Neural String Edit Distance

Model

Black-box architectures vs. Levenshtein distance

- Char-level tasks use the same architectures as e.g, MT
- Overkill: large, hardly interpretable
- Levenshtein distance: transparent, interpretable...

...but weak and not flexible

We fix that!

Wagner-Fischer Algorithm

String s of length n, t of length m, d is a table of $m \times n$

```
for j in range(n):
    d[0, j] := j

  for j in range(n):
    for i in range(m):
      if s[i] = t[j]:
        subs_cost = 0
      else:
        subs_cost = 1

      d[i, j] = min(d[i-1, j] + 1,        # deletion
                    d[i, j-1] + 1,        # insertion
                    d[i-1, j-1] + subs_cost) # substitution

  return d[m, n]
```

Levenshtein Distance Example

Transcribe `kitten` to `sitting`

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

- empty string to empty string costs zero
- first column: empty string $\rightarrow$ `sitting`
- first row: delete `kitten`
- substring `kit` $\rightarrow$ `sittin`
  - we got rid of `ki` and have `sitti` – change `t` $\rightarrow$ `n`
    cost $4 + 1 = 5$
  - we have `sitin` and got rid of `ki` – delete `t`
    cost $5 + 1 = 6$
  - already got rid of `kit` and have `sitin` – add `n`
    cost $3 + 1 = 4 \leftarrow$ **minimum**

Problem of setting the operation costs

Transliteration from latin to cyrilics: `Praha` $\rightarrow$ Прага

- All characters are equivalent, but diffent UTF characters
- Either an expert can write the rules for the characer costs
- Or we can try to learn the weights from data

Learnable Edit Distance: (Ristad and Yianilos, 1998)

- Probabilistic formulation: one multimomial distribution over all possible operations
- Transcription probability (simple modification of the algorithm)
  - Best derivation: Product of most probable opeartion costs (replace min with max and sum log-probabilities)
  - All derivations: Replace max with sum

Expectation – Maximization

- Iterate over training data
- Do inference with curret model (data prob. increases)
- Get expected operation counts (forward-backward algorithm)

- Normalize the expected counts
- With new probability table probability of current derivations increases

More flexible: weights are estimated from the data

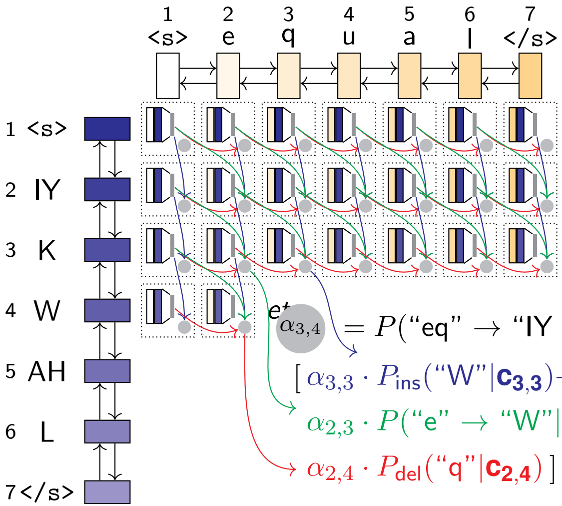
Rigid costs: do not depend on prefix or suffix

Main idea

Do the same thing...

...and *backpropagate* **the objective into a contextualized** *neural representation*.

Model



$$\alpha_{3,4} = P(\text{“eq”} \to \text{“IY K W”}) =$$
$$\big[\, \alpha_{3,3} \cdot P_{\text{ins}}(\text{“W”} \,|\, \mathbf{c_{3,3}}) +$$
$$\alpha_{2,3} \cdot P(\text{“e”} \to \text{“W”} \,|\, \mathbf{c_{2,3}}) +$$
$$\alpha_{2,4} \cdot P_{\text{del}}(\text{“q”} \,|\, \mathbf{c_{2,4}}) \,\big]$$

- Get contextualized representation of input charaters
- Symbol pairs: contatenate their representation and apply projection
- Estimate the insert, delete and substitute operations probabilites from these representations

Training

The original EM algorithm assumes a **discrete operation table**...
...but we have **continuous representations**.

- Expected distribution (forward-backard algorithm) – compared to actual distribution — optimize **KL divergence** between the predicted and expected distribution
- Directly optimize task-specific loss:
  - *String-pair classification*: optimize classification likelihood
  - *String transduction*: optimize output symbol negative log likelihood

Neural String Edit Distance

Cognate Detection

Task

For a pair of IPA strings...

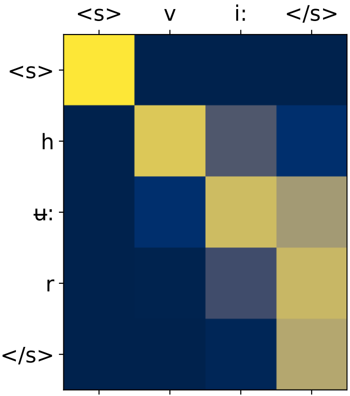
ˈzɛlɛniː	zɛˈɫɛnɪj	✓
ˈɦrubiː	pyknós	×
tu	tam	✓

...decide if they have the same diachronic origin.

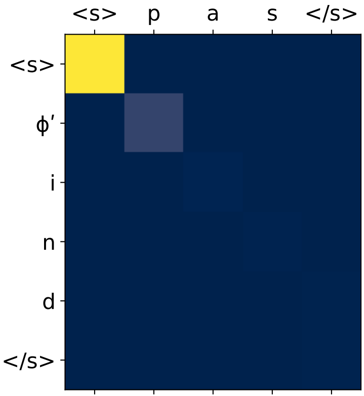
- Databases for Indo-European and Austro-Asiatic languages (Rama et al., 2018)
- Sampled positive and negative pairs, F1-measure for hits
- Use neural string edit distance to estimate the cognate probability

Example: Scores in the dynamic programing table

Cognate



Non-cognate



Results

Method		# Param.	Indo-European			Austro-Asiatic		
			Plain	+ Int. loss	Time	Plain	+ Int. loss	Time
Learnable edit distance		0.2M	32.8 ±1.8	—	0.4h	10.3 ±0.5	—	0.2h
Transformer [CLS]		2.7M	93.5 ±2.1	—	0.7h	78.5 ±0.8	—	0.6h
STANCE	unigram	0.5M	46.2 ±4.9	—	0.2h	16.6 ±0.3	—	0.1h
STANCE	RNN	1.9M	80.6 ±1.2	—	0.3h	15.9 ±0.2	—	0.2h
STANCE	Transformer	2.7M	76.7 ±1.3	—	0.3h	16.7 ±0.3	—	0.2h
ours	unigram	0.5M	78.5 ±1.0	80.1 ±0.8	1.5h	47.8 ±0.7	48.4 ±0.6	0.7h
ours	CNN (3-gram)	0.7M	94.0 ±0.7	93.9 ±0.8	0.9h	77.9 ±1.5	76.2 ±1.9	0.5h
ours	RNN	1.9M	96.9 ±0.6	**97.1 ±0.6**	1.9h	**84.0 ±0.4**	83.7 ±0.5	1.2h
ours	Transformer	2.7M	87.2 ±1.6	87.3 ±1.8	1.6h	69.9 ±1.0	70.7 ±1.1	1.0h

Ablation: IELex

Loss functions	F_1
Complete loss	97.1 $\pm$0.6
— binary XENT for $\alpha_{m,n}$	96.1 $\pm$0.3
— expectation-maximization	96.3 $\pm$0.7

Neural String Edit Distance

Transliteration & Grapheme-to-Phoneme

String Transduction Tasks

Arabic→English Transliteration

- 13k training, 1.5k validation and testing (Rosca and Breuel, 2016)

ساندي	sandy
دايي	daye
سارونى	saronni
أبركرمبي	abercromby
كورت	kurt

Grapheme-to-Phoneme Conversion

- CMUDict dataset (Weide, 2005)
- 108k training, 5k valid., 13k test
- Multiple transcriptions, during evaluation, choose the closest one

PERRON	P EH R AH N
TABUCHI	T AA B UW CH IY
CUVELIER	K Y UW V L IY ER
CONSUMERS'	K AH N S UW M ER Z
KINGDOMS	K IH NG D AH M Z

Evaluation with Word Error Rate (WER) and Character Error Rate (CER)

Model modifications

- Unidirectional representation of the target
- Deletion probability must not depend on the last target character
- Dirty trick: Added attention from the target representaiton to source representation

Results: Transliteration

	Method	# Param.	Arabic → English				
			Plain		+ Interpret. loss		Time
			CER	WER	CER	WER	
	RNN Seq2seq	3.3M	22.0 ±0.2	75.8 ±0.6	—	—	12m
	Transformer	3.1M	22.9 ±0.2	78.5 ±0.4	—	—	11m
ours	unigram	0.7M	31.7 ±1.8	85.2 ±0.9	31.2 ±1.4	85.0 ±0.5	36m
ours	CNN (3-gram)	1.1M	24.6 ±0.6	80.5 ±0.3	24.5 ±0.9	80.1 ±0.9	41m
ours	Deep CNN	3.0M	24.4 ±0.5	80.0 ±0.7	23.8 ±0.3	79.3 ±0.1	52m
ours	RNN	2.9M	24.1 ±0.2	77.0 ±2.0	22.0 ±0.3	77.4 ±0.8	60m
ours	Transformer	3.2M	24.3 ±0.9	79.0 ±0.7	23.9 ±1.6	78.6 ±1.3	1.2h

Results: Grapheme-To-Phoneme

	Method	# Param.	CMUDict Plain CER			+ Interpret. loss CER			Time
			CER	WER	Align.	CER	WER	Align.	
	RNN Seq2seq	3.3M	5.8 ±0.1	23.6 ±0.9	24.5	—	—	—	1.8h
	Transformer	3.1M	6.5 ±0.1	26.6 ±0.3	33.2	—	—	—	1.1h
ours	unigram	0.7M	20.9 ±0.3	67.5 ±1.0	55.7	20.6 ±0.3	66.3 ±0.2	59.5	2.4h
ours	CNN (3-gram)	1.1M	12.8 ±1.0	48.4 ±3.1	35.4	12.8 ±0.2	48.4 ±0.6	38.1	2.5h
ours	Deep CNN	3.0M	10.8 ±0.5	41.4 ±1.9	23.3	10.8 ±0.5	42.1 ±1.6	28.8	2.5h
ours	RNN	2.9M	7.8 ±0.3	31.9 ±1.3	44.7	7.3 ±0.4	33.3 ±1.5	48.9	2.3h
ours	Transformer	3.2M	10.7 ±1.0	41.8 ±3.1	33.3	10.2 ±1.1	43.6 ±3.2	37.9	2.3h

Example: Grapheme-to-Phoneme

graphemes	phonemes	edit operations
GOELLER	G OW L ER	G→G -O -E -L L→OW +L -E R→ER
VOGAN	V OW G AH N	V→V -O G→OW +G +AH -A N→N
ENDLER	EH N D L ER	+EH -E N→N D→D L→L -E R→ER
SWOOPED	S W UW P T	S→S W→W +UW -O -O P→P -E D→T

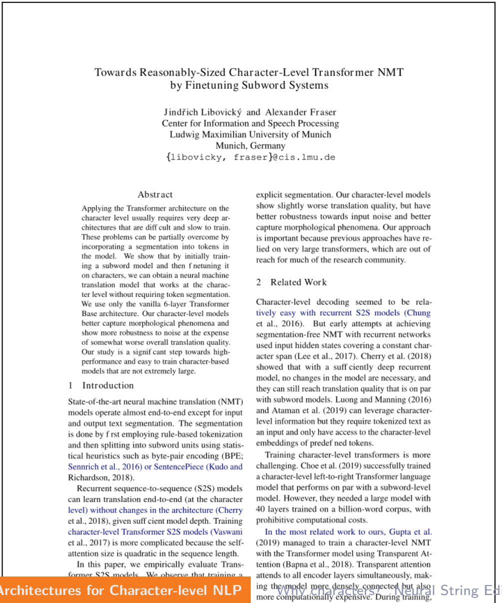
Viterbi decoding with a RNN-based model.

Ablation: RNN model on transliteration

Loss functions	CER	WER
Complete loss	22.5 ±0.3	77.4 ±0.8
— expectation maximization	68.2 ±7.4	93.5 ±1.0
— next symbol NLL	27.2 ±1.4	81.1 ±2.2
— $\alpha_{m,n}$ maximization	23.5 ±1.3	79.2 ±2.5

Character-level Machine Translation

Curriculum Learning for Character-level MT



Towards Reasonably-Sized Character-Level Transformer NMT
by Finetuning Subword Systems

Jindřich Libovický and Alexander Fraser
Center for Information and Speech Processing
Ludwig Maximilian University of Munich
Munich, Germany
{libovicky, fraser}@cis.lmu.de

Abstract

Applying the Transformer architecture on the character level usually requires very deep architectures that are diff cult and slow to train. These problems can be partially overcome by incorporating a segmentation into tokens in the model. We show that by initially training a subword model and then f netuning it on characters, we can obtain a neural machine translation model that works at the character level without requiring token segmentation. We use only the vanilla 6-layer Transformer Base architecture. Our character-level models better capture morphological phenomena and show more robustness to noise at the expense of somewhat worse overall translation quality. Our study is a signif cant step towards high-performance and easy to train character-based models that are not extremely large.

1 Introduction

State-of-the-art neural machine translation (NMT) models operate almost end-to-end except for input and output text segmentation. The segmentation is done by f rst employing rule-based tokenization and then splitting into subword units using statistical heuristics such as byte-pair encoding (BPE; Sennrich et al., 2016) or SentencePiece (Kudo and Richardson, 2018).

Recurrent sequence-to-sequence (S2S) models can learn translation end-to-end (at the character level) without changes in the architecture (Cherry et al., 2018), given suff cient model depth. Training character-level Transformer S2S models (Vaswani et al., 2017) is more complicated because the self-attention size is quadratic in the sequence length.

In this paper, we empirically evaluate Transformer S2S models. We observe that training a

explicit segmentation. Our character-level models show slightly worse translation quality, but have better robustness towards input noise and better capture morphological phenomena. Our approach is important because previous approaches have relied on very large transformers, which are out of reach for much of the research community.

2 Related Work

Character-level decoding seemed to be relatively easy with recurrent S2S models (Chung et al., 2016). But early attempts at achieving segmentation-free NMT with recurrent networks used input hidden states covering a constant character span (Lee et al., 2017). Cherry et al. (2018) showed that with a suff ciently deep recurrent model, no changes in the model are necessary, and they can still reach translation quality that is on par with subword models. Luong and Manning (2016) and Ataman et al. (2019) can leverage character-level information but they require tokenized text as an input and only have access to the character-level embeddings of predef ned tokens.

Training character-level transformers is more challenging. Choe et al. (2019) successfully trained a character-level left-to-right Transformer language model that performs on par with a subword-level model. However, they needed a large model with 40 layers trained on a billion-word corpus, with prohibitive computational costs.

In the most related work to ours, Gupta et al. (2019) managed to train a character-level NMT with the Transformer model using Transparent Attention (Bapna et al., 2018). Transparent attention attends to all encoder layers simultaneously, making the model more densely connected but also more computationally expensive. During training,

Towards Reasonably-Sized Character-Level Transformer NMT by Finetuning Subword Models

Jindřich Libovický

Alexander Fraser

Short paper at EMNLP 2020.

Meta-survey and experiments with architectures

Why don't people use character-level machine translation?

Anonymous ACL submission

Abstract

We present a literature and empirical survey that critically assesses the state of the art in character-level modeling for machine translation (MT). Despite evidence in the literature that character-level systems are comparable with subword systems, they are virtually never used in competitive setups in WMT competitions. We empirically show that even with recent modeling innovations in character-level natural language processing, character-level MT systems still struggle to match their subword-based counterparts. Character-level MT systems show neither better domain robustness, nor better morphological generalization, despite being often so motivated. However, we are able to show robustness towards source side noise and that translation quality does not degrade with increasing beam size at decoding time.

1 Introduction

The progress in natural language processing (NLP) brought by deep learning is often narrated as removing assumptions about the input data and letting the models learn everything end-to-end. One of the assumptions about input data that seems to resist this trend is (at least partially) linguistically motivated segmentation of input data in machine translation (MT) and NLP in general.

For NMT, several papers have claimed parity of character-based methods with subword models, highlighting advantageous features of such systems. Very recent examples include Gao et al. (2020); Banar et al. (2020); Li et al. (2021). Despite this, character-level methods are rarely used as strong baselines in research papers and shared task submissions, suggesting that character-level models input segmentation methods used in WMT shared task submissions. We then systematically compare the most recent character-processing architectures, some of them taken from general NLP research and used for the first time in MT. Further, we propose an alternative two-step decoder architecture that unlike standard decoders does not suffer from a slow-down due to the length of character sequences. Following the recent findings on MT decoding, we evaluate different decoding strategies in the character-level context.

Many previous studies on character-level MT drew their conclusions from experiments on rather small datasets and focused only on quantitatively assessed translation quality without further analysis. To compensate for this, we revisit and systematically evaluate the state-of-the-art approaches to character-level neural MT and identify their major strengths and weaknesses on large datasets.

2 Character-Level Neural MT

The original sequence-to-sequence models used word-based vocabularies of a limited size and thus relatively frequent occurrence of out-of-vocabulary tokens. A typical solution to that problem is subword segmentation (Sennrich et al., 2016; Kudo and Richardson, 2018), which keeps frequent tokens intact and splits less frequent ones into smaller units.

Modeling language on the character level is attractive because it can help overcome several problems of subword models. One-hot representations of words or subwords do not reflect systematic character-level relations between words, potentially harming morphologically rich languages. With subwords, minor typos on the source side lead to radically different input representations resulting in low robustness towards source-side noise (Provilkov et al., 2020; Libovický and Fraser, 2020).

Why don't people use character-level machine translation?

Jindřich Libovický

Helmut Schmid

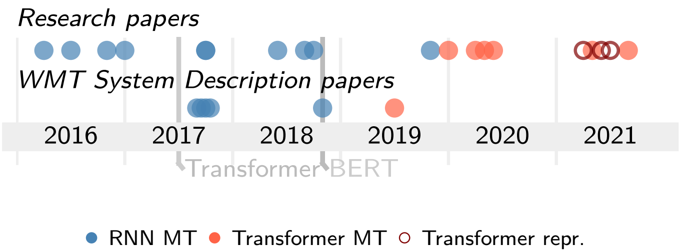
Alexander Fraser

Accepted to Findings of ACL 2022.

Character-level Machine Translation

Characters in literature and at WMT

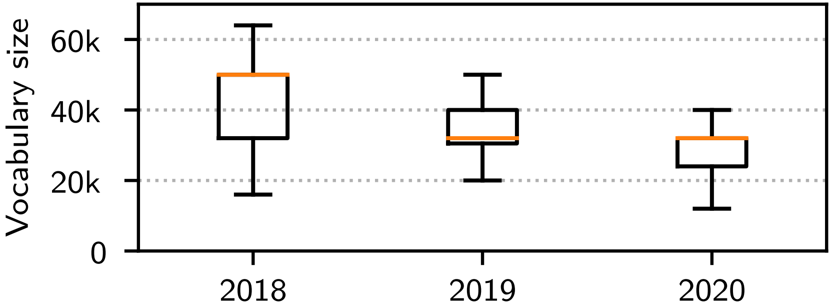
Character-level MT in time



- Research papers often report parity or outperforming subwords
- The results of research papers got never confirmed in the competitive WMT setup
- Suspected reasons: worse quality, 5–6× slower

	2018	2019	2020
Subwords	92%	93%	97%
Morphological	4%	2%	3%
Words	2%	3%	—
Character	2%	2%	—

Vocabulary size in WMT submissions



Decreases in time because of low-resource languages...

Character-level Machine Translation

Curriculum Learning

Main Idea

1. Train a subword model first
   ...so the model knows what words are

2. Finetune it to only use characters

Language Pairs

English ↔ German

- WMT14 data, 4.5M training sentences
- Both Germanic languages, German with more inflection than English

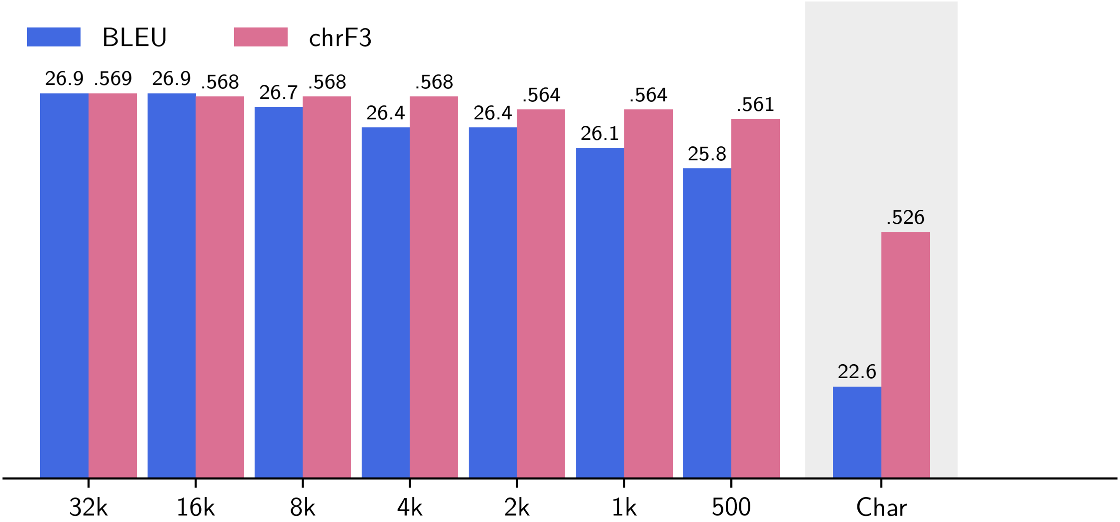
English → Czech

- CzEng 1.7, 15.8M training sentences, tested on WTM18
- Slavic (but still Indo-European) language, rich morphological inflection

English → Turkish

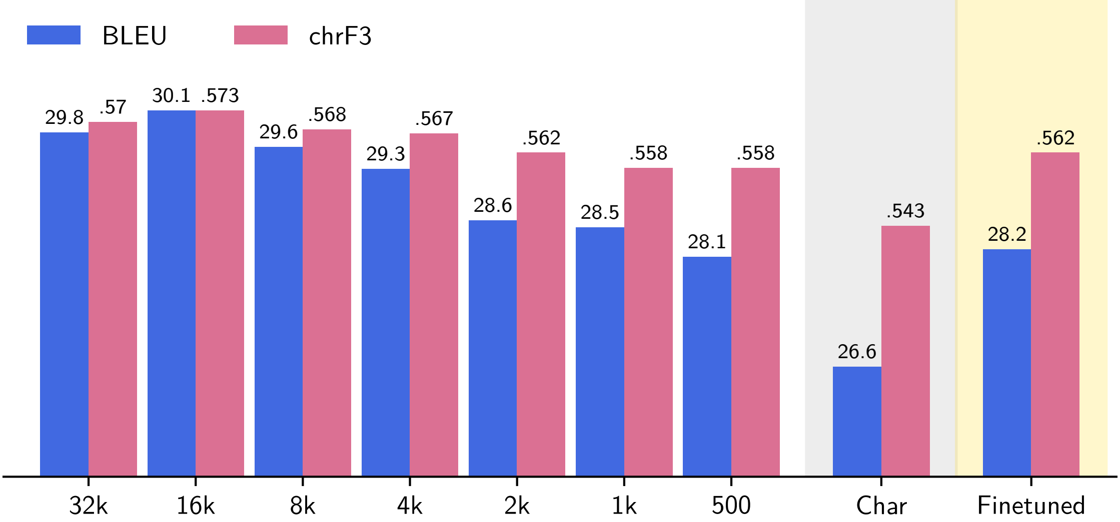
- SETMITES2, 207k training sentences, tested on WMT18
- Turkic language, agglutinative morphology

Results: English → German

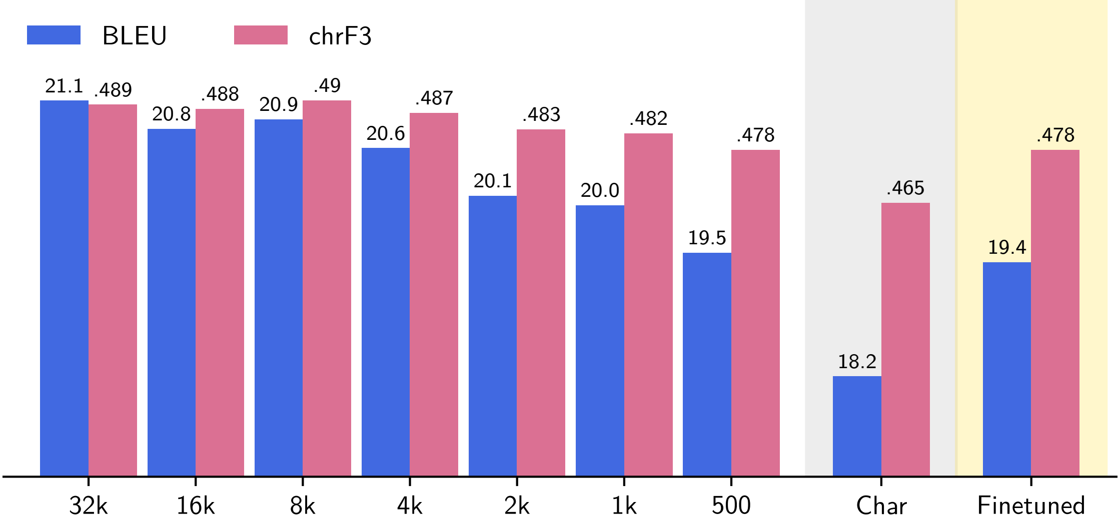


BLEU chrF3

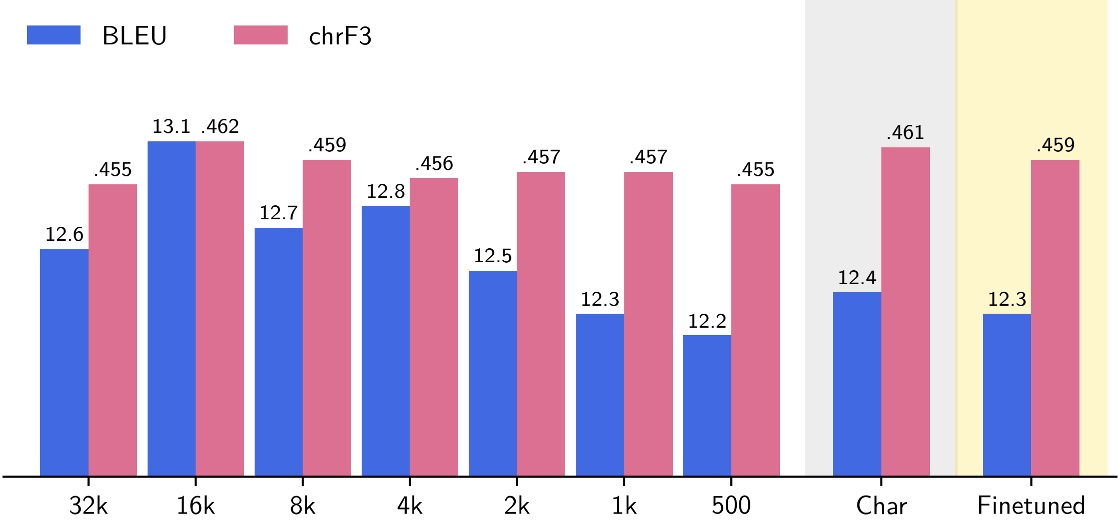
Results: German → English



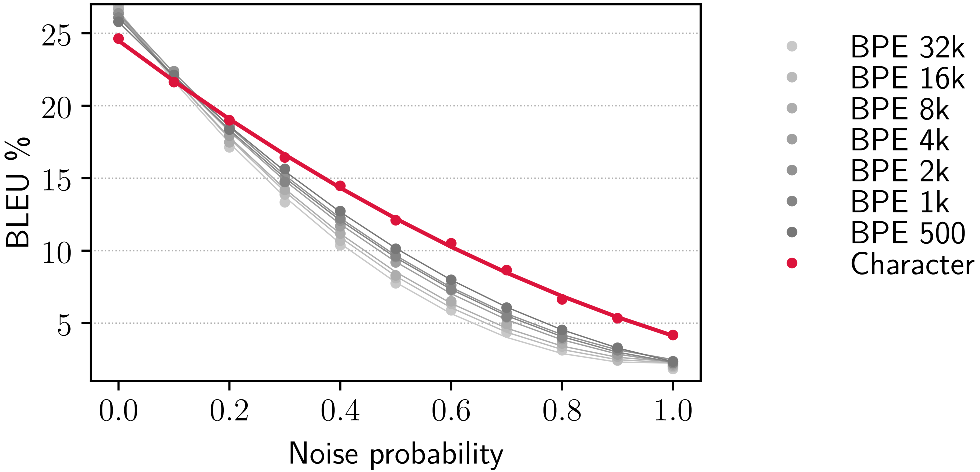
Results: English → Czech



Results: English → Turkish



Noise sensitivity



Semantically rich units are crutial during training

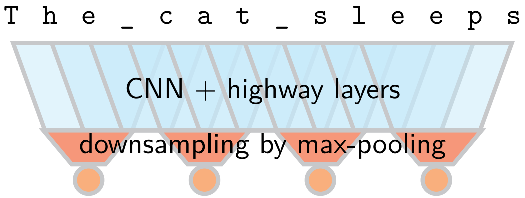
(can be unlearned eventually)

If we want to avoid subwords, we should ge such units from characters.

Character-level Machine Translation

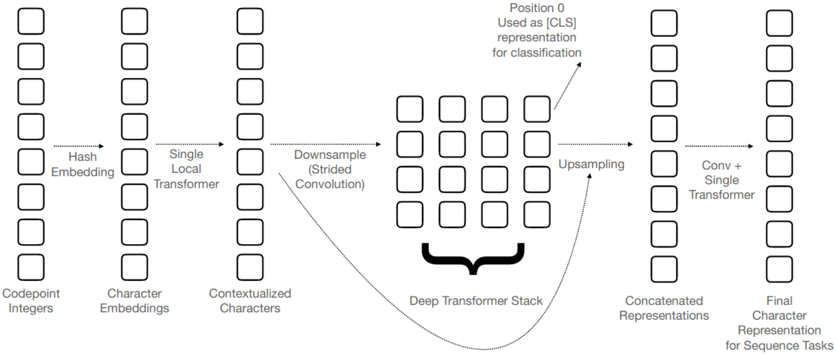
Architecture Innovations

Existing Character-Level Architectures: Convolutional



- Long character-level sequence to word-like units (Lee et al., 2017)
- Convolutions of different kernel sizes, highway layers, max-pooling
- Succesfully used with RNNs and matched BPE performance (not with Transformers)

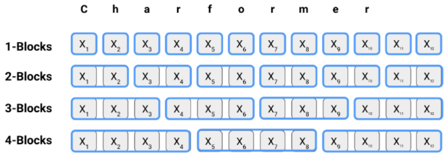
Existing Character-Level Architectures: CANINE



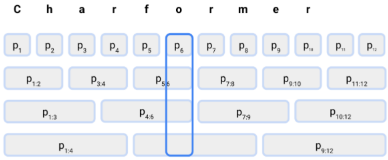
Source: Clark et al. (2021); from Google, pre-print only

- Architecture for pre-trained BERT-like (encoder-only) models; strong multilingual capabilities
- Similar with more modern building blocks

Existing Character-Level Architectures: Charformer



(a) Formation of subword blocks to be scored by F_R. Offsets and/or pre-GBST convolutions not shown.

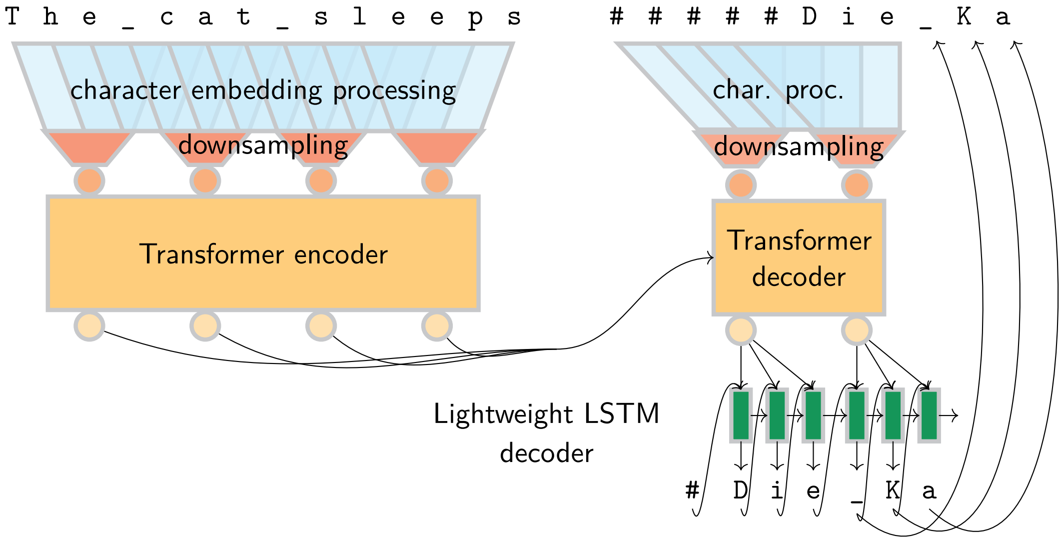


(b) Block scores that have been expanded back to length L. Softmax is taken over block scores at each position i to form block weights for constructing latent subword representations.

Source: Tay et al. (2021); from Google, to appear to ICLR'22

- Parameter-efficient: One convolution and a lot of averaging
- Also for pre-trained representations (encoder-only), matches subwords

State-shrinking and Two-step decoding



Experiment Setup

- IWSLT'14 data for: English $\leftrightarrow$ { Arabic, French, German }
- 200k training sentences, 1.5k validation and test
- Our own implementation in PyTorch

Result: BLEU

Model	Enc. downsample	Dec. downsample	Char. proc. params	From English ar BLEU	From English de BLEU	From English fr BLEU	Into English ar BLEU	Into English de BLEU	Into English fr BLEU
BPE 16k			16516	11.2 ±0.2	27.7 ±0.3	36.4 ±0.3	29.7 ±0.2	31.6 ±0.3	36.2 ±0.3
Vanilla char.			658	13.5 ±0.4	25.6 ±0.7	34.6 ±0.7	27.7 ±0.8	29.4 ±0.7	34.7 ±0.4
Lee-style	3	—	9672	13.1 ±0.5	25.9 ±0.7	35.2 ±0.4	28.0 ±0.4	30.2 ±0.5	35.3 ±0.2
	5	—	9672	12.5 ±0.1	25.0 ±0.4	33.2 ±0.1	24.9 ±4.4	28.9 ±0.3	34.4 ±0.3
	3	3	9646	11.0 ±0.2	23.4 ±0.4	31.7 ±0.5	25.6 ±0.3	28.0 ±0.3	33.3 ±0.4
	5	5	9646	9.4 ±0.5	21.8 ±0.3	28.7 ±1.7	23.7 ±0.3	25.5 ±0.3	30.9 ±0.5
Charformer	3	—	1320	13.3 ±0.3	25.9 ±0.5	32.9 ±0.3	27.3 ±0.5	29.9 ±0.3	35.1 ±0.3
	5	—	1320	12.2 ±0.3	24.2 ±0.6	31.3 ±0.4	25.1 ±0.6	28.1 ±0.4	33.7 ±0.2
	3	3	1165	10.3 ±0.5	23.2 ±0.5	30.6 ±0.4	24.5 ±0.4	27.5 ±0.5	32.6 ±0.3
	5	5	1165	8.4 ±0.2	19.9 ±0.2	27.4 ±0.7	18.4 ±3.1	23.5 ±0.5	29.2 ±0.7
Canine	3	—	6446	12.6 ±0.3	25.4 ±0.5	33.2 ±0.6	26.1 ±0.5	29.1 ±0.4	34.5 ±0.4
	5	—	7470	11.2 ±0.2	22.5 ±0.4	30.5 ±0.5	22.1 ±0.6	27.3 ±0.3	32.5 ±0.5
	3	3	6291	10.3 ±0.5	22.4 ±0.3	30.2 ±0.5	23.7 ±0.9	25.9 ±1.0	32.5 ±0.3
	5	5	7444	6.9 ±0.4	19.1 ±0.4	27.9 ±0.6	15.4 ±0.3	23.2 ±0.2	27.9 ±0.6

chrF and COMET are consistent (in the paper)

- Two-step architecture as fast as BPEs, but low quality
- Downsampling 3 much better than downsampling 5

BPE > Lee-style > Vanilla char. > everything else

Takeaways

- Lee-style encoder works the best
- Two-step decoding matches the speed, but worse quality

Let's try the best option in a competitive setup.

Character-level Machine Translation

Architecture comparison

Competitive data setup

Previous work makes optimistic conclusions based on small and old datasets...
...let's do it properly

 → 

- CzEng 2.0 corpus (Kocmi et al., 2020)
- 61M authentic parallel sentences
  50M back-translated

 → 

- Data mix Edinburgh used for WMT'21 submission (Chen et al., 2021)
- 66M authentic parallel sentence
  52M back-translated

...data almost comparable to best WMT submissions
(tagged back-translation, Transformer BIG architecture, FairSeq)

Extensive evaluation

Character-level methods often motivated by morphological generalization and noise robustness.

- Quality: BLEU, chrF, COMET in News, IT and medical domain
- Gender dataset
- Morpheval: Specific morphological phenomena
- Recall of novel forms and lemmas (in news)
- Quality under sampled noise

Results: English-Czech

	News		IT		Medical		Gender Acc.	Avg. Mor-pheval	Recall of novel		Noisy set chrF
	BLEU	COMET	BLEU	COMET	BLEU	COMET			Forms	Lemmas	
BPE 16k	30.8 ±0.8	.672 ±.022	34.5 ±1.3	.889 ±.022	26.4 ±1.4	.734 ±.037	71.3	86.6	33.7 vs. 63.7	48.5 vs. 71.1	.436 ±.002
BPE to char.	28.4 ±0.8	.597 ±.024	31.4 ±1.2	.821 ±.025	23.6 ±1.3	.674 ±.039	68.9	87.0	34.3 vs.	47.4 vs.	.436 ±.001
Vanilla char.	27.7 ±0.7	.550 ±.026	30.0 ±1.2	.778 ±.028	23.3 ±1.3	.663 ±.039	70.2	86.4	34.4 vs. 61.0	47.4 vs. 68.7	.493 ±.001
Lee-style enc.	28.8 ±0.8	.609 ±.024	31.7 ±1.3	.849 ±.024	24.3 ±1.3	.696 ±.038	65.6	86.6	34.1 vs. 61.7	48.5 vs. 69.2	.497 ±.001

- Characters worse in everything, incl. domain robustness
- No signs of better morphological generalization
- Strictly better for noisy inputs

Results: English-German

	News		IT		Medical		Gender Acc.	Avg. Mor-pheval	Recall of novel		Noisy set chrF
	BLEU	COMET	BLEU	COMET	BLEU	COMET			Forms	Lemmas	
BPE 16k	31.5 ±0.9	.418 ±.021	45.6 ±1.3	.622 ±.021	38.7 ±1.6	.569 ±.034	66.5	90.6	40.2 vs. 72.3	51.0 vs. 67.0	.464 ±.002
BPE to char.	29.1 ±0.8	.360 ±.022	46.5 ±1.3	.617 ±.021	36.0 ±1.4	.513 ±.035	71.2	91.3	45.1 vs. 71.1	50.8 vs. 65.5	.465
Vanilla char.	27.8 ±0.8	.321 ±.023	45.3 ±1.3	.600 ±.022	35.6 ±1.4	.496 ±.036	71.2	91.4	50.7 vs. 64.3	45.1 vs. 70.2	.504 ±.001
Lee-style enc.	29.1 ±0.8	.363 ±.022	46.5 ±1.3	.619 ±.022	36.5 ±1.4	.500 ±.037	74.0	91.5	44.5 vs. 77.1	50.8 vs. 65.5	.515 ±.001

- Same results as for Czech
- Slightly better morphological generalization

Takeways

The only thing where characters are better is noise robustness

Summary

- Neural String Edit Distance can be a viable alternative for character-level tasks
- Machine translation needs semantically rich units and shorter sequences
- Modern architectures that work elsewhere are to weak for MT
- The best character-level architecture: Lee-style encoding
- Many research papers about character-level MT tend to overclaim
- The only advantage of character-level: noise robustness

https://ufal.mff.cuni.cz/jindrich-libovicky

References I

Pinzhen Chen, Jindřich Helcl, Ulrich Germann, Laurie Burchell, Nikolay Bogoychev, Antonio Valerio Miceli Barone, Jonas Waldendorf, Alexandra Birch, and Kenneth Heafield. The University of Edinburgh's English-German and English-Hausa submissions to the WMT21 news translation task. In *Proceedings of the Sixth Conference on Machine Translation*. Association for Computational Linguistics, 2021.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. CANINE: pre-training an efficient tokenization-free encoder for language representation. *CoRR*, abs/2103.06874, 2021. URL https://arxiv.org/abs/2103.06874.

Tom Kocmi, Martin Popel, and Ondřej Bojar. Announcing czeng 2.0 parallel corpus with over 2 gigawords. *CoRR*, abs/2007.03006, 2020. URL https://arxiv.org/abs/2007.03006.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017. doi: 10.1162/tacl_a_00067. URL https://aclanthology.org/Q17-1026.

Taraka Rama, Johann-Mattis List, Johannes Wahle, and Gerhard Jäger. Are automatic methods for cognate detection good enough for phylogenetic reconstruction in historical linguistics? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 393–400, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2063. URL https://aclanthology.org/N18-2063.

Eric Sven Ristad and Peter N. Yianilos. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532, 1998. doi: 10.1109/34.682181. URL https://doi.org/10.1109/34.682181.

Mihaela Rosca and Thomas Breuel. Sequence-to-sequence neural network models for transliteration. *CoRR*, abs/1610.09565, 2016. URL http://arxiv.org/abs/1610.09565.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Prakash Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. *CoRR*, abs/2106.12672, 2021. URL https://arxiv.org/abs/2106.12672.

Robert Weide. The Carnegie-Mellon pronouncing dictionary [cmudict. 0.7]. Pittsburgh, PA, USA, 2005. Carnegie Mellon University. URL http://www.speech.cs.cmu.edu/cgi-bin/cmudict.