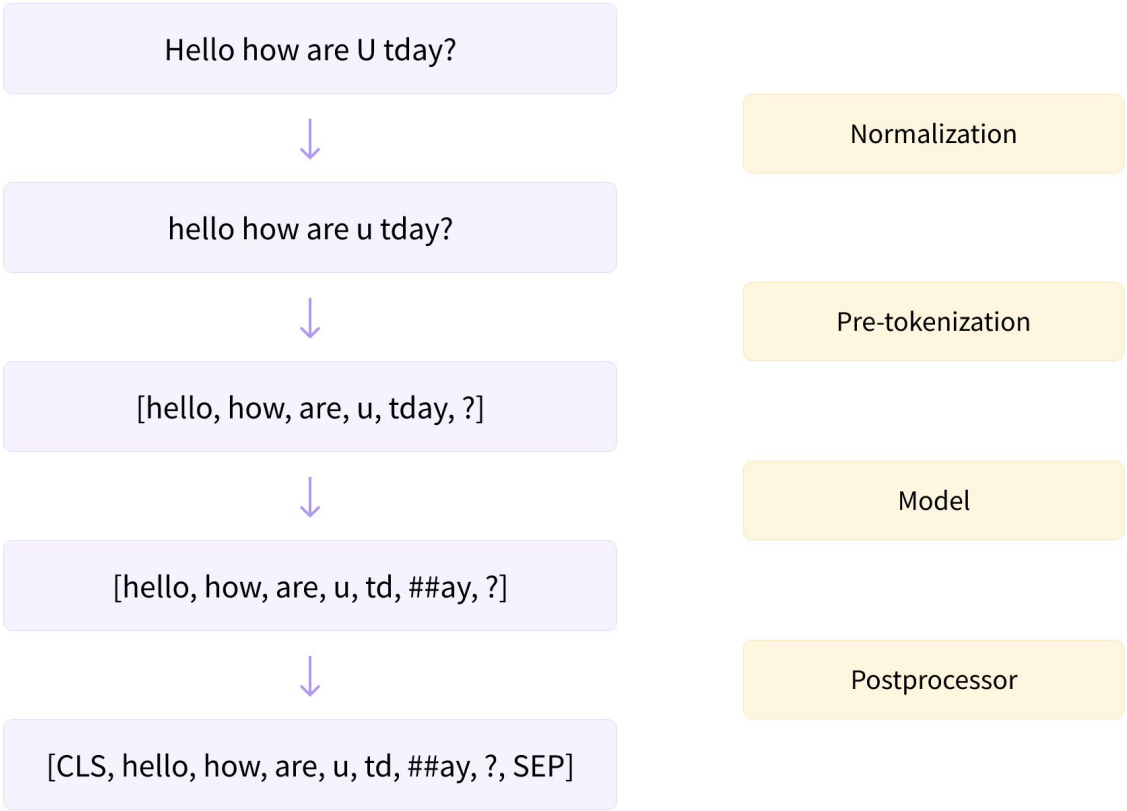
CHARFORMER: FAST CHARACTER TRANSFORMERS VIA GRADIENT-BASED SUBWORD TOKENIZATION

Николаев Максим, Кондратьев Захар и Волгин Даниил

Предобработка текстов



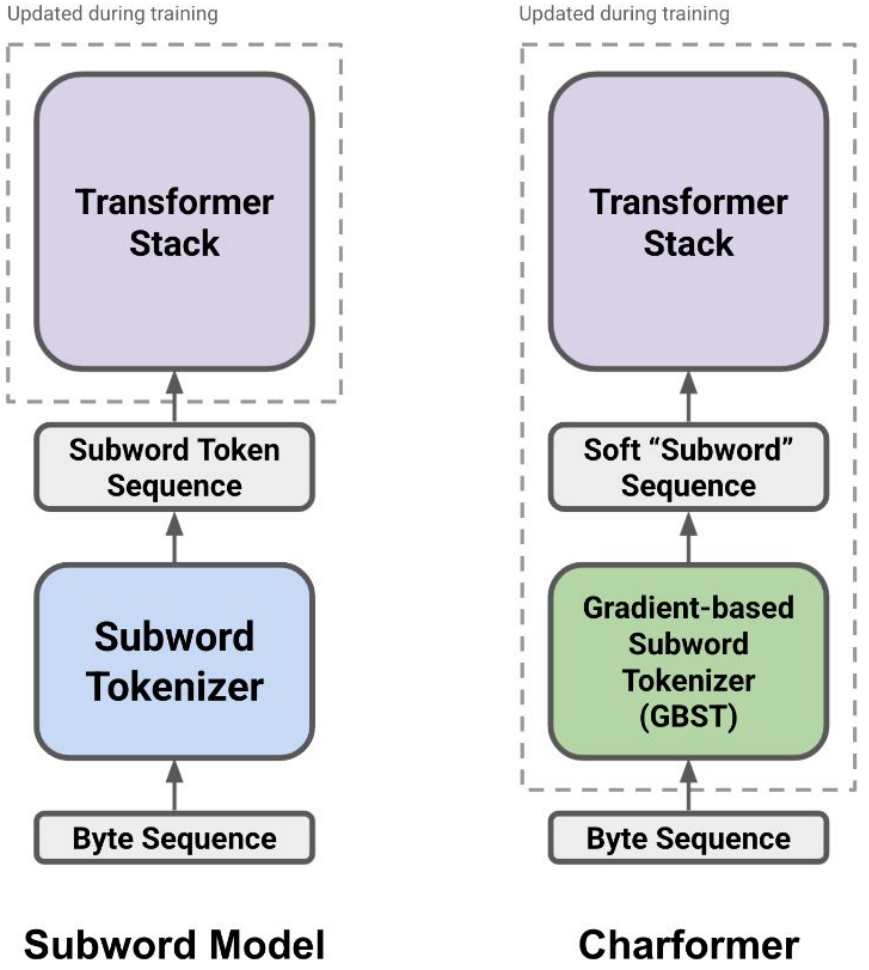
Проблемы токенайзеров

- Разделение на токены исключительно по частоте, без учёта лексики и семантики.
- Некоторые языки не имеют разделения на слова.
- Проблема многоязычных моделей, когда один язык имеет больше токенов.
- Распределение слов на предобучении не совпадает с распределением слов на файнтюнинге.

Итог: предобработка текста превращается в сложную инженерную задачу для получения хорошего качества.

CHARFORMER

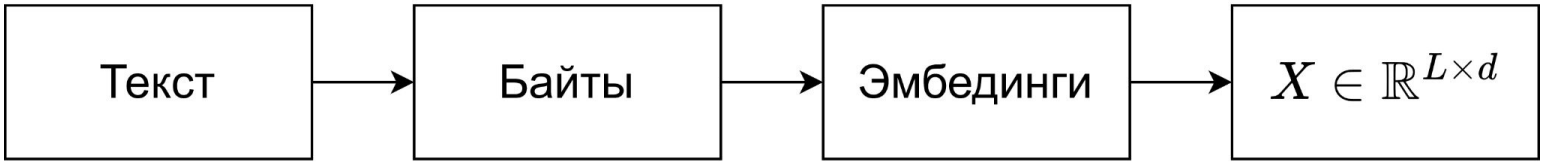
- Работает с байтами.
- Разделяет слова с помощью отбора по блокам подслов-кандидатов.
- Выучивает интерпретируемые латентные подслова и позволяет легко проверить лексические представления.
- Вычислительно эффективен.



https://arxiv.org/pdf/2106.12672.pdf

GRADIENT-BASED SUBWORD TOKENIZATION (GBST)

Пайплайн получения входных данных для GBST:



Получаем матрицу размера
L – количество байтов на d – размерность эмбединга

Основная идея GBST

Хотим научить модель выполнять скрытую сегментацию входных данных по подсловам, выбирая наиболее подходящий **блок подслов** для всех символов.

Определим **блок подслова** длины b как $X_{i:i+b} \in \mathbb{R}^{b \times d}, \ 1 \leqslant i \leqslant L - b$

Также определим функцию проекцию F
На практике F это Average Pooling $F : \mathbb{R}^{b \times d} \rightarrow \mathbb{R}^{d}$

И наконец объединим все блоки подслов:

$$X_b = [F(X_{i:i+b}); F(X_{(i+s):(i+s)+b}); \ldots]$$

s – обычно берут равным b
Строим блоки для b = 1 … m $X_b \in \mathbb{R}^{\frac{L}{b} \times d}$

Основная идея GBST

В таком пайплайне получения блоков подслов мы не рассматриваем все возможные подслова, пример:

Можно уменьшить s и рассмотреть больше подслов, но это увеличит объём вычислений.

Компромисс: перед разбиением на блоки, применить на X 1D свертку. Это эффективно сглаживает блоки подслов.

Бонус: функции проекции F не нужно учитывать порядок.

$$F(\text{Conv1D}([A\ B\ C])) \neq F(\text{Conv1D}([B\ C\ A]))$$

Оценка блоков GBST

Обозначим за $X_{b,i}$ блок длины b, в который попадает i символ.

Тогда введем модель, которая будет оценивать этот блок.	$F_R : \mathbb{R}^d \to \mathbb{R}$
С помощью неё можем оценивать все наши блоки.	$p_{b,i} = F_R(X_{b,i})$
Отсюда можем получить матрицу вероятностей	$P_i = \text{softmax}(p_{1,i}, p_{2,i}, ..., p_{M,i}),$ $P \in \mathbb{R}^{L \times M}$
Также добавим self-attention блок, чтобы учесть вероятности соседних подслов	$\hat{P} = \text{softmax}(PP^T)P$
Наконец получаем новое представление входных данных	$\hat{X} = \sum_b^M \hat{P}_{b,i} X_{b,i}$

Усечение GBST

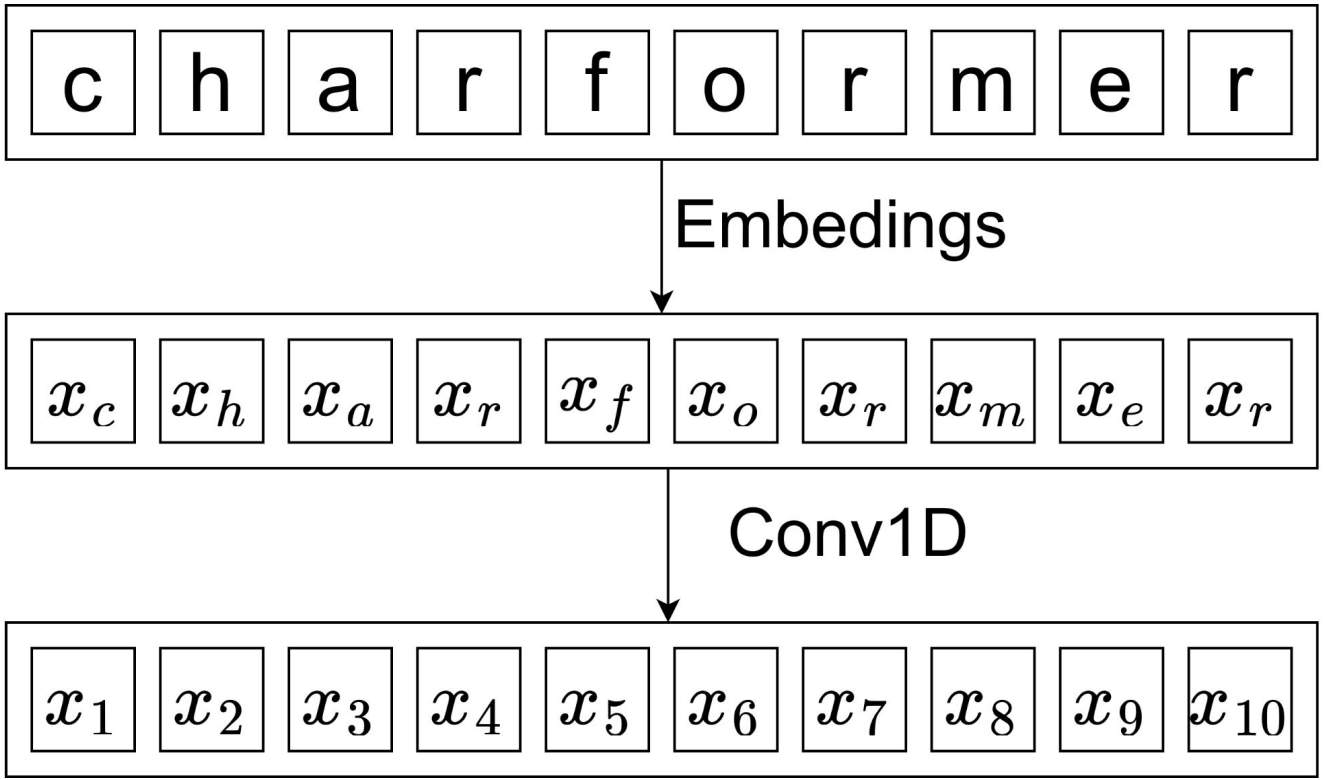
Наконец нужно получить векторное представление подслов, для этого как и в прошлый раз используем Average Pooling:

$$F_D : \mathbb{R}^{L \times d} \rightarrow \mathbb{R}^{\frac{L}{d_s} \times d}$$

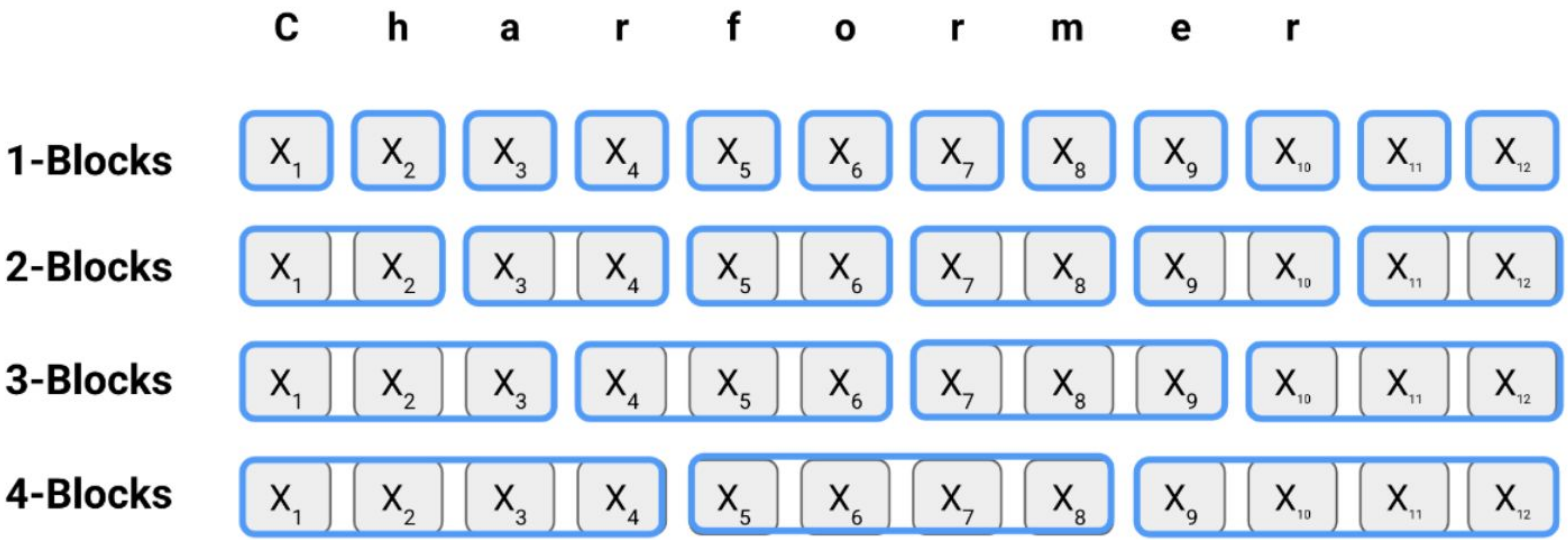
Наша финальная последовательность, которую будем подавать на вход в трансформер:

$$F_D(\hat{X}),\ \hat{X} = [\hat{X}_1, ..., \hat{X}_L]$$

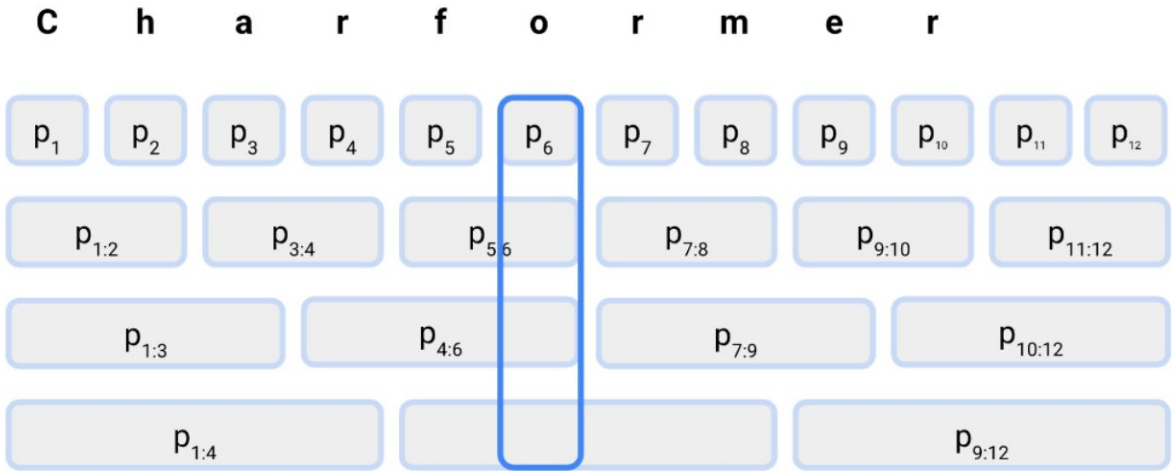
Пример


c h a r f o r m e r
Embedings
x_c x_h x_a x_r x_f x_o x_r x_m x_e x_r
Conv1D
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}

Пример


C h a r f o r m e r
1-Blocks
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12
2-Blocks
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12
3-Blocks
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12
4-Blocks
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12

Пример



$$P_i = \text{softmax}(p_{1,i}, p_{2,i}, ..., p_{M,i})$$

$$\hat{P} = \text{softmax}(PP^T)P$$

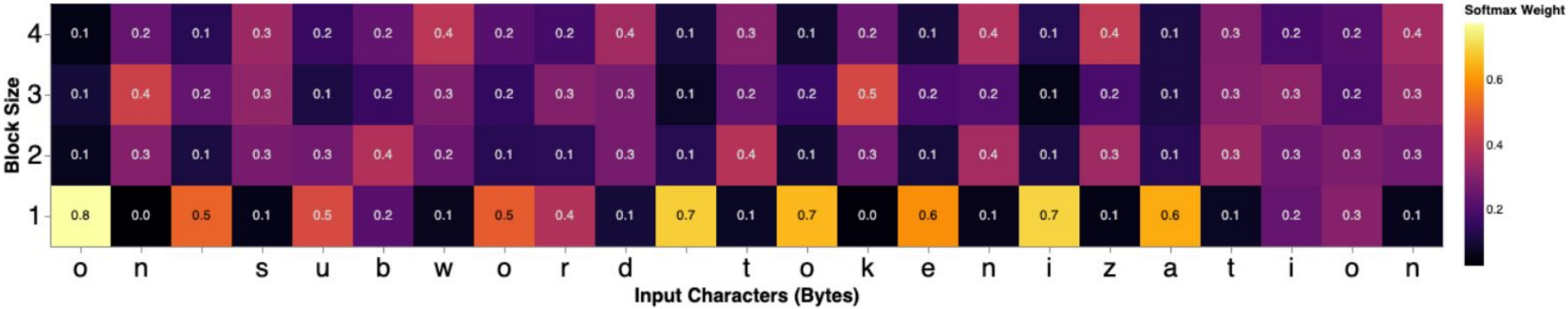
$$\hat{X} = \sum_b^M \hat{P}_{b,i} X_{b,i}$$

усечение

Выход GBST

$$F_D(\hat{X})$$

Пример интерпретации



Немного о трансформере CHARFORMER

- Стандартная Transformer encoder-decoder архитектура.

- В современных многоязычных моделях до 71% параметров занимают эмбединги словаря, в то время как словарь CHARFORMER символьный и потому сильно меньше. Авторы компенсируют это более глубоким и узким энкодером.

- Модели на основе символов имеют более длинные последовательности и обрабатываются на 25% дольше, чем модели на основе подслов. Авторы учитывают это при выравнивании количества параметров с другими моделями.

Эксперименты

Model	$\|\theta\|$	SST-2	MNLI	QNLI	MRPC	QQP	STSB	COLA	AVG
BERT$_{Base,Subword}$	110M	92.7	84.4/-	88.4	86.7/-	-	-	-	-
T5$_{Base,Subword}$	220M	92.7	84.2/84.6	90.5	88.9/92.1	91.6/88.7	88.0	53.8	84.3
Byte-level T5$_{Base}$	200M	91.6	82.5/82.7	88.7	87.3/91.0	90.9/87.7	84.3	45.1	81.5
Byte-level T5+Conv$_{Base}$	205M	89.8	81.1/82.5	89.2	83.6/89.2	90.7/87.7	85.0	47.1	81.2
Byte-level T5+LASC$_{Base}$	205M	90.0	80.0/80.8	87.1	82.8/88.1	89.0/85.4	83.7	25.3	77.0
CHARFORMER$_{Base}$	203M	91.6	82.6/82.7	89.0	87.3/91.1	91.2/88.1	85.3	42.6	81.4
Byte-level T5$_{SBase}$	133M	91.2	83.9/83.7	90.9	85.5/89.2	91.1/88.1	85.7	49.3	82.6
CHARFORMER$_{SBase}$	134M	91.5	83.7/84.4	91.0	87.5/91.4	91.4/88.5	87.3	51.8	83.6

Сравнение CHARFORMER с другими моделями на стандартных английских наборах данных

Эксперименты

Model	Civil Comments	Wiki Comments
$\text{T5}_{Base,Subword}$	81.2 / -	91.5 / -
Byte-level T5_{Base}	82.8 / 78.7	93.2 / 75.4
Byte-level T5+LASC_{Base}	82.9 / 78.2	93.0 / 75.0
CHARFORMER_{Base}	83.0 / 78.8	92.7 / 79.7
$\text{CHARFORMER}_{SBase}$	83.0 / 78.9	93.5 / 75.5

Сравнение на задаче классификации комментариев на WiKi

Эксперименты

Model	IMDb	News
$\text{T5}_{Base,Subword}$	94.2	93.5
$\text{Byte-level T5}_{Base}$	<u>91.5</u>	93.6
$\text{Byte-level T5+LASC}_{Base}$	91.1	93.5
CHARFORMER_{Base}	<u>91.5</u>	<u>94.0</u>
$\text{CHARFORMER}_{SBase}$	94.4	94.1

Классификация на больших документах

Эксперименты

		In-Language	Translate-Train-All				Zero-Shot	
Model	$\|\theta\|$	TyDiQA-GoldP	XQuAD	MLQA	XNLI	PAWS-X	XNLI	PAWS-X
mBERT$_{Base}$ (Subword)	179M	77.6/68.0	-/-	-/-	-	-	65.4	81.9
mT5$_{Base}$ (Subword)	582M	80.8/70.0	75.3/59.7	67.6/48.5	75.9	89.3	75.4	86.4
Byte-level T5$_{Base}$	200M	75.6/65.4	68.6/54.3	61.8/44.4	69.4	87.1	57.4	80.9
Byte-level T5+LASC$_{Base}$	205M	70.6/59.7	66.8/52.1	58.8/41.1	67.9	84.8	55.2	79.0
CHARFORMER$_{Base}$	203M	75.9/65.6	70.2/55.9	62.6/44.9	71.1	87.2	57.6	81.6
CHARFORMER$_{SBase}$	134M	79.1/68.8	73.6/59.0	66.3/48.5	72.2	88.2	66.6	85.2
CHARFORMER$_{SBase,LongPT}$	134M	81.2/71.3	74.2/59.8	67.2/49.4	72.8	88.6	67.8	83.7

Сравнение на многоязычной задаче переводов текстов

Производительность

Model	Batch Size	L	d_s	$\|\theta\|$	Speed (steps/s)	FLOPS
mT5$_{Base}$ (Subword)	1024	1024	-	582M	1.54	1.3×10^{15}
CHARFORMER$_{SBase}$	1024	2048	2	134M	1.98	4.3×10^{14}
CHARFORMER$_{SBase,LongPT}$	2048	2048	2	134M	1.01	4.3×10^{14}

Сравнение CHARFORMER с модель на основе подслов

Производительность

Model	L	d_s	$\|\theta\|$	Speed (steps/s)	FLOPS	Peak Mem.
T5_{Base} (Subword)	512	-	220M	9.3	1.1×10^{13}	-
Byte-level T5_{Base}	1024	1	200M	8.2	2.9×10^{13}	3.09GB
Byte-level T5+LASC_{Base}	1024	4	205M	15	9.9×10^{12}	1.62GB
CHARFORMER_{Base}	1024	2	206M	11	1.6×10^{13}	1.95GB
CHARFORMER_{Base}	1024	3	203M	15	1.1×10^{13}	1.63GB
$\text{CHARFORMER}_{SBase}$	1024	2	134M	14	1.3×10^{13}	1.73GB
$\text{CHARFORMER}_{SBase}$	1024	3	134M	20	8.7×10^{12}	1.34GB

Сравнение CHARFORMER с моделями на основе байтов

Большие модели

Model	TyDiQA-GoldP F1 / EM
mT5_{Large}	85.3 / 75.3
ByT5_{Large}	87.7 / 79.2
CHARFORMER*	86.3 / 77.3

Сравнение моделей с 1.23B параметров

Авторы статьи

- Google Research and DeepMind
- 8 авторов
- Занимаются в основном NLP
- У части авторов есть статьи про задачи, в которых charformer может быть полезен (например при работе с несколькими языками)

Влияющие статьи

1) Rethinking embedding coupling in pre-trained language models, 2020
совпадают 2 автора

Model	Languages	$\|V\|$	N	N_{emb}	%Emb.
mBERT (Devlin et al., 2019)	104	120k	178M	92M	52%
XLM-R_{Base} (Conneau et al., 2020a)	100	250k	270M	192M	71%
XLM-R_{Large} (Conneau et al., 2020a)	100	250k	550M	256M	47%
$\text{BERT}_{\text{Base}}$ (Devlin et al., 2019)	1	30k	110M	23M	21%
$\text{BERT}_{\text{Large}}$ (Devlin et al., 2019)	1	30k	335M	31M	9%

Влияющие статьи

2) ByT5: Towards a token-free future with pre-trained byte-to-byte models, 2021

Подают байты на вход; меняют баланс параметров encoder/decoder частей; предобучают на предсказании нескольких последовательных байт (вырезают в среднем штук 20)

Никак не группируют байты или соответствующие им векторы, то есть вход пропорционален числу байт в тексте.

Влияющие статьи

3) CANINE: Pre-training an Efficient Tokenization-Free Encoder for Language Representation, 2021

Не нужна предобработка; в начале группируют векторы и снижают их число в несколько раз

Подают символы на вход (нужно хеширование + словарь на 16k); группируют с помощью local attention + strided cond; в конце можно аналогично разгруппировать

Достоинства/недостатки

+ Лучше результаты на некоторых задачах
+ Не нужна предобработка входного текста
+ Снижение размера словаря во много раз

- Сэкономленные на словаре параметры придётся отдать на увеличение кодировщика
- Затраты времени на получение эмбедингов

Дальнейшая работа

В статье указывается, что свёртки и другие локальные механизмы могут быть не лучшим решением в многоязычных моделях (например неконкатенативная морфология)

كتب	k-t-b	“write” (root form)
كَتَبَ	**k**a**t**a**b**a	“he wrote”
كَتَّبَ	**k**a**tt**a**b**a	“he made (someone) write”
اِكْتَتَبَ	i**k**ta**t**a**b**a	“he signed up”

Table 1: Non-concatenative morphology in Arabic.[4]

Список источников

- https://arxiv.org/pdf/2106.12672.pdf
- https://huggingface.co/course/chapter6/4
- https://arxiv.org/pdf/2010.12821.pdf
- https://arxiv.org/pdf/2105.13626.pdf
- https://arxiv.org/pdf/2103.06874.pdf