Incorporating Context into Subword Vocabularies

Shaked Yehezkel
Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel
shakedy@mail.tau.ac.il

Yuval Pinter
Department of Computer Science
Ben-Gurion University of the Negev
Beer Sheva, Israel
uvp@cs.bgu.ac.il

Abstract

Most current popular subword tokenizers are trained based on word frequency statistics over a corpus, without considering information about co-occurrence or context. Nevertheless, the resulting vocabularies are used in language models' highly contextualized settings. We present SAGE, a tokenizer that tailors subwords for their downstream use by baking in the contextualized signal at the vocabulary creation phase. We show that SAGE does a better job than current widespread tokenizers in keeping token contexts cohesive, while not incurring a large price in terms of encoding efficiency or domain robustness. SAGE improves performance on English GLUE classification tasks as well as on NER, and on Inference and NER in Turkish, demonstrating its robustness to language properties such as morphological exponence and agglutination.

1 Introduction

Much of the research space in current NLP is focused on advancing models: modifying pretraining objectives, improving network architectures, adding tasks and schemes for downstream evaluation. Limited work is dedicated to a crucial step underlying all modern large language models (LLMs), namely the **tokenization** phase. In order to process a given string of text, an LLM must first obtain a vector representation of the input by segmenting it into tokens. Since out-of-vocabulary (OOV) items inhibit the performance of models, current tokenizers produce tokens which are possibly proper subsegments of input words, known as **subwords**. This method, popularized by systems such as WordPiece (Schuster and Nakajima, 2012), Byte-Pair Encoding (BPE; Sennrich et al., 2016) and UNIGRAMLM (Kudo, 2018), allows any word to be represented by one or more tokens, removing the OOV problem while allowing more flexibility in determining the token vocabulary size, which ultimately affects model speed (mostly through the softmax generation targets) and performance (through better ability to represent less-frequent words).

BPE	_His _son _Raj ash ri _Sud h ak ar _has _p enn ed **_dial og ues** _and _songs _for _some _films _that _were _dubbed _into _Telugu .
SAGE	_His _son _Raj ash r i _Sud h a k a r _has _penn e d **_dial ogues** _and _songs _for _some _films _that _were _dubbed _into _Telugu .
BPE	_This _gene _is _a **_pseud og ene** _in _humans _and _most _other _prim ates .
SAGE	_This _gene _is _a **_pseud ogene** _in _humans _and _most _other _prim ates .
BPE	_The **_St o og es** _work _for _Mir acle _Det ective _Agency ,
SAGE	_The **_St o o g e s** _work _for _Mir acle _Det ective _Agency ,

Table 1: The token *og* is selected by BPE (vocabulary of size 16,000) for achieving the frequency objective, but is discarded by SAGE for failing to be contextually coherent. These examples from the corpus demonstrate some different contexts.

One potential pitfall of both BPE and UNIGRAMLM, as well as their proposed variants (He et al., 2020; Provilkov et al., 2020), is that they are trained on word frequency statistics alone, without considering information about word co-occurrence or contexts. At the same time, the resulting vocabularies are used in highly contextualized settings, the LLMs, where a single subword such as *og* might appear in very different contexts derived from words like *dial og ues* and *pseud og ene*. We propose a system which prepares subwords for their downstream use by baking in the contextualized signal **at the vocabulary creation step**. Our model, SAGE, uses the SKIPGRAM objective (Mikolov et al., 2013) over a corpus as the basis for iteratively eliminating candidate subwords from an initial large vocabulary until the desired vocabulary size has been reached. As Table 1 shows, SAGE succeeds in removing the ambiguous *og* token, facilitating distinct contextualization procedures for the example sentences

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 623–635

(taken from Wikipedia).

We present our algorithm, SAGE, which is predicated on iterative pruning of contextually noisy tokens from the vocabulary, and compare its effects on token properties and context cohesion with BPE both in- and out-of-domain, in English and Turkish. We then evaluate its performance on downstream tasks by training a BERT-based LLM (Devlin et al., 2019) on a vocabulary produced by both tokenizers in both languages, demonstrating substantial improvements on most English GLUE tasks and on NER, as well as Turkish NLI and NER. We emphasize that as opposed to most current tokenizer variants, our model is a "plug and play" substitution for any subword token vocabulary, requiring no modification in the inference protocol (or code) when pre-training or applying an applicable LLM from a popular shared library.[1]

2 Subword Vocabulary Creation

The methods used to tokenize corpus in order to later assign tokens with continuous vectors, or *embeddings*, have evolved over the years. Initially, each word in the corpus was assigned its own embedding (Collobert and Weston, 2008; Mikolov et al., 2013). OOVs, i.e. words not appearing in the original training corpus or below a certain frequency threshold, would receive a special (but identical) "UNK" vector. Subword tokenizers (Schuster and Nakajima, 2012; Wu et al., 2016) were introduced to alleviate this issue, allowing segmentation of all text into embeddable units (assuming no unseen characters, a much more relaxed constraint for languages using alphabetical scripts). The training process used to create a subword vocabulary from which the model then decodes text input involves optimizing an encoding objective over a large corpus. To date, all tokenizers used in practice in large models focus on efficiency and information-theoretic objectives, and reduce the corpus to a unigram frequency count of space-delimited words, reducing calculation time but losing all contextual signal. SAGE reintroduces the contextual dependencies between words into vocabulary creation via a two-stage process, namely over-application of BPE followed by iterative pruning using ideas inspired by UNIGRAMLM and SKIPGRAM. We briefly present these algorithms before tying them together into SAGE.

[1]Our code and models are available at www.github.com/MeLeLbgu/SaGe.

Algorithm 1 Byte-pair encoding vocabulary creation (Gage, 1994; Sennrich et al., 2016)

Input: Corpus C, Vocabulary final size V.
Output: Vocabulary $\mathcal{V}$ of size V (ordered).

procedure BPE(C, V)
  $\mathcal{V} \leftarrow$ All unique characters in C
  while $|\mathcal{V}| < V$ **do** ▷ Merge tokens
    $\langle t_L, t_R \rangle \leftarrow$ Most frequent bigram in C
    $t_{NEW} \leftarrow t_L \oplus t_R$ ▷ Make new token
    $\mathcal{V} \leftarrow \mathcal{V} \oplus [t_{NEW}]$
    C.ReplaceAll($\langle t_L, t_R \rangle, t_{NEW}$)
  end while
return $\mathcal{V}$
end procedure

Byte-Pair Encoding. The BPE algorithm creates a vocabulary "bottom-up", starting with all single characters from the alphabet, iteratively adding tokens until reaching the desired vocabulary size. In each iteration, the added token is the concatenation of the most frequent adjacent pair of existing tokens (see Algorithm 1). The default setting of the algorithm's most popular implementation (Kudo and Richardson, 2018) restricts token addition within word boundaries, facilitating training from unigram frequencies. In addition, LLM tokenizers using BPE (Liu et al., 2019; Radford et al., 2019; Wolf et al., 2020) decode sequences not by applying merges by order of the vocabulary, as originally dictated by the algorithm, but through greedy largest-subsequence left-to-right inference.

Unigram Language Model. UNIGRAMLM offers a top-down vocabulary creation process, starting with an initial vocabulary of all substrings in the input corpus and pruning tokens iteratively until reaching the desired vocabulary size. The pruning procedure involves calculating the overall unigram likelihood of the corpus with the current vocabulary versus a vocabulary lacking the candidate pruning token (see Algorithm 2 for details), which we refer to as the **ablation objective**. Under this system, decoding is ideally performed by considering probabilities of all possible segmentations using, e.g., the Viterbi algorithm; again, common practice is to use left-to-right greedy decoding.

Skipgram Objective. The SKIPGRAM objective (Mikolov et al., 2013) formalizes the relation between a target token t and its context, asking whether context tokens c within a window W_t of pre-defined size can be predicted from t. These predictions are done via sigmoid activation over the inner product of embeddings trained for targets ($\boldsymbol{E}^{(T)}$) and contexts ($\boldsymbol{E}^{(C)}$). When aggregated over

Algorithm 2 UNIGRAMLM vocabulary creation (Kudo, 2018). $n \arg\min_X$ denotes the n bottom-ranked elements in X.

Input: Corpus C, Vocabulary final size V, pruning batch size k.
Output: Vocabulary $\mathcal{V}$ of size V.

procedure UNIGRAMLM(C, V)
  $\mathcal{V} \leftarrow$ All substrings occurring more than once in C
  while $|\mathcal{V}| > V$ **do** ▷ Prune tokens
    $X^{(j)} \leftarrow$ tokenize($C, \mathcal{V}$)
    $\mathcal{L}(\mathcal{V}) \leftarrow \sum_{j=1}^{|C|} \log\left(P(X^{(j)})\right)$
    for all $t \in \mathcal{V}$ **do**: ▷ Calculate ablation objective
      $loss_t \leftarrow \mathcal{L}(\mathcal{V} \setminus \{t\}) - \mathcal{L}(\mathcal{V})$
    end for
    $\mathcal{P} \leftarrow \min(k, |\mathcal{V}| - V) \arg\min_{t \in \mathcal{V}}(loss_t)$
    $\mathcal{V} \leftarrow \mathcal{V} \setminus \mathcal{P}$ ▷ Prune
  end while
return $\mathcal{V}$
end procedure

all tokens in a corpus, SKIPGRAM can be used as a total likelihood measure, approximating its overall contextual cohesion:

$$\mathcal{L}(\mathcal{V}, \mathcal{C}) = -\sum_{t \in \mathrm{tok}(\mathcal{C}, \mathcal{V})} \sum_{c_j \in W_t} \log\left(\sigma(\boldsymbol{E}_t^{(T)} \cdot \boldsymbol{E}_{c_j}^{(C)})\right). \tag{1}$$

As token vocabularies or their inference methods change, so do the target sequences and their contexts, resulting in differences in aggregated likelihood which can then act as scores comparing one tokenization to another. We use this behavior as the ablation objective for SAGE.

3 SAGE Vocabulary Creation

SAGE[2] is a top-down tokenizer, following UNIGRAMLM's general procedure, incorporating a SKIPGRAM objective as its vocabulary trimming rule. Given an initial vocabulary $\mathcal{V}$ and a corpus $\mathcal{C}$, SAGE computes a SKIPGRAM embedding space over $\mathcal{V}$ which provides it with an overall likelihood over $\mathcal{C}$ as in (1). It then proceeds to calculate the *loss* of each token in the vocabulary were it to be removed, eliminating the tokens incurring minimal loss and re-tokenizing the corpus according to the updated vocabulary, repeating this procedure until reaching the desired vocabulary size V. Having learned this vocabulary, downstream inference proceeds exactly as in the other segmentation-based methods, in a greedy left-to-right manner. SAGE can also be adapted to anticipate other decoding algorithms, by changing the re-tokenization steps accordingly.

[2]The name is not an acronym; it is intended to evoke SkipGram while maintaining the "suffix" of *BPE*.

Algorithm 3 SAGE vocabulary creation. $n \arg\min_X$ denotes the n bottom-ranked elements in X.

Input: Corpus C, Vocabulary final size V, basic tokenizer $\mathcal{T}$, overshoot factor n, pruning batch size k, likelihood recalculation frequency m, size of pruning candidate set M, embedding recalculation frequency l.
Output: Vocabulary $\mathcal{V}$ of size V.

procedure SAGE(C, V)
  $\mathcal{V} \leftarrow \mathcal{T}(C, n \cdot V)$
  $i \leftarrow 0$
  while $|\mathcal{V}| > V$ **do**
    if $i \equiv 0 \pmod{l \times m}$ **then**
      $\boldsymbol{E}^{\mathcal{V}} \leftarrow$Word2Vec($\mathcal{V}$) ▷ Embedding table
    end if
    $\mathcal{L}(\mathcal{V}, \mathcal{C}) \leftarrow$ SGObj($\boldsymbol{E}^{\mathcal{V}}, \mathcal{C}$) ▷ Total likelihood (1)
    if $i \equiv 0 \pmod{m}$ **then** ▷ Update bottom set
      for all $t \in \mathcal{V}$ **do**:
        $loss_t \leftarrow \mathcal{L}(\mathcal{V} \setminus \{t\}, \mathcal{C}) - \mathcal{L}(\mathcal{V}, \mathcal{C})$
      end for
      $\mathcal{V}_{bot} \leftarrow M \arg\min_{t \in \mathcal{V}}(loss_t)$
    else ▷ Update losses for bottom set
      for all $t \in \mathcal{V}_{bot}$ **do**:
        $loss_t \leftarrow \mathcal{L}(\mathcal{V} \setminus \{t\}, \mathcal{C}) - \mathcal{L}(\mathcal{V}, \mathcal{C})$
      end for
    end if
    $\mathcal{P} \leftarrow \min(k, |\mathcal{V}| - V) \arg\min_{t \in \mathcal{V}_{bot}}(loss_t)$
    $\mathcal{V}_{bot} \leftarrow \mathcal{V}_{bot} \setminus \mathcal{P}$ ▷ Prune
    $\mathcal{V} \leftarrow \mathcal{V} \setminus \mathcal{P}$
    $i \leftarrow i + 1$
  end while
return $\mathcal{V}$
end procedure

In practice, applying the full process described above introduces multiple sources of considerable computational complexity: for example, calculating the ablation objective for each token in each iteration produces a quadratic amount of calculations over the entire corpus; recalculating embeddings for an updated vocabulary is similarly unreasonable to perform at each iteration. We ameliorate these and other sources of complexity using a series of heuristics found in preliminary experiments to be minimally disruptive to precision of likelihood calculations. We will now describe these heuristics, all depicted in Algorithm 3. First, instead of initializing the vocabulary as the full set of possible character sequences in the corpus, as in UNIGRAMLM, we use any existing noncontextual tokenizer such as BPE to learn a vocabulary larger than V by a factor of n, and begin the pruning process from there. Next, instead of removing a single token from the bottom of the loss-ranked vocabulary, we remove a batch of the k bottom tokens

Sentence fragment	... use of an include directive is when referring to ...
Tokenization using $\mathcal{V}$	_use _of _an _includ **[**e _direct **ive** _is _when**]** _ref er r ing _to
Tokenization using $\mathcal{V} \setminus \{\texttt{\_includ}\}$	_use _of _an _inc l u **[**de _direct **ive** _is _when**]** _ref er r ing _to

Table 2: The effect of retokenization on a context window of width 2 (in brackets) surrounding a target token (in bold). A left-side context token has been replaced as a result of an out-of-window vocabulary ablation.

each time, as does UNIGRAMLM.[3] To avoid frequent loss recalculation, we recompute the entire likelihood set once every m ablation steps, and only keep the bottom M tokens as pruning candidates for the next m steps. Our preliminary experiments support this decision, as we found the ranked list of losses tends to stay relatively stable over dozens of batch-pruning iterations. Lastly, to avoid the costly re-training of the embedding matrix for all tokens given the updated corpus, which only results in minor changes in likelihood during subsequent iterations, we only perform it every l iteration batches, i.e. after the ablation of $k \times m \times l$ tokens. n, k, l, m and M are all algorithm hyperparameters tuned empirically based on desired runtime, corpus size and vocabulary size.

Contextual Loss. In order to calculate the per-token SKIPGRAM likelihood loss, all sentences where a token t occurs need to be re-segmented according to $\mathcal{V} \setminus \{t\}$, and their new likelihoods recorded. To support performing this calculation on a large scale, we maintain a mapping of tokens to sentences containing them, as well as these sentences' current likelihoods. This must be done at the sentence level rather than the window level, since a remaining suffix from an out-of-window re-tokenization may combine with in-window characters and form different token sequence replacements at a given stage. Consider the example in Table 2, where re-tokenization results in the replacement of a context token for a distant target.

Negative Sampling. The original SKIPGRAM objective uses negative samples to estimate context probabilities. Since our application of SKIPGRAM within the vocabulary creation algorithm (independent of the embeddings training procedure) includes only likelihood estimation with no parameter updates, we do not sample negative tokens, a process which would introduce substantial noise and complexity.

[3]As in UNIGRAMLM and other ablation-based vocabularies, single-character tokens are never removed from the vocabulary, in order to allow for all in-alphabet words to be tokenized.

4 SAGE Vocabulary Properties

For an analysis of our modified algorithm's advantages, we trained vocabularies of a pre-determined size using both BPE and SAGE. We selected $|\mathcal{V}| = 16,000$, and obtained corpora for English (750,000 lines from the August 2022 Wikipedia dump) and Turkish (the entire text of the September 2022 Wikipedia dump), opting for languages that share the Latin alphabet but differ in family (Indo-European vs. Turkic) and, crucially, in morphological properties: English is a low-exponence, low-synthesis language, while Turkish features multiple inflectional exponence and high verbal synthesis, as well as highly agglutinative morphology (Bickel and Nichols, 2013a,b). We used the following hyperparameter settings to compute the vocabularies: Initial vocab size $20,000$ (or $n = 1.25$), $l = 4$, $k = 100$, $M = 1500$, $m = 10$. We used the Gensim package to train the SKIPGRAM models (Rehurek and Sojka, 2011), and Sentencepiece (Kudo and Richardson, 2018) to obtain the initial BPE vocabularies.[4] More hyperparameters are detailed in Appendix A.

We present an analysis of the resulting vocabularies, highlighting the advantages and trade-offs exhibited by context-based subword tokenization. Generally speaking, most of the tokens discarded from SAGE's initial vocabulary appear in the baseline BPE's final vocabulary. Among the differences between the vocabularies are many short tokens that appear in BPE's but not SAGE's, proper substrings of longer tokens also appearing in the BPE vocabulary. This is due to BPE's bottom-up merge table construction, which forces retention of the entire chain of tokens created: if *the* is part of the vocabulary, either *th* or *he* must also be there. While essential for the original intended decoding process, actual implementations of greedy decoding have no need for this property. SAGE's initial vocabulary shares this characteristic, but the trimming process allows any token to be ablated,

[4]Since BPE augments its vocabulary iteratively, the baseline BPE vocabulary is a proper subset of that used to initialize SAGE.

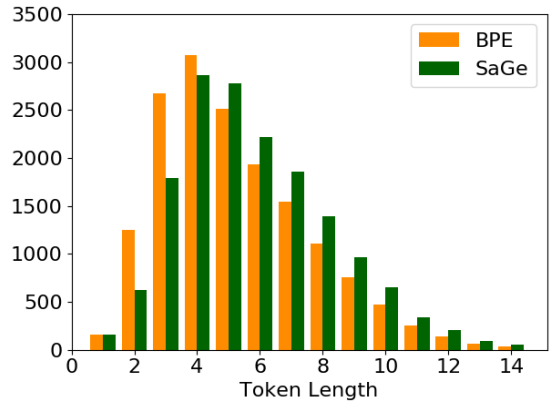

Figure 1: Token length distribution of BPE's vocabulary vs. SAGE's on English.

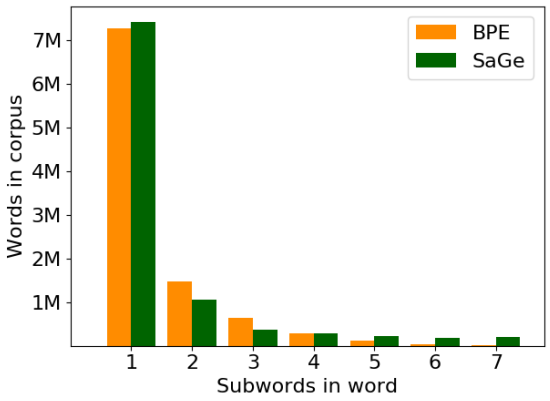

Figure 2: Number of subwords required to tokenize a word, collected over the original English training corpus.

More frequent in			
SAGE		BPE	
e	s	es	ic
ing	ist	ings	ff
ation	ate	ations	ates

Table 3: Tokens with high difference in frequency between tokenizations (English models).

including those in the middle of merge chains. Another difference found between the vocabularies is the strong preference of SAGE for word-initial tokens. 83% of the tokens that appear in SAGE's vocabulary but not in BPE's are word-initial, compared to only 22% of the BPE-only tokens. This is reasonable, since a token surviving SAGE's ablation steps exhibits high loss for the condition of its removal, which is arguably the case when a nearby target word needs to predict a word-initial context.

Token Length. Figure 1 shows a histogram of token lengths (in characters) for the 16,000-token SAGE and BPE vocabularies in English (results on Turkish are similar). SAGE clearly selects longer tokens for its vocabulary, again a sensible outcome given their higher chance of being contextually coherent. The difference is most stark with tokens of length 2 and 3; when considering only tokens appearing in exactly one of the final vocabularies, we find that 56% of BPE-only tokens are of length 2 and 3, while 55% of SAGE-only tokens are of length 5 and above.

Token Frequency. We compute the frequency of tokens in the encoding form of the English training corpus, once using SAGE vocabulary and once using BPE's. In Table 3 we show some of the tokens with the biggest difference in frequency between SAGE and BPE tokenizations. We can see SAGE reverts to single-character tokens considerably more often than BPE (also demonstrated in the last example in Table 1). We view this as a feature of context-based tokenization—its vocabulary is partitioned between (mostly short) tokens that are highly ambiguous in context and (mostly long) tokens that have coherent contexts. At the same time, BPE is rife with tokens that are medially ambiguous contextually, whose resulting embeddings can be neither useful nor completely ignorable, adding noise to the representation sequences. As a result, SAGE breaks down complex suffixes, which in English are compositional, into their constituent morphology. The suffix *ings* is thus dismantled to *ing s*, whereas BPE reserves a token for it, mostly unhelpful in itself.

Subword Fertility. Fertility, as defined in the statistical machine translation literature, refers to the average number of subwords produced per tokenized word. Figure 2 exhibits a histogram of all English corpus words by their subword length, using the BPE vocabulary and the SAGE vocabulary. Although SAGE retains more words as single tokens, it trades them off with more words having five subwords or more, compared with BPE's abundance of words with 2 and 3 subwords. This follows the trend described so far, of SAGE's preference for dismantling unknown words into meaningless single-character tokens rather than confusing, ambiguous length-2 and length-3 tokens. We believe that BPE's behavior harms text understand-

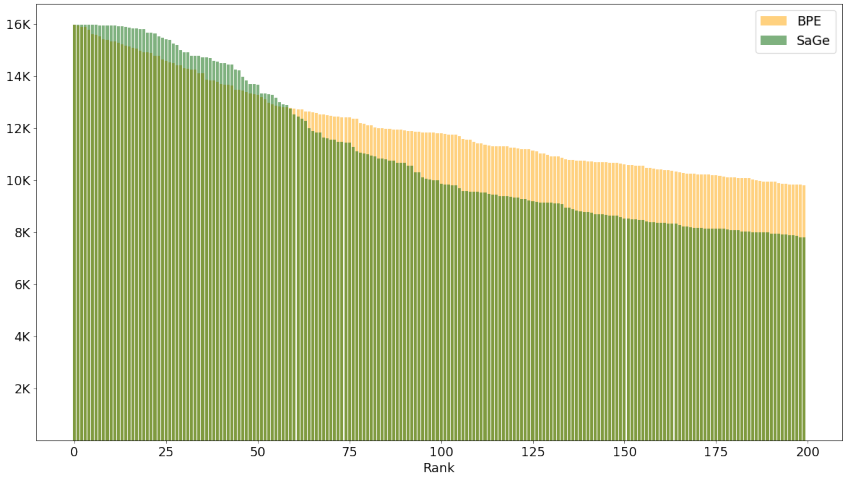

Figure 3: Number of distinct neighbors each token encounters in a width-5 window, top 200, Turkish.

ing in suggesting that these ambiguous fragments (consider "og") have some meaning that an LLM can try and learn, whereas SAGE's single-character breakdown indicates a word that's truly unknown and cannot be inferred by composing constituent in-vocab subwords.

Fertility translates to a trade-off in **encoding efficiency** to SAGE's contextual advantage: a sample of 150K lines from English Wikipedia is encoded by 4 million BPE tokens, optimizing only an information-theoretic objective, whereas SAGE produces 4.5 million. Having said that, this inefficiency might be further offset during LLM pretraining: we propose that contextually coherent tokens will require fewer update steps in order to achieve useful embedding parameters, helping the model converge faster compared to BPE tokens. We leave testing this hypothesis to future work.

Contextual Exponence. To determine the degree to which SAGE effectively optimizes tokens' contextual soundness, which is its ultimate goal, we plot the number of distinct neighbors each token encounters throughout the training corpus, ranked from high to low, in Figure 3. The very top of the ranking is occupied by single-character tokens which are context-null by design, which SAGE makes the most of by placing in almost all contexts. After a few dozen tokens, SAGE's context counts dip below BPE's, a trend which continues all the way through the vocabulary, making up a more contextually coherent set. These findings hold for English as well as Turkish, and replicate when taking a context window of size 2, different from that used during SAGE construction.

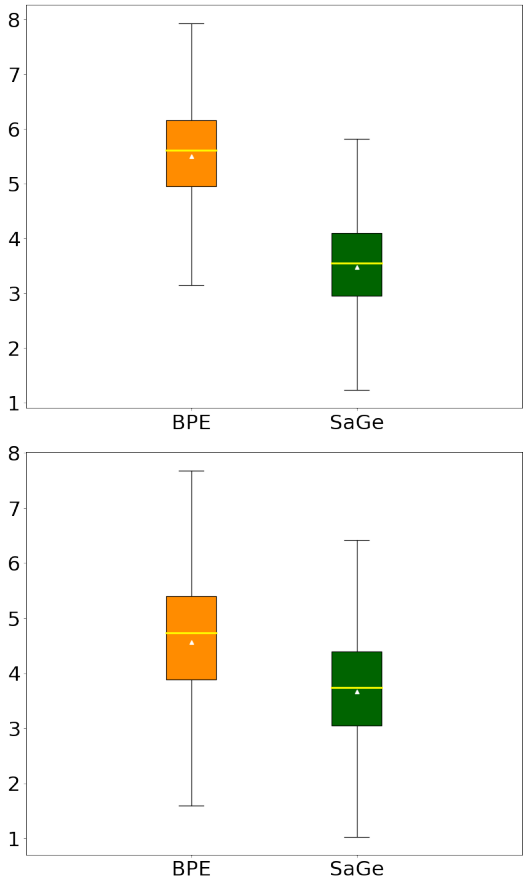

Figure 4: Distribution of token neighbors/frequency ratio for a width-5 window in English (top) and Turkish (bottom); BPE (left) and SAGE (right)

These findings can arguably be attributed to a frequency artifact, where SAGE simply outputs more tokens with lower frequency in order to provide them with fewer contexts. We thus present a normalized analysis in Figure 4, depicting the ratio between each token's number of unique neighbors and its frequency, distributed over the entire vocabulary. SAGE provides substantially lower ratios in both languages, supporting our original claim.

4.1 Robustness to Domain Change

One possible limitation of the SAGE objective is that it increases the reliance on the original training corpus compared to word-count-only algorithms. In and of itself, this should not necessarily be viewed as a problem, assuming the collected corpus is a faithful representative of an LLM's use case.[5] To this end, we collected comparable corpora from non-Wikipedia domains and ran our analysis on the SAGE and BPE vocabularies trained on Wikipedia. Our findings suggest that while SAGE loses its relative advantage in context-dependence over BPE, it does not fall behind it (i.e. it has not overfit to the Wikipedia domain). We present a fertility chart for an English corpus of 7.5M words from Quora questions[6] in Figure 5, depicting similar trends to that on Wikipedia (Figure 2) but with smaller differences between SAGE and BPE; the neighbor-to-frequency ratio aggregation chart in Figure 6 differs from Figure 4 (top) substantially but shows that SAGE and BPE tokens do not diverge significantly on this measure. We repeated the experiment on English legal text centered on US congress bills (Henderson et al., 2022) and on a 2.6M-word Turkish corpus of online reviews,[7] and observed similar trends.

These results indicate that while a considerable amount of the longer tokens preferred by SAGE were selected to optimize contextuality in the source domain, as it was designed to do, there is no "short blanket" effect for text originating in different domains. This could either be due to wide-scope advantages of some of the tokens selected by SAGE, or due to an intrinsic deficiency in BPE's long-tail tokens, or a combination of both.

[5]Indeed, existing literature recommends adding pre-training steps on new domains before fine-tuning models for them (e.g., Han and Eisenstein, 2019).

[6]https://huggingface.co/datasets/chenghao/quora_questions

[7]https://huggingface.co/datasets/cansen88/turkishReviews_5_topic

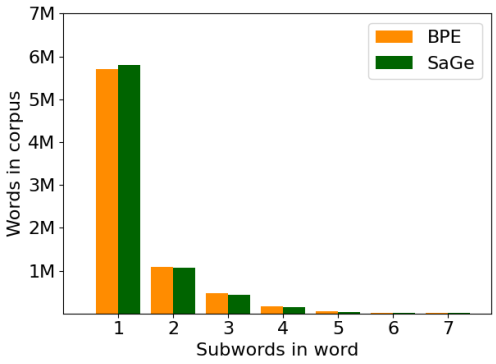

Figure 5: Number of subwords required to tokenize a word using the original Wikipedia-trained vocabularies, collected over a English Quora questions corpus.

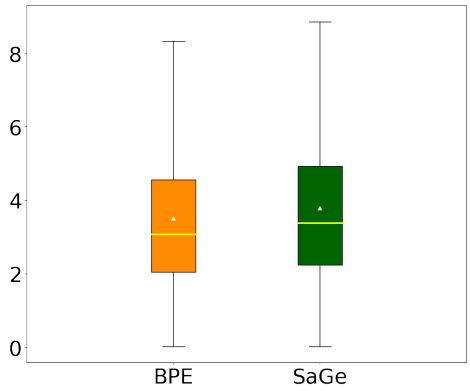

Figure 6: Distribution of token neighbors/frequency ratio for a width-5 window in English, based on a Wikipedia-trained vocabulary and collected over a English Quora questions corpus.

5 Downstream Evaluation

In order to evaluate the utility of our tokenization algorithm for major NLP tasks, we compare SAGE to a BPE vocabulary of the same size by means of pre-training a BERT-parameterized model (Devlin et al., 2019) using an expedited training scheme (Izsak et al., 2021). We then evaluate the LLM's performance both on sequence classification via the English GLUE benchmark (Wang et al., 2018) and the Turkish partition of XNLI (Conneau et al., 2018), and on named entity recognition in English (Wang et al., 2019) and Turkish (Al-Rfou et al., 2015). We use the default settings from Huggingface's library implementations of the fine-tuning processes (Wolf et al., 2020) and do not perform hyperparameter tuning for either model.

We present our results on sequence-level tasks in Table 4. SAGE tokenization improves performance on nearly all tasks with particularly sub-

	MRPC (F1)	MNLI (Acc %)	COLA (Matt.)	QNLI (Acc %)	SST2 (Acc %)	STSB (Pear.)	QQP (Acc %)	XNLI$_{tur}$ (Acc %)
BPE	.7918	62.76	.0777	66.17	80.54	.3094	82.75	41.20
SAGE	.8004	64.00	.0985	74.83	79.85	.3387	84.69	46.46

Table 4: Performance on sequence-level tasks for BERT models trained on different 16k-size vocabularies. XNLI$_{tur}$ is Turkish, the rest are English GLUE tasks. All results averaged over three runs on the dev set with different seeds.

	English	Turkish
BPE	.7142	.4660
SAGE	.7502	.5475

Table 5: Performance (F1) on NER tasks of BERT Turkish and English models trained on different subword vocabularies of size 16,000. All results averaged over three runs on the dev set with different seeds.

stantial improvements (1.3–8 accuracy points) on NLI datasets. Results on NER are presented in Table 5, again showing SAGE's dominance over BPE. Due to the length of the training pipeline leading from vocabulary creation through pre-training to fine-tuning, it is difficult to find individual examples where difference in tokenization leads to direct changes in prediction; we attribute the consistent overall gains in downstream performance mostly to the LLM pre-training step, where the design of SAGE's context-friendly vocabulary enables a more coherent contextual signal to flow through the transformer layers during backpropagation. We note that in general, our models fare worse on GLUE tasks compared to Izsak et al. (2021). We attribute this in part to the smaller token vocabulary size, and more substantially to the smaller pre-training corpus we used in our experiments.

6 Related Work

In recent years, a growing body of research has demonstrated the shortcomings of existing tokenization algorithms in the context of representing linguistic phenomena in different languages across different tasks (Banerjee and Bhattacharyya, 2018; Klein and Tsarfaty, 2020; Hakimi Parizi and Cook, 2020; Rust et al., 2021; Maronikolakis et al., 2021; Mielke et al., 2021; Hofmann et al., 2021), as well as the statistical properties affecting their downstream performance (Bostrom and Durrett, 2020). Our work addresses the concerns raised in this line of work by introducing an improved subword vocabulary creation method which leverages the contextual aspects of the main intended use case, namely LLMs. Previous work towards this goal includes algorithms which offer robustness within an existing subword vocabulary (Provilkov et al., 2020; He et al., 2020; Hiraoka, 2022), necessitating modification of either training, inference, or both procedures in the context of LLMs. Others have considered tuning the *size* of a subword vocabulary (Salesky et al., 2020), or selecting from an enlarged set of possible segmentations (Asgari et al., 2020), for optimizing performance on downstream tasks.

Some alternative tokenization methods focus on the application of a model which considers the expected downstream tasks together with the pre-training corpus (Hiraoka et al., 2020), to the degree of jointly optimizing the tokenizer with the downstream model (Hiraoka et al., 2021). In addition to the massive changes in training and inference procedures this approach incurs, we note that it is difficult to apply to large contextualized models due to the long path from tokenization to prediction; SAGE overcomes this problem by "nudging" only the LLM vocabulary itself towards a contextualization-friendly segmentation.

The concept of subword tokenization made its rise alongside that of contextualized representations, meaning that little work exists where SKIP-GRAM or other static models are trained over proper subword segmentations. Recently, Kaushal and Mahowald (2022) did so for a proof-of-concept of a spelling prediction model, in lieu of training full LLMs. To our knowledge, no work to date has used a static embedding-based objective to score token sequence likelihood for a separate task (as we do for vocabulary trimming).

Finally, we acknowledge the recent efforts to do away with tokenization altogether, be it through character-only (Clark et al., 2022) or byte-only (Xue et al., 2022) models, or through encoding characters visually and passing them through a vision model (Salesky et al., 2021; Rust et al., 2022).

These represent an even more radical departure from the established application of LLMs, and we look forward to testing their abilities against our improved contextual subword tokenization methods. We note that while these models have been facing issues regarding scaling, mostly on the decoding side, SAGE vocabularies are ready to be used immediately within existing popular LLM implementations. Furthermore, recent work has shown the limited utility of character-level transformers in semantic tasks, even for morphologically rich languages with nontrivial orthography-morphology relations (Keren et al., 2022).

7 Conclusion

In this work, we introduced SAGE, a context-aware tokenizer built using insights from BPE, UNIGRAMLM, and SKIPGRAM, and showed that it achieves better results when used in an LLM-pretrain-then-fine-tune schema on two typologically distant languages on both the sequence and token levels. We believe that further investigation into incorporating context in tokenization models can improve results even further, and intend to also extend our efforts toward other languages and writing systems, as well as to multilingual tokenizers. For example, we plan to apply SAGE in the context of Abjads like Hebrew and Arabic, as well as languages written in alphasyllabaries such as Devanagari.

Within SAGE itself, there is room for improvement. The algorithm is still relatively slow, taking roughly a day to run on a strong CPU, making it difficult to apply to a truly large corpus, to start from a larger initial vocabulary, or to conduct exhaustive search over the hyperparameters. We intend to keep optimizing it, and continue evaluation against other subword and character-only schemas.

Limitations

We acknowledge several limitations of SAGE, a novel algorithm still in its development stages. First, scaling the vocabulary creation framework up from corpus-level unigram statistics to context dependence incurs many points where linear factors turn into quadratic, and worse. We introduced several heuristics to alleviate this issue in §3, however SAGE still takes longer to train compared to BPE and other tokenizers, by roughly a factor of ten. While having no effect on downstream pretraining and fine-tuning steps, it does mean hyperparameters are more difficult to tune. Second, the prohibitive resources required to implement a full LLM pipeline has limited our downstream evaluation setup to ten individual tasks on two languages. Ideally, as more languages with more diverse scripts and typological properties are examined, better generalizations can be made about the utility of integrating context into subword tokenizer vocabularies. Finally, we still do not have a well-formed theory of integrating multiple domains, languages, or scripts together into a single vocabulary. This question has interested researchers in recent years (e.g., Chung et al., 2020; Rust et al., 2021; Zhang et al., 2022), yet a tokenizer-internal solution (as opposed to data balance manipulation) still seems to have eluded the community. This question affects SAGE more than other tokenizers, given its reliance on context, which changes starkly when considering multiple sources of text in unison.

Acknowledgments

We thank Jacob Eisenstein and Cassandra Jacobs for work on earlier versions of the high-level idea, and Timo Schick and Lütfi Kerem Senel for fruitful conversations in early stages of the project. We thank Marco Cognetta, Michael Elhadad, Omer Levy, and attendees of ISCOL 2022 for comments and suggestions on more recent versions of the work. We thank the reviewers for their helpful comments. We thank Kaj Bostrom, Peter Izsak, and Tamar Levy for helping us obtain and operate resources for training our models.

References

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-ner: Massive multilingual named entity recognition. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 586–594. SIAM.

Ehsaneddin Asgari, Masoud Jalili Sabet, Philipp Dufter, Christopher Ringlstetter, and Hinrich Schütze. 2020. Subword sampling for low resource word alignment. *arXiv preprint arXiv:2012.11657*.

Tamali Banerjee and Pushpak Bhattacharyya. 2018. Meaningless yet meaningful: Morphology grounded subword-level nmt. In *Proceedings of the Second Workshop on Subword/Character Level Models*, pages 55–60.

Balthasar Bickel and Johanna Nichols. 2013a. Exponence of selected inflectional formatives. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*.

Max Planck Institute for Evolutionary Anthropology, Leipzig.

Balthasar Bickel and Johanna Nichols. 2013b. Inflectional synthesis of the verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4536–4546, Online. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

Ali Hakimi Parizi and Paul Cook. 2020. Evaluating sub-word embeddings in cross-lingual models. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2712–2719, Marseille, France. European Language Resources Association.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4238–4248, Hong Kong, China. Association for Computational Linguistics.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3042–3051, Online. Association for Computational Linguistics.

Peter Henderson, Mark S. Krass, Lucia Zheng, Neel Guha, Christopher D. Manning, Dan Jurafsky, and Daniel E. Ho. 2022. Pile of law: Learning responsible data filtering from the law and a 256gb open-source legal dataset.

Tatsuya Hiraoka. 2022. Maxmatch-dropout: Subword regularization for wordpiece. *arXiv preprint arXiv:2209.04126*.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2020. Optimizing word segmentation for downstream task. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1341–1351, Online. Association for Computational Linguistics.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2021. Joint optimization of tokenization and downstream model. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 244–255, Online. Association for Computational Linguistics.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3594–3608, Online. Association for Computational Linguistics.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train BERT with an academic budget. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10644–10652, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ayush Kaushal and Kyle Mahowald. 2022. What do tokens know about their characters and how do they know it? *arXiv preprint arXiv:2206.02608*.

Omri Keren, Tal Avinari, Reut Tsarfaty, and Omer Levy. 2022. Breaking character: Are subwords good enough for mrls after all? *arXiv preprint arXiv:2204.04748*.

Stav Klein and Reut Tsarfaty. 2020. Getting the ##life out of living: How adequate are word-pieces for modelling complex morphology? In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 204–209, Online. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Antonis Maronikolakis, Philipp Dufter, and Hinrich Schütze. 2021. Wine is not v i n. on the compatibility of tokenizations across languages. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2382–2399, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2):2.

Phillip Rust, Jonas F Lotz, Emanuele Bugliarello, Elizabeth Salesky, Miryam de Lhoneux, and Desmond Elliott. 2022. Language modelling with pixels. *arXiv preprint arXiv:2207.06991*.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

Elizabeth Salesky, David Etter, and Matt Post. 2021. Robust open-vocabulary translation from visual text representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7235–7252, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Elizabeth Salesky, Andrew Runge, Alex Coda, Jan Niehues, and Graham Neubig. 2020. Optimizing segmentation granularity for neural machine translation. *Machine Translation*, 34(1):41–59.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019. CrossWeigh: Training named entity tagger from imperfect annotations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5154–5163, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu,

Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Shiyue Zhang, Vishrav Chaudhary, Naman Goyal, James Cross, Guillaume Wenzek, Mohit Bansal, and Francisco Guzman. 2022. How robust is neural machine translation to language imbalance in multilingual tokenizer training? In *Proceedings of the 15th biennial conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 97–116, Orlando, USA. Association for Machine Translation in the Americas.

Final Vocab Size	16K
Initial Vocab Size	20K
k (tokens to prune each batch)	100
M (size of pruning candidate set)	1500
m (likelihood recalculation frequency)	10
l (embedding recalculation frequency)	4
SAGE window size	5
Word2Vec window size	5
Word2Vec vector dimension	50
Word2Vec negative samples	15

Table 6: Hyperparameters for vocabulary creation.

A Hyperparameters

In Table 6, 7, and 8, we present the hyperparameters used for training the various elements in our experiments.

B Computing Resources

For our experiments we used Quadro RTX 8000 GPU.

layer_norm_type	pytorch
model_type	bert-mlm
hidden_act	gelu
hidden_size	1024
num_hidden_layers	24
num_attention_heads	16
intermediate_size	4096
hidden_dropout_prob	0.1
attention_probs_dropout_prob	0.1
encoder_ln_mode	pre-ln
lr	1e-3
train_batch_size	4032
train_micro_batch_size_per_gpu	32
lr_schedule	time
curve	linear
warmup_proportion	0.06
gradient_clipping	0.0
optimizer_type	adamw
weight_decay	0.01
adam_beta1	0.9
adam_beta2	0.98
adam_eps	1e-6
total_training_time	24.0
optimizer_type	adamw
validation_epochs	3
validation_epochs_begin	1
validation_epochs_end	1
validation_begin_proportion	0.05
validation_end_proportion	0.01
validation_micro_batch	16
deepspeed	yes
data_loader_type	dist

Table 7: Hyperparameters for pre-training BERT-architecture models using the academic-budget-bert code (Izsak et al., 2021).

max_seq_length	128
evaluation_strategy	steps
per_device_train_batch_size	16
gradient_accumulation_steps	1
per_device_eval_batch_size	16
learning_rate	5e-5
weight_decay	0.1
max_grad_norm	1.0
lr_scheduler_type	polynomial
warmup_steps	50

Table 8: Hyperparameters for fine-tuning tasks using scripts from the academic-budget-bert package.