CANINE: Pre-training an Efficient Tokenization-Free Encoder for Language Representation

Jonathan H. Clark, Dan Garrette, Iulia Turc, John Wieting
Google Research
{jhclark,dhgarrette,iuliaturc,jwieting}@google.com

Abstract

Pipelined NLP systems have largely been superseded by end-to-end neural modeling, yet nearly all commonly-used models still require an explicit tokenization step. While recent tokenization approaches based on data-derived subword lexicons are less brittle than manually engineered tokenizers, these techniques are not equally suited to all languages, and the use of any fixed vocabulary may limit a model's ability to adapt. In this paper, we present CANINE, a neural encoder that operates directly on character sequences—without explicit tokenization or vocabulary—and a pre-training strategy that operates either directly on characters or optionally uses subwords as a soft inductive bias. To use its finer-grained input effectively and efficiently, CANINE combines downsampling, which reduces the input sequence length, with a deep transformer stack, which encodes context. CANINE outperforms a comparable mBERT model by 2.8 F1 on TYDI QA, a challenging multilingual benchmark, despite having 28% fewer model parameters.

1 Introduction

End-to-end neural models have generally replaced the traditional NLP pipeline, and with it, the error cascades and feature engineering common to such systems, preferring instead to let the model automatically induce its own sophisticated representations. Tokenization, however, is one of few holdovers from that era, with nearly all commonly-used models today requiring an explicit preprocessing stage to segment a raw text string into a sequence of discrete model inputs. Broadly speaking, tokenizers are generally either carefully constructed systems of language-specific rules, which are costly, requiring both manual feature engineering and linguistic expertise, or data-driven algorithms such as Byte Pair Encoding (Sennrich et al., 2016), WordPiece (Wu et al., 2016), or SentencePiece (Kudo and Richardson, 2018) that split strings based on frequencies in a corpus, which are less brittle and easier to scale, but are ultimately too simplistic to properly handle the wide range of linguistic phenomena that can't be captured by mere string-splitting (§2.1).

The degree of sophistication required to accurately capture the full breadth of linguistic phenomena, along with the infeasibility of writing such rules by hand across all languages and domains, suggests that explicit tokenization itself is problematic. In contrast, an end-to-end model that operates directly on raw text strings would avoid these issues, instead learning to compose individual characters into its own arbitrarily complex features, with potential benefits for both accuracy and ease of use. While this change is conceptually very simple—one could replace the subword vocabulary in a model like BERT (Devlin et al., 2019) with a vocabulary made solely of individual characters—doing so leads to two immediate problems. First, the computational complexity of a transformer (Vaswani et al., 2017), the main components in BERT as well as other state-of-the-art models such as GPT (Radford et al., 2019; Brown et al., 2020) and T5 (Raffel et al., 2020), grows quadratically with the length of the input. Since standard subword models have roughly four characters per subword on average, the 4x increase in input sequence length would result is a significantly slower model. Second, simply switching to a character vocabulary yields empirically poor results (§4.2).

In order to enable tokenization-free modeling that overcomes these obstacles, we present

CANINE: **C**haracter **A**rchitecture with **N**o tokenization **I**n **N**eural **E**ncoders.

Code and checkpoints will be made available on GitHub at caninemodel.page.link/code.

CANINE. CANINE is a large language encoder with a deep transformer stack at its core. Inputs to the model are sequences of Unicode characters.[1] To represent the full space of Unicode characters[2] without a vocabulary, we employ a hashing strategy. To avoid the potential slowdown from increasing the model's sequence length, we pass the inputs through strided convolutions to downsample the relatively large input sequence length into a much smaller sequence length in the deep transformer stack.

Like BERT, we pre-train CANINE on the Masked Language Model (MLM) and Next Sentence Prediction (NSP) tasks. For the MLM task, CANINE offers two options:

1. A fully character-level pre-training loss that autoregressively predicts characters within masked spans.

2. Alternatively, rather than predicting items at the same granularity as the input (i.e. individual characters), we instead predict the identities of subword tokens. Critically, however, this tokenization is used only for the target output of the pre-training task; the tokens are never passed as input to the encoder, and the tokenizer and subword vocabulary can be safely discarded after pre-training. By reading characters yet predicting subword tokens, we are effectively converting the hard token-boundary constraint found in other models into a soft *inductive bias* in CANINE. This design choice also results in a sharper decoupling of the training stages since pre-training no longer ties *fine-tuning* tasks to a particular tokenization and vocabulary.

In this article, we contribute:

- the first pre-trained tokenization-free deep encoder;
- an efficient model architecture that directly encodes long sequences of characters with speed comparable to vanilla BERT; and
- a model that performs no tokenization on the input, thus avoiding that lossy *information bottleneck*.

[1]We consider splitting on Unicode characters to be tokenization-free because it depends only on the (deterministic) process defined by the Unicode standard, and not on any models, hand-crafted rules, or external linguistic knowledge.

[2]Unicode defines 1,114,112 total "codepoints", of which only 143,698 are assigned to characters as of Unicode 13.0. This covers 154 scripts and over 900 languages.

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] The root contains only consonants; when conjugating, vowels, and sometimes consonants, are interleaved with the root. The root is not separable from its inflection via any contiguous split.

2 Motivation

2.1 Linguistic pitfalls of tokenization

Subword tokenizers are the de-facto standard in modern NLP. These include Byte Pair Encoding (BPE), WordPiece, and SentencePiece, which use vocabularies derived from statistical analyses of a corpus: a common string is more likely to be memorized as a unit, whereas rare strings are split into smaller constituents. While successfully adopted by state-of-the-art models (Devlin et al., 2019; Raffel et al., 2020; Brown et al., 2020), subword tokenizers are no panacea, with issues arising in both monolingual and multilingual contexts.

Even in high-resource languages, subword models still tend to struggle on challenging domains, such as informal text which often includes typos, spelling variation,[5] transliteration, or emoji (O'Connor et al., 2010). BERT, which uses WordPiece tokenization, is sensitive to corruptions of the input, both natural typos (Sun et al., 2020) and adversarial manipulations (Pruthi et al., 2019), with some of the loss attributable to corrupted strings no longer being covered by the vocabulary.

Subword algorithms are limited to only simple word-splitting operations. While this is perhaps a reasonable approach for a language with impoverished morphology such as English, it is much less appropriate in the face of phenomena like agglutinative morphology (Turkish, Greenlandic), non-concatenative morphology (Arabic, Hebrew), reduplication (Tagalog, Kiswahili), compounding (German, Japanese), consonant mutation (Welsh), vowel harmony (Finnish), and more.

Even seemingly safe heuristics used by these

[5]For example, speakers of Spanish and Italian may drop accents when typing.

algorithms, such as splitting on whitespace and punctuation, are problematic when applied to languages that do not use spaces between words (Thai, Chinese) or use punctuation as consonants (Hawaiian[6], Twi[7]). While SentencePiece does offer the option to skip whitespace splitting, it is not typically used due to poor empirical performance[8] showing that, when unconstrained by traditional word boundaries, it can produce arbitrarily long spans and degrade model performance perhaps due to excessive memorization (§2.2).

Fixed vocabulary methods can also force modelers to choose between difficult preprocessing tradeoffs: should one keep accents, casing, etc. and avoid destructive preprocessing?—Or keep such orthographic information and risk important words dropping out of the frequency-based vocabulary altogether due to the presence of multiple variants of otherwise-similar words? For instance, mBERT initially removed all diacritics, thus eliding tense information in Spanish[9] and conflating many unrelated words in Vietnamese.[10]

Finally, using a fixed vocabulary during pretraining also creates complications for downstream tasks, which are subsequently tied to the same tokenizer and vocabulary, even if not well-suited for the target domain and/or endtask. Boukkouri et al. (2020) showed that BERT's Wikipedia+BooksCorpus WordPiece vocabulary results in excessive segmentation when fine-tuning on medical data, diminishing the benefit of pretraining as a strategy.

2.2 Enabling better generalization

Much as Tenney et al. (2019) showed that large encoders learn elements of the classic NLP pipeline, it seems natural to let the model discover tokenization as well. With this in mind, we seek an approach that can better generalize beyond the orthographic forms encountered during pre-training.

In terms of scientific inquiry, we would like to know whether we can build models that learn how to *compose* words where appropriate, and *memorize* them where memorization is needed. Large frequency-derived vocabularies partially mitigate this problem by simply memorizing more, but language inherently requires aspects of both memorization and composition. If we model language without tokenizers, we can then study how and when our models are capable of these behaviors, potentially allowing models to go beyond our intuitions of how language *ought* to behave. By building a model that directly engages with these issues within the small scale of word composition, we hope to enable future work studying these problems at larger scales such as phrasal constructions.

Large vocabulary embedding matrices also suffer from the problem that they will likely have many infrequent vocabulary elements for which good embeddings will not be learned, since embeddings that are rarely accessed during pretraining will not be updated much beyond their random initializations. This can lead to missed opportunities for generalization. For instance, subword tokenizers like SentencePiece and byte-level BPE (Wang et al., 2019) prevent out-of-vocabulary tokens via byte-fallback; byte-level tokens might be poorly estimated given they are updated only in the absence of alternative segmentations. Generalization is also hindered for vocabulary elements that are slight orthographic variations, where one is very infrequent. For example, a model may estimate a very good embedding for a common vocabulary element *kitten* but a poor embedding for the less frequent element *kittens* since the model has no knowledge that they are related. On the other hand, a character-based model in which both words co-estimate shared weights should allow even infrequent words to receive good representations, provided they have some degree of overlap with more frequent words. While intuitively practitioners may assume that subword algorithms separate words into units that are semantically/linguistically reasonable, yielding a consistent root plus affixes; however, this is often not the case (see Table 3 in results).

2.3 Reducing engineering effort

Mature tokenizers often include years of hand-engineered rules around special cases such as hash

[4] From https://en.wikipedia.org/wiki/Arabic_verbs#Derivational_categories,_conjugations

[6] Hawaiian uses an apostrophe to indicate a glottal stop.

[7] Informal Twi uses a right paren) to represent the letter ɔ.

[8] https://github.com/google/sentencepiece/blob/master/doc/experiments.md

[9] Spanish past tense uses an accented final vowel.

[10] Vietnamese uses diacritics to indicate tones—often the only difference among several unrelated content words.

tags (O'Connor et al., 2010), email addresses, URLs, and handling unknown words;[11] even fairly minimal modern tokenizers include initial word-splitting heuristics followed by a specific algorithm and vocabulary for further breaking these tokens into subwords.

Preprocessing heuristics also change over time. Some of these tokenizer improvements may be intended to have large effects on the overall vocabulary and so should be kept with a particular model version to avoid mismatches with older vocabularies while other tokenizer improvements may be incremental fixes intended to roll out to existing models immediately, complicating versioning.

Modern pre-trained models also have many requirements throughout their lifecycle: Between the time a model is pre-trained, fine-tuned, and served—potentially months or years apart—its weights and model implementation may be converted to be compatible with another toolkit, its fine-tuning data may be tokenized in a different way, and the natural distribution of words may be quite different. All of these things introduce ample opportunities for mismatches to arise between tokenization and the vocabulary from pre-training. Yet this same pre-training paradigm presents an advantage for character-level models: we now have access to a far larger amount of (unsupervised) data to learn word composition from characters; without transfer learning, this has historically been impractical for many tasks having little supervised data.

3 CANINE

CANINE consists of three primary components: (1) a vocabulary-free technique for embedding text; (2) a character-level model that is efficient by means of downsampling and upsampling; and (3) an effective means of performing masked language modeling on a character-level model.

3.1 Model

CANINE is designed to be a minimally modified variant of the deep transformer stack found in modern encoders such as GPT, (m)BERT, XLM, and XLM-R such that its architecture is easily usable in other models in this family. The simplest implementation of such a model at the character-level would be to simply feed in characters at each position rather than subwords. However, this approach would result in far more sequence positions given the same input text, leading to linearly more compute in the transformer's feed forward layers and quadratically more compute in the transformer's self-attention layers.

The overall form of the CANINE model is the composition of a downsampling function DOWN, a primary encoder ENCODE, and an upsampling function UP;[12] given an input sequence of character embeddings $\mathbf{e} \in \mathbb{R}^{n \times d}$ with length n and dimensionality d:

$$\mathbf{Y}_{\text{seq}} \leftarrow \textsc{Up}\left(\textsc{Encode}\left(\textsc{Down}(\mathbf{e})\right)\right)$$

where $\mathbf{Y}_{\text{seq}} \in \mathbb{R}^{n \times d}$ is the final representation for sequence prediction tasks. Similarly, for classification tasks, the model simply uses the zeroth element of the primary encoder:

$$\mathbf{y}_{\text{cls}} \leftarrow \left[\textsc{Encode}\left(\textsc{Down}(\mathbf{e})\right)\right]_0$$

Preprocessing: Like existing models, the input to CANINE must ultimately be represented as a sequence of integers, but because the nature of characters is well-defined and standardized by Unicode, preprocessing code that would typically be hundreds or thousands of lines can be replaced by a very simple procedure: just iterate over the characters in the input string, and return their codepoint integer values (i.e., a single line[13] in Python). Furthermore, because codepoint values are part of the Unicode Standard, they are documented publicly, already supported by programming languages, and will not change over time, unlike arbitrary vocabulary-based IDs.

Character hash embeddings: CANINE embeds these codepoint integers using multiple hash functions, a vocabulary-free generalization of the word hash embedding trick (Svenstrup et al., 2017). This allows CANINE models to represent all 143k Unicode characters[14] with a relatively small number of parameters. And because the model always

[11] For example, should a subword containing an unknown character be a separate token, or should the unknown character be separated as its own token?

[12] Enveloping the attention stack between downsampling and upsampling layers is similar to the Funnel-Transformer (Dai et al., 2020), which operates on WordPiece. However, many of its design choices (e.g., average pooling, their residual structure) did not work for us, since we are dealing with fundamentally different representations.

[13] Python preprocessing: `[ord(c) for c in text]`

[14] https://unicode.org/versions/Unicode13.0.0

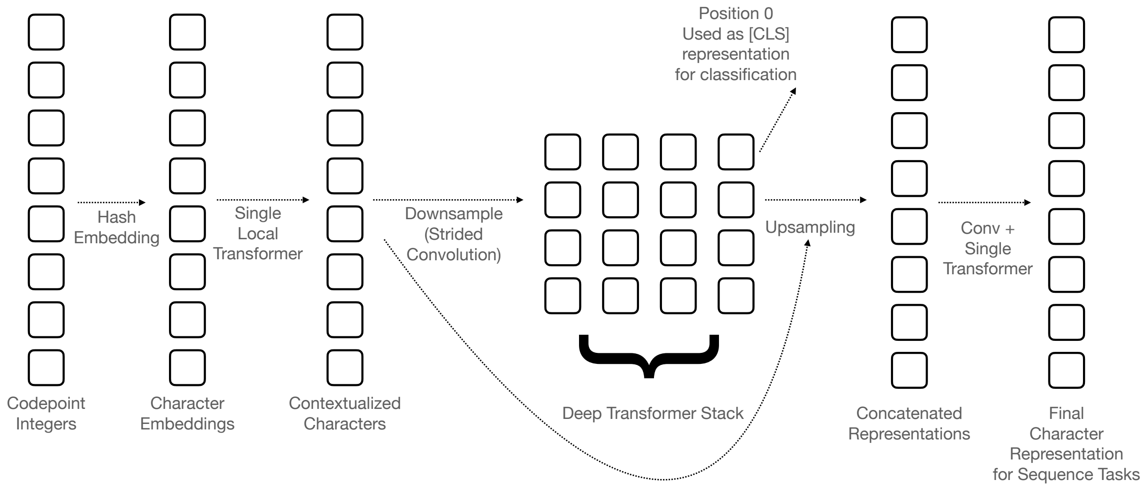

Figure 1: CANINE neural architecture.

supports all codepoints, it is possible to learn representations during fine-tuning for characters (and, by extension, words, scripts, etc) never seen during pre-training, while still leveraging what pre-training learned about word composition and sentence structure.

More formally, given a single codepoint[15] $x_i \in \mathbb{N}$, we look up an embedding slice from each of B hash buckets, each of which has dimensionality $d' = {}^{d}/_{K}$ such that each hash function performs a LOOKUP into its own embedding matrix in $\mathbb{R}^{B \times d'}$:

$$e_i \leftarrow \bigoplus_{k}^{K} \text{LOOKUP}_k \left(\mathcal{H}_k(x_i) \,\%\, B,\ d' \right)$$

where $\oplus$ denotes vector concatenation. That is, a single character embedding of dimension d is created by concatenating K slices of size d', where each slice is taken from a different embedding matrix, and determined by a different hash function. While each individual hash function is subject to hash collisions, the effect of this on the model is minimal since it only affects a small portion of the codepoint's overall embedding. We refer to these as the character embeddings $\mathbf{e} \in \mathbb{R}^{n \times d}$. In our experiments, we use $d = 768$, $K = 8$, and $B = 16\text{k}$.

Downsampling: To make CANINE efficient, we use a multi-part downsampling strategy. First, we encode characters using a single-layer block-wise local attention transformer. This model performs self-attention only within each block of a pre-defined size,[16] saving the quadratic cost of attention while leveraging the linguistic intuition that word composition—i.e., the kind of composition relevant in the lowest layers of the model (Tenney et al., 2019)—tends to happen at a very local level. Next, we use a strided convolution to reduce the number of sequence positions to be similar to that of a word piece model.[17] (Jiang et al., 2020) Given character embeddings $\mathbf{e} \in \mathbb{R}^{n \times d}$ with a sequence length of n characters and dimensionality d, we use a strided convolution to downsample by a rate of r:

$$\mathbf{h}_{\text{init}} \leftarrow \text{LOCALTRANSFORMER}_1(\mathbf{e})$$
$$\mathbf{h}_{\text{down}} \leftarrow \text{STRIDEDCONV}(\mathbf{h}_{\text{init}},\ r)$$

We refer to this output as the *downsampled positions*: $\mathbf{h}_{\text{down}} \in \mathbb{R}^{m \times d}$ where $m = {}^{n}/_{r}$ is the number of downsampled positions. In our experiments, we use $r = 4$ and $n = 2048$ such that $m = 512$, giving CANINE's primary encoder—the transformer stack—the same form as mBERT.

Deep transformer stack: After downsampling, CANINE applies a deep transformer stack with L layers to the resulting downsampled positions. This is the same as the core of BERT and derivative models and remains the core of CANINE in that it accounts for the vast majority of its compute and parameters. We note that this middle portion of the model could easily be replaced with any other sequence-to-sequence model including those with better compute performance such as

[15] Conceptually, each **codepoint** is a character; however, the definition of a Unicode codepoint is precise and unambiguous.

[16] We use a window of 128 characters in our experiments.

[17] In our experiments, we found a downsampling rate of 4X to result in high quality with a speed comparable to BERT.

Performer (Choromanski et al., 2020), Big Bird (Zaheer et al., 2020), RFA (Peng et al., 2021), ETC (Ainslie et al., 2020), etc. This portion of the model yields a new downsampled representation $\mathbf{h}'_{\text{down}} \in \mathbb{R}^{m \times d}$:

$$\mathbf{h}'_{\text{down}} \leftarrow \text{TRANSFORMER}_L(\mathbf{h}_{\text{down}})$$
$$\mathbf{y}_{\text{cls}} = [\mathbf{h}'_{\text{down}}]_0$$

Upsampling: While the above architecture is sufficient for classification tasks, sequence prediction tasks require that the model expose an output layer with the same sequence length as the input (i.e. characters remain the input and output "API" of the model for tasks such as tagging and span prediction). We reconstruct a character-level output representation by first concatenating the output of the original character transformer (above) with the downsampled representation output by the deep transformer stack (i.e. each downsampled position is associated with exactly 4 characters for a downsampling rate of 4, and thus the positions of the deep transformer stack are replicated before concatenation). More formally,

$$\mathbf{h}_{\text{up}} \leftarrow \text{CONV}\left(\mathbf{h}_{\text{init}} \oplus \mathbf{h}'_{\text{down}}, w\right)$$
$$\mathbf{y}_{\text{seq}} \leftarrow \text{TRANSFORMER}_1\left(\mathbf{h}_{\text{up}}\right)$$

where $\oplus$ indicates vector concatenation of the representations (i.e. not sequences) such that CONV projects from $\mathbb{R}^{n \times 2d}$ back to $\mathbb{R}^{n \times d}$ across a window of w characters.[18] Applying a final transformer layer (standard, not local) yields a final sequence representation $\mathbf{y}_{\text{seq}} \in \mathbb{R}^{n \times d}$.

Residual connections: While the initial character encoder (before downsampling) and final character encoder (after upsampling) both represent character *positions*, they conceptually have very different purposes in the network. Intuitively, we think of the initial character encoder as composing characters to create a more word-like representation, while the final character encoder is extracting the in-context representation that's relevant for predicting the "meaning" of the content at each position; CANINE must be able to deal with additional ambiguity here since a single downsampled position may span more than one word. Because of the different roles of these induced features, we do *not* use residual connections between $\mathbf{h}_{\text{init}}$ and $\mathbf{h}_{\text{up}}$.

[18] We use $w = 4$ in our experiments.

3.2 Pre-training

Recent pre-trained models ranging from BERT to T5 have largely used variations on a masked language model (MLM) task (also known as a *span corruption task*) as an unsupervised pre-training loss function—a means of generating synthetic examples that are not from any realistic task, yet prepare a model to learn realistic tasks in future phases of training (i.e., fine-tuning). The CANINE pre-training procedure retains the MLM task, but with modifications that are relevant in the context of a character-based model. CANINE offers two pre-training strategies, both of which yield a tokenization-free model following pre-training. In our experiments, we use only one loss at a time.

3.2.1 Autoregressive Character Loss

Span-wise masking: CANINE's autoregressive character loss masks several character spans within each sequence. These spans are chosen based on whitespace boundaries. No punctuation splitting nor other heuristics are used. All characters within the masked span are replaced by a special "mask" codepoint in the input.[19] Random replacements are chosen from the vocabulary of same-length subwords.

Span prediction: CANINE autoregressively predicts the masked characters. The order of the masked positions is shuffled such that masked context is not necessarily revealed left-to-right, but rather a single character at a time.

3.2.2 Optional Subword Loss

Span-wise masking: Rather than using whitespace to determine boundaries for contiguous mask spans, in CANINE's subword loss each span corresponds to a single subword. As with the autoregressive loss, all characters within the masked span are replaced with a special mask codepoint. No random subword replacement is performed as there is no subword vocabulary.[20]

Span prediction: Within each masked character span, CANINE's subword loss randomly selects a character position where the model will make a prediction; the model predicts the identity of the masked subword via softmax. The associated subword embeddings are discarded after pre-training.

[19] We use Unicode characters in the private use block such that the input remains a valid Unicode string.

[20] Though we expect that future work on vocabulary-free random replacement may improve quality.

Model	Input	MLM	r	Length	Example / sec	Params	TyDiQA SelectP	TyDiQA MinSpan
mBERT (public)	Subwords	Subwords	–	512	–	179M	63.1	50.5
mBERT (ours)	Subwords	Subwords	–	512	9000	179M	63.2	51.2
	Chars	Single Chars	1	2048	925	127M	59.5 (-3.7)	43.7 (-7.5)
	Chars	Subwords	1	2048	900	127M	63.8 (+0.6)	50.2 (-1.0)
Canine-S	Chars	Subwords	4	2048	6400	127M	**66.0** (+2.8)	52.5 (+1.3)
Canine-C	Chars	Autoreg. Chars	4	2048	6050	127M	65.7 (+2.5)	**53.0** (+1.8)

Table 2: Direct comparison between mBERT (rows 1–2) and CANINE (rows 5–6) on TyDi QA. Public mBERT results are taken from the TyDi QA paper. Rows 3 and 4 show simple baselines that yield inefficient / low-quality performance. Despite operating on 4x more sequence positions, CANINE remains comparable to mBERT in terms of speed. Pre-training example/sec are shown for our reported hardware (see Setup, §4.1). r represents the ratio for downsampling. Parameters are calculated at fine-tuning time. All results are averaged over 3 fine-tuning replicas. TYDI QA scores are F1 scores, macro-averaged across languages. Deltas from our mBERT (the most comparable baseline) are shown in parentheses.

3.2.3 Targeted Upsampling

By design, each final character representation (after upsampling) is a function of the output of the initial character encoder (before downsampling) and the output of the deep transformer stack—there are no inter-position dependencies across the upsampled sequence. This depends on the upsampler using position-wise feed-forward projections and a single transformer layer. During pre-training, we leverage this design to improve speed by only performing upsampling on the sequence positions that will be used by the MLM task $\mathbf{p}$. More formally, we use the following equivalent form of the UP function during pre-training:

$$\mathbf{h}^*_{\text{up}} \leftarrow \text{GATHER}\left(\mathbf{p},\ \mathbf{h}_{\text{up}}\right)$$
$$\mathbf{y}^*_{\text{seq}} \leftarrow \text{TRANSFORMER}_1\left(\text{Q} = \mathbf{h}^*_{\text{up}}, \text{KV} = \mathbf{h}_{\text{up}}\right)$$

3.2.4 Modularity

Unlike previous models, CANINE removes both the vocabulary and tokenization algorithm as fossilized parts of the final model that must be replicated during fine-tuning and prediction. Regardless of which pre-training loss is chosen (characters or subwords), the use of these components in CANINE is limited to a detail of the pre-training procedure—an *inductive bias* of the loss function—that is then discarded. The fine-tuning and prediction phases of the model lifecycle never have any knowledge of what vocabulary or tokenization algorithm (if any) were used in pre-training. This allows the model to natively process untokenized data, or even process data that has been pre-processed by different tokenizers, a situation that would otherwise introduce a significant skew between training phases.

4 Experiments

4.1 Experimental Setup

TYDI QA: Primary Tasks TYDI QA is a dataset of information-seeking questions in 11 typologically diverse languages (Clark et al., 2020). Questions are written before answers, leading to less lexical and morphological overlap between the questions and their answers, which are drawn from each language's Wikipedia. We evaluate on the two primary tasks.[21]

Passage Selection Task (SELECTP): Given a list of the passages in a Wikipedia article, return either the index of the passage that answers the question, or return `NULL` if the article contains no acceptable answer.

Minimal Answer Span Task (MINSPAN): Given the full text of a Wikipedia article, return the start and end byte indices of the minimal span that completely answers the question. Alternatively, a system may indicate that the article does not contain an answer, or return `YES` or `NO` for yes/no type questions.

Direct Comparison with mBERT In order to determine which pre-training architecture produces better quality downstream predictions, we compare CANINE to mBERT, which we re-implemented and re-trained in order to hold as many variables as possible constant. Note that we

[21]As opposed to the simplified TYDIQA-GOLDP task, which is part of the XTREME meta-benchmark.

intentionally do *not* compare against public pre-trained checkpoints that use different pre-training corpora since (a) this would be a major confounding variable and (b) most publicly available pre-trained models are simply instantiations of BERT, including XLM-R[22] and X-STILTS.[23]

Setup We pre-train on the multilingual Wikipedia data of mBERT, which includes 104 languages. Similarly, we reuse mBERT's exponential smoothing technique to weight the languages within the pre-training samples. We train for 124k steps with batch size 4096 (2.5 passes over the data) using the LAMB optimizer (You et al., 2020) with a linearly decayed learning rate of 0.018 where 2.5% of the steps are used for warm-up. We use a sequence length of 512 for mBERT, and 2048 for CANINE, which results in 512 downsampled positions in its core deep transformer stack. We pre-train on 64 Cloud TPUs (v3). For both mBERT and CANINE-S (CANINE with the subword loss), we select 15% of subwords for the MLM loss and predict up to 80 output positions; 80% of these are masked in the input, 10% are randomly replaced, and 10% are unmodified. For CANINE-C (CANINE with the autoregressive character loss), we select 15% of contiguous spans for the MLM loss and predict up to 320 output characters, and no random replacement is performed. For TYDI QA, we use a maximum answer length of 100 characters, which is approximately the 99$^{\text{th}}$ percentile answer length.

4.2 Results

Our main result is shown in Table 2. CANINE-S (CANINE with the subword loss) improves over mBERT in the TYDI QA SELECTP task by 2.8 F1, while using about 30% fewer parameters. Similarly, CANINE-C (CANINE with the autoregressive character loss), improves over mBERT by 2.5 F1.

We also present results from some ablation models as additional baselines in rows 3-4 of Table 2. First, for row 3, we simply replace BERT's subword vocabulary with a pure character vocabulary, which makes characters both the input granularity and the unit of masking and prediction for the MLM task, and observe that not only is the model 10X slower than subword-based BERT, but the quality also suffers greatly. Then, for row 4, we modify that model to use subwords for masking and MLM predictions, while keeping characters as the input granularity, and we see a substantial quality improvement, though pre-training remains extremely slow. Finally, by comparing to the full CANINE model in row 5, we can see that adding the downsampling strategy improves speed by 700%, and also leads to an additional small bump in quality. We speculate that this additional quality gain comes from giving the model a better inductive bias toward more word-like units within the deep transformer stack.

Analysis CANINE fares particularly well on morphologically rich languages such as Kiswahili. Table 3 shows examples where CANINE outperforms mBERT on the TYDI QA SELECTP task. In particular, we observe examples where Kiswahili's rich morphology does not hinder the matching process for CANINE.

4.3 Ablations

In Table 4, we consider minor modifications to the final CANINE architecture, and evaluate effect of each on the downstream quality of the model.

Attending directly to $\mathbf{h}'_{\text{down}}$ We use $\mathbf{h}_{\text{up}}$ as the query sequence to the transformer in order to get a character-like sequence while using $\mathbf{h}'_{\text{down}}$ as the key-value sequence to the transformer results in far less attention computation such that:

$$\mathbf{y}^{+}_{\text{seq}} = \text{TRANSFORMER}_1\big(\text{Q} = \mathbf{h}_{\text{up}}, \text{KV} = \mathbf{h}'_{\text{down}}\big)$$

While this change reduces the overall FLOPS of the model, it does not have a major effect on pre-training throughput while it does substantially degrade quality.

Number of hash buckets We reduce the number of hash buckets from 16k to 8k, which significantly hinders the model's performance in the MINSPAN task.

Character vocab We switch from our hash-based no-vocabulary strategy to using a normal character vocabulary (which we derive from the pre-training corpus). We observe that this underperforms the hashing approach. We speculate that this might be due to skew between the pre-training corpus and the final downstream task.

[22]XLM-R instantiates BERT with a larger pre-training corpus, larger model size, and larger vocabulary size.

[23]X-STILTS performs English fine-tuning on an existing XLM-R checkpoint. (Phang et al., 2020)

Question	Passage Answer
Chelsea ina **milikiwa** na nani?	Kwa kawaida Chelsea huvaa jezi ya blu, kaptula blu na soksi nyeupe. Nembo ya klabu imebadilishwa mara nyingi kulingana na wakati na kuboresha muonekano wa klabu. Nembo ya sasa inaonesha picha ya simba akiwa amebeba mkuki. Tangu Julai 2003, Chelsea imekuwa iki**milikiwa** na Bilionea wa Kirusi, Roman Abramovich.
Who owns Chelsea?	*Chelsea usually wear blue jerseys, blue shorts and white socks. The club logo has been changed many times over time and improved the club's appearance. The current emblem shows a picture of a lion carrying a spear. Since July 2003, Chelsea has been owned by Russian billionaire Roman Abramovich.*
Kampuni **isambazayo** umeme nchini Kenya inaitwaje? *What is the name of the company that distributes electricity in Kenya?*	Kenya Power and Lighting (KPLC) ni kampuni inayohusika na maambukizi ya umeme na **usambazaji** wa umeme nchini Kenya. *Kenya Power and Lighting (KPLC) is a company responsible for electricity transmission and distribution in Kenya.*

Table 3: Kiswahili examples in which CANINE improved over mBERT in the TYDI QA SELECTP task. On examining the mBERT's subword tokenization, we observe that the segmentations do not align well, putting more pressure on the model to combine them and more opportunities for some embeddings to be poorly estimated. **Top:** The model must match a key word in the question *milikiwa* (*own*) to a morphological variant in the answer *iki-milikiwa* (*to be owned*). mBERT's WordPiece segmentation produces *milik -iwa* and *iki -mi -iki -wa* for these, respectively. **Bottom:** The model must match *i-sambaza-yo* (*distributes*) in the question with *u-sambaza-ji* (*distribution*). mBERT's WordPiece segmentation produces *isam -ba -za -yo* and *usa -mba -zaj -i*.

Input character dimension We reduce the embedding size of the initial character encoder (i.e. the embedding size of $\mathbf{h}_{\text{init}}$ and $\mathbf{e}$—not $\mathbf{h}_{\text{up}}$ nor $\mathbf{y}_{\text{seq}}$) and observe that quality falls off rapidly.

No initial transformer We remove the local transformer from $\mathbf{h}_{\text{init}}$ and similarly observed a marked reduction in quality.

Increased downsampling While more aggressive downsampling (a factor of 5X or 6X, rather than 4X) brings substantial speed gains, the passage-level quality degrades substantially and the minimal span predictions suffer even more.

No position-limited MLM When we do not use the trick of applying the final character transformer ($\mathbf{y}_{\text{seq}}$) only to the positions that will be computed by the MLM task, we observe a large reduction in speed. Since this model is theoretically equivalent in terms of operations, we show only the speed for exposition.

5 Related Work

5.1 Improvements to subword tokenization

Further improvements to standard subword tokenization like Byte Pair Encoding (BPE) (Sennrich et al., 2016), WordPiece (Wu et al., 2016), and SentencePiece (Kudo and Richardson, 2018) have been proposed. Subword regularization (Kudo, 2018) and BPE-dropout (Provilkov et al., 2020) recognize that deterministic segmentation during training limits the ability to leverage morphology and word composition; instead, they sample at random one of the multiple tokenizations of the training input, made possible by the inherent ambiguity of subword vocabularies. Wang et al. (2021) recently expanded on this paradigm to enforce consistency of predictions over different segmentations. Unigram LM (Kudo, 2018), a segmentation technique that builds its vocabulary top-down, was shown to align with morphology better than BPE on modern pre-trained encoders (Bostrom and Durrett, 2020).

Other efforts have built hybrid models that operate on multiple granularities, combining characters with whitespace-separated tokens (Luong and Manning, 2016) or different subword vocabularies (Zhang and Li, 2020).

5.2 Character-level models

Following the larger NLP trend, character-level n-gram models (Huang et al., 2013; Wieting et al., 2016; Bojanowski et al., 2017) have mostly been replaced by neural networks. While generally lagging behind their word-level counterparts, character-level models are crucial for morphologically rich languages and special linguistic phenomena (Garrette and Baldridge, 2013).

For language modeling Character language models (CLMs) have used vanilla RNN architec-

Condition	Example / sec	TyDiQA SelectP F1	TyDiQA MinSpan F1
Attend to $\mathbf{h}'_{\text{down}}$ (instead of $\mathbf{h}_{\text{up}}$)	6400	64.5	52.2
8k codepoint hash buckets (instead of 16k)	6400	64.1 (-0.4)	50.5 (-1.7)
Character vocab (no hashing)	6400	64.6 (+/-)	51.2 (-1.0)
Input character dim 384 (instead of 768)	6600	62.9 (-1.2)	49.3 (-1.2)
Input character dim 192 (instead of 768)	6400	61.7 (-2.4)	47.3 (-3.2)
No initial character transformer	6700	63.2 (-1.4)	48.3 (-2.9)
Downsample by a factor of 5 (instead of 4)	7000	62.9 (-1.7)	49.2 (-2.0)
Downsample by a factor of 6 (instead of 4)	9200	62.7 (-1.9)	47.6 (-3.6)
Don't limit final character transformer to MLM positions	5200	—	—
CANINE-S	6400	66.0	52.5

Table 4: Ablation experiments on the CANINE model. Deltas are shown in parentheses with regard to the top-most experiment, which serves as the baseline configuration for all experiments in this table. Each result is averaged over 3 fine-tuning and evaluation replicas.

tures to produce distributions over sequences of characters in a purely *tokenization-free* manner (Sutskever et al., 2011; Graves, 2013; Hwang and Sung, 2017; Radford et al., 2018). Hierarchical RNNs modeled the assumption that language operates on increasing layers of abstraction: Chung et al. (2017) jointly trained a sub-module to segment the character-level input into larger spans at each layer of a stacked LSTM.

Due to the consistent lag in performance behind their word-level counterparts, attention shifted from pure CLMs towards merely *character-aware* models, still reliant on traditional tokenization. Some hybrid models processed the input at character level, but predicted words from a closed vocabulary (Kim et al., 2016; Gerz et al., 2018). Others reintroduced explicit tokenization on the input side, and either generated bursts of character sequences that formed an open vocabulary (Kawakami et al., 2017) or used a character-only generator as a fallback when the main closed-vocabulary word generator produced a rare or unknown token (Matthews et al., 2019; Mielke and Eisner, 2019). Especially after the popularization of the inherently ambiguous subword vocabularies like BPE, several studies removed the assumption of a single correct input segmentation and marginalized over all possible segmentations (Van et al., 2017; Buckman and Neubig, 2018; Grave et al., 2019).

Coming full circle, Kawakami et al. (2019) induced a lexicon without any explicit supervision, reverting back to pure CLMs. In a revitalized effort to bring them on-par with coarser granularities, researchers leveraged external resources such as grounding in vision (Kawakami et al., 2019) or multi-task learning together with supervised morphology tasks (Blevins and Zettlemoyer, 2019).

After the transformer (Vaswani et al., 2017) replaced RNNs as the dominant architecture in NLP, character-level models followed. Al-Rfou et al. (2019) showed that byte-level vanilla Transformers significantly underperform their word-level counterparts. A similar finding was reported by Radford et al. (2019). Although the gap has been reduced (Choe et al., 2019), subword transformers remain the status quo for pure language modeling.

For specific tasks In parallel with LM efforts, the neural machine translation (NMT) community sought to solve its open-vocabulary problem via character-level modeling. Luong and Manning (2016) proposed a hybrid model that operated mainly at word level, but consulted a character-level LSTM for unknown words; this was a practical compromise, as their character-only model took 3 months to train. Lee et al. (2017) enabled pure character NMT by shortening the input length via convolutional, pooling, and highway layers. Notably, their many-to-English model outperformed its subword counterpart and most bilingual baselines, with a 35% increase in training time (on a single GPU) compared to a baseline BPE-to-char model. CANINE has a similar motivation, but operates in the context of pre-trained transformers; training is 7x faster compared to a char-to-char baseline (on TPU v3), and has a 28% increase in training time over mBERT (Table 2).

Character-level information has been leveraged for many other end tasks as well, including: text classification (Zhang et al., 2015; Zhang and Lecun, 2017), part-of-speech tagging and named entity recognition (Gillick et al., 2016; Akbik et al., 2018; Pinter et al., 2019), named entity detection (Yu et al., 2018), dependency parsing (Vania et al., 2018), and machine reading comprehension (Hewlett et al., 2018). Character information proved particularly useful for low-resource languages (Xie et al., 2018), phenomena such as code-switching and transliteration (Ball and Garrette, 2018), and rich morphology (Vania and Lopez, 2017), which had previously received special modeling including adaptor grammars (Botha and Blunsom, 2013).

For transfer learning Token-based models have also been augmented with character-level information in the context of transfer learning, where encoders trained with unsupervised objectives are repurposed to solve downstream tasks. Pinter et al. (2017) addressed the out-of-vocabulary problem of static pre-trained word embeddings by training a model to map the surface of a word to its pre-trained representation, and used it on unknown words. ELMo (Peters et al., 2018), a bidirectional LSTM model, applied character convolutions to its whitespace-separated input tokens. CharacterBERT (Boukkouri et al., 2020) ported this technique to BERT (Devlin et al., 2019), augmenting its existing WordPiece-tokenized input. Consistent with previous observations that feeding characters into a transformer stack comes with a huge computational cost while not improving over tokenization-based approaches (Al-Rfou et al., 2019), a BERT model fine-tuned for semantic parsing achieved gains only when characters *complemented* subwords (van Noord et al., 2020).

5.3 Multilingual models

Recently, multilingual NLP has been dominated by deep pre-trained multilingual models whose subword vocabularies are shared across languages. Such models borrow their architectures from monolingual predecessors and apply joint training in over 100 languages, either with unsupervised LM objectives: mBERT (Devlin et al., 2019), mT5 (Xue et al., 2020), or with additional translation objectives: XLM (Lample and Conneau, 2019), XLM-R (Conneau et al., 2020).

Despite unprecedented quality and engineering convenience, these models suffer from *the curse of multilinguality* (Conneau et al., 2020) and disfavor low-resource languages. Modeling-based solutions injected per-language adapters in the network (Artetxe et al., 2020; Pfeiffer et al., 2020). Others revisited the shared multilingual vocabulary: Chung et al. (2020) formed language clusters and unioned per-cluster vocabularies, while Xu et al. (2020) added a pre-pre-training stage to their Chinese-Japanese model, replacing characters with their morphological clusters. In order to accommodate for languages unseen during pre-training, Wang et al. (2020) proposed to extend the original vocabulary and continue pre-training.

While shared subword vocabularies proved to be a practical compromise that allows handling multiple languages within the same network, they are suboptimal when targeting a specific language; recent work reports gains from customized single-language vocabularies (Delobelle et al., 2020).

To the best of our knowledge, CANINE is the first character-level pre-trained deep encoder that is entirely tokenization-free, in both monolingual and multilingual literature.

6 Future Work

In this work, we have focused on evaluating CANINE on established community benchmarks. However, CANINE has the potential to be even more effective on noisy text, such as that encountered on the web or on social media where misspellings and creative use of orthography are much more common—this is true even for isolating languages[24] such as English. Similarly, CANINE is designed with the linguistic properties of morphologically rich languages in mind, including agglutinative, infix, and polysynthetic morphology. Further evaluation is needed to test prediction quality under these conditions.

CANINE also opens up the opportunity to use multiple knowledge sources as sources of inductive bias at pre-training time, even if they have inconsistent token boundaries. For example, it is possible to use multiple segmenters, vocabularies, NER systems, etc. in the MLM task since all boundaries can trivially be expressed in terms of character boundaries and downstream tasks need not have any knowledge of those components' tokenization schemes when used in CANINE.

[24]Isolating languages tend to have a low morpheme-to-word ratio due to using very little inflectional morphology.

7 Conclusion

In this article, we described CANINE, which is, to our knowledge, the first pre-trained encoder for language understanding that uses a tokenization-free, vocabulary-free model, while surpassing the quality of models built on top of heuristic tokenizers. CANINE eliminates many engineering pitfalls for practitioners, and opens up new research directions for the community.

Acknowledgements

The authors wish to thank Noah Constant, Rami Al-Rfou, Kristina Toutanova, Kenton Lee, Ming-Wei Chang, and Tim Dozat for their feedback on this work. We would also like to thank Martin Njoroge and Nanjala Misiko for their consultations on the Swahili examples, Diana Akrong for consulting on Twi orthography, and Waleed Ammar for consulting on Arabic morphology.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding Long and Structured Inputs in Transformers. In *Proc. of EMNLP*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proc. of COLING*.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. In *Proc. of AAAI*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proc. of ACL*.

Kelsey Ball and Dan Garrette. 2018. Part-of-Speech Tagging for Code-Switched, Transliterated Texts without Explicit Language Identification. In *Proc. of EMNLP*.

Terra Blevins and Luke Zettlemoyer. 2019. Better character language modeling through morphology. In *Proc. of ACL*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

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

Jan A. Botha and Phil Blunsom. 2013. Adaptor Grammars for learning non-concatenative morphology. In *Proc. of EMNLP*.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Junichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters. In *Proc. of COLING*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proc. of NeurIPS*.

Jacob Buckman and Graham Neubig. 2018. Neural lattice language models. *TACL*.

Dokook Choe, Rami Al-Rfou, Mandy Guo, Heeyoung Lee, and Noah Constant. 2019. Bridging the Gap for Tokenizer-Free Language Models. *arXiv preprint arXiv:1908.10322*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Rethinking Attention with Performers. *arXiv preprint arXiv:2009.14794*.

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving Multilingual Models with Language-Clustered Vocabularies. In *Proc. of EMNLP*.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical Multiscale Recurrent Neural Networks. *arXiv preprint arXiv:1609.01704v7*.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for Information-Seeking Question Answering in Typologically Diverse Languages. *TACL*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proc. of ACL*.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. 2020. Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. In *Proc. of NeurIPS*.

Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. RobBERT: a Dutch RoBERTa-based Language Model. *arXiv preprint arXiv:2001.06286*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL*.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proc. of NAACL*.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018. Language Modeling for Morphologically Rich Languages: Character-Aware Modeling for Word-Level Prediction. *TACL*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual Language Processing From Bytes. In *Proc. of NAACL*.

Edouard Grave, Sainbayar Sukhbaatar, Piotr Bojanowski, and Armand Joulin. 2019. Training hybrid language models by marginalizing over segmentations. In *Proc. of ACL*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2018. Byte-Level Machine Reading across Morphologically Varied Languages. In *Proc. of AAAI*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*.

Kyuyeon Hwang and Wonyong Sung. 2017. Character-Level Language Modeling with Hierarchical Recurrent Neural Networks. In *Proc. of ICASSP*.

Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. ConvBERT: Improving BERT with Span-based Dynamic Convolution. In *Proc. of NeurIPS*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2017. Learning to create and reuse words in open-vocabulary neural language modeling. In *Proc. of ACL*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2019. Learning to Discover, Ground and Use Words with Segmental Neural Language Models. In *Proc. of ACL*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-Aware Neural Language Models. In *Proc. of AAAI*.

Taku Kudo. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proc. of ACL*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proc. of EMNLP: System Demonstrations*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. In *Proc. of NeurIPS*.

Jason Lee, Eth Zürich, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *TACL*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proc. of ACL*.

Austin Matthews, Graham Neubig, and Chris Dyer. 2019. Using Morphological Knowledge in Open-Vocabulary Neural Language Models. In *Proc. of NAACL*.

Sebastian J. Mielke and Jason Eisner. 2019. Spell once, summon anywhere: A two-level open-vocabulary language model. In *Proc. of AAAI*.

Rik van Noord, Antonio Toral, and Johan Bos. 2020. Character-level Representations Improve DRS-based Semantic Parsing Even in the Age of BERT. In *Proc. of EMNLP*.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory Search and Topic Summarization for Twitter Introduction and Description. In *Proceedings of the International AAAI Conference on Web and Social Media*.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random feature attention. In *Proc. of ICLR*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer. *arXiv preprint arXiv:2005.00052*.

Jason Phang, Iacer Calixto, Phu Mon Htut, Yada Pruksachatkun, Haokun Liu, Clara Vania, Katharina Kann, and Samuel R Bowman. 2020. English Intermediate-Task Training Improves Zero-Shot Cross-Lingual Transfer Too. In *Proc. of AACL*.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proc. of EMNLP*.

Yuval Pinter, Marc Marone, and Jacob Eisenstein. 2019. Character Eyes: Seeing Language through Character-Level Taggers. In *Proc. of BlackboxNLP*.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-Dropout: Simple and Effective Subword Regularization. In *Proc. of ACL*.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proc. of ACL*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2018. Learning to generate reviews and discovering sentiment. *OpenReview*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of ACL*.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-BERT: BERT is not robust on misspellings! Generating nature adversarial samples on BERT. *arXiv preprint arXiv:2003.04985*.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proc. of ICML*.

Dan Svenstrup, Jonas Meinertz Hansen, and Ole Winther. 2017. Hash Embeddings for Efficient Word Representations. In *Proc. of NeurIPS*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proc. of ACL*.

Bart Van, Merriënboer Mila, Amartya Sanyal, Hugo Larochelle Google, Brain Yoshua, Bengio Mila, and Cifar Fellow. 2017. Multiscale sequence modeling with a learned dictionary. *arXiv preprint arXiv:1707.00762v2*.

Clara Vania, Andreas Grivas, and Adam Lopez. 2018. What do character-level models learn about morphology? The case of dependency parsing. In *Proc. of EMNLP*.

Clara Vania and Adam Lopez. 2017. From Characters to Words to in Between: Do We Capture Morphology? In *Proc. of ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2019. Neural machine translation with byte-level subwords. In *Proc. of AAAI*.

Xinyi Wang, Sebastian Ruder, and Graham Neubig. 2021. Multi-view subword regularization. *arXiv preprint arXiv:2103.08490*.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020. Extending multilingual BERT to low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proc. of EMNLP*.

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*.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proc. of EMNLP*.

Canwen Xu, Tao Ge, Chenliang Li, and Furu Wei. 2020. UnihanLM: Coarse-to-Fine Chinese-Japanese Language Model Pretraining with the Unihan Database. In *Proc. of AACL*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mT5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In *Proc. of ICLR*.

Xiaodong Yu, Stephen Mayhew, Mark Sammons, and Dan Roth. 2018. On the strength of character language models for multilingual named entity recognition. In *Proc. of EMNLP*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Proc. of NeurIPS*.

Xiang Zhang and Yann Lecun. 2017. Which Encoding is the Best for Text Classification in Chinese, English, Japanese and Korean? *arXiv preprint arXiv:1708.02657v2*.

Xiang Zhang, Junbo Zhao, and Yann Lecun. 2015. Character-level convolutional networks for text classification. In *Proc. of NeurIPS*.

Xinsong Zhang and Hang Li. 2020. AMBERT: A Pre-trained Language Model with Multi-Grained Tokenization. *arXiv preprint arXiv:2008.11869*.