Char2Subword: Extending the Subword Embedding Space Using Robust Character Compositionality

Gustavo Aguilar,[‡*] **Bryan McCann,**[†**] **Tong Niu,**[†]
Nazneen Rajani,[†] **Nitish Keskar,**[†] **Thamar Solorio**[‡]
[‡]University of Houston, Houston, TX
[†]Salesforce Research, Palo Alto, CA
[‡]{gaguilaralas, tsolorio}@uh.edu
[†]{bmccann, tniu, nazneen.rajani, nkeskar}@salesforce.com

Abstract

Byte-pair encoding (BPE) is a ubiquitous algorithm in the subword tokenization process of language models as it provides multiple benefits. However, this process is solely based on pre-training data statistics, making it hard for the tokenizer to handle infrequent spellings. On the other hand, though robust to misspellings, pure character-level models often lead to unreasonably long sequences and make it harder for the model to learn meaningful words. To alleviate these challenges, we propose a character-based subword module (char2subword)[1] that learns the subword embedding table in pre-trained models like BERT. Our char2subword module builds representations from characters out of the subword vocabulary, and it can be used as a drop-in replacement of the subword embedding table. The module is robust to character-level alterations such as misspellings, word inflection, casing, and punctuation. We integrate it further with BERT through pre-training while keeping BERT transformer parameters fixed–and thus, providing a practical method. Finally, we show that incorporating our module to mBERT significantly improves the performance on the social media linguistic code-switching evaluation (LinCE) benchmark.

1 Introduction

Byte-pair encodings (BPE) is a ubiquitous algorithm in the tokenization process among transformer-based language models such as BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), RoBERTa (Liu et al., 2019), and CTRL (Keskar et al., 2019). This method addresses the open vocabulary problem by segmenting unseen or rare words into smaller subword units while keeping a reasonable vocabulary size (Huck et al., 2017; Kudo, 2018; Wang et al., 2019). However, BPE and its variants are sensitive to small perturbations in the text, potentially distorting the sentences' meaning (Jones et al., 2020) (see Figure 1). Moreover, this tokenization process is rigid to changes such as adding more subwords to the vocabulary or correcting the segmentation splits. That is because the tokenization relies on the original corpus where the vocabulary was generated (e.g., Wikipedia), resulting in a fixed set of subword pieces tied to an embedding lookup table (Bostrom and Durrett, 2020). Although these aspects are not a problem with clean and properly formatted text, that is not the case when the text presents substantial noise (e.g., Wikipedia vs. social media). Noisy text can result in extensive subword pieces per word (see Figure 1), preventing the models from capturing the meaning effectively and adapting to such domains. This is particularly prominent on social media text (Baldwin et al., 2015; Eisenstein, 2013a,b), where the noise permeates even across languages and in code-switching scenarios (Singh et al., 2018; Aguilar et al., 2018; Molina et al., 2016; Das, 2016).

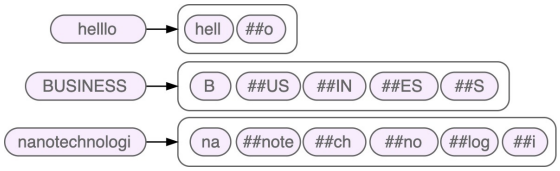

Figure 1: Examples of subword tokenization from OOV words. The word *helllo* changes its meaning (e.g., *hell*), *BUSINESS* is split almost to characters, and *nanotechnologi* do not resemble any of its morphemes.

This paper proposes a character-to-subword (char2subword) module trained to handle rare or unseen spellings robustly while being less restrictive to a particular tokenization method. Our method works as a drop-in alternative to the embedding table in pre-trained language models like mBERT. It improves performance and reduces the number of embedding parameters by 45% without sacrificing

[*]Work performed as summer intern at Salesforce.
[**]Work performed as manager while at Salesforce.
[1]The code is available at https://github.com/salesforce/char2subword

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1640–1651
November 7–11, 2021.

inference speed. We train our module to approximate the embedding table using characters from the original vocabulary words and subwords. This procedure leverages transfer learning from the pre-trained embedding table rather than starting from scratch—thus, saving precious computational time and resources. Besides, the subword vocabulary provides enough character-level patterns to learn from already-segmented tokens. We integrate our module with mBERT's transformer layers even further by continuing to train with the pretraining data and the MLM objective. Once our char2subword is adapted to the pre-trained language model, we evaluate the overall model performance by fine-tuning it on downstream tasks. We show our method's effectiveness by outperforming mBERT on the social media linguistic code-switching (LinCE) benchmark (Aguilar et al., 2020), where the fine-tuning domain deviates substantially from the pre-training domain. The results show that the char2subword module can also capture intra-word code-switching. At the sentence level, the model can relate words from the same language to support language prediction.

We highlight our main contributions as follows:

1. We introduce char2subword, a new parameter-efficient and open-vocabulary module that extends the domain-constrained and fixed vocabulary in mBERT (or any pre-trained model relying on subwords) while preserving the semantics of the multilingual embedding space.
2. We show the character compositionality capabilities of our module by handling noise robustly at the character level while being language-independent and flexible to different tokenization.
3. We analyze the advantages of our model on downstream tasks and demonstrate its practical use and adaptability to other domains despite of vocabulary changes.

2 Related Work

Word representations Most of the initial ground-breaking advances in NLP relied on word embedding representations from methods like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). They showed the capability of arranging words in a continuous high-dimensional space encoding semantic relationships and meaning (Goldberg and Levy, 2014). However, rare words are weakly represented in such space, and OOV words are not representable. To alleviate that, researchers proposed word representations using recursive neural networks guided by morphology (Luong et al., 2013), as well as morpheme embeddings as a prior distribution over probabilistic word embeddings (Bhatia et al., 2016). Regardless, the challenges persist in noisy text, where users do not follow the canonical word forms (Eisenstein, 2013b). Such problems are aggravated in social media due to the inherently multilingual environment. More words per language are required, while the spelling noise is persistent across languages.

Character representations While character-level systems proved strong for text classification (Conneau et al., 2017), they were not as successful on multilingual tasks like neural machine translation (NMT) initially (Neubig et al., 2013; Chung et al., 2016). Even when the performance was satisfactory, such systems had to process long sequences of characters resulting in a very slow process (Costa-jussà and Fonollosa, 2016; Ling et al., 2015b). Additionally, languages have different writing systems and specific properties encoded at the character level. While some of those properties may be captured effectively on morphologically rich languages (e.g., Czech and Arabic), properties from other languages are not more impactful than using words (e.g., English) (Cherry et al., 2018). These challenges are also applicable to our case since we conduct our study on multilingual data with typologically different languages.

Hybrid representations Using words or characters has shown advantages and disadvantages on both ends. Researchers tried to get the best of both worlds by combining characters and words in a hybrid architecture (Luong and Manning, 2016) where the default was based on static word embeddings that backed off to characters if the word was unknown. Parallel efforts focused on character-aware neural language models (Kim et al., 2016) where the meaning is contextually enriched by highway networks (Srivastava et al., 2015), as well as character-based LSTM language models that build intermediate word representations from character-level LSTMs (Ling et al., 2015a). Most successful contextualized word embeddings built out of characters are the language models ELMo (Peters et al., 2018) and Flair (Akbik et al., 2018). Building models from characters can easily adapt to social media domains (Akbik et al., 2019), including

code-switching data (Aguilar and Solorio, 2020).

Subword models Sennrich et al. (2016) proposed subword tokenization using the byte-pair encoding (BPE) algorithm to balance the use of characters and words. BPE automatically chooses a vocabulary of subwords given the desired vocabulary size. This procedure recursively builds subwords upon characters using the word frequencies (Sennrich et al., 2016). Another greedy variation of BPE can select the longest prefix to segment words (Wu et al., 2016). Alternatively to the greedy versions, the segmentation can happen in a stochastic way; drawing segmentation candidates at different points of a word can improve generalization (Kudo, 2018). The WordPiece variation of BPE is used in NMT and language models such as multilingual BERT (Devlin et al., 2019). Regardless of the variant, these methods handle the out-of-vocabulary problem by breaking down unseen or rare words into pieces that are in the vocabulary. The problem is that BPE can generate subword pieces that are not linguistically plausible. The BPE tokenization is not ideal for social media domains because its rules do not necessarily apply across domains, particularly the ones with substantial noise and spelling differences (Bostrom and Durrett, 2020).

Compositional models The idea of composing OOV vectors has been explored before (Ling et al., 2015a; Plank et al., 2016). However, learning such vectors requires a large corpus and long computing time (i.e., processing characters). Pinter et al. (2017) proposed learning OOV words from a pre-trained word embedding dictionary. They treat every word from the dictionary as a sequence of characters and output a single vector that mimicks the associated word embedding in the dictionary. Schick and Schütze (2019) improved this method by introducing *attentive mimicking* to account for context, besides the surface form of the word.

3 Method

Given a word w, a subword model produces a sequence of subword pieces $\boldsymbol{s} = (s_0, s_1, \ldots, s_n)$, such that the concatenation of all the segments from $\boldsymbol{s}$ fully reconstructs the word w. Regardless of whether a subword piece represents a character in a word or not, all the pieces are treated as semantic units within a sentence.[2] Such pieces come from a rule-based system that does not take into account semantics or morphology during the tokenization. Thus, the subword tokenization has a significant impact on the semantic abstraction from upper layers in pre-trained models like mBERT.

To alleviate such problems, we build word representations out of characters. The char2subword module allows flexible tokenization patterns, where the model can split by spaces, use the original tokenization method, or employ a different tokenization process as defined by the user. There are two main phases in our proposed method: approximating subword embeddings with the char2subword module (i.e., ideally replicating the embedding space $\boldsymbol{E}$) (Section 3.1), and contextually integrating the char2subword module into the pre-trained model (Section 3.2).

3.1 Approximating the subword embedding

Consider a subword s_i from the vocabulary $\mathcal{V}$ and a subword embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$. We learn a parameterized function $f_\theta : \mathbb{R}^{|c| \times 1} \rightarrow \mathbb{R}^d$ that maps the sequence of characters $\boldsymbol{c}_i = (c_{i1}, c_{i2}, \ldots, c_{i|s_i|})$ from the subword s_i to its corresponding embedding vector $\boldsymbol{e}_i \in \boldsymbol{E}$:

$$\hat{\boldsymbol{e}}_i = f_\theta(\boldsymbol{c}_i) \quad s.t. \;\; \hat{\boldsymbol{e}}_i \approx \boldsymbol{e}_i$$

To accomplish this, we design an objective function that fulfills our four desiderata; we want the embeddings to: (i) preserve their angular distances, (ii) be similar in L^2 norm to prevent magnitude disruptions in upper layers of mBERT, (iii) have similar neighbors in cosine-distance space, and (iv) ultimately map to the same tokens in embedding space. We thus optimize f_θ by minimizing the overall objective function $\mathcal{L}(\cdot)$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{c}_i, \boldsymbol{e}_i, \boldsymbol{y}_i, f_\theta) = \mathcal{L}_{cos}(\boldsymbol{e}_i, f_\theta(c_i)) + L^2(\boldsymbol{e}_i, f_\theta(c_i)) \\ + \mathcal{L}_{nbr}(\boldsymbol{e}_i, f_\theta(c_i)) + \mathcal{L}_{ce}(\boldsymbol{y}_i, f_\theta(c_i))\end{aligned}$$

The four objectives of the loss function correspond to the four aforementioned desired properties. The first objective, $\mathcal{L}_{cos}(\cdot)$, is the cosine distance between the target and the predicted embedding vectors $\boldsymbol{e}_i$ and $\hat{\boldsymbol{e}}_i$. By using an angular distance function, we encourage the model to replicate the semantic relationships and vector arrangements

[2] Previous studies (Clark et al., 2019; Rogers et al., 2020) showed that BERT learns syntax and parsing within its self-attention probabilities. That is evidence that subwords need to preserve semantics when fed into such layers. This suggests that subword pieces broken down to the character level can prevent the model from exploiting linguistic properties.

in the original embedding space of $\boldsymbol{E}$:

$$\mathcal{L}_{cos}(\boldsymbol{e}_i, \hat{\boldsymbol{e}}_i) = 1 - \frac{\boldsymbol{e}_i \cdot \hat{\boldsymbol{e}}_i}{||\boldsymbol{e}_i|| \, ||\hat{\boldsymbol{e}}_i||}$$

The second objective is the L^2 norm or euclidean distance between the vectors $\boldsymbol{e}_i$ and $\hat{\boldsymbol{e}}_i$. The previous objectives do not regulate the magnitude of the predicted vector $\hat{\boldsymbol{e}}_i$, allowing that to be a degree of freedom for f_θ. By using the L^2 norm, we penalize the model for generating a vector $\hat{\boldsymbol{e}}_i$ with a different magnitude than $\boldsymbol{e}_i$. Regulating the magnitude is important to approximate the vector arrangements in the embedding space. We hypothesize that slightly different properties in the embedding $\boldsymbol{E}$ can magnify differences at the upper layers of mBERT.

The third objective, $\mathcal{L}_{nbr}(\cdot)$, is the mean squared error (MSE) of cosine distances generated between the k-th closest neighbors to $\boldsymbol{e}_i$ versus the distances of the same neighbors with respect to $\hat{\boldsymbol{e}}_i$:

$$(\boldsymbol{n}_1, \dots \boldsymbol{n}_k) = topk(\boldsymbol{e}_i, \boldsymbol{E})$$

$$\mathcal{L}_{nbr}(\boldsymbol{e}_i, \hat{\boldsymbol{e}}_i) = \frac{1}{k} \sum_{j=1}^{k} (dis(\boldsymbol{e}_i, \boldsymbol{n}_j) - dis(\hat{\boldsymbol{e}}_i, \boldsymbol{n}_j))^2$$

where $topk(\cdot, \cdot)$ retrieves the k-th closest neighbors according to the cosine distances among all the subword vectors in $\boldsymbol{E}$. The core idea of this objective is to force distances between $\hat{\boldsymbol{e}}_i$ and the neighbors $\boldsymbol{n}_*$ to be as similar as possible to the distances between the same neighbors and $\boldsymbol{e}_i$.

The final objective is the cross-entropy loss $\mathcal{L}_{ce}(\cdot)$. We use $\boldsymbol{E}$ as fixed parameters to project linearly from the embedding to the vocabulary. This loss term forces the model to learn accurate embedding representations such that they map to the original subwords from the vocabulary $\mathcal{V}$:

$$\mathcal{L}_{ce}(\boldsymbol{y}_i, \hat{\boldsymbol{e}}_i) = -\sum_{j}^{|\mathcal{V}|} \boldsymbol{y}_{ij} \log \hat{\boldsymbol{y}}_{ij}$$

$$s.t. \quad \hat{\boldsymbol{y}}_i = \text{softmax}(\hat{\boldsymbol{e}}_i \cdot \boldsymbol{E}^\top)$$

Char2subword module We model f_θ using the transformer architecture (Vaswani et al., 2017). The module processes a sample as a sequence of characters $\boldsymbol{c}_i = (c_{i1}, c_{i2}, \dots, c_{iM})$ of a subword s_i of length M.[3] We represent the sequence $\boldsymbol{c}_i$ as the sum between the character embeddings and sinusoidal positional encodings. We pass the resulting sequence of character vectors $\boldsymbol{X}_0$ to a stack of l attention layers, each with k attention heads. On top of the l attention layers, we add a linear layer $\boldsymbol{W}_e \in \mathbb{R}^{d' \times d}$ followed by max-pooling and a layer normalization for the final output $\hat{\boldsymbol{e}}_i$ (see full definition in Appendix A).

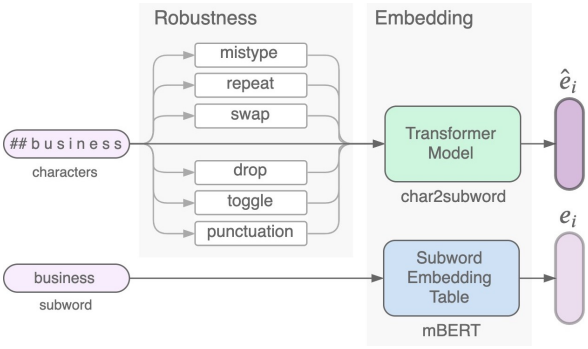

Figure 2: The char2subword module approximates the mBERT subword embedding table. We incorporate noise in every word with single-character operations.

Operation	Description
`mistype`	replace a random character of a subword by randomly choosing from its nearby keys according to a keyboard layout
`repeat`	repeat a random character of a subword
`swap`	randomly choose a character and swap it with the next character in a subword
`drop`	randomly drop a character of a subword
`toggle`	toggle the case of a randomly chosen character from a given subword
`punct`	randomly add a punctuation mark commonly used within words (e.g., dashes, periods)

Table 1: Single-character operations to incorporate noise in the approximation stage. The operations are applied to every word in the vocabulary that exceeds the four characters.

Character-level robustness The flexibility of the char2subword module makes it easier to teach the model text invariance because the inputs are now processed at the character-level. We augment the subword vocabulary $\mathcal{V}$ by introducing natural single-character misspellings during training. We apply one operation at a time and only to subwords that exceed four characters to reduce the chance of ambiguity between valid subwords. The operations are described in Table 1 and the high-level view of the approximation appears in Figure 2.[4]

[3] To distinguish between words and subwords, we prepend '##' to the sequence $\boldsymbol{c}_i$ in the case of full words.

[4] For the `mistype` operation, we use over 100 keyboard layouts to cope with the languages in mBERT.

3.2 Pre-training with the char2subword

The previous techniques leverage the pre-trained knowledge in the embedding matrix $\boldsymbol{E}$. However, the char2subword module may not be integrated with the pre-trained mBERT's upper layers since it has only seen individual subwords without context. To alleviate that, we pre-train the char2subword module along with mBERT (Gururangan et al., 2020). We do not update parameters in upper layers of mBERT since the goal is to provide the char2subword module as a drop-in alternative for $\boldsymbol{E}$ on the publicly available pre-trained models.[5]

Following Liu et al. (2019), we use a dynamic masked language modeling (MLM) objective (see Figure 3). We randomly choose 15% of the subword tokens and mask them at the character level. We replace 80% of the characters with `[MASK]`, 10% with randomly chosen characters and the remaining 10% is left unchanged. We feed characters to the char2subword module and make predictions from the subword vocabulary $\mathcal{V}$.[6] We pre-train the char2subword model with 1M sequences of 512 subword tokens from Wikipedia (200K sequences for each English, Spanish, Hindi, Nepali, and Arabic text). Using gradient accumulation, we update parameters with an effective batch size of 2,000 samples. Note that the model does not require extensive pre-training since 1) the upper-layer parameters are initialized from the pre-trained mBERT checkpoint and kept fixed during training, and 2) the char2subword module is initialized from the embedding approximation phase. Thus, pre-training the model for a few epochs is sufficient.

3.3 Fine-tuning

Once the char2subword module has been optimized, we evaluate the pre-trained model with the char2subword module on downstream NLP tasks. Specifically, we experiment with two scenarios: the full and the hybrid modes.

Full mode This mode completely replaces the subword embedding table in mBERT (i.e., the set of parameters and vectors) with the char2subword module. The idea of this setting is to evaluate how well approximated was the embedding space originally in $\boldsymbol{E}$. Intuitively, if the char2subword replicates the embedding space in $\boldsymbol{E}$ perfectly, then the overall model should behave about the same as the original mBERT model. Nevertheless, this setting does not tokenize further a word; hence, the input sequence tends to be shorter and more meaning-preserving (i.e., too many subword pieces for a single word can degrade its meaning).

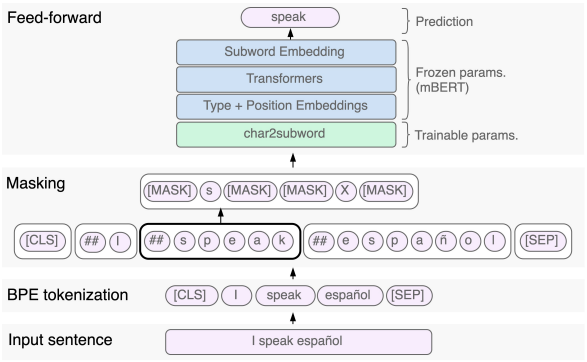

Figure 3: An example of an input and output of the pre-training setting with a masked language modeling (MLM) objective at the character level.

Hybrid mode Unlike the full mode, this mode does not replace the subword embedding table. Instead, it uses the subword embedding vectors by default for full words (i.e., not subword pieces). The model backs off to character-based embeddings from the char2subword module when a word as a whole does not appear in the vocabulary. This method focuses specifically on subwords rather than full words, effectively preventing words from being broken down into pieces.

4 Experiments

Embedding approximation The goal of the approximation experiments is to replicate the original subword embedding table while ensuring robustness at the character level. We experiment with the objective functions described in Section 3.1. We use the average precision to determine the best method (we also provide the accuracy for reference).[7] The experiments 1.1-1.4 show the results of each objective separately (see Table 2). Notably, the cross-entropy objective is the most relevant to ensure high precision (58% vs. 28.5% of the cosine objective). Combining all the objectives gives an average precision of 60% (experiment 1.9). Although experiment 1.6 and 1.9 perform very close

[5] While the study focuses on mBERT, this method can be applied to other pre-trained models like RoBERTa or XLM-R.

[6] We project the internal representations per word onto the vocabulary space using $\boldsymbol{E}$ (without updating its parameters).

[7] Using accuracy to determine the best method can mislead the interpretation of the model's capabilities. Accuracy is not ideal in this scenario since the goal is to approximate an embedding space rather than merely predicting vocabulary subwords given their characters.

Exp.	$\mathcal{L}_{ce}$	$\mathcal{L}_{cos}$	L^2	$\mathcal{L}_{nbr}$	Acc.	Prec@1	Prec@15	Avg Prec
1.1	✓				99	99.6	43.9	58.1
1.2		✓			62	41.8	24.2	28.5
1.3			✓		45	18.2	12.2	13.5
1.4				✓	43	25.5	17.1	19.6
1.5	✓	✓			96	96.1	41.2	55.1
1.6	✓		✓		95	99.1	46.6	59.9
1.7	✓	✓	✓		95	98.6	46.7	59.8
1.8	✓			✓	98	97.4	42.6	56.5
1.9	✓	✓	✓	✓	95	98.3	47.1	**60.0**

Table 2: The results of approximating the subword embedding table from mBERT using different objectives (✓). The accuracy denotes the capability of the model to predict a subword out of its characters. Precision @ k measures the overlap between the k ground-truth neighbors for a vector $\boldsymbol{e}_i$ (that represents subword s_i) and the k neighbors of the predicted vector $\hat{\boldsymbol{e}}_i$.

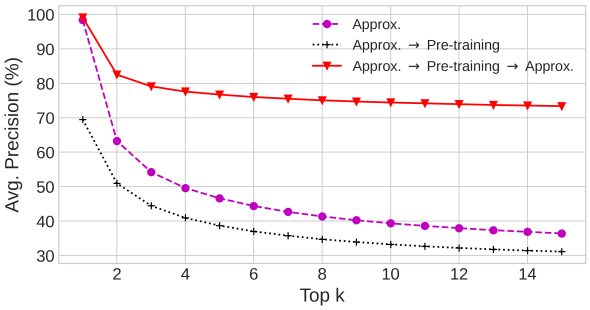

Figure 4: The precision up to 15 neighbors combining the approx. and pre-training phases in different ways.

(59.9% vs. 60%), the latter still preserves more neighbors along the top k expected neighbors.

After optimizing a char2subword module (experiment 1.9), we contextualize it in the pre-training phase (Section 3.2). The results show that the precision at k drops substantially (Figure 4, "Approx. → Pre-training"). However, when restarting approximation after pre-training, the model performs far better than the initial approximated version reaching an average precision of 82.4% (Figure 4, "Approx. → Pre-training → Approx."). This improvement shows the need for contextualization for the original char2subword module. Contextualization by itself does not guarantee that the module will resemble the same embedding space as in $\boldsymbol{E}$ (i.e., nothing that forces the module to optimize for that). However, it aligns better the semantics of the space facilitating the approximation.

Character-level robustness is another essential aspect when optimizing the char2subword. We add single-character edits to the training phase described in Section 3.1. Table 3 shows the neighbors of the word *business* and its variations. When fed *business* without noise, the char2subword modules (with and without noise) retrieve semantically-related neighbors. However, when the word is capitalized, the neighbors are not related to the word *business* for the char2subword without noise. Also, the subword tokenization for *BUSINESS* becomes *B-US-INE-SS*, which distorts the meaning of the original word. Regardless, the char2subword with noise is resilient to the capitalization pattern and capable of maintaining the meaning.[8]

Fine-tuning experiments Once the module is adapted to mBERT, we benchmark the model in the full and hybrid modes (see Section 3.3) using the LinCE benchmark (Aguilar et al., 2020). Particularly, we focus on language identification (LID), part-of-speech (POS) tagging, named entity recognition (NER), and sentiment analysis (SA). Table 4 shows the results of the experiments using the full and hybrid modes. Also, we include ELMo's test scores[9] as baseline since ELMo composes its representations from characters. For each proposed model, we use the approximated and pre-trained (i.e., "Approx. → Pre-training → Approx.") versions of the char2subword module. The language identification results are not a strong indicator of improvement since the scores are all very close.[10] Nevertheless, it is important to note that the model, regardless of the version, can perform on par with the mBERT baseline. This suggests that the char2subword representations are compatible with the rest of the mBERT model (i.e., mBERT transformer layers).

For the POS and NER tasks, we see improvements compared to mBERT. The hybrid pre-trained experiment for Hindi-English is significantly better than the baseline for both POS (89.64% vs. 87.86%) and NER (74.91% vs. 72.94%). One of the reasons for this performance boost is due to the noise that splitting transliterated Hindi (i.e., Romanized Hindi) generates for the baseline. On the contrary, the char2subword compresses the transliterated words into a single vector, reducing the noise in the model. The NER results for Spanish-English (es-en) and Modern Standard Arabic-Egyptian Ara-

[8]The char2subword module never sees a subword from the vocabulary with more than a single character edit (i.e., we defined the robustness procedure this way). That means that the word *BUSINESS* never appeared in training for the model.

[9]ritual.uh.edu/lince/leaderboard

[10]The average score for LID across language pairs is 95.71% for mBERT (baseline) and 95.80% for char2subword module (hybrid, pre-trained).

Input	Model	Neighbors
business	mBERT	(business, 1.0), (Business, 0.61), (businesses, 0.47), (businesses, 0.47), (bisnis, 0.46)
	Char2subword	(business, 0.82), (Business, 0.50), (businesses, 0.43), (бизнес , 0.40), (bisnis, 0.38)
	Char2subowrd + noise	(business, 0.80), (Business, 0.61), (businesses, 0.53), (бизнес, 0.43), (negocios, 0.39)
bsusinessses	Char2subword	(businesses, 0.42), (companies, 0.33), (opportunities, 0.32), (industries, 0.31)
	Char2subword + noise	(businesses, 0.79), (companies, 0.53), (shops, 0.52), (corporations, 0.50), (employees, 0.49)
BUSINESS	Char2subword	(ASEAN, 0.25), (RSS, 0.24), (FCC, 0.24), (WEB, 0.2403), (Austrália, 0.2360)
	Char2subword + noise	(Business, 0.53), (business, 0.32), (Marketing, 0.31), (Corporate, 0.31), (Communications, 0.30)

Table 3: Neighbors from the mBERT subword embedding table using different embedding vectors to represent the word *business* and its modifications (e.g., $topk(\boldsymbol{e}, \boldsymbol{E})$) . For the mBERT OOV words *bsusinessses* and *BUSINESS*, the tokenizer breaks the words as *b-sus-iness-ses* and *B-US-INE-SS*, respectively.

			LID (W. F_1)				POS (Acc.)		NER (F_1)			SA (W Acc.)
Method	**Adaptation**	**Avg**	es-en	hi-en	ne-en	msa-arz	es-en	hi-en	es-en	hi-en	msa-arz	es-en
Validation set results												
mBERT	N/A	83.86	98.23	96.37	**96.67**	91.55	**97.29**	87.86	62.66	72.94	78.93	56.10
Full	Approx.	83.59	98.16	95.79	96.45	91.63	96.93	89.04	62.02	70.79	79.13	55.98
Full	Pre-trained	83.89	98.20	96.97	96.47	91.48	96.91	89.38	61.23	71.98	79.42	56.82
Hybrid	Approx.*	84.33	**98.24**	**96.98**	96.50	91.48	97.16	88.95	**64.26**	72.68	80.10	56.98
Hybrid	Pre-trained*	**84.60**	98.18	96.75	96.37	**91.64**	97.03	**89.64**	63.32	**74.91**	**80.45**	**57.71**
Test set results												
ELMo	N/A	79.52	97.93	95.43	95.90	86.53	96.34	86.71	52.58	68.79	56.68	52.88
mBERT	N/A	82.23	**98.36**	94.24	**96.32**	**91.55**	**97.07**	86.30	64.05	72.57	65.39	56.43
Hybrid	Pre-trained*	**83.03**	98.33	**96.23**	96.19	91.19	96.88	**88.23**	**64.65**	**73.38**	**66.13**	**59.07**

* Statistically significant with respect to the mBERT baseline, with p-value < 0.01 in student's t-test (Dror et al., 2018).

Table 4: Results on the LinCE benchmark. Full refers to the full mode where the model only uses the char2subword to embed the input. Hybrid means that the model uses the subword embedding table by default and backs off to the char2subword module for OOV words, instead of splitting them. For this table, pre-trained means that the model was approximated after the pre-training phase (i.e., "Approx. → Pre-training → Approx."). The languages involved are English (en), Spanish (es), Hindi (hi), Nepali (ne), Modern Standard Arabic (msa), and Egyptian Arabic (arz). The best results on each language pair are in bold, and the test scores are in italics.

bic (msa-arz) also exceed the baseline (64.26% vs. 62.66%). Although there is no transliteration in these language pairs, there is still much noise coming from social media user-generated language. Also, pre-training the char2subword on Spanish and Arabic data improves the model's representations and robustness for such languages.

5 Analysis

Attention for language identification Figure 5 shows the visualization for the Spanish-English LID task with an intra-sentential code-switching example (i.e., code-switching at the clause level of a sentence utterance). The example shows that the strongest connections at the word level (Figure 5 (left)) happen for words in the same language. Particularly, the word *consequencias* is slightly ambiguous since its morphology overlaps substantially with both the English and Spanish versions. With the context from the surrounding Spanish words, the model can determine that the word is Spanish.

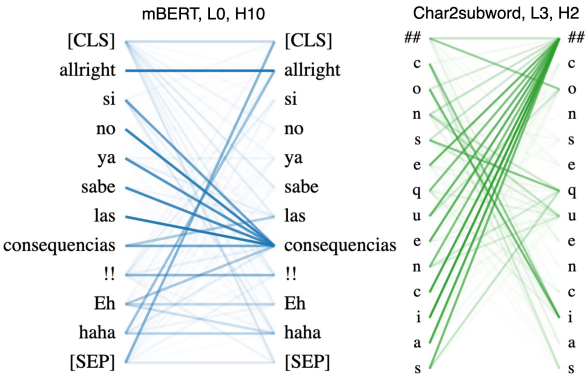

Figure 5: Attention visualization from a Spanish-English tweet. Translation: *"Alright, otherwise you know the consequences!! Eh, haha."*

Although there are more patterns captured among all the heads in mBERT, this pattern suggests that words of the same language can provide contextual support along with the sentence.

In addition to the contextual support, character-level attention plays an important role when build-

ing the word representation. Particularly for this word, the ambiguity is introduced due to the letter *q*. Note that the char2subword module creates strong connections with this letter and parts where more ambiguity could happen. For example, the letter *i* happens where the suffixes *-cias* (Spanish) and *-ces* English could complete the word).

Error analysis By inspecting the mistakes of the model in the confusion matrix for the Spanish-English LID development set, we noticed 112 English words predicted as Spanish, and 101 Spanish words predicted as English (see Table 6 in Appendix B for the confusion matrix). Out of the 101 English words, 63 were processed by the char2subword module (i.e., via backoff). Most of these errors come from words that heavily overlap in morphology between the two languages. For example, the words *imagine, rodeos, superego, tacos* are exact spellings between the languages, while the words *apetite* and *pajamas* change one letter between the languages (e.g., *apetito, pijamas* in Spanish). These errors suggest that the robustness may create some ambiguity when it comes to detecting the text's language. That is, single-character differences can denote one or another language, but the robustness operations (Table 1) can blur such distinction during the approximation phase. Other words are interjections that are spelled the same way (e.g., *oh*, *eh*, and *Muahahahahaha*). Also, there are cases where the ground-truth labels are wrong. E.g., the word *larges* in the the sentence "*La puerta esta abierta para que te larges porque no te has ido*"[11] was correctly predicted as Spanish based on the context (i.e., the correct spelling is *largues*, which translates to *get out*).

Subword sequence lengths Sequences from the subword tokenization are the same length or longer than the original sequence of tokens. Quantifying that tells us about the opportunity that the char2subword mBERT model has in practice. Table 5 shows the statistics of the original sequence lengths (Tokens) and the sequence lengths after the subword tokenization (Subword). Note that the average sequence lengths tend to duplicate across datasets. This can potentially explain a larger gap in performance for NER and POS tagging tasks than in LID. The former tasks require more semantics, which aligns with the fact that subwords degrade meaning by splitting into many pieces.

[11] "The door is open for you to leave, why haven't you left?"

			Original		Tokenized	
Task	**Lang.**	**Seqs.**	**Mean$_{\pm\text{Std}}$**	**Range**	**Mean$_{\pm\text{Std}}$**	**Range**
LID	es-en	3.3K	$12.1_{\pm 7.7}$	[1, 39]	$21.1_{\pm 12.0}$	[1, 69]
	hi-en	744	$20.8_{\pm 24.1}$	[1, 225]	$31.4_{\pm 32.9}$	[4, 278]
	ne-en	1.3K	$14.5_{\pm 6.3}$	[3, 34]	$28.5_{\pm 10.8}$	[3, 63]
	msa-arz	1.1K	$19.7_{\pm 6.5}$	[2, 36]	$43.5_{\pm 14.4}$	[2, 93]
NER	es-en	10K	$12.1_{\pm 7.6}$	[1, 45]	$25.7_{\pm 14.2}$	[1, 120]
	hi-en	314	$17.0_{\pm 6.3}$	[4, 34]	$40.5_{\pm 13.6}$	[7, 74]
	msa-arz	1.1K	$20.2_{\pm 6.7}$	[2, 38]	$44.5_{\pm 14.8}$	[3, 112]
POS	es-en	4.2K	$7.7_{\pm 6.0}$	[2, 90]	$9.9_{\pm 7.8}$	[2, 127]
	hi-en	160	$21.7_{\pm 5.2}$	[5, 37]	$41.3_{\pm 12.2}$	[7, 93]

Table 5: Statistics across the development sets comparing sequence lengths before (e.g., **Original**) and after (e.g., **Tokenized**) subword tokenization.

Parameters vs. efficiency The subword lookup table in mBERT provides immediate access for the tokenized text to the embedding space, making such a table very convenient. However, this access is highly restricted to a predefined vocabulary, and, in the case of multilingual models, such vocabulary has to have adequate coverage for all the languages involved. Models like mBERT or XLM-R (Conneau et al., 2020) use more than 100 languages, which translates into a large number of parameters just to enable the text to be vectorized. More specifically, mBERT has 177M parameters in total while only its subword embedding table ($|\mathcal{V}| = 119K$) occupies 91M parameters—more than 50% of all the parameters of the model.[12] The char2subword module, on the other hand, reduces the number of parameters to 50M, about 45% less than the subword embedding table, while also capable of handling misspellings and inflections robustly. Nevertheless, this module requires more computation time to come up with subword-level embedding representations.

Adversarial attacks We assess the robustness of the char2subword by using the TextAttack library (Morris et al., 2020). Particularly, we apply the DeepWordBug recipe (Gao et al., 2018) to the es-en sentiment analysis validation set. The attack consists of character-level transformations on the highest-ranked words that minimizes the edit distance of the perturbation. Notably, the char2subword module is more resilient than mBERT to these attacks; mBERT loses 16.78 points of weighted accuracy (56.10 → 39.32), while char2subword + mBERT drops 12.41 points (57.71 → 45.30). Most of the attacks that affect

[12] For XLM-R base (278M) and large (559M), the percentages are 65% and 49%, respectively.

the prediction on mBERT are entities. Intuitively, this is reasonable since the BPE splits such cases into many subword pieces, while the char2subword sticks to the name words and leverage context.

6 Conclusion

We provide a novel, flexible, and robust method to expand the mBERT subword embedding table. The char2subword module provides more control at the tokenization level, and it can generate word embeddings without being restricted to a fixed vocabulary or segmentation method. Also, the char2subword module gives the possibility to refine a language or domain of interest (i.e., by pre-training the char2subword module) while preserving its multilingual properties. Finally, this method is not limited to code-switching; the char2subword module is a general approach that can be applied to any word or subword-based pre-trained model.

Acknowledgments

This work was partially funded by the National Science Foundation under grant #1910192.

References

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Named entity recognition on code-switched data: Overview of the CALCS 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147, Melbourne, Australia. Association for Computational Linguistics.

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France. European Language Resources Association.

Gustavo Aguilar and Thamar Solorio. 2020. From English to code-switching: Transfer learning with strong morphological clues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8033–8044, Online. Association for Computational Linguistics.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. 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 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Timothy Baldwin, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China. Association for Computational Linguistics.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 490–500, Austin, Texas. Association for Computational Linguistics.

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.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, Valencia, Spain. Association for Computational Linguistics.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Amitava Das. 2016. Tool contest on POS tagging for code-mixed Indian social media (Facebook, Twitter, and Whatsapp) text. Retrieved 05-10-2019.

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.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Jacob Eisenstein. 2013a. Phonological factors in social media writing. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 11–19, Atlanta, Georgia. Association for Computational Linguistics.

Jacob Eisenstein. 2013b. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Matthias Huck, Simon Riess, and Alexander Fraser. 2017. Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 56–67, Copenhagen, Denmark. Association for Computational Linguistics.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2752–2765, Online. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Yoon Kim, Yacine Jernite, D. Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *AAAI*.

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.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015b. Character-based neural machine translation. *CoRR*, abs/1511.04586.

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. *CoRR*, abs/1907.11692.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany. Association for Computational Linguistics.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49, Austin, Texas. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Graham Neubig, Taro Watanabe, Shinsuke Mori, and Tatsuya Kawahara. 2013. Substring-based machine translation. *Machine Translation*, 27(2):139–166.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

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

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Timo Schick and Hinrich Schütze. 2019. Attentive mimicking: Better word embeddings by attending to informative contexts. 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 489–494, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018. A Twitter corpus for Hindi-English code mixed POS tagging. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 12–17, Melbourne, Australia. Association for Computational Linguistics.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019. Multilingual neural machine translation with soft decoupled encoding. In *International Conference on Learning Representations*.

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, Lukasz 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. *CoRR*, abs/1609.08144.

A Char2subword Module Definition

We model the char2subword module f_θ using the Transformer architecture (Vaswani et al., 2017). The module processes a sample as a sequence of characters $\boldsymbol{c}_i = (c_{i1}, c_{i2}, \ldots, c_{iM})$ of a subword s_i of length M.[13] We represent the sequence $\boldsymbol{c}_i$ as the sum between the character embeddings and sinusoidal positional encodings. We pass the resulting sequence of character vectors $\boldsymbol{X}_0$ to a stack of l attention layers, each with k attention heads. The j-th attention layer receives the input $\boldsymbol{X}_j$ and it outputs $\boldsymbol{X}_{j+1}$ by applying two subsequent components: multi-head attention and feed-forward layers. The multi-head attention is defined as follows:

$$\begin{aligned}
\text{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d'}})\boldsymbol{V} \\
\text{MultiHead}(\boldsymbol{X}) &= [\text{head}_1; \ldots; \text{head}_k]\boldsymbol{W}^O \\
\text{where head}_i &= \text{Attn}(\boldsymbol{X}\boldsymbol{W}_j^{Q_i}, \boldsymbol{X}\boldsymbol{W}_j^{K_i}, \boldsymbol{X}\boldsymbol{W}_j^{V_i}) \\
\boldsymbol{X}'_j &= \text{MultiHead}(\boldsymbol{X}_j)
\end{aligned}$$

The feed-forward component linearly projects $\boldsymbol{X}'_j$ using $\boldsymbol{W}_{j1} \in \mathbb{R}^{d' \times 4d'}$ followed by a GELU activation function (Hendrycks and Gimpel, 2016). The projection is passed to another linear transformation such that the result $\boldsymbol{X}'_j$ is mapped back to $\mathbb{R}^{d'}$:

$$\text{FFN}(\boldsymbol{X}'_j) = \text{GELU}(\boldsymbol{X}'_j\boldsymbol{W}_{j1} + \boldsymbol{b}_{j1})\boldsymbol{W}_{j2} + \boldsymbol{b}_{j2}$$

Each component normalizes its input $\bar{\boldsymbol{X}}_j = \text{LayerNorm}(\boldsymbol{X}_j)$ using layer normalization (Ba et al., 2016). We add the normalized input to the output of the component as in a residual connection (He et al., 2016):

$$\begin{aligned}
\boldsymbol{X}'_j &= \text{MultiHead}(\bar{\boldsymbol{X}}_j) + \bar{\boldsymbol{X}}_j \\
\boldsymbol{X}_{j+1} &= \text{FFN}(\bar{\boldsymbol{X}'}_j) + \bar{\boldsymbol{X}'}_j
\end{aligned}$$

Following (Vaswani et al., 2017), we preserve the dimension d' of the character embedding throughout the attention layers. On top of the l attention layers, we add a linear layer $\boldsymbol{W}_e \in \mathbb{R}^{d' \times d}$ followed by max-pooling and a layer normalization for the final output $\hat{\boldsymbol{e}}_i$:

$$\hat{\boldsymbol{e}}_i = \text{LayerNorm}(\text{maxpool}(\boldsymbol{X}_l\boldsymbol{W}_e + \boldsymbol{b}_e))$$

[13] To distinguish between words and subwords, we prepend '##' to the sequence $\boldsymbol{c}_i$ in the case of full words.

	Ground-truth							
Pred.	**amb.**	**fw**	**lang1**	**lang2**	**mixed**	**ne**	**other**	**unk**
amb.	0	0	21	16	0	0	1	1
fw	0	1	0	1	0	0	0	0
lang1	14	0	16K	101	0	74	14	17
lang2	13	0	112	14K	0	51	5	3
mixed	0	0	1	4	0	1	0	0
ne	3	0	110	96	1	597	7	1
other	1	0	13	6	1	3	7K	4
unk	0	0	8	10	0	3	3	8

Table 6: The confusion matrix on the development set of the LID task for Spanish-English. The labels are lang1 (English), lang2 (Spanish), mixed (partially in both languages), ambiguous (either one or the other language), fw (a language different than lang1 and lang2), ne (named entities), other, and unk (unrecognizable words).

B Analysis

In Table 6, we provide the confusion matrix of the pre-trained char2subword model on the Spanish-English LID development set.