Learning Mutually Informed Representations for Characters and Subwords

Yilin Wang[*]
Harvard University
yilin_wang@g.harvard.edu

Xinyi Hu[*]
Carnegie Mellon University
xinyih2@alumni.cmu.edu

Matthew Gormley
Carnegie Mellon University
mgormley@cs.cmu.edu

Abstract

Most pretrained language models rely on subword tokenization, which processes text as a sequence of subword tokens. However, different granularities of text, such as characters, subwords, and words, can contain different kinds of information. Previous studies have shown that incorporating multiple *input* granularities improves model generalization, yet very few of them *outputs* useful representations for each granularity. In this paper, we introduce the *entanglement model*, aiming to combine character and subword language models. Inspired by vision-language models, our model treats characters and subwords as separate modalities, and it generates mutually informed representations for *both* granularities as output. We evaluate our model on text classification, named entity recognition, POS-tagging, and character-level sequence labeling (intraword code-switching). Notably, the entanglement model outperforms its backbone language models, particularly in the presence of noisy texts and low-resource languages. Furthermore, the entanglement model even outperforms larger pre-trained models on all English sequence labeling tasks and classification tasks. We make our code publically available.[1]

1 Introduction

Since the emergence of pretrained language models (LMs) like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), subwords tokenization have become the prevailing approach to tokenization. Common techniques include byte-pair-encoding (BPE) (Sennrich et al., 2016), WordPiece (Wu et al., 2016), and SentencePiece (Kudo and Richardson, 2018), which create word-sized character n-grams for the LM to learn reusable representations. However, subword tokenization has limitations: the number and vocabulary of subwords must be predetermined during pretraining. Consequently, tasks involving noisy text or low-resource languages often require meticulous engineering to achieve satisfactory performance.

A less studied alternative is tokenizing at the character or byte level. Pretrained LMs like CANINE (Clark et al., 2022), Charformer (Tay et al., 2022), and ByT5 (Xue et al., 2022) utilize character-level tokenization. Though such models usually require careful design to handle longer sequences resulting from fine-grained tokenization, they offer advantages such as better incorporation of morphology and avoidance of tokenization overfitting to the pretraining corpus domain.

Previous studies have shown that incorporating both character and subword (or full word) representations can enhance model generalization. However, most studies focused on using characters to enhance or refine word representations (Aguilar et al., 2018; Sanh et al., 2019; Shahzad et al., 2021; Wang et al., 2021; Ma et al., 2020; Tay et al., 2022). However, these models, unlike the character-level pretrained language models mentioned earlier, do not generate usable character-level representations.

In this paper, we argue that character and subword representations are distinct yet complementary. We introduce a novel model, named the *entanglement model*, which combines a pretrained character LM and a pretrained subword LM. Inspired by techniques from the vision-language models (specifically ViLBERT (Lu et al., 2019a)), we treat characters and subwords as two modalities and leverage cross-attention to learn new representations by iteratively attending between the character and subword sides of the model. The result is a simple, yet general approach for bringing together the fine-grained representation afforded by characters with the rich memory of subword representations.

We evaluate our entanglement model on a variety of *tasks* (named entity recognition (NER), part-of-speech (POS) tagging, and sentence classification), *domains* (noisy and formal text), and

[1]https://github.com/TonyW42/noisy-IE

languages (English and ten African languages). We also evaluate the entanglement model on character-level tasks (intraword code-switching), which cannot be processed by subword models. Empirically, our model consistently outperforms its backbone models and previous models that incorporate character information. On English sequence labeling and classification tasks, the entanglement model even outperforms larger pre-trained models. Further, we found that the usage of subword-aware character representations yields performance gains, compared to using a character-only model.

In order to better understand the effectiveness of our model, we also explore two natural extensions: (1) incorporating positional embeddings that explicitly align the characters and subwords and (2) masked language model (MLM) pretraining of the entanglement model. We find that these augmentations of the model are unnecessary, suggesting that our model is capable of learning positional alignment between characters and subwords on its own and leveraging the substantial pretraining of the backbone models *without* costly pretraining of our entanglement cross-attention layers.

2 Methods

We propose a novel *entanglement model* that allows information exchange between pretrained character models and subword models, which is facilitated by two separate sets of co-attention modules. Our intention is for each layer of co-attention to further entangle the subword and character representations. The model thereby builds subword representations that are character-aware and character representations that are subword-aware which can be used on both character-level and word-level tasks.

We apply the model to sequence labeling and text classification assuming a dataset of N samples and K classes, $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^{n_i}$ is a sequence of words of length n_i with label $\mathbf{y}^{(i)}$. For sequence labeling, the label $\mathbf{y}^{(i)} \in \{1, 2, \cdots, K\}^{n_i}$ is a vector with the same length as $\mathbf{x}^{(i)}$. For text classification, the label $\mathbf{y}^{(i)} \in \{1, 2, \cdots, K\}$ is an integer.

2.1 The Entanglement Model

Figure 1 shows the architecture of the entanglement model. We describe the model for a single training example $(\mathbf{x}, \mathbf{y})$, we first tokenize it into a subword sequence $\mathbf{x}^s \in \mathbb{R}^{n^s}$ and a character sequence $\mathbf{x}^c \in \mathbb{R}^{n^c}$, where n^s, n^c refers to the length of the subword and character sequences respectively. We then feed $\mathbf{x}^c$ through a character encoder and $\mathbf{x}^s$ through a subword encoder to obtain contextualized representations $H^s \in \mathbb{R}^{n^s \times d}$ and $H^c \in \mathbb{R}^{n^c \times d}$, where d is the embedding size for the contextualized representations. Then, we feed H^s and H^c through m (separate) co-attention modules to facilitate information exchange between character and subword representations, which outputs a character-aware subword embedding H^s_* and a subword-aware character embedding H^c_*. When using H^s_* for inference, we call the experiment to use the *subword side* (SUBW). When using H^c_* for inference, we call the experiment to use the *character side* (CHAR)

While having separate encoders for characters and subwords allows better modeling of the features unique to each granularity, the cross-attention block inside the co-attention module allows the representations for characters and words to learn from each other. During training, the information exchange happens not only in the co-attention modules but also in the backbone text encoders through the flow of the gradient.

2.2 The Co-attention Module

A co-attention module consists of two transformer blocks (Vaswani et al., 2017a). The first transformer block, named CO-TRM, features a cross-attention layer that uses one modality to query the other, which facilitates information exchange between the two modalities. Figure 2 demonstrates the structure of the CO-TRM module. The second transformer block, named TRM, features a self-attention layer, which is the same as the transformer layers in the backbone encoders.

Let $H^s_0 = H^s$ and $H^c_0 = H^c$ be the output of the pretrained LMs and H^s_i and H^c_i be the subword and character embeddings output by the i^{th} co-attention module. Given H^s_i and H^c_i the subword-side co-attention module outputs the next-layer hidden states H^s_{i+1} as:

$$C^s_{i+1} = \text{CO-TRM}(Q = H^s_i, K = H^c_i, V = H^c_i)$$
$$H^s_{i+1} = \text{TRM}(Q = K = V = C^s_{i+1})$$

Where C^s_{i+1} refers to the intermediate representation output by the CO-TRM module. Similarly, the character side co-attention module outputs the next-layer hidden states H^s_{i+1} as:

$$C^c_{i+1} = \text{CO-TRM}(Q = H^c_i, K = H^s_i, V = H^s_i)$$
$$H^c_{i+1} = \text{TRM}(Q = K = v = C^c_{i+1})$$

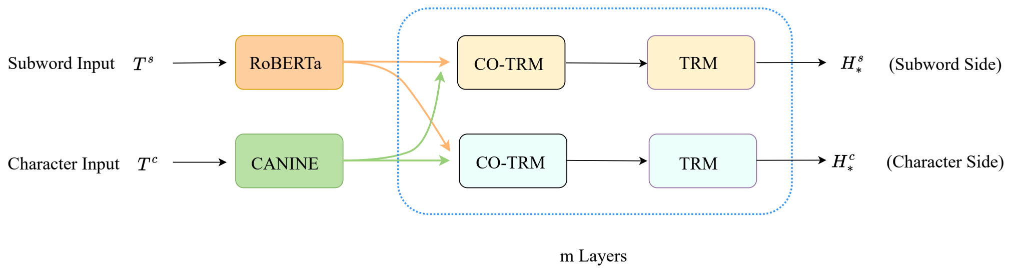

Figure 1: Architecture of the entanglement model.

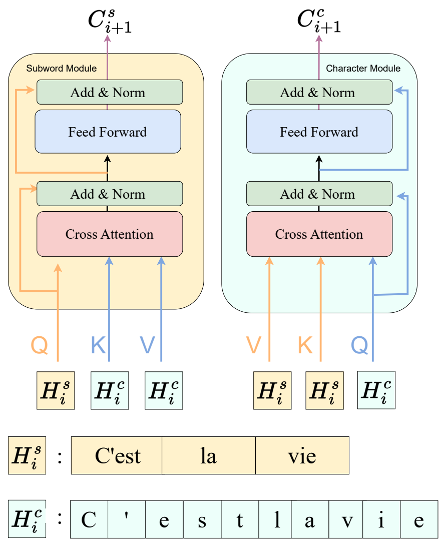

Figure 2: Architecture of the CO-TRM block inside the co-attention module.

2.3 Sequence Labeling

The Subword Side When training our model through the subword side, we pass the character-aware subword embedding H_*^s through a linear classification layer and a softmax layer to obtain the output probabilities $\hat{p}^s \in \mathbb{R}^{n_s \times K}$ for each subword:

$$\hat{p}^s = \text{Softmax}\left(H_*^s W^s\right) \qquad W^s \in \mathbb{R}^{d \times K}$$

We then select the output probabilities for the first subword as the prediction for each word, which creates word-level output probabilities $\hat{p}^w \in \mathbb{R}^{n \times K}$.

The Character Side Similarly, when training our model on the character side, we use the subword-aware character embedding H_*^c to obtain the output probabilities $\hat{p}^c \in \mathbb{R}^{n_c \times K}$ for each character:

$$\hat{p}^c = \text{Softmax}\left(H_*^c W^c\right) \qquad W^c \in \mathbb{R}^{d \times K}$$

We then select the output probabilities for the first character as the prediction for each word to get word-level output probabilities $\hat{p}^w \in \mathbb{R}^{n \times K}$.

Loss and Inference We then train the model under cross-entropy loss:

$$\mathcal{L}(\hat{p}^w, \boldsymbol{y}) = \sum_{j=1}^{n} \sum_{k=1}^{K} \boldsymbol{y}_k^j \log(\hat{p}_k^{w,j})$$

where $\boldsymbol{y}_k^j$ refers to the one-hot encoding of the label on the word j and $\hat{p}_k^{w,j}$ refers to output probability of that word on class k.

For inference, we will take the class with the highest output probability as the predicted label for each word. i.e.,

$$\hat{y}^j = \text{argmax}_k \quad \hat{p}_k^{w,j}$$

2.4 Text Classification

The Subword and Character Sides For text classification, the procedure for the subword side and the character side is the same: We take $h \in \mathbb{R}^d$, the first argument of either H_*^s or H_*^c, which is the embedding for the `[CLS]` token, and pass it through a linear and $\tanh$ layer. We then pass this output through a linear classification layer and a softmax function to obtain the output probabilities $\hat{p} \in \mathbb{R}^K$:

$$\hat{p} = \text{Softmax}(W^c(\sigma(W^p h)))$$

where $W^p \in \mathbb{R}^{d \times d}$, $W^c \in \mathbb{R}^{d \times K}$, and $\sigma(\cdot)$ refers to the $\tanh(\cdot)$ function.

Loss and Inference We then train the model under cross-entropy loss:

$$\mathcal{L}(\hat{p}, \boldsymbol{y}) = \sum_{k=1}^{K} \boldsymbol{y} \log(\hat{p}_k)$$

where $\boldsymbol{y}$ refers to the one-hot encoding of the label of the sample text and $\hat{p}_k$ refers to output probability of that sample on class k.

For inference, we take the class with the highest output probability as the predicted label for each sample. i.e.,

$$\hat{y} = \text{argmax}_k \quad \hat{p}_k$$

2.5 Comparison with Previous Work

Our model architecture draws partial inspiration from ViLBERT (Lu et al., 2019b), a pretrained vision-language model. However, unlike ViLBERT, our model capitalizes on the capabilities of pre-trained character and subword models, eliminating the need for additional pretraining steps and resulting in faster training times.

Other studies have investigated combining character and word embeddings. The ACE model (Wang et al., 2021) uses neural architecture search to find a subset of 11 embeddings, which are concatenated to form word representations. Unlike our model, ACE relies on fixed word embeddings, lacks learned character representations, and requires computationally intensive search. Our model is more efficient and learns a fine-grained representation of characters and subwords.

3 Experimental Setup

3.1 Datasets and Tasks

We evaluate our model on four tasks: named entity recognition (NER), part-Of-Speech (POS) tagging, intraword code-switching and text classification.

English sequence labeling: For NER, We utilize the WNUT-17 dataset (Derczynski et al., 2017) and the CONLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003), which respectively contains noisy user-generated texts from social media and formal writings sourced from the Reuters news. For POS-tagging, we use TweeBank (Jiang et al., 2022), which contains noisy texts from Twitter.

Multilingual NER: We use the MasakhaNER dataset (Adelani et al., 2021), which offers NER tasks for 10 low-resourced African languages.

Character-level sequence labeling: We also use the Spanish-Wixarika and Turkish-German data of Mager et al. (2019) on *intraword* code-switching. Since the language switch exists within a word, the *intraword* segmentations cannot be predicted by subword models because the morpheme boundaries might not align with subword boundaries. We formulate it as a character-level sequence labeling task.

Text classification: The WNUT-2020 shared task #2 dataset (Nguyen et al., 2020a) focuses on identifying informative English tweets related to COVID-19. Additionally, we use the TweetEval dataset (Barbieri et al., 2020), a comprehensive benchmark for evaluating tweet classification.

3.2 Experimental Details

For our experiments, we utilize the CANINE-s[2] (Clark et al., 2022) as the underlying character encoder backbone. For multilingual sequence labeling tasks, we employ XLM-R$_{\text{base}}$ (Conneau et al., 2020) as the subword encoder backbone, while for all other tasks, we use RoBERTa$_{\text{base}}$ (Zhuang et al., 2021) as the subword encoder backbone.

During model training, we employ the Adam optimizer with an initial learning rate of 2e-5 and a linear scheduler. The number of maximum epochs varies for each dataset: 25 for TweetEval and 50 for all other datasets. We select the model with the best performance on the validation set and evaluate it on the test set. Due to the small scale of MasakhaNER, we run each experiment three times with different seeds and report the average results.

We evaluate the entanglement model against four baselines: the backbone text and character model, a larger pre-trained subword model, and CharBERT (Ma et al., 2020), a previous subword model that incorporates character information.

In our result tables, we employ bold to highlight the best outcome achieved by either our baselines or the entanglement model, while † denotes the state-of-the-art performance. We keep the numbers from prior work in greyscale in all following tables.

4 Results

We conduct an extensive analysis of our model's performance on various sequence labeling and text classification tasks. We evaluate the effectiveness of our model on both formal and noisy English texts, as well as low-resourced languages, in order to assess its capabilities across different scenarios. Moreover, for each task, we report the performance of different configurations of our model, such as utilizing the subword or character side and varying the number of co-attention modules. This approach enables us to examine the robustness of our modules under different hyperparameter settings.

4.1 English Sequence Labeling

Table 1 shows the results of our model on two English NER datasets: WNUT-17 (noisy text) and CONLL-03 (formal text). Across all experiments, our model consistently outperforms the backbone

[2]The best CANINE model from (Clark et al., 2022) employs character n-gram embeddings. However, the corresponding pretrained model is not released by Google, so we use the available model: CANINE-s.

Model		WNUT-17	CONLL-03
ACE		-	94.60[†]
CL-KL		60.45[†]	-
RoBERTa$_{large}$		57.10	92.31
RoBERTa$_{base}$		56.38	91.93
CharBERT		53.63	92.07
CANINE-s		24.27	86.23
Side	**#C**		
CHAR	1	40.45	89.09
	2	39.77	89.57
	3	39.46	89.43
	4	**42.42**	**89.74**
SUBW	1	57.80	91.81
	2	**57.97**	92.21
	3	57.14	92.07
	4	56.28	**92.23**

Table 1: F1 on English NER tasks. Both sides of the entanglement model outperform the corresponding backbone models, and the subword side outperforms RoBERTa$_{large}$ (which has more parameters) and CharBERT. #C means the number of co-attention modules.

models on both the subword and character sides. Interestingly, the improvement is more pronounced for WNUT-17 compared to CONLL-03, indicating that our model excels at handling noisy text. Additionally, we observe that the character side exhibits a more significant improvement than the subword side, suggesting that the character model benefits greatly from co-attending with the subword model. Although our models do not surpass the state-of-the-art (SOTA) performance, it is important to note that the SOTA models either rely on external context (CL-KL), employ neural architecture search across a broader range of models (ACE), or a linear chain CRF layer (ACE), making them less directly comparable to our model. Table 2 showcases the results of our model on TweeBank. Overall, we observe minimal differences between the entanglement model and the RoBERTa baseline.

4.2 Multilingual NER

The results of our model on MasakhaNER are presented in Table 3. Again, we observe that our model outperforms the baseline models on both the subword and character side, with a more substantial improvement on the character side. The performance boost for certain languages, such as Luo

Model		TweeBank	WNUT-20
BERTweet		95.20	-
NutCracker		-	90.96[†]
CharBERT		93.59	88.08
RoBERTa$_{base}$		95.41	88.93
RoBERTa$_{large}$		94.50	89.21
Side	**#C**		
SUBW	1	95.39	89.14
	2	**95.52**[†]	**89.98**
	3	95.42	88.86

Table 2: Accuracy on TweeBank and F1 on WNUT-20. The entanglement model outperforms RoBERTa$_{base}$, RoBERTa$_{large}$, and CharBERT on these tasks. #C refers to the number of co-attention modules.

(LUO) and Wolof (WOL), appears more substantial. Luo consists of additional consonants and nine vowels (Adelani et al., 2021), which might be better processed by the character model. Wolof's morphology is derivationally rich (Ka, 1987), which may suggest that our model performs better on morphologically rich languages because it effectively leverages the character model.

Motivated by the performance gap between XLM-R and its larger variant, XLM-R$_{large}$, we experimented with the entanglement model using XLM-R$_{large}$ as the foundational subword backbone. To reconcile the embedding dimension mismatch between the two backbones (768 for CANINE-S and 1024 for XLM-R$_{large}$), we employed a fully-connected linear layer to upscale CANINE's character embeddings before passing them to the co-attention layers. As illustrated in the bottom panel of Table 3, when the entanglement model utilize XLM-R$_{large}$ as the backbone, its performance surpasses the standalone XLM-R$_{large}$ model, and it archives SOTA performance across most languages.

4.3 Character-level Sequence Labeling

Table 4 shows the results of our model on intra-word code-switching tasks. We see that the entanglement model outperforms CANINE-s across all tasks and specifications, and it outperforms the previous SOTA SegRNN (Mager et al., 2019) in most tasks. The performance gain is more substantial for "Mixed" words, which contain intraword code-witching.

Model		**AMH**	**HAU**	**IBO**	**KIN**	**LUG**	**LUO**	**PCM**	**SWA**	**WOL**	**YOR**	**Avg**
PIXEL		47.7	82.4	79.9	64.2	76.5	66.6	78.7	79.8	59.7	70.7	70.62
CANINE-c$_{+n\text{-grams}}$		50.0	88.0	85.0	72.8	79.6	74.2	88.7	83.7	66.5†	79.1	76.76
CANINE-s		32.70	74.38	71.79	55.92	69.98	53.75	66.17	73.37	57.82	61.00	61.69
XLM-R$_{base}$		71.69	90.05	84.79	73.35	78.33	73.98	87.96	86.46	63.43	77.56	78.76
XLM-R$_{large}$		75.51	91.06	83.85	**76.61**†	78.09	**77.08**†	90.08	88.87	65.58	79.50	80.62
Side	**#C**											
CHAR	1	41.99	79.12	74.00	59.23	70.48	61.17	75.06	78.25	59.19	63.45	66.20
	2	39.17	78.33	74.48	58.50	69.02	56.20	74.50	77.24	53.11	61.78	64.24
	3	41.14	79.00	73.81	58.89	70.53	55.56	73.67	77.58	57.71	59.67	64.76
SUBW	1	70.44	89.66	85.17	73.65	77.76	75.88	87.74	87.35	64.73	76.35	78.87
	2	72.83	89.89	84.71	72.53	78.44	75.94	88.01	86.54	65.66	77.25	79.18
	3	71.79	89.45	84.38	73.86	77.03	74.60	87.61	87.39	64.77	76.76	78.76
SUBW	1	74.01	91.35	84.33	74.83	79.08	75.89	**90.60**†	**89.58**†	65.40	77.81	80.29
(XLM-R$_{large}$)	2	74.14	90.67	85.03	72.52	79.93	75.40	90.10	89.62	**66.13**	78.29	80.18
	3	**76.67**†	**91.90**†	**85.83**†	73.42	**80.16**†	75.24	88.98	88.60	65.82	**80.49**†	**80.71**†

Table 3: MasakhaNER F1 score for Multilingual NER results. The first 2 panels (CHAR, SUBW) refers to the two sides of EM trained with XLM-R as the backbone. The bottom panel utilizes EM with XLM-R$_{large}$ as the backbone. Both sides of the best entanglement model consistently outperform the corresponding backbone models (XLM-R$_{base}$ and CANINE-S). EM with XLM-R$_{large}$ as the subword backbone archives SOTA performance on 6 out of 10 languages. #C means the number of co-attention modules. The last column Avg indicates the macro average F1 score of all the 10 African languages.

Evaluation	**All**	**All**	**MIX**	**MIX**
Data	**S-W**	**G-T**	**S-W**	**G-T**
SegRNN	92.40†	93.60	84.6	72.9
CANINE-s	90.84	94.12	82.97	72.44
Side-#C				
CHAR-1	**91.24**	94.60	**86.23**†	**74.21**†
CHAR-2	91.17	94.74	84.42	73.82
CHAR-3	91.00	94.39	84.05	71.26
CHAR-4	91.00	**94.86**†	84.42	72.63

Table 4: Character accuracy on code-switching tasks. The entanglement model outperforms CANINE-s and previous studies across all sub-tasks, and it outperforms SegRNN (Mager et al., 2019) except (All, S-W). "All" means the accuracy of all data, and "MIX" means the accuracy of words with intraword switching. S-W refers to Spanish-Wixarica, G-T refers to German-Turkish.

4.4 Classification

Table 5 presents the results of our model on the WNUT-2020 shared task #2, demonstrating its superiority over the baseline RoBERTa model and achieving performance close to state-of-the-art (the NutCracker model (Kumar and Singh, 2020)).

Furthermore, Table 5 showcases the results of the TweetEval benchmark, where our model outperforms the backbone models that have not been pretrained on this type of noisy text. For some subtasks, our performance is competitive with BERTweet (Nguyen et al., 2020b), which is pretrained on Twitter text. We also observe that the improvement on the character side is more substantial than the subword side.

4.5 Discussion

Character Models In most tasks, we see that the performance of CANINE-s is not comparable with RoBERTa. This perhaps explains the observation that the improvement of our model on the character is usually much more substantial than the subword side. Thus, our model might benefit from a different (potentially stronger) character model, such as Charformer (Tay et al., 2022) and ByT5 (Xue et al., 2022), and we leave it for future research.

Number of Co-attention Modules Generally, we observe that using two co-attention modules appears to be the optimal choice for the subword side, while one co-attention module appears to suffice for the character side. Although in certain tasks using 4 co-attention modules yields the highest performance, these additional benefits of more co-attention modules appear minimal.

Efficiency Our entanglement model requires 2-3 times the memory of a single backbone model. The

Model		Emoji	Emotion	Hate	Irony	Offsensive	Sentiment	Avg
RoB-RT		31.4	79.5[†]	52.3	61.7	80.5	72.6	63.00
BERTweet		33.58	78.88	**53.87**	**80.53**	80.17	68.53	**65.93**
CANINE-s		26.27	61.72	43.51	61.96	73.63	61.72	54.80
RoBERTa$_{\text{base}}$		33.36	78.55	50.49	73.14	78.05	68.28	63.65
RoBERTa$_{\text{large}}$		34.25	**81.87**	51.08	70.75	80.29	71.40	64.94
CharBERT		30.68	75.56	48.11	68.72	70.95	**71.62**	60.94
Side	**#C**							
CHAR	1	31.47	66.43	46.25	69.38	**81.24** [†]	70.39	60.86
	2	31.00	77.46	50.05	67.46	80.41	70.86	62.87
SUBW	1	33.56	79.31	50.19	73.95	80.57	70.62	64.70
	2	**34.38** [†]	78.65	52.60	73.69	80.02	71.41	65.13
(Pretrain)	1	30.33	74.02	44.81	59.87	78.27	66.42	58.95

Table 5: F1 on TweetEval. Both sides of the best entanglement model (EM) outperform the corresponding backbone models (XLM-R$_{\text{base}}$ and CANINE-S) and CharBERT across all tasks except Sentiment. For all models we have evaluated (not including BERTweet, which is pre-trained on Twitter text), EM performs the best for 4 out of 6 subtasks. The last column Avg indicates the macro average F1 across 6 tasks.

2-COTRM entanglement model contains around 290M parameters, whereas its subword backbone (RoBERTa$_{\text{base}}$) contains 125M parameters. Yet, our model contains fewer parameters than larger pre-trained models like RoBERTa$_{\text{large}}$ (354M parameters). Our model has higher parallelizability than RoBERTa$_{\text{large}}$, as the computation of the character and subword model is independent before the co-attention module. Empirically, the runtime of the entanglement model is roughly 1.72 times of RoBERTa$_{\text{base}}$ and 0.54 times of RoBERTa$_{\text{large}}$.

Baseline Table 1, 2, 5 shows that the entanglement model outperforms RoBERTa$_{\text{large}}$, which is pretrained and has more parameters, across all English classification and sequence labeling tasks. Table 3 shows that the entanglement model with XLM-R as the backbone failed to outperform XLM-R$_{\text{large}}$, so maybe more pretraining is required for lower-resourced languages. We see that EM with XLM-R$_{\text{large}}$ as the backbone outperforms XLM-R$_{\text{large}}$. Also, the entanglement model outperforms CharBERT (an English-only model) across all English classification and sequence labeling tasks, suggesting that our model more effectively leverages the ability of both character and subword models.

5 Model Extensions

In this section, we explore two natural extensions that demonstrate how the simplicity of our model eliminates the need for additional complexity.

Positional Embeddings We experimented with several ways to add positional embeddings (PE) in the co-attention module. Details on PE training are in appendix A. From table 6, we see that for WNUT-17, adding PEs hurts the model's performance. In CONLL-03, strategy C has a slight improvement in the model's performance, though it appears very marginal. This suggests that the entanglement model autonomously learns the translation between subword PEs and character PEs.

MLM pretraining We pretrain a 1-layer entanglement model on 8% of WikiText-103 (Merity et al., 2016) and Bookcorpus (Zhu et al., 2015). Details on pretraining are in appendix B. From table 7 & 5, we see that the pre-trained model fails to outperform the standard, un-pretrained entanglement model. This suggests that pretraining does not appear to help the model generalize. Nevertheless, it is also possible that the scale of pretraining is not large enough for it to exhibit a positive influence, and we leave it to future work.

6 Related Work

Many existing studies have investigated learning subward representations from multiple granularities of input (§6.1), and many studies has explored learning character representations (§6.2). Comparatively few works have explored outputting represen-

Strategy	WNUT-17	CONLL-03
No PEs	**57.80**	91.81
A	56.23	91.88
B	57.06	91.79
C	56.94	**92.15**

Table 6: NER F1 results for different kinds of positional embeddings (PEs). For WNUT-17, adding PEs decreases model performance. For CONLL-03, adding PE A and C leads to a very marginal performance boost.

Dataset	RL	CB	EM-P	EM
TweeBank	94.50	93.59	93.97	**95.52**
WNUT-20	89.21	88.08	88.08	**89.98**
WNUT-17	56.38	53.63	51.71	**57.97**
CONLL-03	92.31	92.07	91.21	**92.23**

Table 7: A more direct comparison between EM and larger pre-trained models (RoBERTa$_{\text{large}}$, RL), another character-aware subword model (CharBERT, CB), and pre-trained EM. The standard EM outperforms these three models across all these four tasks.

tations at multiple granularities (§6.3). Our model draws inspiration from studies in multimodal machine learning (§6.4) and facilitates information exchange between subword and character representations through a co-attention module.

6.1 Multiple Granularities of Input

Several previous studies have explored the use of multiple granularities in input representation. Charformer (Tay et al., 2022) uses a data-driven method to learn subword representation from characters. CharBERT Ma et al. (2020) learns two *subword-level* representations, respectively containing subword and character-level information. The ACE model (Wang et al., 2021) employs neural architecture search to determine the optimal combination of embeddings. Sanh et al. (2019) merge embeddings from various text granularities before inputting them into the encoder (Sanh et al., 2019). Shahzad et al. (2021) and Aguilar et al. (2018) employing separate encoders to extract contextualized representations for different granularities of text, which are later combined during inference. All these studies produce subword-level representations, but they produce no useful representations for other text granularities.

6.2 Character Representation Learning

Models like CANINE (Clark et al., 2022) and ByT5 (Xue et al., 2022) directly pre-train a character-level transformer to obtain character representations. However, character models could be hard to train as they assume less structure about the text. To mitigate this issue, Sun et al. (2023) uses a hierarchical structure to integrate word boundary information in the character model. Huang et al. (2023) learns character representation inside a subword model by treating characters as type variables in a causal model. Studies found that incorporating linguistic features of the characters, such as phonetic information (Matsuhira et al., 2023), Chinese character shape and Pinyin (Sun et al., 2021; Wei et al., 2023), can yield performance gains.

6.3 Multiple Granularities of Output

In contrast to the extensive research on processing multiple granularities as input, there have been limited studies proposing models that generate multiple granularities of output. In speech recognition, Sanabria and Metze (2018) train a single model to simultaneously produce text transcripts at different granularities, specifically characters, and subwords with varying vocabulary sizes. Srinivasan et al. (2019) employs a shared encoder but separate decoders for different output granularities, allowing decoders to generate outputs concurrently. Kremer et al. (2018)optimize different models for distinct granularities of text jointly, using a combined loss.

6.4 Multimodal NLP

Prior research has demonstrated the potential benefits of incorporating non-linguistic modalities in various NLP tasks. For instance, ChineseBERT (Sun et al., 2021) incorporates Pinyin and glyph information of Chinese characters during pretraining, leading to performance boosts in Chinese NLP tasks. Our work draws inspiration from vision-language models. Models like VisualBERT (Li et al., 2019) and VL-BERT (Su et al., 2020) learn a shared representation space for both images and language, utilizing a single transformer as the encoder for both modalities. In contrast, our model utilizes pretrained subword and character models and employs the co-attention module, as adopted by ViLBERT (Lu et al., 2019b), to facilitate information exchange between the two granularities.

7 Conclusion

In this paper, we introduce a novel entanglement model to effectively combine character and subword language models using co-attention modules. Unlike many prior works, our model produces mutually informed representations of subwords and characters, which could be used to process both subword and character-level tasks. Its architecture is model-agnostic, and it opens new directions for pretraining and scaling up. Empirically, our model has demonstrated improvements over the baseline models on various sequence labeling and text classification tasks. Our entanglement model achieves state-of-the-art results on various tasks/settings: POS tagging on TweeBank, NER on Ibo and Wolof from MasakhaNER, and intra-word code-switching on German-Turkish. Notably, the improvement of our model is most significant for noisy texts and low-resourced, morphologically rich languages. Furthermore, the entanglement model outperforms larger pretrained subword models, which have higher parameter counts, on most tasks. While our model features a simple architecture, incorporating extensions like positional embeddings or additional pretraining do not improve its performance, which implies that the model's structure facilitates the learning of relevant information during fine-tuning, rendering additional complexities unnecessary.

8 Limitations

Computational Efficiency Although our model demands greater computational resources and may have a slower optimization process compared to the backbone model (e.g., RoBERTa), it still faster than previous models like ACE, which utilize multiple embeddings from different models through neural architecture search. Moreover, in §4.5, we demonstrate that the performance of our model is comparable, if not superior, to the RoBERTa$_{\text{large}}$ model, which has a higher parameter count.

Model Extension Our model is designed to accommodate a maximum of two backbone models, and there is no straightforward way to extend it for the utilization of three or more backbone models. While exploring the entanglement model with three backbone models could be an intriguing avenue for researchers interested in word-level modeling, it's worth noting that the majority of current language modeling primarily focuses on the two levels (character or subword) employed in our entanglement models.

Pretraining During the pretraining phase, we observed a rapid decrease in the character MLM loss compared to the subword MLM loss. To introduce more challenging training objectives, one option is to mask out an entire word's worth of characters instead of just a single character at a time. This strategy could potentially encourage the model to capture more nuanced details in the text, leading to potential performance improvements. Additionally, due to limitations in computational resources, we performed pretraining on a subset (approximately 8%) of the corpus instead of conducting a full-scale pretraining. As such, further investigation into this approach is left to future research.

References

David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen H. Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Aremu Anuoluwapo, Catherine Gitau, Derguene Mbaye, Jesujoba Alabi, Seid Muhie Yimam, Tajuddeen Rabiu Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane MBOUP, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima DIOP, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, and Salomey Osei. 2021. MasakhaNER: Named entity recognition for African languages. *Transactions of the Association for Computational Linguistics*, 9:1116–1131.

Gustavo Aguilar, Adrian Pastor López-Monroy, Fabio González, and Thamar Solorio. 2018. Modeling noisiness to recognize named entities using multitask neural networks on social media. 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 1401–1412, New Orleans, Louisiana. Association for Computational Linguistics.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval:

Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

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

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.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

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

Jing Huang, Zhengxuan Wu, Kyle Mahowald, and Christopher Potts. 2023. Inducing character-level structure in subword-based language models with type-level interchange intervention training. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12163–12180, Toronto, Canada. Association for Computational Linguistics.

Hang Jiang, Yining Hua, Doug Beeferman, and Deb Roy. 2022. Annotating the Tweebank corpus on named entity recognition and building NLP models for social media analysis. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 7199–7208, Marseille, France. European Language Resources Association.

Omar Ka. 1987. *Wolof phonology and morphology: A non-linear approach*. Ph.D., University of Illinois at Urbana-Champaign, United States – Illinois. ISBN: 9798206799132.

Jan Kremer, Lasse Borgholt, and Lars Maaløe. 2018. On the inductive bias of word-character-level multi-task learning for speech recognition.

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

Priyanshu Kumar and Aadarsh Singh. 2020. NutCracker at WNUT-2020 task 2: Robustly identifying informative COVID-19 tweets using ensembling and adversarial training. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 404–408, Online. Association for Computational Linguistics.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019a. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019b. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. CharBERT: Character-aware pre-trained language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Manuel Mager, Özlem Çetinoğlu, and Katharina Kann. 2019. Subword-level language identification for intra-word code-switching.

Chihaya Matsuhira, Marc A. Kastner, Takahiro Komamizu, Takatsugu Hirayama, Keisuke Doman, Yasutomo Kawanishi, and Ichiro Ide. 2023. Ipa-clip: Integrating phonetic priors into vision and language pretraining.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Dat Quoc Nguyen, Thanh Vu, Afshin Rahimi, Mai Hoang Dao, Linh The Nguyen, and Long Doan. 2020a. Wnut-2020 task 2: Identification of informative covid-19 english tweets.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020b. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.

Matthew E. 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.

Ramon Sanabria and Florian Metze. 2018. Hierarchical multi task learning with ctc.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6949–6956. Issue: 01.

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.

Moemmur Shahzad, Ayesha Amin, Diego Esteves, and Axel-Cyrille Ngonga Ngomo. 2021. Inferner: an attentive model leveraging the sentence-level information for named entity recognition in microblogs. In *The international FLAIRS conference proceedings*, volume 34.

Tejas Srinivasan, Ramon Sanabria, and Florian Metze. 2019. Multitask learning for different subword segmentations in neural machine translation.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. Vl-bert: Pre-training of generic visual-linguistic representations.

Li Sun, Florian Luisier, Kayhan Batmanghelich, Dinei Florencio, and Cha Zhang. 2023. From characters to words: Hierarchical pre-trained language model for open-vocabulary language understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3605–3620, Toronto, Canada. Association for Computational Linguistics.

Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. 2021. ChineseBERT: Chinese pretraining enhanced by glyph and Pinyin information. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2065–2075, Online. Association for Computational Linguistics.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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

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

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Automated concatenation of embeddings for structured prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2643–2660, Online. Association for Computational Linguistics.

Xiao Wei, Jianbao Huang, Hang Yu, and Qian Liu. 2023. PTCSpell: Pre-trained corrector based on character shape and Pinyin for Chinese spelling correction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6330–6343, Toronto, Canada. Association for Computational Linguistics.

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

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

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

A Positional Embeddings

Since we are co-attending subword and character embeddings, it appears beneficial to re-introduce positional information in the co-attention modules. We do this by adding positional embeddings (PEs) to the character and subword embeddings output by the backbone encoder before they are passed to the co-attention modules. Since the character and subword sequence typically have different lengths, it is necessary to have strategies for translating between the character-level and subword-level PEs. We consider three potential strategies: Using strategy A, each character inherits the PE of the subword it belongs to. Using strategy B, each subword inherits the PE of its first character. Using strategy C, the PE of each subword is the average of its character's PEs. Table 8 demonstrates the three strategies on a small example. We experiment with an entanglement model with 1 co-attention module and sinusoidal absolute PEs used by the original transformer (Vaswani et al., 2017b).

word position	**A**	**dog**	**sat**
strategy A	1	2	3
strategy B	1	2	5
strategy C	1	(2+3+4)/3	(5+6+7)/3

char position	**A**	**d**	**o**	**g**	**s**	**a**	**t**
strategy A	1	2	2	2	3	3	3
strategy B	1	2	3	4	5	6	7
strategy C	1	2	3	4	5	6	7

Table 8: Strategies for mappings the PEs. The unhighlighted PEs are derived from the highlighted PEs by the rule specified.

B Pretraining

Since the co-attention modules are essentially transformer blocks, our model could be pretrained. To investigate the effect of pretraining on our model, we pretrain the model on a subset of the combined corpus of WikiText-103 (Merity et al., 2016) and Bookcorpus (Zhu et al., 2015). The model is trained on three types of objectives: subword-level masked language modeling (MLM) loss, character-level MLM loss, and a novel character-word matching loss that aims to align the representation space of the output character and subword embeddings, described below. Table 9 displays the results of the pre-trained model on WNUT-17 using different amount of data for pretraining, and we see that the model seems to perform worse when more data is used in pretraining.

B.1 Character-word Matching

In order to align the representation space of character and word embeddings, we propose a contrastive learning objective named character-subword matching, which is used during our pretraining step Figure 3 presents a visualization of the character-subword matching objective. For each character in T^c, we record a label for the subword that it belongs to. For example, consider the sentence `A la carte`. Character `A` would be labeled 1 and `l,a` would be labeled 2. We call the label sequence L^c

We compute the pairwise similarity (scaled dot product) between each subword-character pair, and we create a similarity matrix $S = H^s_* \cdot H^c_*$, where $S[i,j] = H^s_*[i] \cdot H^c_*[j]/a$, where a is a trainable constant. Therefore, we formulate the contrastive loss as follows:

$$\mathcal{L}_c = \sum_{j=1}^{m} \text{CrossEntropyLoss}(S, \text{ref} = L^c)$$

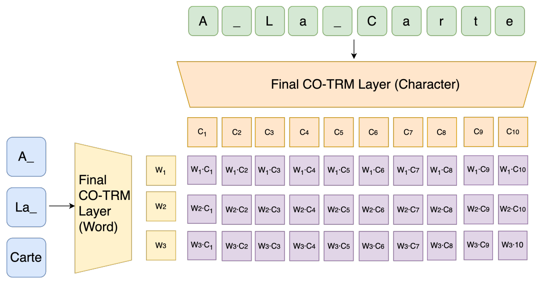
A
_
L
a
_
C
a
r
t
e
Final CO-TRM Layer (Character)
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10
A_
La_
Carte
Final CO-TRM Layer (Word)
W1
W2
W3
W1·C1 W1·C2 W1·C3 W1·C4 W1·C5 W1·C6 W1·C7 W1·C8 W1·C9 W1·C10
W2·C1 W2·C2 W2·C3 W2·C4 W2·C5 W2·C6 W2·C7 W2·C8 W2·C9 W2·C10
W3·C1 W3·C2 W3·C3 W3·C4 W3·C5 W3·C6 W3·C7 W3·C8 W3·C9 W3·10

Figure 3: Character-word matching loss

B.2 Optimization

To learn the parameters of our model, we optimize the model over three objectives. For MLM, we randomly masked out 15% of the tokens in the subword and character sequence. Take a single piece of text $(\mathbf{x})$ for example. We respectively compute the subword-level and character-level MLM loss as

Model	% data	# epoch	F1
RoBERTa$_{\text{base}}$	-	-	56.38
EM(#C = 1)	-	-	**57.80**
Side | **#C**			
SUBW | 1	$\sim 0.18\%$	15	56.59
1	$\sim 8\%$	1	51.71
2	$\sim 8\%$	1	52.25
6	$\sim 8\%$	1	53.14

Table 9: F1 scores for pretrained entanglement model in WNUT-17. % data refers to the % of the corpus used for pretraining. EM(#C = 1) refers to the un-pretrained entanglement model with 1 co-attention module.

follows:

$$\mathcal{L}_{mlm}^{sub}(\mathbf{x}) = -\sum_{t=1}^{n_i^s} \log(\mathbf{x}_t^s | \mathbf{x}_{\neq t}^s, \theta)$$

$$\mathcal{L}_{mlm}^{char}(\mathbf{x}) = -\sum_{t=1}^{n^c} \log(\mathbf{x}_t^c | \mathbf{x}_{\neq t}^c, \theta)$$

where $\mathbf{x}_t^s, \mathbf{x}_t^c$ are respectively subword and character tokens, and n^s, n^c are respectively the number of subword tokens and character tokens. $\mathbf{x}_{\neq t}^s$, $\mathbf{x}_{\neq t}^c$ means the complete character/subword sequence without token $\mathbf{x}_t^s, \mathbf{x}_t^c$ and other masked-out tokens, and θ refers to the parameters in our model.

Our model is then pretrained over the three objectives:

$$\mathcal{L}(\mathbf{x}) = \mathcal{L}_c(\mathbf{x}) + \mathcal{L}_{mlm}^{sub}(\mathbf{x}) + \mathcal{L}_{mlm}^{char}(\mathbf{x})$$

We pretrain our model on a random subset of the combined corpus of WikiText-103 (Merity et al., 2016) and Bookcorpus (Zhu et al., 2015). The model is trained for 8 hours on 4 Tesla V100 GPUs with 32 GB memory. The initial learning rate is 2e-5 and we used an Adam optimizer and a linear scheduler.