Parameter-Efficient Korean Character-Level Language Modeling

Marco Cognetta
Tokyo Institute of Technology
cognetta.marco@gmail.com

Sangwhan Moon
Tokyo Institute of Technology
sangwhan@iki.fi

Lawrence Wolf-Sonkin
Google
wolfsonkin@google.com

Naoaki Okazaki
Tokyo Institute of Technology
okazaki@c.titech.ac.jp

Abstract

Character-level language modeling has been shown empirically to perform well on highly agglutinative or morphologically rich languages while using only a small fraction of the parameters required by (sub)word models. Korean fits nicely into this framework, except that, like other CJK languages, it has a very large character vocabulary of 11,172 unique *syllables*. However, unlike Japanese Kanji and Chinese Hanzi, each Korean syllable can be uniquely factored into a small set of subcharacters, called *jamo*.

We explore a "three-hot" scheme, where we exploit the decomposability of Korean characters to model at the syllable level but using only jamo-level representations. We find that our three-hot embedding and decoding scheme alleviates the two major issues with prior syllable- and jamo-level models. Namely, it requires fewer than 1% of the embedding parameters of a syllable model, and it does not require tripling the sequence length, as with jamo models. In addition, it addresses a theoretical flaw in a prior three-hot modeling scheme.

Our experiments show that, even when reducing the number of embedding parameters by $> 99.6\%$ (from 11.4M to just 36k), our model suffers no loss in translation quality compared to the baseline syllable model.

1 Introduction

Subword modeling has been used for Korean to reduce the required vocabulary size due to its agglutinative nature and morphological richness. Several works have characterized the many subword tokenization strategies for Korean (Park et al., 2020; Moon and Okazaki, 2020).

(Sub)character modeling has been employed for translation in Asian languages (Nguyen et al., 2017; Ngo et al., 2019; Yu et al., 2017). Likewise, it has found use in generic Korean NLP tasks (Cho et al., 2019; Choi et al., 2017). Further, Stratos (2017) investigated exploiting the hierarchical nature of Korean characters for modeling. However, nearly all of these works are restricted to the input side only, either because the downstream task is a classification task or it is a translation task where Korean is the *source* language, and so syllable generation is unnecessary. An exception is Song et al. (2018), where the authors propose a "multi-hot" scheme that appears on both the embedding and decoding sides of a sequence-to-sequence denoising autoencoder applied to Korean spelling correction.

The baseline approaches of syllable- and jamo-level modeling each have downsides. Syllable models must have embeddings for all 11,172 syllables to provide full coverage, which requires an enormous number of parameters. On the other hand, jamo-level modeling requires fewer than 70 embeddings, but the sequence lengths become 3x longer, which greatly slows inference in attention-based models. Here, we propose a three-hot model[1] that addresses both of these issues, as well as an issue in the architecture from Song et al. (2018).

We emphasize that our goal is specifically to investigate parameter-efficient character-level modeling, and not to compare with (sub)word models.

1.1 Korean Syllabary

$\mathcal{V}_i$	ㄱ ㄲ ㄴ ㄷ ㄸ ㄹ ㅁ ㅂ ㅃ ㅅ ㅆ ㅇ ㅈ ㅉ ㅊ ㅋ ㅌ ㅍ ㅎ
$\mathcal{V}_v$	ㅏ ㅐ ㅑ ㅒ ㅓ ㅔ ㅕ ㅖ ㅗ ㅘ ㅙ ㅚ ㅛ ㅜ ㅝ ㅞ ㅟ ㅠ ㅡ ㅢ ㅣ
$\mathcal{V}_f$	ㄱ ㄲ ㄳ ㄴ ㄵ ㄶ ㄷ ㄹ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅁ ㅂ ㅄ ㅅ ㅆ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ $\oslash$

Table 1: The three jamo classes. $\oslash \in \mathcal{V}_f$ represents the *absence* of a final consonant.

[1]Our three-hot implementation has been packaged as a library alongside code to reproduce our experiments: https://github.com/mcognetta/ThreeHotKoreanModeling.

a) 기계번 → 역

b) ㄱ ㅣ ∅ ㄱ ㅖ ∅ ㅂ ㅓ ㄴ → ㅇ → ㅕ → ㄱ

c) (ㄱ , ㅣ , ∅) (ㄱ , ㅖ , ∅) (ㅂ , ㅓ , ㄴ) → (ㅇ , ㅕ , ㄱ)

Figure 1: Modeling scenarios for predicting the syllable 역 (*yeok*:'decode'), from the context "기계번"(*gigyebeon*:'machine translate') using a) syllable, b) jamo, and c) three-hot modeling. Notice that in jamo modeling, the prediction is spread over three time-steps.

Korean syllables are made up of three subcharacters: an initial consonant, a vowel, and an optional final consonant, collectively known as jamo. There are 19, 21, and 28 initial consonants, vowels, and final consonants (including the lack of a final consonant, represented by $\oslash$), denoted $\mathcal{V}_i$, $\mathcal{V}_v$, and $\mathcal{V}_f$, respectively, shown in Table 1. These classes are disjoint, and even visually similar jamo in $\mathcal{V}_i$ and $\mathcal{V}_f$ (e.g., ㄱ) are distinct subcharacters.

Any Korean syllable can be uniquely decomposed into three jamo, and any choice $(i, v, f) \in \mathcal{V}_i \times \mathcal{V}_v \times \mathcal{V}_f$ corresponds to a unique syllable. For example, 한: (ㅎ, ㅏ,ㄴ) and 무: (ㅁ,ㅜ,$\oslash$).

Specific details about Korean jamo and their Unicode representations are in Appendix A.

2 Three-hot Modeling

Naïve syllable-level and jamo-level modeling use one-hot encodings to represent each token. We propose a three-hot scheme, where jamo triplets representing syllables are consumed and generated at each step. Figure 1 gives examples of each of the considered modeling schemes.

2.1 Embedding

In three-hot modeling, we learn embeddings for jamo and combine them to produce a syllable embedding as:

$$emb_s = emb_i + emb_v + emb_f,$$

where $+$ is vector addition. We also experimented with combining $emb_{i,v,f}$ via concatenation, but this method requires more parameters and we observed no performance difference.

Similar factored embeddings have been considered before in language-agnostic settings (Svenstrup et al., 2017) and for Korean specifically (Stratos, 2017).

2.2 Independent Three-hot Decoding

Song et al. (2018) proposed a three-hot decoder for a sequence-to-sequence autoencoder which produces syllables by predicting three jamo subcharacters as a three-hot vector. In their scheme, they predict the individual jamo *independently*, as:

$$\mathcal{P}(s \mid h) \approx \mathcal{P}(i \mid h) \times \mathcal{P}(v \mid h) \times \mathcal{P}(f \mid h).$$

However, a syllable's jamo are *not* independent, and thus this method does not capture the true joint probability distribution:

$$\mathcal{P}(s \mid h) = \mathcal{P}(i, v, f \mid h).$$

2.3 Conditional Three-hot Decoding

To properly model the true joint distribution, we factor it as:

$$\mathcal{P}(s \mid h) = \mathcal{P}(i \mid h) \times \mathcal{P}(v \mid i, h) \times \mathcal{P}(f \mid i, v, h).$$

Given a context embedding h from a base language model, we first predict the initial consonant, π_i, from $\texttt{softmax}(I(h_i))$, where $I : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{V}_i|}$ and h_i is a vector generated from h and an initial vector h_0. Then, we generate a continuous embedding for π_i via $\texttt{embedding}_i : \mathbb{R}^{|\mathcal{V}_i|} \to \mathbb{R}^d$. The continuous embedding is combined with h_i and used to predict the vowel jamo, π_v. Likewise, π_v is re-embedded and used to predict the final consonant, π_f. As such, a triplet (π_i, π_v, π_f) is generated similarly to a three-step, unrolled RNN:

$$\begin{aligned}
h_i &= \texttt{tanh}(W_e h + W_h h_0) \\
\pi_i &\sim \texttt{softmax}(I(h_i)) \\
emb_i &= \texttt{embedding}_i(\pi_i) \\
h_v &= \texttt{tanh}(W_e emb_i + W_h h_i) \\
\pi_v &\sim \texttt{softmax}(V(h_v)) \\
emb_v &= \texttt{embedding}_v(\pi_v) \\
h_f &= \texttt{tanh}(W_e emb_v + W_h h_v) \\
\pi_f &\sim \texttt{softmax}(F(h_f))
\end{aligned} \tag{1}$$

In the decoding layer, we define $\texttt{embedding}_{i,v}$, but not $\texttt{embedding}_f$. This is because once the prediction π_f has been made, decoding for that time-step is done, and so an embedding emb_f is not needed to generate any more jamo.

2.3.1 Parameter Reduction via Weight Sharing and Diagonal RNNs

A common way to reduce parameter counts is to share weights between the embedding and decoding layers. For jamo and syllable models, this is

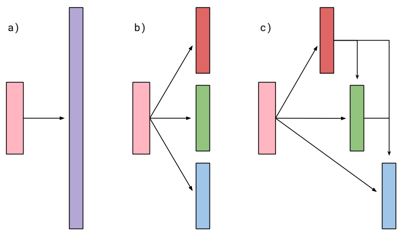

Figure 2: The three decoding types: `a)` one-hot decoding (jamo or syllable), `b)` independent three-hot decoding (Song et al., 2018), and `c)` conditional three-hot decoding (ours). The light pink box is a contextual embedding h, while red, green, and blue are $\mathcal{V}_i$, $\mathcal{V}_v$, and $\mathcal{V}_f$, respectively.

straightforward, as the one-hot embedding table can be reused for the final linear output layer.

For independent three-hot models, the embedding weights from Section 2.1 can also be reused in the final output layers. For conditional three-hot models, the embedding weights can be used for both $\texttt{embedding}_{i,v}$ and I, V, F in Equation 1, so that only the matrices W_e and W_h are unshared. Additionally, we hypothesize that, due to the strictly bounded context length Equation 1, a fully dense RNN transition matrix may be excessive, and so we experiment with replacing W_e and W_h with diagonal matrices (Sübakan and Smaragdis, 2017).

2.4 Prediction Order

In multi-label classification, it has been observed that the order in which labels are generated can impact a model's accuracy (Read et al., 2021). For three-hot decoding, we experiment with the six triplet-generation permutations to see if there is any significant difference. This does not require any modification of the training data or architecture, as we need only permute the order of Equation 1.

2.5 Triplet Representation

Three-hot decoding models output jamo triplets representing a full syllable. However, the model may need to process non-Korean characters, which do not have a three-hot jamo decomposition. To unify the representations of Korean and non-Korean tokens, we add all non-Korean tokens to $\mathcal{V}_i$ and introduce a padding character $\mathbf{x} \in \mathcal{V}_{v,f}$. Then, a non-Korean token c is represented by $(c, \mathbf{x}, \mathbf{x})$.

To ensure a fair comparison between the three-hot decoding schemes and the jamo model, no logic is used to prohibit the generation of degenerate triplets, such as a non-Korean symbol followed by jamo subcharacter. Syllable and jamo models do not generate $\mathbf{x}$ when predicting non-Korean tokens, but jamo models are trained to predict $\oslash$ if a syllable does not have a final consonant.

3 Parameter Counts

Table 2 shows the parameter counts for each of the architectures. Since the three-hot models use jamo embeddings, they have the same number of embedding parameters as the jamo-level models. The conditional three-hot models have additional parameters for the three-stage decoding step: two internal transition matrices W_e and W_h (Equation 1), which dominate the decoding layer's parameter count. But, when the transition matrices are diagonalized and weight sharing is used, the total number of parameters is only 1k more than the shared-weight jamo and independent three-hot models.

The syllable models contain several orders of magnitude more parameters than any other architecture. In the most extreme case, unshared syllable vs. shared, diagonal three-hot, the latter has only $\frac{36k}{11.4M} \approx 0.32\%$ as many embedding parameters.

Our underlying LM is a transformer model with 15.7M parameters on the target side[2], <u>not</u> counting those for embedding or generation. Thus, the unshared syllable-level embedding/decoding layers increase the target-side parameter count by 73%, while the unshared jamo, independent, and diagonal conditional three-hot increase it by less than 2%, and non-diagonal conditional three-hot by less than 6%.

4 Evaluation Metrics

Since we are modeling at two different granularities, perplexity is not immediately comparable between the different architectures. To unify them, we use bits-per-jamo (BPJ) as our granularity-agnostic metric.

We use BLEU and chrF to evaluate the quality of the translations produced by our models. Since our models work on different granularities, we canonicalize their outputs for comparison.

For syllable-level modeling, we split all syllables into their jamo subcharacters and leave all non-Korean-syllable tokens as is. For three-hot models, triplets are flattened to a string of three characters. We remove the **x** pad from non-Korean tokens and replace degenerate triplets with a special `BAD` token. For all models, all jamo are converted to a

[2]See Appendix B for architecture details.

	Embedding	Decoding	Total	BPJ	BLEU	chrF
Syllable (unshared)	5.7M	5.7M	11.4M	33.9	14.1	38.1
Syllable (shared)	5.7M	-	5.7M	0.342	14.0	38.1
Jamo (unshared)	35k	35k	70k	0.355	13.7	37.8
Jamo (shared)	35k	-	35k	0.356	14.1	38.0
Three-hot (Ind., unshared)	35k	35k	70k	0.555	7.9	28.9
Three-hot (Ind., shared)	35k	-	35k	0.556	8.3	29.4

	Embedding	Decoding	Total	BPJ	BLEU	chrF
Three-hot (IVF, unshared)	35k	579k	614k	0.287	14.3	38.1
Three-hot (IVF, shared)	35k	524k	559k	0.292	14.1	38.1
Three-hot (IVF, diag., unshared)	35k	71k	106k	0.293	14.2	38.2
Three-hot (IVF, diag., shared)	35k	1k	36k	0.306	14.0	38.0
Three-hot (FIV, unshared)	35k	579k	614k	0.289	14.0	37.9
Three-hot (FIV, shared)	35k	524k	559k	0.294	14.1	37.8
Three-hot (FIV, diag., unshared)	35k	71k	106k	0.294	14.0	37.9
Three-hot (FIV, diag., shared)	35k	1k	36k	0.304	13.8	37.8

Table 2: Parameter counts (when $d = 512$, as in our experiments) and metrics (BPJ, BLEU, and chrF) for each of the architectures. For brevity, only some decoding orders are listed here. The decoding column lists only parameters that are not shared with the embedding layer. A complete set of metrics is given in Appendix C.

canonical *compatibility jamo* format (see Appendix A). Finally, we remove all punctuation.

For BLEU, we use the standard n-gram order of 4, since this operates on the word-level (in our experiments, contiguous token sequences surrounded by whitespace). The default chrF character n-gram order is 6, which we interpret as meaning 6 *syllables* for Korean. Since we measure our metrics on decomposed jamo sequences, we use a jamo n-gram order of 18.

5 Experiments

We use an English-Korean news translation dataset from AI Hub[3] with 400k sentences. We remove all sentence pairs that have more than 100 syllables on the Korean side, leaving 365k , from which we select 5k and 5k for testing and validation.

As our only goal is to evaluate the effectiveness of three-hot modeling compared to naïve syllable- and jamo-level modeling, we use the same base transformer model architecture and hyperparameters in all experiments and only change the target-side embedding and decoding layers. The complete hyperparameter list is given in Appendix B.

In total, we train 30 models: unshared and shared weights for syllable, jamo, and three-hot independent modeling (6 total) and $\{\text{unshared, shared}\} \times \{\text{dense, diagonal}\}$ for each of the 6 conditional three-hot prediction orders (24 total). Each model is trained for 50 epochs, and the epoch with the lowest bits-per-jamo on the validation set is used.

For inference, we use a beam size of 15 for syllable models and 8 for jamo models. Three-hot decoding is two-staged: an internal beam search constructs triplets jamo-by-jamo, and the highest probability triplets are added to the outer beam as syllables. For the independent three-hot models, we use beam size of 5 and an internal beam size of 3. For the conditional three-hot models, we use a beam size of 15 and an internal beam size of 4. These hyperparameters were determined by a sweep search on the validation set.

[3]https://aihub.or.kr/

6 Results and Analysis

Table 2 lists the metrics for each model type.

Independent three-hot modeling performs the worst across all metrics. On translation in particular, we suspect that it is because the model can generate high probability but meaningless triplets, especially in scenarios where several reasonable syllable continuations exist. The highest probability triplet may contain individual jamo that come from each of the true high probability syllable continuations, but which are combined in a way that forms a meaningless character. Such triplets quickly saturate the beam with gibberish contexts and degrade the translation quality.

For bits-per-jamo, conditional three-hot models vastly outperform all other architectures. However, this has some caveats. Since the syllable model must predict all three jamo simultaneously, the bits-per-jamo is spread evenly throughout the component jamo. On the other hand, conditional three-hot modeling has a relatively difficult time predicting the first jamo, but the subsequent elements are increasingly easy due to the additional conditioning (Appendix C lists BPJ-per-jamo-class for each model). Jamo models must deal with a longer context (and thus less focused attention) and the output layer can output any class of jamo or non-Korean tokens at any time step, which makes modeling less structured and more difficult than conditional three-hot and syllable-level models.

We observed that the independent three-hot models had the highest perplexity on $\mathcal{V}_v$, followed by $\mathcal{V}_i$, then $\mathcal{V}_f$. Since this family generates jamo independently, it approximates the true entropy of the marginal subcharacter distributions. Thus, we conjectured that (f, i, v) order would be the best and (v, i, f) the worst for model accuracy, but found that no prediction order consistently outperformed the others.

For translation, the BLEU and chrF scores for jamo, syllable, and conditional three-hot models are extremely close. We found that, as we moved from unshared, dense three-hot models to shared, diagonal models, there was a slight loss in BPJ performance, but even in the diagonal, shared case, all three-hot models perform on-par with syllable models on translation ($\pm$ 0.3 BLEU and $\pm$ 0.3 chrF), supporting our claims that the syllable models are massively overparameterized, and using fully dense W_e and W_h matrices in Equation 1 is unnecessary.

7 Conclusion

We presented a conditional three-hot decoder for Korean character-level language models. Our model addresses several issues with other Korean (sub)character-level modeling schemes. Compared to syllable-level models, it uses only a small fraction of the number of parameters, and, unlike jamo-level models, it does not triple the sequence length, avoiding the resulting inference time increase with attention mechanisms. It also addresses a theoretical flaw in a prior three-hot decoding scheme, where a syllable's subcharacters were generated independently. Finally, we proposed several variants of our model to further reduce parameters.

On a character-level translation task, we found that all of our conditional three-hot models perform on-par with jamo and syllable models, even when using as little as $\sim$0.3% of the embedding parameters of a syllable model. They also outperform the prior independent three-hot model in every metric.

These results suggest that conditional three-hot modeling is an efficient and principled method of character-level Korean language modeling.

8 Acknowledgments

These research results were obtained partially from the commissioned research (No. 225) by National Institute of Information and Communications Technology (NICT), Japan.

9 Limitations

One limitation of this work is that it is specific to the writing system typically used in Korean, Hangul. A similar idea could be used for subword modeling in other scripts, but the idea presented here draws specifically on the hierarchical and compositional nature of Hangul syllables that is unique to the script itself. A second is that we did not compare to state-of-the-art non-character-level Korean translation models. However, this was on purpose, as the point of this paper is specifically to investigate efficient character-level modeling.

References

Won-Ik Cho, Seok Min Kim, and Nam Soo Kim. 2019. Investigating an effective character-level embedding in korean sentence classification. *CoRR* abs/1905.13656.

Sanghyuk Choi, Taeuk Kim, Jinseok Seol, and Sang-goo Lee. 2017. A syllable-based technique for word embeddings of Korean words. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*. Association for Computational Linguistics, Copenhagen, Denmark, pages 36–40.

Sangwhan Moon and Naoaki Okazaki. 2020. Jamo pair encoding: Subcharacter representation-based extreme Korean vocabulary compression for efficient subword tokenization. In *Proceedings of the 12th Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, pages 3490–3497.

Thi-Vinh Ngo, Thanh-Le Ha, Phuong-Thai Nguyen, and Le-Minh Nguyen. 2019. How transformer revitalizes character-based neural machine translation: An investigation on japanese-vietnamese translation systems. *CoRR* abs/1910.02238.

Viet Nguyen, Julian Brooke, and Timothy Baldwin. 2017. Sub-character neural language modelling in Japanese. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*. Association for Computational Linguistics, Copenhagen, Denmark, pages 148–153.

Kyubyong Park, Joohong Lee, Seongbo Jang, and Dawoon Jung. 2020. An empirical study of tokenization strategies for various korean NLP tasks. Association for Computational Linguistics, pages 133–142.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2021. Classifier chains: A review and perspectives. *J. Artif. Intell. Res.* 70:683–718.

Chisung Song, Myungsoo Han, Hoon Young Cho, and Kyong-Nim Lee. 2018. Sequence-to-sequence autoencoder based korean text error correction using syllable-level multi-hot vector representation. In *Proceedings of HCLT (in Korean)*. pages 661–664.

Karl Stratos. 2017. A sub-character architecture for korean language processing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics, pages 721–726.

Y. Cem Sübakan and Paris Smaragdis. 2017. Diagonal rnns in symbolic music modeling. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2017, New Paltz, NY, USA, October 15-18, 2017*. pages 354–358.

Dan Svenstrup, Jonas Meinertz Hansen, and Ole Winther. 2017. Hash embeddings for efficient word representations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. pages 4928–4936.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint embeddings of Chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 286–291.

A Hangul and Jamo

Korean is divided into three Unicode blocks: syllables[4], jamo[5], and *compatability* jamo[6].

The syllable block contains a codepoint for each of the 11,172 unique syllables and the jamo block contains one codepoint for each of the initial, vowel, and final jamos, separated by class. Table 1 shows all of the jamo. Note that some visually identical jamo exist in both the initial and final consonant classes (e.g., ㄱ). These are technically distinct jamo, hence the need for two unique codepoints. The compatibility jamo block contains a codepoint for each consonant and vowel, but merges visually identical ones from across different classes. In total there are 51 compatibility jamo. Conversion from syllables to jamo and compatibility jamo is deterministic using basic modular arithmetic on the codepoints[7]. Likewise, it is trivial to recompose jamo into syllables. Since compatibility jamo has ambiguities about which class a specific jamo belongs to, it must be composed greedily from left to right using a state machine. However, in all cases, assuming that that jamo sequence is valid, both jamo and compatibility sequences can be unambiguously recomposed into syllables.

In our experiments, we always convert from syllables/triplets/jamo to compatibility jamo, and never from compatibility jamo, so all ambiguities are avoided even in the case of invalid jamo sequences.

B Baseline Transformer LM

A single baseline transformer model configuration was used across all experiments, with only the target side embedding and decoding layers swapped out. Table 3 lists the relevant hyperparameters. All models were trained on an NVIDIA RTX A6000 GPU using PyTorch 1.12.

Hyperparameter	**Value**
English Vocabulary	30k BPE
Embedding Dimension	512
Feed Forward Dimension	512
Encoder Layers	6
Decoder Layers	6
Heads	8
Optimizer	ADAM
Learning Rate	1e-4
Betas	(0.9, 0.98)
Batch Size (tokens)	4k (10k for Jamo)
Epochs	50

Table 3: Hyperparameters for the base transformer language model.

C Full Metrics

The full metrics for all experients are listed in Table 4. This is the complete version of Table 2.

[4]https://en.wikipedia.org/wiki/Hangul_Syllables

[5]https://en.wikipedia.org/wiki/Hangul_Jamo_(Unicode_block)

[6]https://en.wikipedia.org/wiki/Hangul_Compatibility_Jamo

[7]https://en.wikipedia.org/wiki/Korean_language_and_computers#Hangul_in_Unicode

	Embedding	Decoding	Total	BPJ	BPJ (i)	BPJ (v)	BPJ (f)	BLEU	chrF
Syllable (unshared)	5.7M	5.7M	11.4M	0.339	-	-	-	14.1	38.1
Syllable (shared)	5.7M	-	5.7M	0.342	-	-	-	14.0	38.1
Jamo (unshared)	35k	35k	70k	0.355	-	-	-	13.7	37.8
Jamo (shared)	35k	-	35k	0.356	-	-	-	14.1	38.0
Three-hot (Ind., unshared)	35k	35k	70k	0.555	0.588	0.599	0.478	7.9	28.9
Three-hot (Ind., shared)	35k	-	35k	0.556	0.589	0.600	0.478	8.3	29.4
Three-hot (IVF, unshared)	35k	579k	614k	0.287	0.556	0.208	0.099	14.3	38.1
Three-hot (IVF, shared)	35k	524k	559k	0.292	0.561	0.214	0.103	14.1	38.1
Three-hot (IVF, diag., unshared)	35k	71k	106k	0.293	0.561	0.211	0.108	14.2	38.2
Three-hot (IVF, diag., shared)	35k	1k	36k	0.306	0.571	0.226	0.121	14.0	38.0
Three-hot (IFV, unshared)	35k	579k	614k	0.288	0.556	0.127	0.183	14.0	37.7
Three-hot (IFV, shared)	35k	524k	559k	0.293	0.563	0.131	0.186	13.9	37.8
Three-hot (IFV, diag., unshared)	35k	71k	106k	0.293	0.560	0.136	0.183	14.1	38.1
Three-hot (IFV, diag., shared)	35k	1k	36k	0.305	0.569	0.153	0.195	14.1	38.1
Three-hot (VIF, unshared)	35k	579k	614k	0.288	0.175	0.590	0.100	14.0	38.2
Three-hot (VIF, shared)	35k	524k	559k	0.293	0.179	0.597	0.102	14.0	38.3
Three-hot (VIF, diag., unshared)	35k	71k	106k	0.294	0.178	0.595	0.109	14.1	37.9
Three-hot (VIF, diag., shared)	35k	1k	36k	0.305	0.190	0.605	0.120	14.3	38.1
Three-hot (VFI, unshared)	35k	579k	614k	0.287	0.114	0.589	0.160	14.1	38.1
Three-hot (VFI, shared)	35k	524k	559k	0.292	0.119	0.596	0.162	14.3	38.2
Three-hot (VFI, diag., unshared)	35k	71k	106k	0.294	0.126	0.595	0.161	13.9	38.1
Three-hot (VFI, diag., shared)	35k	1k	36k	0.306	0.142	0.606	0.170	13.8	37.8
Three-hot (FIV, unshared)	35k	579k	614k	0.289	0.262	0.128	0.478	14.0	37.9
Three-hot (FIV, shared)	35k	524k	559k	0.294	0.266	0.132	0.484	14.1	37.8
Three-hot (FIV, diag., unshared)	35k	71k	106k	0.294	0.264	0.137	0.482	14.0	37.9
Three-hot (FIV, diag., shared)	35k	1k	36k	0.304	0.276	0.151	0.486	13.8	37.8
Three-hot (FVI, unshared)	35k	579k	614k	0.289	0.116	0.273	0.478	14.3	38.2
Three-hot (FVI, shared)	35k	524k	559k	0.293	0.120	0.277	0.482	14.1	38.3
Three-hot (FVI, diag., unshared)	35k	71k	106k	0.294	0.125	0.276	0.481	14.1	38.0
Three-hot (FVI, diag., shared)	35k	1k	36k	0.305	0.143	0.287	0.487	14.2	38.0

Table 4: The complete metrics for Table 2. For each model architecture and prediction order, parameter counts (when $d = 512$, as in our experiments) and metrics (BPJ, BLEU, and chrF) are given. Additionally, for three-hot models, the BPJ-per-jamo-class is provided. The decoding column lists only unshared parameters.