Break it Down into BTS: Basic, Tiniest Subword Units for Korean

Nayeon Kim[1*], **Jun-Hyung Park**[1*], **Joon-Young Choi**[2],
Eojin Jeon[2], **Youjin Kang**[1], **SangKeun Lee**[1,2]
[1]Department of Computer Science and Engineering [2]Department of Artificial Intelligence
Korea University, Seoul, Republic of Korea
{lilian1208, irish07, johnjames, skdlcm456, yjkang10, yalphy}@korea.ac.kr

Abstract

We introduce **B**asic, **T**iniest **S**ubword (**BTS**) units for the Korean language, which are inspired by the invention principle of *Hangeul*, the Korean writing system. Instead of relying on 51 Korean consonant and vowel letters, we form the letters from **BTS** units by adding strokes or combining them. To examine the impact of **BTS** units on Korean language processing, we develop a novel **BTS**-based word embedding framework that is readily applicable to various models. Our experiments reveal that **BTS** units significantly improve the performance of Korean word embedding on all intrinsic and extrinsic tasks in our evaluation. In particular, **BTS**-based word embedding outperforms the state-of-the-art Korean word embedding by 11.8% in word analogy. We further investigate the unique advantages provided by **BTS** units through in-depth analysis. Our code is available at `https://github.com/irishev/BTS`.

1 Introduction

Distributed word representations, known as word embeddings, play a fundamental role in natural language processing (NLP). With their capability to formulate semantic and syntactic relationships between words, word embeddings serve as a useful feature for various NLP applications, such as machine translation (Qi et al., 2018), sentence classification (Kim, 2014), and named entity recognition (Lample et al., 2016). Initial approaches to word embeddings, such as Word2vec (Mikolov et al., 2013a,b) and GloVe (Pennington et al., 2014), represent each word in a vocabulary using distinct vectors. To improve the quality of word representations, many approaches have considered the internal structure of words with subword information. (Kim et al., 2016; Wieting et al., 2016; Bojanowski et al., 2017; Sasaki et al., 2019; Zhao et al., 2020).

For non-English languages, prior studies (Park et al., 2018; Chen et al., 2020) have demonstrated that word embeddings should be designed to consider their linguistic structures. In particular, Korean has a unique and complex linguistic structure as a morphologically rich and agglutinative language (Song, 2006). In the realm of Korean word embedding, decomposing a word into *jamo* letters[1] is believed to be the best feasible option for capturing syntactic and semantic information of Korean words, which has been empirically supported by Park et al. (2018).

However, previous approaches have overlooked the academic background of the Korean alphabet, *Hangeul*. According to *Hunminjeongeum*[2], *Hangeul* involves five basic consonants (ㄱ, ㄴ, ㅁ, ㅅ, ㅇ) and three basic vowels (ㆍ, ㅡ, ㅣ). Other letters are made from the basic letters by adding strokes or combining together (National Hangeul Museum, 2018). This unique invention principle of *Hangeul* indeed helps to understand the Korean linguistic structure. Although this principle is well-recognized by the Korean linguistic community, its application to Korean NLP is yet to be explored.

In this paper, we firstly bring attention to the overlooked background of *Hangeul* for Korean NLP. We propose novel decomposition units to analyze the Korean linguistic structure called **B**asic, **T**iniest **S**ubword (**BTS**) units, based on the invention principle of *Hangeul*. **BTS** units involve the basic consonants and vowels of *Hangeul* and represent all *jamo* letters by themselves or through expansion. We implement this concept by developing a simple-yet-effective **BTS**-based word embedding framework. To verify the effectiveness of **BTS** units, we evaluate our **BTS**-based embeddings on the intrinsic tasks of word analogy and

[*] Authors contributed equally.

[1] In Korean linguistics, *jamo* is defined phonologically and distinguished from the corresponding letters used in writing. We indicate them as '*jamo* letters' in this paper.

[2] *Hunminjeongeum* refers to the name of a book describing the background of *Hangeul*.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 7007 - 7024
December 7-11, 2022 ©2022 Association for Computational Linguistics

similarity, and the extrinsic task of sentiment analysis. The results demonstrate that our embeddings consistently outperform the state-of-the-art Korean word embeddings on all evaluation tasks. Through our in-depth analysis, we verify that **BTS** units improve both syntactic and semantic information in Korean words, leading to consistent performance improvements. We summarize our contributions as follows:

- We introduce novel basic units for Korean called **BTS** units, based on the invention principle of *Hangeul*. To the best of our knowledge, our work is the first to investigate and examine the deep insights of **BTS** units in Korean NLP.[3]
- We develop a simple-yet-effective word embedding framework based on **BTS** units that is readily applicable to various NLP models.
- We demonstrate the effectiveness of **BTS** units through extensive experiments and verify that our **BTS**-based embeddings outperform the state-of-the-art Korean word embeddings on all intrinsic and extrinsic tasks in our evaluation.

2 Related Work

2.1 Language-Specific Word Embeddings

Word embeddings have long been studied in the NLP field, providing useful features for various tasks. Mikolov et al. (2013a,b) and Pennington et al. (2014) have proposed to assign continuous representations to each word by training word vectors based on the co-occurrence of words. However, such word-level approaches are unsuitable for morphologically rich languages that tend to have structurally and semantically similar yet non-identical words. To handle this problem, many works have focused on assigning vectors to character-level subwords (Kim et al., 2016; Wieting et al., 2016). FastText (Bojanowski et al., 2017) has proposed to train vectors corresponding to subword n-grams by using the summation of the subword vectors as the word vector.

It has recently been demonstrated that linguistic structures of languages must be considered in word embeddings (Park et al., 2018; Chen et al., 2020). Since the aforementioned methods were specifically designed for English and other Latin script-based languages, there have been studies on language-specific word embeddings that take the linguistic structure of non-English languages into account. For example, previous works on Chinese and Japanese embeddings have introduced diverse sub-elements of characters (Sun et al., 2014; Yu et al., 2017; Cao et al., 2018; Karpinska et al., 2018).

2.2 Word Embedding for Korean NLP

There have been studies on subword information in Korean, which is an agglutinative language. Various methods have been proposed to obtain subwords in morpheme units and inter-character structure units. Stratos (2017) has demonstrated the utility of decomposing Korean into consonants and vowels by applying the results to sentence parsing. Choi et al. (2017) have suggested learning word representations using character-level embeddings.

However, as each Korean character is a combination of consonants and vowels, it can be decomposed into smaller units. Park et al. (2018) have suggested effective subword-level Korean word representations based on n-gram extraction from Korean words decomposed into consonants and vowels. To better capture the distinct characteristics of the Korean language, we propose to decompose Korean words into the basic units of *Hangeul*.

3 Method - BTS

In this section, we introduce **BTS** units for Korean and present the implementation details.

3.1 Linguistic Structure of Korean Words

Korean words are formed by an explicit hierarchical structure, as shown in Figure 1(a). First, a Korean word can be decomposed into one or more characters. Each Korean character corresponds to a syllable, whereas English syllables are formed by a sequential combination of characters. Korean characters in turn can be decomposed into three *jamo* letters, i.e., consonants and vowels in Korean. Specifically, each Korean character strictly consists of the initial sound 'chosung', the middle sound 'joongsung', and the final sound 'jongsung'. Chosung and jongsung are consonants, while joong-

[3] Korean input systems in mobile phones have adopted the backgrounds of *Hangeul* for efficiency, but they are limited to either a vowel-oriented variant, e.g., *Cheonjiin* input system, or a consonant-oriented variant, e.g., *Naratgeul* (or *EZ Hangeul*). Unlike the input systems in mobile phones, the **BTS** units involve both vowels and consonants alike, truly conforming to the invention principle of *Hangeul*.

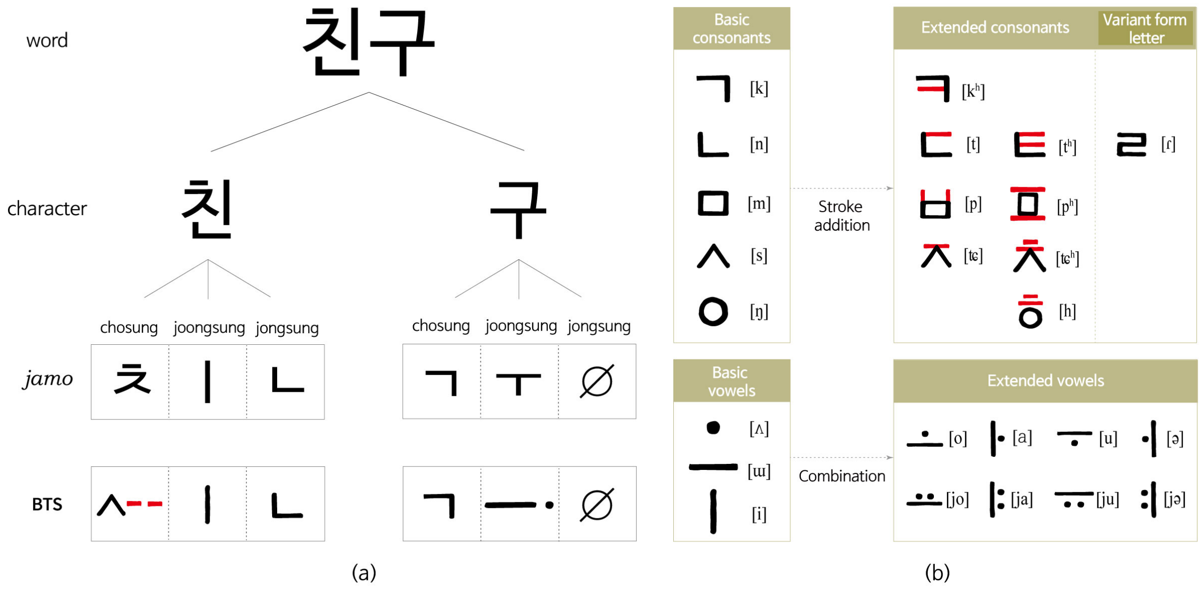


Figure 1: (a) Decomposition of the Korean word "친구$_{friend}$" into characters, *jamo* letters and the proposed **BTS** units. Each *jamo* letter is further decomposed into basic consonants and vowels. (b) Demonstration of basic consonant extension (via stroke addition) and basic vowel combination. In modern Korean, five basic consonants, nine extended consonants, two basic vowels (excluding '· [ʌ]') and eight combined vowels are used.

sung is a vowel. Chosung and joongsung are necessary for forming a character, whereas jongsung can be empty. A character is written with chosung at the top, joongsung at the right or below of chosung, and jongsung (if not empty) at the bottom. See Appendix A.2 for additional details.

3.2 BTS-Level Decomposition

We decompose Korean words into **BTS** units, based on the invention principle of *Hangeul*. **BTS** units comprise eight basic units (ㄱ, ㄴ, ㅁ, ㅅ, ㅇ, ·, ㅡ, ㅣ), each of which is an n-stroke letter defined as an atomic unit in *Hunminjeongeum*.

BTS-level consonants

Modern *Hangeul* consonants consist of five basic consonants (ㄱ, ㄴ, ㅁ, ㅅ, ㅇ) and nine extended consonants (ㅋ, ㄷ, ㅌ, ㄹ, ㅂ, ㅍ, ㅈ, ㅊ, ㅎ). The nine extended consonants are formed by adding stroke(s), red lines in Figure 1(b), to the five basic consonants.

Specifically, adding strokes to the basic consonants gives first-level extended consonants (ㅋ, ㄷ, ㅂ, ㅈ) and second-level extended consonants (ㅌ, ㅍ, ㅊ, ㅎ), which express stronger sounds of basic consonants (National Hangeul Museum, 2018). We decompose consonants into the basic consonants and a symbol for adding a stroke '-', e.g., 'ㅋ' is decomposed into 'ㄱ-'.

A total of 30 consonant letters are used for chosung and jongsung, which are written using 14 consonants as in their original form or by combining them. It is noteworthy that the consonant 'ㄹ' is an extension of 'ㄴ', but is not a stronger sound (National Hangeul Museum, 2018). For this reason, we have an exception in this work - we deliberately take 'ㄹ' as a basic consonant.

BTS-level vowels

Modern *Hangeul* vowels are formed by the combination of three basic vowels (·, ㅡ, ㅣ)[4]. By combining the basic vowels, first-level combined vowels (ㅗ, ㅏ, ㅜ, ㅓ) and second-level combined vowels (ㅛ, ㅑ, ㅠ, ㅕ) are created. For example, 'ㅑ' can be decomposed into 'ㅏ' and '·', where 'ㅏ' consists of 'ㅣ' and '·'. As a result, 'ㅑ' can be broken down into 'ㅣ··' (National Hangeul Museum, 2018). Note that the basic vowel '·' is distinct from the aforementioned stroke letter '-'.

A total of 21 vowel letters are used as joongsung, and written using 10 vowels (excluding '·') as in their original form or by combining them.

Throughout this paper, we refer to **BTS**-level decomposition as the decomposition of Korean words into basic consonants, strokes, and basic vowels.

[4] According to Sino-Korean pronunciation, '·', 'ㅡ', and 'ㅣ' are readable as '*cheon* (天$_{heaven}$)', '*ji* (地$_{earth}$)', and '*in* (人$_{human}$)', respectively.

Decomposition Level	Subword Sequence
word	친구
character	친, 구
jamo (Park et al., 2018)	ㅊ, ㅣ, ㄴ, ㄱ, ㅜ
stroke (ours)	ㅅ, -, -, ㅣ, ㄴ, ㄱ, ㅜ
cji (ours)	ㅊ, ㅣ, ㄴ, ㄱ, ㅡ, ·
BTS (ours)	ㅅ, -, -, ㅣ, ㄴ, ㄱ, ㅡ, ·

Table 1: Examples of subword decomposition for the word '친구$_{\text{friend}}$' sorted by level of decomposition units

We consider both consonant-only **BTS** decomposition (denoted as **stroke**-level) and vowel-only **BTS** decomposition (denoted as **cji**-level, short for ***cheonjiin***). Examples of the proposed decomposition methods are presented in Table 1. Appendix B shows the details of the decomposition of 51 *jamo* letters into **BTS** units.

3.3 BTS-Level N-grams Extraction

We can extract n-grams from **BTS**-level decomposed Korean words at five levels: **BTS**-level, **stroke**-level, **cji**-level, *jamo*-level, and character-level. Subword n-grams from these levels are to be integrated for word representations. Unlike the *jamo*-level decomposition presented by Park et al. (2018), where each character is ensured to consist of three *jamo* letters, there is no such fixed length for characters in **BTS**-level decomposition. For example, characters '가' and '카' are decomposed into tri-gram 'ㄱ ㅣ·' and four-gram 'ㄱ- ㅣ·', respectively. For clear distinction, we insert the character separator ' / ' at the end of each character. We then add '<' and '>' to indicate the start and the end of each word. Consider the following **BTS**-level decomposed sequence of '친구$_{\text{friend}}$' as an example:

{<, ㅅ, - , - , ㅣ, ㄴ, /, ㄱ, ㅡ, · , /, >}.

BTS-level n-grams. We extract **BTS**-level subword n-grams from the decomposed sequence. **BTS** tri-grams for '친구$_{\text{friend}}$' are extracted as follows:

{<, ㅅ, - }, { ㅅ, - , - }, {- , - , ㅣ},
{- , ㅣ, ㄴ }, { ㅣ, ㄴ, /}, { ㄴ, /, ㄱ },
{/, ㄱ, ㅡ}, { ㄱ, ㅡ, · }, {ㅡ, · , /}, {· , /, >}.

***Jamo*-level n-grams.** To treat all *jamo* letters independently, we can extract *jamo* n-grams by re-composing the decomposed sequence. For example, *jamo*-level tri-grams for '친구$_{\text{friend}}$' are re-composed as follows:

{<, ㅅ, - , - , ㅣ}, { ㅅ, - , - , ㅣ, ㄴ },
{ ㅣ, ㄴ, /}, { ㄴ, /, ㄱ }, {/, ㄱ, ㅡ, · },
{ ㄱ, ㅡ, · , /}, {ㅡ, · , /, >}.

Character-level n-grams. We extract character-level n-grams from the decomposed sequence, by splitting the sequence by the separator ' / '. For example, the character-level uni-grams and bi-grams of '친구$_{\text{friend}}$' are extracted from the decomposed sequence as follows:

{ ㅅ, - , - , ㅣ, ㄴ, /}, { ㄱ, ㅡ, · , /},
{ ㅅ, - , - , ㅣ, ㄴ, /, ㄱ, ㅡ, · /}.

3.4 BTS-Based Subword Representations

We begin this section with a brief explanation of the word-level Skip-gram model (Mikolov et al., 2013a). The goal of the model is to learn the word vectors corresponding to each word. Suppose we are given a sequence of words $\{w_1, w_2, w_3, ..., w_T\}$ from the training corpus, where T is the length of the sequence. The model is trained to predict words $\{w_{t+j}\}_{-c\le j\le c, j\ne 0}$ within a context, based on the target word w_t. Here, c denotes the size of the context window. Formally, the objective of the model is to maximize the following log-likelihood expression:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\log p(w_{t+j}|w_t). \qquad (1)$$

While $p(w_{t+j}|w_t)$ can be defined as a softmax function, the computation of precise softmax outputs is infeasible. Therefore, we adopt negative sampling (Mikolov et al., 2013b), approximating the log probability to the following binary logistic loss function:

$$\log\left(1+e^{-s(w_t,w_{t+j})}\right)+\sum_{n\in\mathcal{N}_{t,j}}\log\left(1+e^{s(w_t,n)}\right), \qquad (2)$$

where $\mathcal{N}_{t,j}$ is the set of all negative examples, and s is a scoring function that assigns a real number to each word pair. In the Skip-gram model, each word w is represented by a unique vector. Thus, the value $s(w_t, w_{t+j})$ of function s is computed as the dot product between the input vector corresponding to w_t and the output vector corresponding to w_{t+j}.

In the subword information Skip-gram model (Bojanowski et al., 2017), we compute the vector representation of a word as an average of the vector representations of its n-grams. Therefore, the scoring function s is defined as:

$$s(w_t, w_{t+j}) = \frac{1}{|G_t|} \sum_{g_t \in G_t} \mathbf{z}_{g_t}^\top \mathbf{v}_{t+j}, \qquad (3)$$

where G_t denotes the set of n-grams extracted from the word w_t. $\mathbf{z}_{g_t}$ is the vector representation corresponding to each n-gram $g_t \in G_t$. $\mathbf{v}_{t+j}$ is the output vector corresponding to the word w_{t+j}.

We extract character-level, *jamo*-level, and one of stroke-, cji-, or **BTS**-level n-grams from a Korean word as described in Section 3.3. The word representation is then computed as the average of vectors corresponding to each n-gram. As different levels of n-grams are utilized based on the model setting, the scoring function s is defined as follows:

$$s(w_t, w_{t+j}) = \frac{1}{|U_t|} \sum_{g_t \in U_t} \mathbf{z}_{g_t}^\top \mathbf{v}_{t+j}. \qquad (4)$$

Here, U_t denotes the union set of n-grams extracted from diverse levels of subwords, based on the model settings. If we train a SISG model with character- and **BTS**-level n-grams, an embedding of a word is calculated by averaging the vectors of the whole word, character-level n-grams, and **BTS**-level n-grams.

4 Experiments

In this section, we demonstrate the effectiveness of **BTS** units through the evaluation of intrinsic and extrinsic tasks. See Appendix C for details on datasets in these experiments.

4.1 Corpus

We collect a corpus of Korean documents from three sources to enable the model to learn rich representations in diverse domains. The sources are: 1) 2020 newspaper corpus provided by the MODU corpus[5], 2) Korean Wikipedia [6], and 3) 21st century Sejong corpus (Kim et al., 2007). The training corpus is constructed by randomly mixing sentences from all three sources, and the resulting corpus contains 19M sentences with 1.13M unique vocabulary words.

4.2 Experimental Settings

Models. We experiment with our proposed stroke-level, cji-level, and **BTS**-level Subword Information Skip-Gram models (i.e., SISG(stroke), SISG(cji), SISG(**BTS**), respectively) and compare their performances to those of the baseline models. Additionally, we experiment with models trained on multiple levels of subword n-grams. Specifically, these models are trained on integration of one of *jamo*- or character-level subword n-grams, and one of stroke-, cji-, or **BTS**-level n-grams. For instance, SISG(jm+**BTS**) is a model trained on integrated subword n-grams of *jamo*- and **BTS**-level.

Baseline. We use the word-level Skip-Gram model (SG) (Mikolov et al., 2013a,b), character- and *jamo*-level Subword Information Skip-Gram models (SISG(ch) and SISG(jm), respectively) (Bojanowski et al., 2017; Park et al., 2018) as our baselines. We include models trained on both character- and *jamo*-level subword n-grams to baseline models, i.e., SISG(ch4+jm) and SISG(ch6+jm) from Park et al. (2018). Due to the difference in the training corpus, we re-implement and evaluate the baselines based on the code released by Park et al. (2018). Note that we have fixed some bugs (e.g., composing Korean characters and extracting n-grams) in the released code.

N-gram range. For our baseline models, we follow character- and *jamo*-level n-grams settings of Park et al. (2018). For our proposed models, we search the best-performing n-gram range settings from 2 or 3 through 20 for each task. We provide specific results for each hyperparameter setting in Appendix D.

Hyperparameter settings. We train all of our models and baselines on 300-dimensional embeddings, 5 epochs, a window size of 5, a learning rate of 0.025, and 5 negative samples. We filter out words that appear less than 10 times in the corpus and apply subsampling (Mikolov et al., 2013b) with a threshold of $t = 10^{-4}$. Additionally, a hash function from Bojanowski et al. (2017) is used to map all n-grams to integers ranging from 1 to 10^7.

4.3 Evaluation Tasks

We evaluate SISG(**BTS**), its variants, and baselines on two intrinsic tasks of word analogy and word similarity, and one extrinsic task of sentiment analysis. We report the average scores of five runs with different random seeds in these experiments.

[5] https://corpus.korean.go.kr/

[6] https://dumps.wikimedia.org/kowiki/

Model	Semantic					Syntactic					Sem.	Syn.	Avg.
	Capt	Gend	Name	Lang	Misc	Case	Tense	Voice	Form	Honr	Avg.	Avg.	
SG	0.463	0.531	0.585	0.435	0.644	0.533	0.612	0.543	0.677	0.538	0.532	0.581	0.556
SISG(ch)	0.417	0.460	0.554	0.374	0.561	0.234	0.472	0.456	0.545	0.357	0.473	0.413	0.443
SISG(jm)	0.413	0.430	0.510	0.346	0.557	0.164	0.351	0.364	0.415	0.297	0.451	0.318	0.385
SISG(ch4+jm)	0.402	0.432	0.506	0.337	0.556	0.152	0.346	0.361	0.404	0.294	0.447	0.311	0.379
SISG(ch6+jm)	0.404	0.430	0.502	0.337	0.556	0.151	0.345	0.364	0.400	0.295	0.446	0.311	0.378
SISG(stroke)	0.347	0.368	0.448	0.309	0.481	0.154	0.324	0.352	0.380	**0.260**	0.391	0.294	0.342
SISG(cji)	0.347	0.368	0.447	0.312	0.485	0.156	0.321	0.355	0.374	0.268	0.392	0.295	0.343
SISG(**BTS**)	**0.342**	**0.360**	**0.440**	**0.306**	**0.473**	**0.151**	**0.319**	**0.348**	**0.370**	0.267	**0.384**	**0.291**	**0.338**
SISG(jm+stroke)	0.343	0.360	0.446	**0.303**	0.476	0.151	0.316	0.351	**0.363**	0.256	0.386	0.287	**0.337**
SISG(jm+cji)	0.350	0.383	0.444	0.312	0.497	0.152	0.319	0.349	0.387	0.265	0.397	0.294	0.346
SISG(jm+**BTS**)	**0.339**	0.362	**0.443**	0.305	**0.475**	0.153	0.327	0.355	0.369	0.264	**0.385**	0.294	0.339
SISG(ch4+stroke)	0.346	**0.358**	0.451	**0.303**	0.477	0.153	0.319	0.352	0.372	0.260	0.387	0.291	0.339
SISG(ch4+cji)	0.352	0.382	0.445	0.316	0.502	0.153	0.322	0.347	0.389	0.266	0.399	0.296	0.347
SISG(ch4+**BTS**)	0.346	0.389	0.444	0.311	0.499	0.159	0.338	0.358	0.407	0.269	0.398	0.306	0.352
SISG(ch6+stroke)	0.348	0.381	0.447	0.309	0.492	0.153	0.328	0.352	0.391	0.266	0.395	0.298	0.347
SISG(ch6+cji)	0.349	0.376	0.452	0.312	0.486	0.158	0.328	0.362	0.384	0.273	0.395	0.301	0.348
SISG(ch6+**BTS**)	0.348	0.372	0.447	0.307	0.490	**0.146**	**0.314**	**0.337**	0.378	**0.253**	0.393	**0.286**	0.339

Table 2: Word analogy evaluation results. The upper half of the table shows the results for the baseline models and models trained using a single decomposition, whereas the lower half shows the results for models trained using multiple decompositions.

Word Analogy

We evaluate the semantic and syntactic features of word vectors on the Korean word analogy dataset (Park et al., 2018), in which questions are in the form of A is to B as C is to D, where D is the target word to predict. Consistent with Park et al. (2018), we employ the 3COSADD method and measure the distance between prediction and target vectors. We report the results for each category, the average of semantic and syntactic categories, and the overall average. We empirically observe that overfitting occurs in syntactic feature evaluation results. These results exhibit different trends compared to the other experimental results. Therefore, we report word analogy results for the model that performs best on semantic tasks, i.e., the model with the lowest average distance on 5 semantic tasks.

Word Similarity

We evaluate the trained word vectors on how well they formulate the relationship between words on the Korean version of the WS-353 dataset (Finkelstein et al., 2002). Each word similarity question comprises a pair of words and human-annotated similarity labels ranging from 0 to 10 (e.g., 호랑이$_{\text{tiger}}$, 고양이$_{\text{cat}}$ $= 7.17$). We report Spearman's correlation coefficient between human-annotated labels and the cosine similarities of word representations. A higher correlation coefficient indicates higher consistency between word similarity and human judgments.

Model	Similarity
SG	0.591
SISG(ch)	0.665
SISG(jm)	0.675
SISG(ch4+jm)	0.687
SISG(ch6+jm)	0.684
SISG(stroke)	0.703
SISG(cji)	**0.707**
SISG(**BTS**)	**0.707**
SISG(jm+stroke)	0.708
SISG(jm+cji)	0.703
SISG(jm+**BTS**)	0.707
SISG(ch4+stroke)	0.704
SISG(ch4+cji)	0.700
SISG(ch4+**BTS**)	0.704
SISG(ch6+stroke)	0.703
SISG(ch6+cji)	**0.714**
SISG(ch6+**BTS**)	0.709

Table 3: Word similarity evaluation results.

Sentiment Analysis

To highlight the efficacy of our trained word vectors on extrinsic tasks, we evaluate them on Naver Sentiment Movie Corpus (NSMC) [7]. We report the accuracy, precision, recall, and F1 score values of trained models. We train a single-layer LSTM model with 300 hidden dimensions and 0.5 dropout rates topped with a sigmoid activation function on the final LSTM state. We train 10 epochs of the dataset and use 0.5 dropout probability, 32 batch size, a cross-entropy error, an Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$, and 0.001 learning rate. The embedding layer contains representations for words in the train set and is frozen during training.

[7] https://github.com/e9t/nsmc

Model	Acc.	Prc.	Rec.	F1
SG	78.07	0.818	0.738	0.776
SISG(ch)	81.03	0.876	0.732	0.797
SISG(jm)	81.83	0.865	0.762	0.810
SISG(ch4+jm)	81.67	**0.879**	0.740	0.804
SISG(ch6+jm)	81.66	0.861	0.763	0.809
SISG(stroke)	82.44	0.878	0.758	0.814
SISG(cji)	**82.50**	0.862	0.781	**0.820**
SISG(**BTS**)	82.18	0.843	**0.798**	**0.820**
SISG(jm+stroke)	82.46	0.867	0.773	0.817
SISG(jm+cji)	82.23	**0.870**	0.764	0.813
SISG(jm+**BTS**)	82.21	0.861	0.775	0.815
SISG(ch4+stroke)	82.64	0.869	0.773	0.818
SISG(ch4+cji)	82.35	0.866	0.772	0.816
SISG(ch4+**BTS**)	82.33	0.854	0.786	0.818
SISG(ch6+stroke)	82.86	0.865	0.785	0.823
SISG(ch6+cji)	82.42	0.855	0.789	0.821
SISG(ch6+**BTS**)	**83.80**	0.857	**0.819**	**0.838**

Table 4: NSMC evaluation results. Acc., Prc., and Rec. refer to accuracy, precision, and recall, respectively.

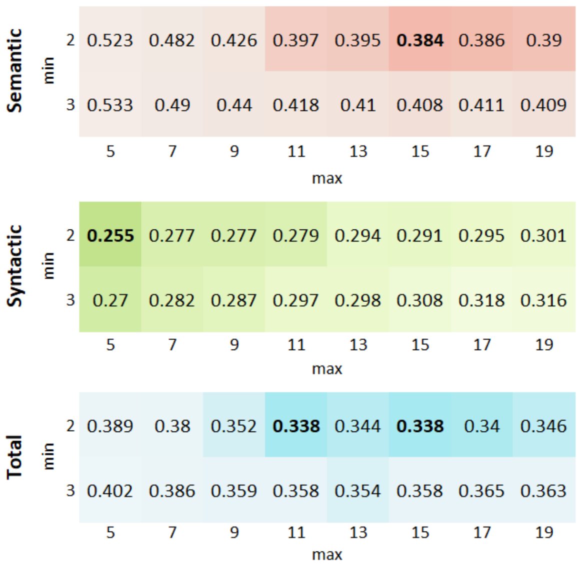


Figure 2: Semantic, syntactic and overall average performance of SISG(**BTS**) models on word analogy task by subword n-gram ranges. The darker, the better.

4.4 Results

Word analogy

The results for word analogy are shown in Table 2. The upper half of Table 2 shows that SISG(**BTS**) outperforms all baselines in whole categories. This demonstrates the significance of **BTS** units in capturing information from Korean words. While SISG(stroke) and SISG(cji) models also outperform all baselines, SISG(**BTS**) yields the greatest performance improvements in most categories. This reveals that both **BTS**-level consonants and **BTS**-level vowels advance performance and complement each other.

Further experiments are conducted to examine the power of **BTS** units when combined with other-level n-grams. As shown in the lower half of Table 2, we observe that the performance of all our combined models is worse than or comparable to that of SISG(**BTS**). This indicates that **BTS** units effectively capture the syntactic and semantic information for word analogy, and cover the information captured by character- and *jamo*-level n-grams.

For the baselines, we observe that our experimental results are not necessarily consistent with those of prior work (Park et al., 2018). For example, contrast to the prior work, we observe that SISG(ch) outperforms SG on semantic categories. We suspect that this is due to the different training corpus and bugs in the original code.

Word similarity

The results for word similarity are shown in Table 3. Consistent with the word analogy results, SISG(stroke), SISG(cji), and SISG(**BTS**) show performance improvements over the baselines. Specifically, SISG(cji) and SISG(**BTS**) outperform the previous state-of-the-art Korean embedding (SISG(ch4+jm)) by 3%. In addition, we observe that SISG(ch6+cji) achieves the best result, which shows that adding a wide range of character n-grams provides an advantage in capturing the information about word similarity.

Sentiment analysis

The results of sentiment analysis are reported in Table 4. SISG(**BTS**) consistently outperforms the baseline models, and adding character-level subwords improves the accuracy and F1 scores. These results demonstrate the effectiveness of **BTS**-level decomposition for the extrinsic task in Korean.

5 Analysis

5.1 N-gram Range Analysis

We analyze how the performance of SISG(**BTS**) depends on the subword n-gram range on word analogy task.

First, the overall average performance of SISG(**BTS**) on word analogy is maximized when we use n-gram ranges of 2-11 and 2-15. These are longer n-gram ranges compared with 3-6 of SISG(jm), which are explainable in that decomposing into **BTS** units generates longer sequences than those of SISG(jm).

Query	SG		SISG(ch)		SISG(jm)		SISG(**BTS**)	
	NN words	Equiv	NN words	Equiv	NN words	Equiv	NN words	Equiv
케잌	버터크림	✗	프랄린	✗	케익	✓	케익	✓
	버터크림을	✗	태슬	✗	케일	✗	케익을	✓
	카나페	✗	피넛	✗	케인	✗	케이크$_{\text{cake}}$	✓
찐한	사랑법이	✗	틀막	✗	찐득한	✗	찐득한	✗
	애절함	✗	3MC의	✗	찐찌	✗	진한$_{\text{strong}}$	✓
	애틋하면서도	✗	잔망	✗	찐	✗	찐	✗
웬지	괜스레	✗	어쩐지	✗	어쩐지	✗	웬일	✗
	허전하고	✗	웬일인지	✗	왠지$_{\text{for some reason}}$	✓	웬일이지	✗
	초조하고	✗	웬일이지	✗	웬일인지	✗	웬일이니	✗
뒤치닥거리	*n/a*	*n/a*	푸닥거리	✗	푸닥거리	✗	뒤치다꺼리$_{\text{cover for}}$	✓
	n/a	*n/a*	복닥거리는	✗	푸닥거리를	✗	뒤치다꺼리를	✓
	n/a	*n/a*	노닥거리는	✗	노닥거리고	✗	푸닥거리	✗

Table 5: Top-3 nearest neighbor words of given queries for each model. Queries are words containing common typos. ✓ in Equiv denotes that the query word and neighbor word are semantically equivalent, annotated by humans[8]. In the case of Korean standard words that have the intended meaning of query words, we put the meaning in English in subscripts (e.g., 케이크$_{\text{cake}}$, 진한$_{\text{strong}}$, etc.). *n/a* indicates that the trained vector for the query word does not exist, i.e., '뒤치닥거리' is an out-of-vocabulary word.

In addition, for SISG(**BTS**), word vectors including **BTS** bi-grams generally perform better than those excluding **BTS** bi-grams. According to Figure 2, **BTS** bi-grams improve performance on all evaluation criteria. As every extended consonant and vowel letter can be decomposed into at least two **BTS** units, we speculate that including **BTS** bi-grams provides better information for each *jamo* letter in word analogy.

Another observation is that the semantic and syntactic analogy performances of models show an opposite trend. As we include longer **BTS** n-grams in word vectors, the semantic analogy performance tends to increase, while the syntactic analogy performance tends to decrease. This is clear from Figure 2, which shows that the best performing n-gram ranges for semantic and syntactic analogy are 2-15 and 2-5, respectively.

5.2 Nearest Neighbor Words Analysis

While we have verified that SISG(**BTS**) effectively enriches word vectors with syntactic and semantic information, we further observe interesting results that show unique advantages provided by **BTS** units. We conduct qualitative analysis on the top-3 nearest neighbors for Korean words containing common typos. We report the results in Table 5 with human judgment on whether query words and nearest neighbor words are semantically equivalent.

As seen in a few examples in Table 5, **BTS** units show a strong capability to handle typos and out-of-vocabulary (OOV) words. We observe that SISG(**BTS**) shows better capability in identifying the meanings of words containing typos, compared to the baselines.

First, given '케잌', of which the intended meaning is 'cake', SISG(**BTS**) correctly finds the standard word '케이크$_{\text{cake}}$' with other words, such as '케익' semantically equivalent but containing different typos and '케익을' formed by appending objective case '을' to '케익'. In contrast, SG and SISG(ch) models find words related to cake such as '버터크림$_{\text{butter cream}}$' or '프랄린$_{\text{praline}}$', but fail to find the semantically equivalent words.

In addition, Koreans occasionally make intentional typos to emphasize the meaning, as can be seen in '찐한' coined by replacing 'ㅈ' with 'ㅉ'. SISG(**BTS**) finds the standard word '진한$_{\text{strong}}$' in this case.

We observe cases where SISG(**BTS**) tends to over-prioritize overlapping letters like other SISG models do. For example, all nearest neighbor words of '웬지' in SISG(**BTS**) include the character '웬', while the target word '왠지$_{\text{for some reason}}$' does not contain it. Mitigating such a problem would be the future work for word embeddings with subword information.

Finally, in the case of OOV words, given an OOV word '뒤치닥거리', SISG(**BTS**) effectively finds the standard word '뒤치다꺼리$_{\text{cover for}}$' with intended meanings, while other models fail to do so. This result suggests a promising direction for further exploration of **BTS**. We leave this for future works.

[8] Human annotations are given by a majority vote from 3 Korean native graduate students, excluding the authors.

6 Conclusion

Until now, among the Korean NLP community, various efforts have been made to reflect the unique writing system of Korean to NLP systems, but they have overlooked how *Hangeul* was created. In this paper, we have developed an unprecedented approach to Korean NLP inspired by the invention principle of *Hangeul*, leading to novel **BTS** units for analyzing the Korean linguistic structure. Instead of relying on the conventional approach of using Korean consonant and vowel letters, we have formed letters from **BTS** units by adding strokes or combining them together. We have shown that our **BTS** units effectively improve the quality of Korean word embeddings, which is a fundamental and versatile component of NLP models. We have demonstrated the efficacy of **BTS** units for both intrinsic and extrinsic tasks by outperforming existing Korean word embeddings methods on whole tasks in our evaluation. We hope that our **BTS** will *light the way up like dynamite*[9] for future research on Korean NLP.

Limitations

While we have shown that our **BTS** units facilitate performance improvements for Korean NLP tasks, there are some limitations that present avenues for future research. First, the efficacy of **BTS** units on Transformer-based Korean language models remains unexplored. We believe that the two-dimensional linguistic structure of Korean necessitates the investigation or even re-designing of a vanilla Transformer architecture, which takes an input sequence in a one-dimensional and monolayer form, as well as tokenization units. Future work on both topics would be of ultimate significance for Korean NLP, triggering new directions in various areas including Korean natural language understanding (Park et al., 2021), sub-character and morpheme-aware tokenization (Mielke et al., 2021), and multilingual NLP.

Additionally, we expect that our **BTS** units help improve the quality of representations for addressing typos and OOV words, as observed in Table 5, although we have not fully examined the efficacy of **BTS** units for typos and OOV words. Future work in this direction is expected to be another promising avenue for Korean NLP.

[9] Excerpted from the lyrics of 'Dynamite' sung by BTS, a popular South Korean boy band.

Acknowledgement

We thank the anonymous reviewers for their helpful comments. This work was supported by the Basic Research Program through the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2020R1A4A1018309), National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2021R1A2C3010430) and Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00079, Artificial Intelligence Graduate School Program (Korea University)).

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Shaosheng Cao, Wei Lu, Jun Zhou, and Xiaolong Li. 2018. cw2vec: Learning chinese word embeddings with stroke n-gram information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 1, pages 5053–5061.

Hong-You Chen, Sz-Han Yu, and Shou-de Lin. 2020. Glyph2vec: Learning chinese out-of-vocabulary word embedding from glyphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2865–2871.

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

Lev Finkelstein, Evgeniy Gabrilovich, Y. Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20:116–131.

Marzena Karpinska, Bofang Li, Anna Rogers, and Aleksandr Drozd. 2018. Subcharacter information in Japanese embeddings: When is it worth it? In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*.

Hung-gyu Kim, Beom-mo Kang, and Jungha Hong. 2007. 21st century sejong corpora (to be) completed. *The Korean Language in America*, 12:31–42.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 2741–2749.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Yongeun Lee. 2006. *Sub-syllabic constituency in Korean and English*. Ph.D. thesis, Northwestern University.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *CoRR*, abs/2112.10508.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: Annual Conference on Neural Information Processing Systems 2013*, pages 3111—-3119.

National Hangeul Museum. 2018. A guide to hunminjeongeum. `https://www.hangeul.go.kr/fileDownload.do?saved_filename=BBS/EAEB416F-DD45-E105-137B-36DD215662EE.pdf&filename=AGuideToHunminjeongeum_Ver2.4_20181010.pdf`.

Sungjoon Park, Jeongmin Byun, Sion Baek, Yongseok Cho, and Alice Oh. 2018. Subword-level word vector representations for Korean. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2429–2438.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won Ik Cho, Jiyoon Han, Jangwon Park, Chisung Song, Junseong Kim, Yongsook Song, Tae Hwan Oh, Joohong Lee, Juhyun Oh, Sungwon Lyu, Young kuk Jeong, Inkwon Lee, Sang gyu Seo, Dongjun Lee, Hyunwoo Kim, Myeonghwa Lee, Seongbo Jang, Seungwon Do, Sunkyoung Kim, Kyungtae Lim, Jongwon Lee, Kyumin Park, Jamin Shin, Seonghyun Kim, Lucy Park, Alice H. Oh, Jung-Woo Ha, and Kyunghyun Cho. 2021. Klue: Korean language understanding evaluation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*.

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.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (Volume 2: Short Papers)*, pages 529–535.

Shota Sasaki, Jun Suzuki, and Kentaro Inui. 2019. Subword-based Compact Reconstruction of Word Embeddings. 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 3498–3508.

Jae Jung Song. 2006. *The Korean language: Structure, use and context*. Routledge.

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)*, pages 721–726.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced chinese character embedding. In *International Conference on Neural Information Processing*, pages 279–286.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1504–1515.

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 (EMNLP)*, pages 286–291.

Jinman Zhao, Shawn Zhong, Xiaomin Zhang, and Yingyu Liang. 2020. Pbos: Probabilistic bag-of-subwords for generalizing word embedding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 596–611.

A Invention principle of *Hangeul*

We explain the invention principle of *Hangeul*, the Korean alphabet, based on the literature distributed by National Hangeul Museum (2018).

A.1 Historical Backgrounds

Until the Middle Ages, there was no official writing system for Korean, so it was written in Classical Chinese. To remedy the lack of a writing system and designated alphabets for writing Korean, King Sejong the Great, the 4th monarch of the Joseon dynasty (1397-1450), invented *Hangeul*. The invention was finished in 1443, and *Hunminjeongeum*, the book with detailed explanations of *Hangeul*, was published in 1446. Unlike other writing systems whose invention principle and period are unknown, those of *Hangeul* are revealed by *Hunminjeongeum*. In addition, *Hunminjeongeum* refers to the original name of *Hangeul*.

A.2 Korean Writing System

In Korean, consonants and vowels that make up a syllable are combined and written as syllable blocks, whereas in English, letters make up words through the sequential combination of characters. In this study, we consider syllable blocks as being equivalent to characters.

Each Korean character is made up of at least one consonant and vowel. When we construct a syllable block, it generally follows the order of C_1V C_2 (Lee, 2006), where C_1, C_2 and V are short for consonant and vowel, respectively. In the C_1V C_2 structure, each component refers to special terms: 1) C_1 to the initial sound, 'chosung', 2) V to the middle sound, 'joongsung', and 3) C_2 to the final sound, 'jongsung'. Joongsung is placed at the right of or below chosung, and jongsung (if used) is placed below chosung and joongsung. Note that jongsung is not mandatory to form a Korean character. See Figure A.3.

A.3 Consonants and Vowels of *Hangeul*

In Korean, we refer to consonants as '*jaeum*', and vowels as '*moeum*'. Consonants and vowels, grouped together, are called *jamo*s. We describe how *Hangeul jamo*s are formed based on the invention principle of *Hangeul*.

Consonant extension via stroke addition. Consonants in modern Korean consist of five basic consonants (ㄱ, ㄴ, ㅁ, ㅅ, ㅇ) and nine extended consonants (ㅋ, ㄷ, ㅌ, ㄹ, ㅂ, ㅍ, ㅈ, ㅊ, ㅎ). The basic consonants were formed by imitating the shape of vocal organs (See Figure A.5(a)), and the extended consonants were formed by adding strokes to the basic consonants. For example, 'ㅋ' is a single stroke extension of 'ㄱ' that expresses the aspirated sound of 'ㄱ'. Figure 1 (b) depicts how other consonants were extended from basic consonants. In addition, double consonants are formed by combining the same consonants (e.g., ㅈ + ㅈ→ ㅉ), and consonant clusters are created by combining different consonants (e.g., ㄴ+ ㅎ → ㄶ). Double consonants and consonant clusters are used for constructing characters.

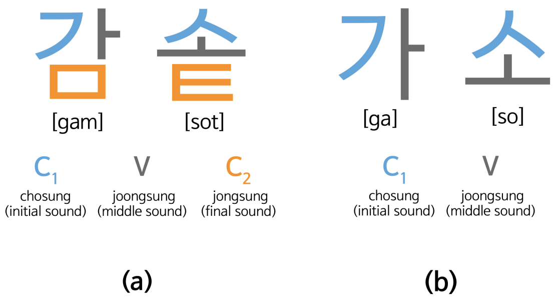


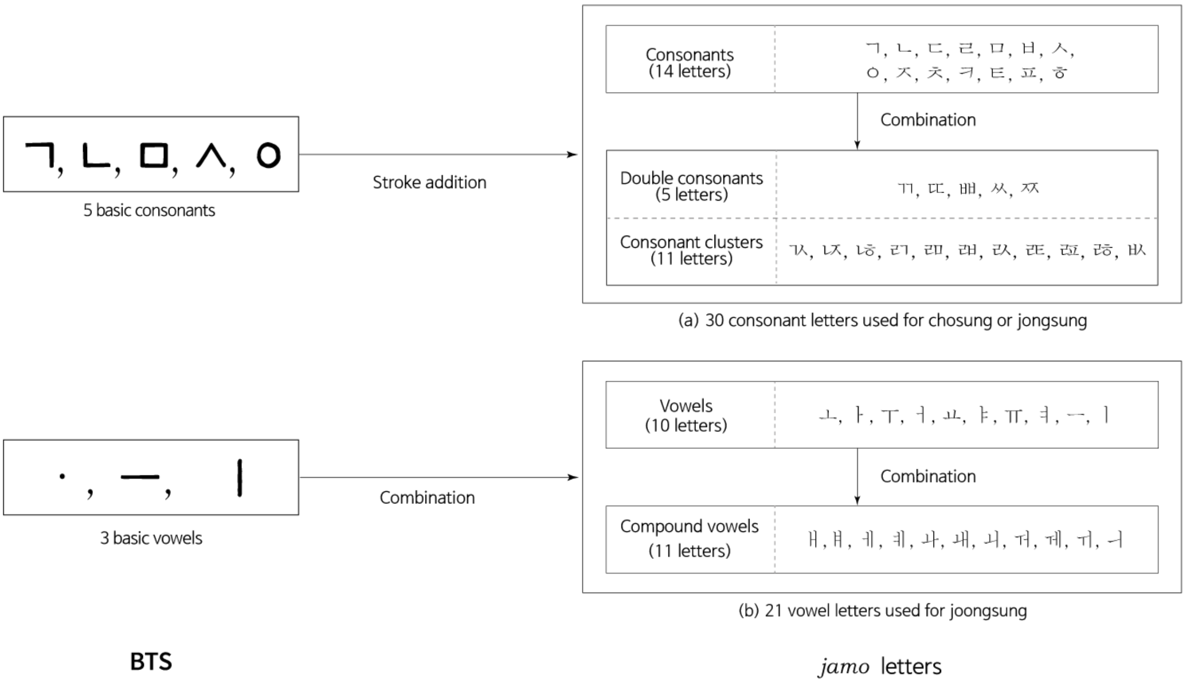
Figure A.3: Structures of Korean characters. C_1, C_2 is short for consonants and V is short for vowels. C_1, V, and C_2 refer to chosung, joongsung, and jongsung, respectively. A Korean character generally follows the C_1VC_2 structure as shown in (a), but jongsung is not mandatory for forming a Korean character, as shown in (b).

Vowel creation via combination. *Hangeul* vowels are formed through the combination of three basic vowels (ㆍ, ㅡ, ㅣ). Each of the basic vowels represents the shape of the round, heaven 'ㆍ[ʌ]', flat earth 'ㅡ[ɯ]', and standing human 'ㅣ[i]'[10], and are pronounced as 'cheon(天$_{\text{heaven}}$)', 'ji(地$_{\text{earth}}$)', and 'in(人$_{\text{human}}$)', respectively in Sino-Korean pronunciation. See Figure A.5.

More vowels are created by combining the basic vowels. 'ㆍ' is placed either above or below the letter 'ㅡ' and on the either left or right side of 'ㅣ', resulting in four vowels 'ㅗ, ㅏ, ㅜ, ㅓ'. The further addition of 'ㆍ' creates four more vowels 'ㅛ, ㅑ, ㅠ, ㅕ' (National Hangeul Museum, 2018). For example, 'ㅑ' is created by combining 'ㅏ' and 'ㆍ', while 'ㅏ' consists of 'ㅣ' and 'ㆍ'. Therefore, 'ㅑ' can be broken down into 'ㅣㆍㆍ'.

Additionally, compound vowels can be formed

[10] We follow International Phonetic Alphabet (IPA) notation for the pronunciation of Korean words. Refer to https://www.internationalphoneticassociation.org/.



(a) 30 consonant letters used for chosung or jongsung

(b) 21 vowel letters used for joongsung

Figure A.4: 51 letters used to construct Korean characters. *jamo* letters consist of (a) 30 consonant letters and (b) 21 vowel letters. **BTS** units represent all *jamo* letters by adding strokes or combining them.

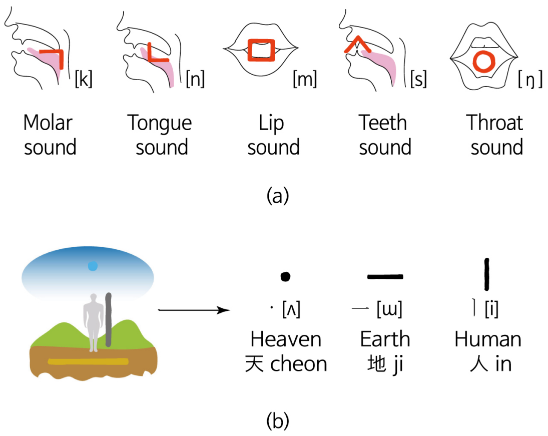


Figure A.5: Invention principle of basic letters. (a) Basic consonants were designed to resemble the shape of the vocal organs. (b) The basic letters depict the shape of a round heaven, flat earth, and a standing human.

by combining multiple vowels. For example, a compound vowel 'ㅘ[wa]' is formed by combining 'ㅗ' and 'ㅏ' where 'ㅗ' is further decomposed into '·' and 'ㅡ', and 'ㅏ' is further broken down into 'ㅣ' and '·'. For this reason, we can decompose 'ㅘ' into '· ㅡ ㅣ ·'.

Consonant and vowel letters. There are 51 *jamo* letters used in Korean characters, including 30 consonant letters and 21 vowel letters. 30 consonant letters consist of 5 basic consonants, 9 extended consonants (via stroke addition), 5 double consonants, and 11 consonant clusters. Double consonants and consonant clusters result from further combinations of basic/extended consonants. All of the basic/extended consonants and double consonants can be used for chosung, resulting in 19 possible letters. For jongsung, a total of 27 consonants can be used, including all basic/extended consonants, two double consonants ('ㄲ' and 'ㅆ'), and all consonant clusters. 21 vowel letters consist of 10 vowels (created via basic vowel combination) and 11 compound vowels, all of which can be used as joongsung. Specific letters are presented in Figure A.4.

B Full Comparison of *jamo*-level and **BTS**-level Decomposition

We provide a full comparison of how each Korean *jamo* letter is decomposed by *jamo*- and **BTS**-level subword.

consonant letters		vowel letters	
jamo-level	**BTS**-level	*jamo*-level	**BTS**-level
ㄱ	ㄱ	ㅏ	ㅣ·
ㄴ	ㄴ	ㅑ	ㅣ··
ㄷ	ㄴ-	ㅓ	· ㅣ
ㄹ	ㄹ	ㅕ	·· ㅣ
ㅁ	ㅁ	ㅗ	· ㅡ
ㅂ	ㅁ-	ㅛ	·· ㅡ
ㅅ	ㅅ	ㅜ	ㅡ·
ㅇ	ㅇ	ㅠ	ㅡ··
ㅈ	ㅅ-	ㅡ	ㅡ
ㅊ	ㅅ--	ㅣ	ㅣ
ㅋ	ㄱ-		
ㅌ	ㄴ--		
ㅍ	ㅁ--		
ㅎ	ㅇ-		
ㄲ	ㄱ ㄱ	ㅐ	ㅣ· ㅣ
ㄸ	ㄷ ㄷ	ㅒ	ㅣ·· ㅣ
ㅃ	ㅁ-ㅁ-	ㅔ	· ㅣ ㅣ
ㅆ	ㅅ ㅅ	ㅖ	·· ㅣ ㅣ
ㅉ	ㅅ-ㅅ-	ㅘ	· ㅡ ㅣ·
ㄳ	ㄱ ㅅ	ㅙ	· ㅡ ㅣ· ㅣ
ㄵ	ㄴ ㅅ-	ㅚ	· ㅡ ㅣ
ㄶ	ㄴ ㅇ-	ㅝ	ㅡ·· ㅣ
ㄺ	ㄹ ㄱ	ㅞ	ㅡ·· ㅣ ㅣ
ㄻ	ㄹ ㅁ	ㅟ	ㅡ· ㅣ
ㄼ	ㄹ ㅁ-	ㅢ	ㅡ ㅣ
ㄽ	ㄹ ㅅ		
ㄾ	ㄹ ㄴ--		
ㄿ	ㄹ ㅁ--		
ㅀ	ㄹ ㅇ-		
ㅄ	ㅁ-ㅅ		

Table B.1: Full comparison between *jamo*-level and **BTS**-level decomposition of *jamo* letters. Above the line are 14 consonants and 10 vowels, and below the line are double consonants, consonant clusters, and compound vowels.

C Dataset Details

C.1 Corpus

We provide detailed information on three sources comprising the training corpus.

2020 Newspaper Corpus. The 2020 newspaper corpus contains approximately 0.7M articles extracted from 35 media sources, covering eight domains (political, economic, social, life, IT/science, entertainment, sports culture, and beauty/health). The corpus contains approximately 133M words and 9M sentences.

Korean Wikipedia. The Korean Wikipedia corpus covers 0.4M documents, containing 69M words and 7M sentences.

21st Century Sejong Corpus. The 21st century Sejong corpus was developed from 1998 to 2007 under the national research project "21st Century Sejong Project". It is composed of written modern Korean corpus, spoken-transcript modern Korean corpus, and parallel corpus (e.g., Korean-English, Korean-Japanese) collected from newspapers, magazines, books, and monologue/dialogue scripts. We use a filtered version of the corpus that contains 31M words and 3M sentences.

C.2 Word Analogy

The Korean word analogy evaluation dataset (Park et al., 2018) consists of 10 categories, five for semantic feature evaluation and five for syntactic feature evaluation. Each category contains 1K items, and each item consists of four words that evaluate the semantic and syntactic features of learned word representations. We provide details regarding the word analogy dataset released by Park et al. (2018) by presenting an example for each category.

Semantic :

- Capital-Country (Capt): Pairs of a country name and its capital city

  베를린$_{\text{Berlin}}$: 독일$_{\text{Germany}}$ = 서울$_{\text{Seoul}}$: 대한민국$_{\text{Korea}}$

- Male-Female (Gend): Pairs of corresponding male-female words

  고모부$_{\text{uncle}}$: 고모$_{\text{aunt}}$ = 남편$_{\text{husband}}$: 아내$_{\text{wife}}$

- Name-Nationality (Name): Pairs of a celebrity name and nationality

  고흐$_{\text{Gogh}}$: 네덜란드$_{\text{Netherlands}}$ = 링컨$_{\text{Lincoln}}$: 미국$_{\text{USA}}$

- Country-Language (Lang): Pairs of a country name and official language :

  브라질$_{\text{Brazil}}$: 포르투갈어$_{\text{Portuguese language}}$ = 일본$_{\text{Japan}}$: 일본어$_{\text{Japanese language}}$

- Miscellaneous (Misc): Various semantic features not included in the above four categories

  개$_{\text{dog}}$: 강아지$_{\text{puppy}}$ = 닭$_{\text{chicken}}$: 병아리$_{\text{chick}}$

Syntactic :

- Case: Pairs of a noun and the noun with case marker attached

  여정$_{\text{journey}}$: 여정을$_{\text{journey+case(을)}}$ = 마술사$_{\text{magician}}$: 마술사를$_{\text{magician+case(를)}}$

- Tense: Pairs of the present tense and past tense of a verb

  돕다$_{\text{help}}$: 도왔다$_{\text{helped}}$ = 사용하다$_{\text{use}}$: 사용했다$_{\text{used}}$

- Voice: Pairs of active voice and passive voice of a verb

  거절했다$_{\text{rejected}}$: 거절당했다$_{\text{was/were rejected}}$ = 먹었다$_{\text{ate}}$: 먹혔다$_{\text{was/were eaten}}$

- Verb ending form (Verb): Pairs of a verb and the verb with the ending form attached

  사다$_{\text{buy}}$: 사면서$_{\text{buy+form(면서)}}$ = 잡다$_{\text{grab}}$: 잡으면서$_{\text{grab+form(으면서)}}$

- Honorific (Honr): Pairs of morphological variation for verbs in Korean.

  갔다$_{\text{went}}$: 가셨다$_{\text{went+honorific}}$ = 이끌고$_{\text{lead}}$: 이끄시고$_{\text{lead+honorific}}$

C.3 Word Similarity

The Korean version of the WS-353 dataset (Finkelstein et al., 2002) was released by Park et al. (2018). It consists of 353 questions annotated by 14 Korean native speakers, asking about the similarity between pairs of words.

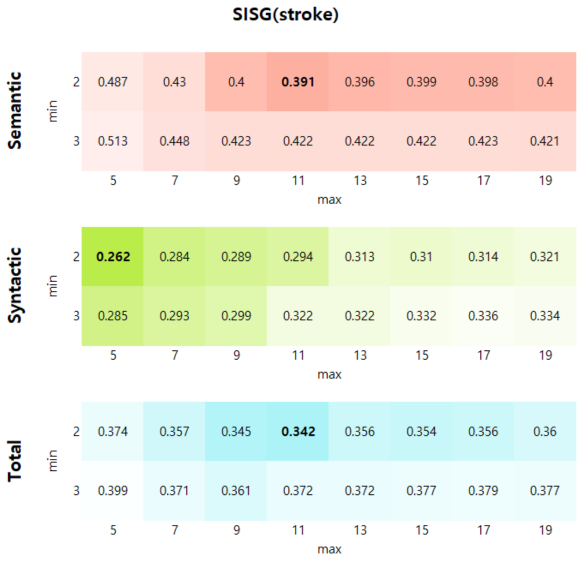
C.4 Sentiment Analysis

Naver Sentiment Movie Corpus (NSMC) dataset consists of 150K training and 50K test samples, scraped from the Naver Movies website. Each sample contains a movie review sequence in Korean and a binary score (0 for negative, 1 for positive). We sample 100K train, 25K validation, and 25K test samples from the dataset, balancing the ratio of labels among each set. Since the original source of the dataset is inevitably noisy, we preprocess the dataset so that only Korean, English, and numbers remain in the dataset.

D Word Analogy Results by N-gram Range

We report the experimental results of word analogy by n-gram range.

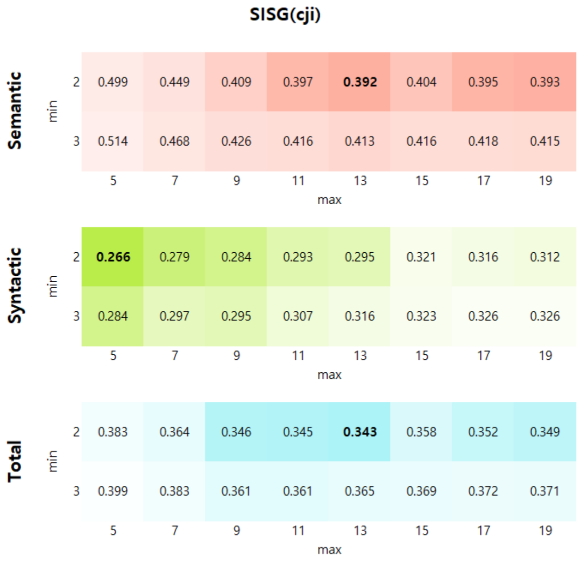
D.1 Single Decomposition Models



SISG(stroke)

	min	5	7	9	11	13	15	17	19
Semantic	2	0.487	0.43	0.4	**0.391**	0.396	0.399	0.398	0.4
	3	0.513	0.448	0.423	0.422	0.422	0.422	0.423	0.421
Syntactic	2	**0.262**	0.284	0.289	0.294	0.313	0.31	0.314	0.321
	3	0.285	0.293	0.299	0.322	0.322	0.332	0.336	0.334
Total	2	0.374	0.357	0.345	**0.342**	0.356	0.354	0.356	0.36
	3	0.399	0.371	0.361	0.372	0.372	0.377	0.379	0.377

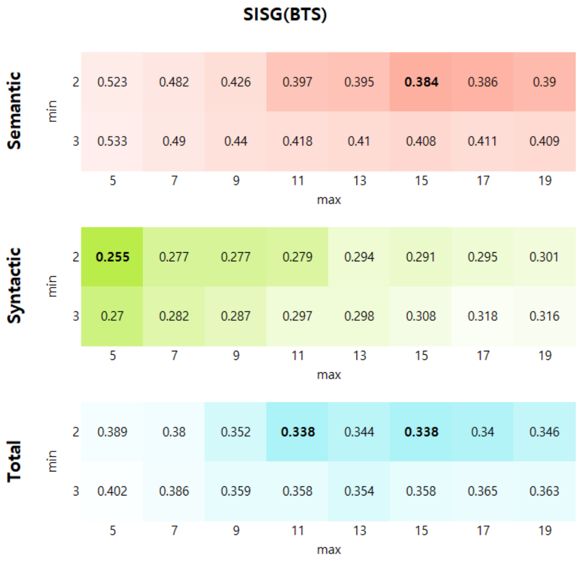
Figure D.1: SISG(stroke)



SISG(cji)

	min	5	7	9	11	13	15	17	19
Semantic	2	0.499	0.449	0.409	0.397	**0.392**	0.404	0.395	0.393
	3	0.514	0.468	0.426	0.416	0.413	0.416	0.418	0.415
Syntactic	2	**0.266**	0.279	0.284	0.293	0.295	0.321	0.316	0.312
	3	0.284	0.297	0.295	0.307	0.316	0.323	0.326	0.326
Total	2	0.383	0.364	0.346	0.345	**0.343**	0.358	0.352	0.349
	3	0.399	0.383	0.361	0.361	0.365	0.369	0.372	0.371

Figure D.2: SISG(cji)



SISG(BTS)

	min	5	7	9	11	13	15	17	19
Semantic	2	0.523	0.482	0.426	0.397	0.395	**0.384**	0.386	0.39
	3	0.533	0.49	0.44	0.418	0.41	0.408	0.411	0.409
Syntactic	2	**0.255**	0.277	0.277	0.279	0.294	0.291	0.295	0.301
	3	0.27	0.282	0.287	0.297	0.298	0.308	0.318	0.316
Total	2	0.389	0.38	0.352	**0.338**	0.344	**0.338**	0.34	0.346
	3	0.402	0.386	0.359	0.358	0.354	0.358	0.365	0.363

Figure D.3: SISG(**BTS**)

D.2 *Jamo*-Integrated Models

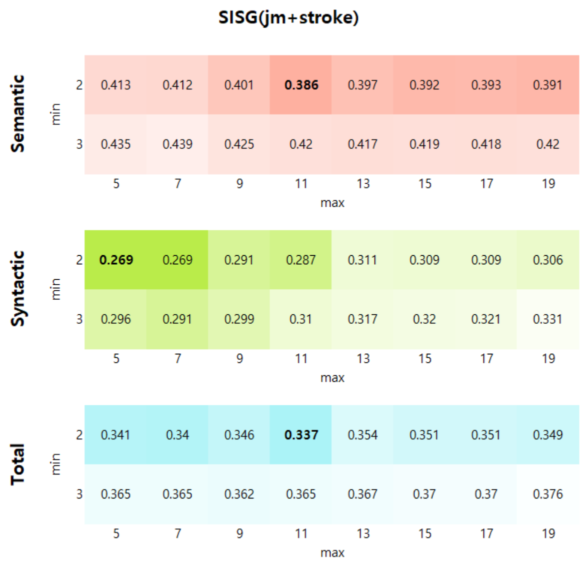


Figure D.4: SISG(jm+stroke)

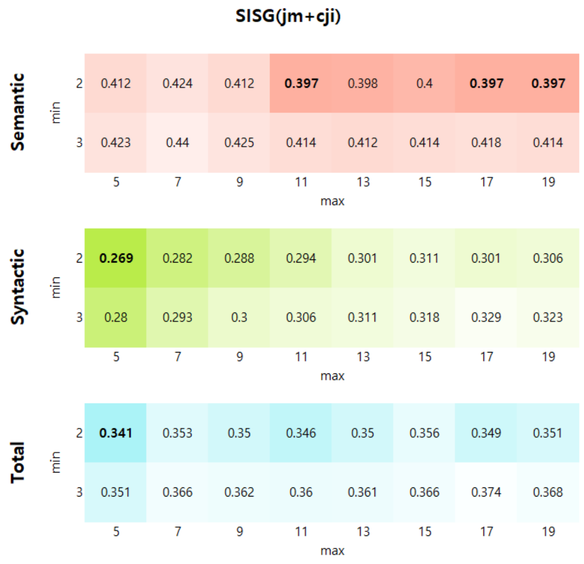


Figure D.5: SISG(jm+cji)

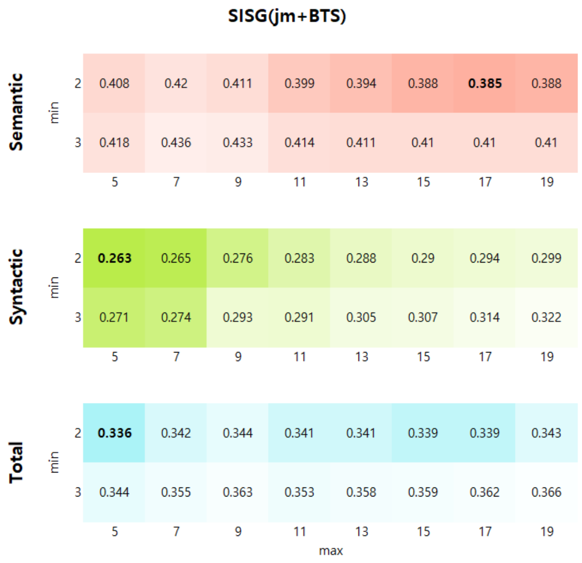


Figure D.6: SISG(jm+**BTS**)

D.3 Character(1-4) - Integrated Models

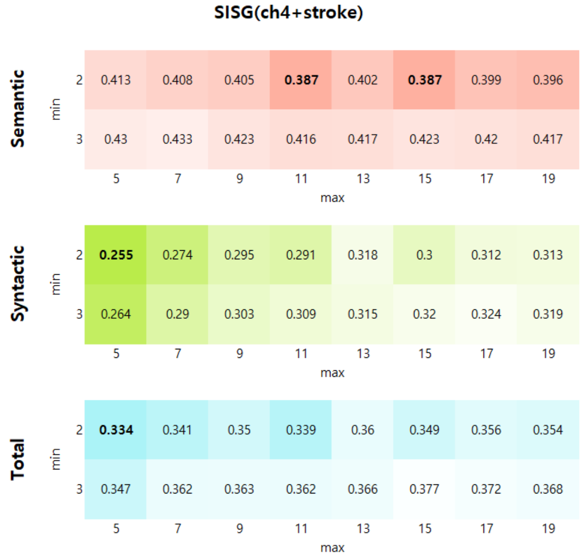


Figure D.7: SISG(ch4+stroke)

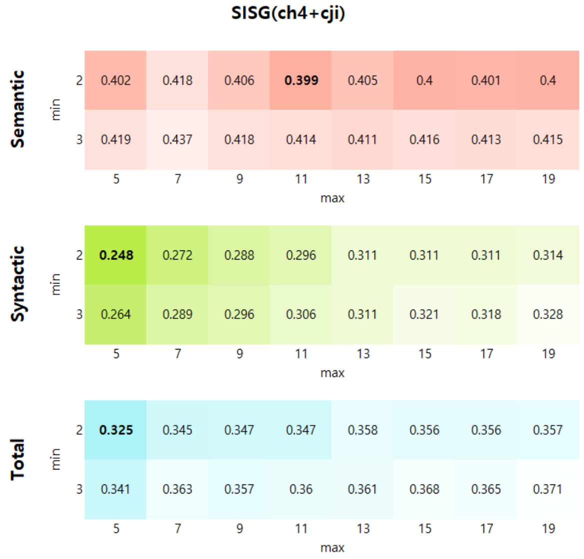


Figure D.8: SISG(ch4+cji)

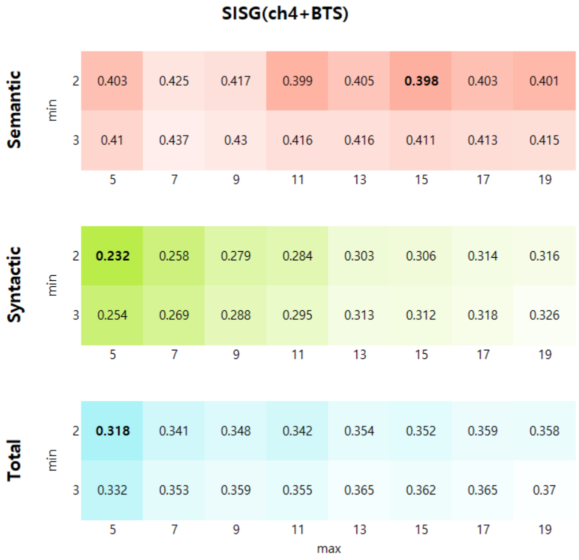


Figure D.9: SISG(ch4+**BTS**)

D.4 Character(1-6) - Integrated Models

For SISG(ch6+stroke), the best performing model reported in Table 2 was experimented on n-gram range of 2-10, so the results may vary with Figure D.10.

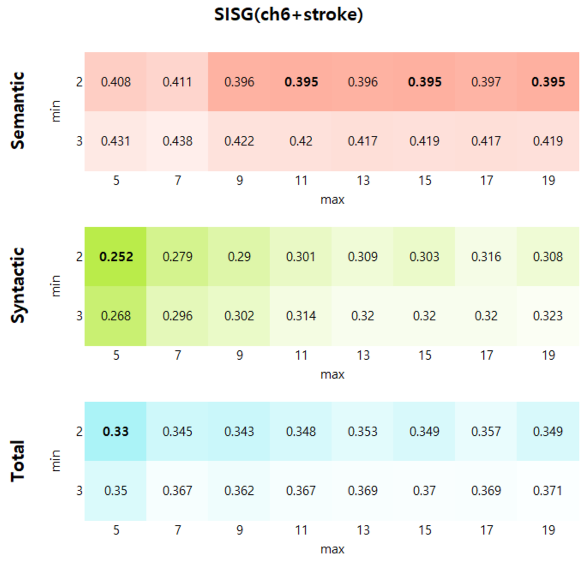


Figure D.10: SISG(ch6+stroke)

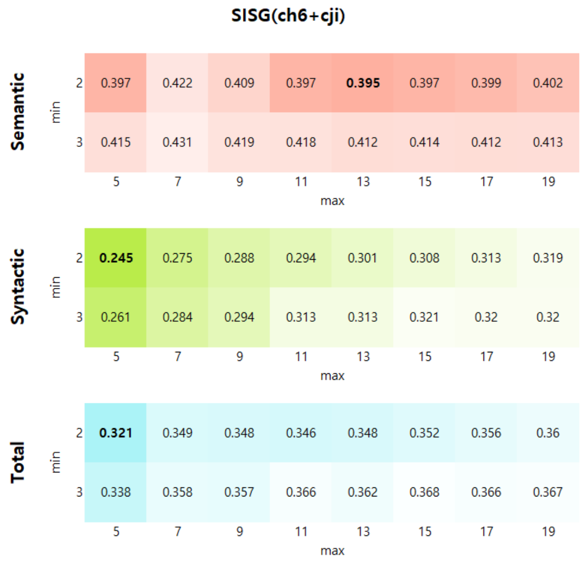


Figure D.11: SISG(ch6+cji)

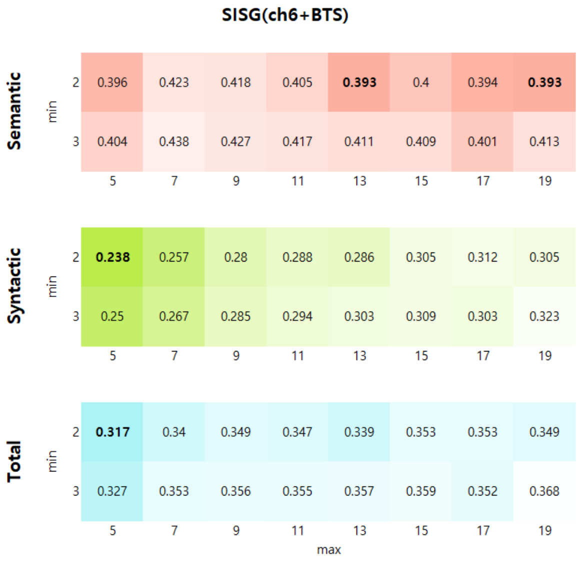


Figure D.12: SISG(ch6+BTS)