Post-hoc Vocabulary Expansion for Embedding Models via Definition-Learning

Anonymous ACL submission

Abstract

Embedding models have become a core component of modern Natural Language Processing (NLP). However, most widely used embedding models adopt tokenizers that rely on frequency- or probability-based segmentation, which overlooks the morphological structure of agglutinative languages such as Korean. Moreover, embedding model vocabulary is fixed after training and cannot be extended to incorporate new domain-specific words. To address these challenges, we propose an embedding framework that integrates morpheme-aware tokenization with definition-based vocabulary expansion and is readily extensible to other languages and backbone models. Also, we introduce a SLERP-based training objective to improve embedding alignment during dimensionality reduction. After training, the framework enables vocabulary expansion by generating token embeddings from definition sentences without additional fine-tuning. Experimental results show that our approach improves downstream task performance by $0.1710 \sim 0.2182$ in terms of Micro-F1 and Micro-Recall@K.

1 Introduction

In recent years, embedding models (Devlin et al., 2019; Liu et al., 2019) have provided a foundation for learning sentence semantics. However, most embedding models have been designed and trained primarily for English and exhibit degraded performance when applied to non-English languages (Wu and Dredze, 2020), especially agglutinative languages such as Korean (Sohn, 2001). Unlike English (Comrie, 1989), which expresses grammatical meaning mostly through word order and auxiliary verbs, Korean forms words by combining stems with various particles and endings to express grammatical and semantic information. This structure makes tokenization challenging, as modern tokenizers, such as WordPiece (WP) (Wu et al., 2016) and SentencePiece (SP) (Kudo and Richardson, 2018), rely on frequency and probabilistic criteria rather than morpheme boundaries. As a result, Korean grammatical structures are distorted and meaningful units are fragmented (Jeon et al., 2023).

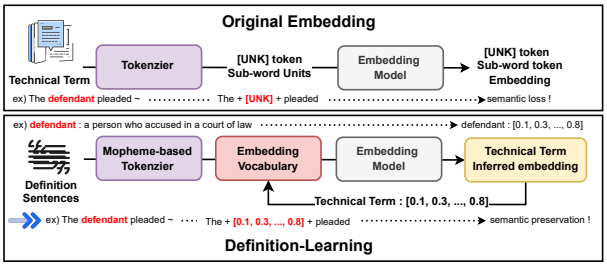

Figure 1: Comparison between the original embedding model, which treats rare words as unknown, and the proposed Definition-Learning framework, which infers their embeddings from definitions.

In addition, specialized domains contain many technical terms that are absent from general-domain corpora, making it difficult for the model to learn accurate representations during training. Existing tokenizers handle such words by replacing them with the [UNK] token or by segmenting them into subword units, which either leads to a complete loss of semantic information due to the out-of-vocabulary (OOV) problem or prevents the model from fully capturing their semantics. Most importantly, because the tokenizer rules and embedding model vocabulary are fixed after pretraining, incorporating new domain-specific words requires large-scale data collection and fine-tuning.

Existing study (Tang and Yang, 2024) has shown that embedding models trained on general-domain data exhibit performance degradation in specialized domains and argues for the necessity of more sophisticated models for such domains. To overcome this limitation, (Ruzzetti et al., 2022) has explored leveraging definition sentences, and its usefulness has been demonstrated. However, their approaches rely on fixed token selection and do not account for multiple definitions associated with a single token.

To address the limitations caused by improper tokenization and semantic loss of domain-specific terms, we propose a Definition-Learning embedding framework that employs the morpheme-based MeCab tokenizer (Kudo, 2005), as illustrated in Figure 1. With Definition-Learning, token embeddings are generated by dynamically attending to informative tokens across multiple definition sentences. This framework is motivated by the way humans learn new words: people first acquire general linguistic patterns through exposure to natural language (PINE, 2005). Once this foundation is established, they expand their lexicon by learning new words through definitions without relearning the language (Aitchison, 2012). Inspired by this process, our framework learns general semantic representations during pre-training and subsequently infers embeddings for unseen tokens from definitions. To enable more accurate alignment, we introduce a Spherical Linear Interpolation (SLERP)-based training objective during dimensionality reduction, which preserves geometric structure and aligns inferred token embeddings with the existing vocabulary embedding space. Once this framework is trained, new tokens can be added to the model's vocabulary without additional fine-tuning or modifying the original model architecture. This approach maintains model stability while providing a scalable and efficient solution for handling OOV words and domain-specific terms. The key contributions of this study are as follows:

- We propose a Definition-Learning framework integrated with a morpheme-based tokenizer to generate vocabulary token embeddings [1] for [UNK] tokens from definition sentences.
- We design a loss function based on an SLERP-based geodesic distance approximation to train the dimensionality reduction when aggregating definition sentences into token embedding.
- We improve embedding classification performance by $0.1710 \sim 0.2182$ in Micro-F1 and retrieval performance by $0.1843 \sim 0.2087$ in Micro-Recall@K compared to baseline model.

2 Related Work

2.1 Tokenizers and Embedding Models

Widely used tokenizers such as WP and SP rely on frequency- and probability-based segmentation, which fails to capture the morphological structure of agglutinative languages (Jeon et al., 2023) and degrades performance in embedding models (Lee et al., 2024). Moreover, these tokenizers perform tokenization based on learned rules, resulting in fixed vocabularies that reduce the domain adaptability of embedding models (Islam et al., 2022). To address these issues, prior works (Kudo, 2005; Kakao, 2019) have explored morpheme-based tokenizers for agglutinative languages.

Early word embedding research introduced neural distributed models such as Continuous Bag-of-Words (CBOW) and Skip-gram (Mikolov et al., 2013). With the Transformer architecture (Vaswani et al., 2017), BERT (Devlin et al., 2019), which employs Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), became the dominant paradigm. However, following studies showing that NSP provides limited benefits, Robustly Optimized BERT Approach (RoBERTa) (Liu et al., 2019), which adopts dynamic masking with MLM, has been widely used as the backbone of recent state-of-the-art (SOTA) embedding models, such as BAAI General Embedding (BGE) (Multi-Granularity, 2024) and E5 (Wang et al., 2022).

2.2 Embedding Models for Expert Domains

Embedding models perform well in general domains, but their performance drops when applied to expert domains. Tang and Yang (2024) demonstrated that even SOTA models exhibit performance degradation in specialized domains such as finance. This study showed that the degradation was due to the models' inability to capture domain-specific semantic relations and emphasized the need for sophisticated embedding methods. Most existing approaches address this issue through domain-specific fine-tuning in areas such as finance (Araci, 2019), law (Chalkidis et al., 2020), and biomedical (Lee et al., 2020). However, such approaches require additional data collection, substantial computational resources, and considerable training time.

Ruzzetti et al. (2022) proposes inferring vocabulary token embeddings from definitions by selecting core tokens and processing them with either a feed-forward (FF) layer or a BERT model. However, core tokens are selected through fixed structural analysis rather than dynamically based on the input definition, and the approach does not consider cases where a single token is associated with multiple definitions, such as polysemy or homonymy.

These limitations, including fixed tokenization

[1] In this paper, a vocabulary token embedding refers to the static embedding vector retrieved from the embedding lookup table when the token is used as input.

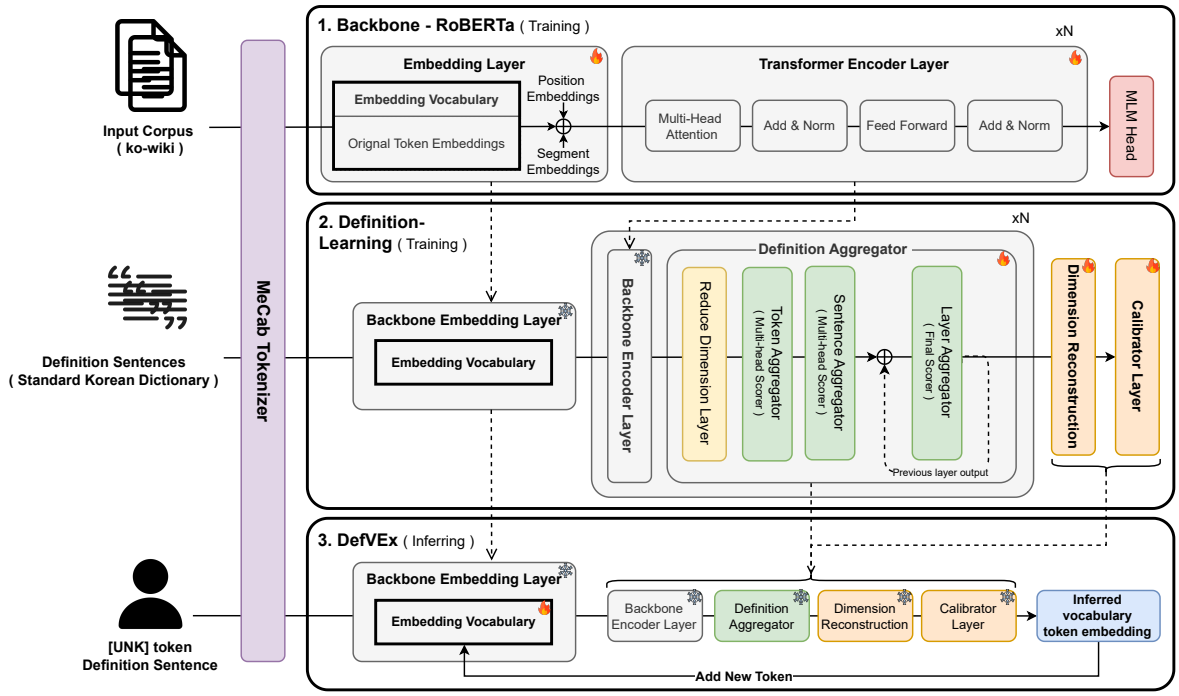

Figure 2: Overall architecture of the proposed Definition-Learning framework and DefVEx model. The model operates on MeCab and extends the original RoBERTa structure with Dimension Aggregator, Dimension Reconstruction, and Calibrator Layer for Definition-Learning. The backbone remain frozen during Definition-Learning training.

rules, limited domain adaptability, and static structural analysis of definition sentences, highlight the need for a more flexible framework that can dynamically leverage multiple definition sentences to generate vocabulary token embeddings.

3 Method

3.1 Model Overview

The architecture of the proposed **DefVEx : Definition-based Vocabulary Expansion** model, which integrates the MeCab tokenizer with Definition-Learning, is illustrated in Figure 2. To reflect morphological richness of Korean, input sentences are tokenized using the MeCab-ko[2], which adopts the Sejong Corpus[3]-based POS tag system for fine-grained grammatical categorization. Unlike WP or SP, MeCab is a dictionary-based tokenizer with a user-extensible vocabulary, which determines segmentation based on the surface form, POS tag, and priority stored in its dictionary. Therefore, users can add new words or merge multiple morphemes by updating MeCab's dictionary, with the changes immediately reflected in tokenization without any retraining. This design enables flexible vocabulary extension and provides a suitable foundation for Definition-Learning.

DefVEx separates general language understanding from new-word inference: (1) a RoBERTa-base (Liu et al., 2019) backbone provides general language understanding, and (2) additional modules are trained on definitions to generate vocabulary token embeddings for unseen words. During Definition-Learning, the backbone is frozen to preserve general language understanding. After training stages are completed, users can input arbitrary definition sentences into the DefVEx model and add new vocabulary token embeddings to the embedding vocabulary without additional fine-tuning. In our setting, the input consists of definition sentences for a target word, so it is shaped as $[N, S, H]$, where N is the number of definitions, S is the sentence length, and H is the hidden size. Definition-Learning consists of three stages: (1) the Definition Aggregator for dimensional compression and aggregation, (2) the Dimension Reconstruction to restore dimension, and (3) the Calibrator Layer to align with the vocabulary embedding space.

3.2 Definition Aggregator

The Definition Aggregator integrates the backbone encoder layer outputs of the definition sentences into a single vector, as illustrated in Figure 3.

[2]MeCab-ko is a Korean-adapted version of MeCab.

[3]The Sejong Corpus is a large-scale Korean linguistic corpus developed by the National Institute of Korean Language.

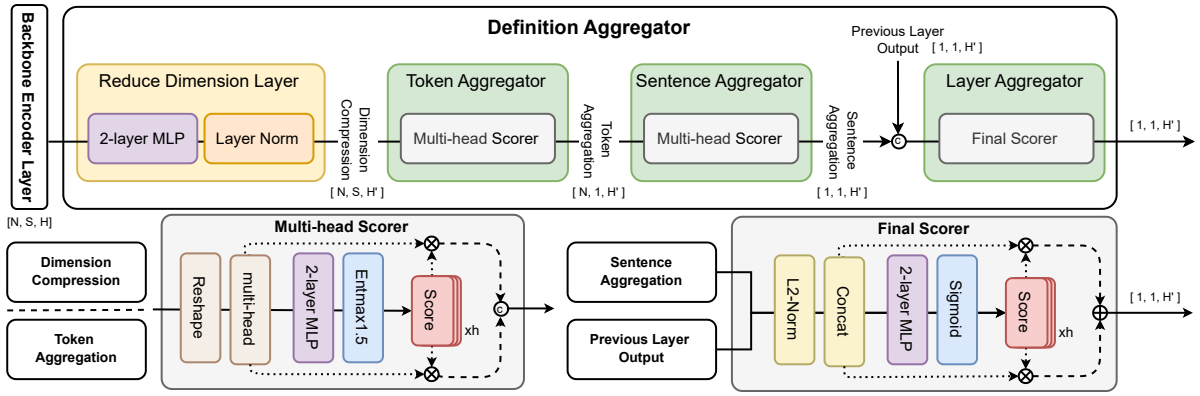

Figure 3: Architecture of Definition Aggregator. At each layer, the backbone output is processed sequentially through the Reduce Dimension Layer and the Token, Sentence, and Layer Aggregators to generate the vocabulary token embedding. ⓒ, ⊗, and ⊕ denote concatenation, element-wise multiplication and element-wise addition.

Yin and Shen (2018) demonstrated that word embeddings lie on a lower-dimensional manifold, and high dimensionality captures semantically irrelevant variation. Therefore, dimensionality reduction helps isolate the core meaning of a definition sentence while suppressing noise. Hwang et al. (2023) further show that shallow nonlinear transformations are sufficient to preserve semantics while reducing dimensionality. Based on these findings, we design the Reduce Dimension Layer as a lightweight nonlinear multi-layer perceptron (MLP) layer that reduces the dimension to $H' = H/4$.

After dimensionality reduction, the representations are aggregated in three stages. The Token Aggregator and Sentence Aggregator take as input the outputs of the Reduce Dimension Layer and the Token Aggregator, respectively, and generate the Token Aggregation and Sentence Aggregation outputs. Each Aggregator adopts a Multi-head Scorer as its weighting mechanism, implemented as an MLP-based module, to identify important semantic components. Finally, the Layer Aggregator merges the current block output with accumulated representations from previous blocks by computing layer-level weights through an MLP-based Final Scorer to produce the final representation.

In the Token aggregator, each token representation already encodes contextual information from the backbone, so additional attention is redundant. Moreover, although attention mechanisms are effective at modeling dependencies among inputs, neither definition sentences for the same word nor layer-wise outputs exhibit meaningful semantic interactions that justify such mechanisms. Therefore, we adopt MLP-based scoring to independently evaluate each representation without pairwise interactions. We use Entmax1.5 instead of softmax, as it can assign zero weights to low-score components, which makes it effective for compression.

3.3 Dimension Reconstruction and Alignment

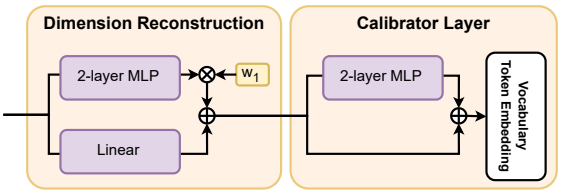

Figure 4: Residual MLP-based architecture for the Dimension Reconstruction and Calibrator Layer.

After semantic inference is accumulated across all layers, the resulting vocabulary token embedding is processed through the Dimension Reconstruction to restore its original dimensionality and then the Calibrator Layer to align with the vocabulary embedding space, as illustrated in Figure 4.

Dimension Reconstruction restores the representation using a **residual skip-projection**. This provides an advantage for preserving information, and promotes smoother overall training convergence.

Finally, we add the Calibrator Layer with a **residual MLP block** to remap the reconstructed representation into the vocabulary embedding space. Because the Calibrator Layer is designed for alignment, it uses the same-dimensional projections instead of the dimensionality-expanding MLP from standard Transformer feed-forward networks to prevent unnecessary transformation.

3.4 Training Objective

In the Reduce Dimension Layer, it is crucial to preserve the semantic relational structure in the

original space. Wang et al. (2021) show that retaining both angular and geometric information during low-dimensional projection leads to more coherent representations. In addition, Sala et al. (2018) report that Euclidean distance cannot properly capture positional relationships on a nonlinear manifold. Based on these observations, we design a loss function that jointly preserves cosine similarity and **an approximate geodesic distance**. Since computing the exact geodesic distance is expensive, we leverage the fact that the backbone encoder output is normalized by Layer Normalization (LN). LN maps vectors onto a hyper-ellipsoid, interpretable as a shifted hyper-sphere of radius $\sqrt{H}$. Given this property, we approximate the geodesic distance using a **SLERP-based segmented integral**. Derivations are given in Appendix A.

Using this method, we compute pairwise cosine similarities and distances for all tokens. To increase sensitivity to distance distortions among neighbor vectors, we apply the relative error to the distance term. The projection loss function is defined in Equations (1)–(3). The relative error between the exact values and their SLERP-based approximations remains small across the entire angular range. The error analysis is provided in Appendix B.

$$\mathcal{L}_{\text{projection}} = \mathcal{L}_{\text{deg}} + \mathcal{L}_{\text{dist}} \tag{1}$$

$$\mathcal{L}_{\text{deg}} = \|\cos(x_i, x_j) - \cos(z_i, z_j)\|_F^2 \tag{2}$$

$$\mathcal{L}_{\text{dist}} = \left\| \frac{\sqrt{\frac{H'}{H}}\,\text{dist}(x_i, x_j) - \text{dist}(z_i, z_j)}{\sqrt{\frac{H'}{H}}\,\text{dist}(x_i, x_j) + \epsilon} \right\|_F^2 \tag{3}$$

where α_1 and α_2 are weighting coefficients, x_i and x_j are original space embeddings, and z_i and z_j are corresponding embeddings in the reduced space.

We introduce an Embedding Alignment (EA) loss to jointly supervise aggregation and reconstruction-alignment. To better distinguish similar words beyond cosine loss, we include a contrastive loss (Oord et al., 2018). Moreover, because the vocabulary embedding space is not normalized, we add a magnitude alignment term to match the embedding vector magnitude. The EA loss function is described in Equations (4)–(5).

$$\mathcal{L}_{\text{norm}} = \left[\, \|\hat{e}\| - \|e\| - 0.05\,\|e\| \,\right]_+^2 \tag{4}$$

$$\mathcal{L}_{\text{EA}} = \lambda_{\text{norm}}\mathcal{L}_{\text{norm}} + \lambda_{\cos}\big(1 - \cos(\hat{e}, e)\big) + \lambda_{\text{nce}}\mathcal{L}_{\text{InfoNCE}} \tag{5}$$

where $\hat{e}$ and e denote the inferred and label vectors, and $[x]_+ = \max(x, 0)$ denotes the hinge operator.

4 Experiments

4.1 Dataset

We use three types of datasets. First, the backbone RoBERTa model is trained on the Korean Wikipedia (Ko-wiki) (lovit, 2021). Second, for Definition-Learning, we use definition sentences from the Standard Korean Dictionary[4]. Third, we evaluate the quality of vocabulary token embeddings using the legal-domain LCUBE dataset (Hwang et al., 2022), which includes classification (casename, casename+) and retrieval (statute, statute+) tasks. In addition, we also measure general semantic representation quality using the Korean Semantic Textual Similarity (KorSTS) (Ham et al., 2020) benchmark. Detailed information of all datasets are summarized in Appendix C.

4.2 Experiments Settings

For the backbone model, we follow the same architecture and training setup as the original RoBERTa-base. Before training, we filter tokens that appear at least 100 times in the Ko-wiki corpus when tokenized with MeCab, resulting in 90,729 tokens included in the vocabulary. Definition-Learning uses the same training settings as the backbone. More details are provided in Appendix D.

Ruzzetti et al. (2022) evaluated inferred token embeddings using cosine similarity to label embeddings. The reported values of $0.3 \sim 0.5$ are comparable to that of nearest-neighbor vocabulary token embeddings inferred by our Definition-Learning. As a result, cosine similarity alone makes it difficult to determine whether embeddings are accurately inferred from definition sentences. Therefore, we evaluate Definition-Learning by expanding the vocabulary with inferred OOV tokens and assessing downstream task performance. A detailed analysis of this evaluation is provided in Appendix E.

For downstream evaluation, sentence embeddings are generated from the last-layer hidden states $[S, H]$. If a sentence exceeds the maximum length, it is split into segments, whose $[S, H]$ representations are averaged into a single $[S, H]$. Finally, token representations are averaged to $[H]$ and L2-normalized to form the final sentence embedding.

We use three evaluation metrics: Spearman correlation for Semantic Textual Similarity (STS), Micro-F1 for classification, and Micro-Recall@K (K=5) for retrieval. Recall-based metrics are more suitable because retrieval queries have 2-4 labels.

[4] https://stdict.korean.go.kr/main/main.do.

Metric	Filter Ratio	KorSTS	Casename	Casename+	Statute	Statute+
Vocab Size	no-filter	91,330(+601)	97,252(+6,523)	100,489(+9,760)	91,867(+1,138)	92,599(+1,870)
	0.5	91,330(+601)	97,249(+6,520)	100,486(+9,757)	91,863(+1,134)	92,595(+1,866)
	0.1	91,330(+601)	97,219(+6,490)	100,454(+9,725)	91,856(+1,127)	92,581(+1,852)
Params	no-filter	156.76M(+1.76M)	174.11M(+19.11M)	183.59M(+28.59M)	158.33M(+3.33M)	160.48M(+5.48M)
	0.5	156.76M(+1.76M)	174.10M(+19.10M)	183.58M(+28.58M)	158.32M(+3.32M)	160.47M(+5.47M)
	0.1	156.76M(+1.76M)	174.01M(+19.01M)	183.49M(+28.49M)	158.30M(+3.30M)	160.43M(+5.43M)
Excluded [UNK]	–	393	35,938	77,662	16,306	4,039

Table 1: Vocabulary size, parameter count, and the number of excluded [UNK] tokens after filtering.

Model	Tokenizer	Filter Ratio	Params	Vocab	KorSTS	Casename	Casename+	Statue	Statue+
RoBERTa	WP	–	110M	32,000	**0.5193**	0.0749	0.2793	0.2783	0.0262
RoBERTa(Ours)	Mecab	–	110M	32,000	0.4807	0.2537	0.4274	0.4087	0.0913
RoBERTa(Ours)	Mecab	–	155M	90,729	0.4928	0.2778	0.4402	0.4807	0.1608
DefVEx(Ours)	Mecab	no-filter	155M+	90,729+	<u>0.4991</u>	0.2703	0.4467	0.4398	0.1277
DefVEx-0.5(Ours)	Mecab	0.5	155M+	90,729+	<u>0.4991</u>	<u>0.2873</u>	**0.4503**	<u>0.4845</u>	<u>0.2083</u>
DefVEx-0.1(Ours)	Mecab	0.1	155M+	90,729+	<u>0.4991</u>	**0.2931**	<u>0.4490</u>	**0.4870**	**0.2105**

Table 2: Embedding downstream task performance comparison. Top-2 results are highlighted in **bold** and <u>underline</u>.

4.3 Quantitative evaluation

To evaluate Definition-Learning, we first classify tokens in the test sentences. This procedure includes: (1) tokenizing with MeCab, (2) identifying [UNK] tokens, and (3) checking whether each [UNK] token has a definition in the dictionary. The full process is described in Appendix F.

Among the [UNK] tokens identified above, we handle only those that appear word-initially, such as **a** in $A = \mathbf{a} + \#\#b$. In addition, we evaluate filtering strategies based on token frequency. We compute token frequencies per test set and apply filter ratios of 0.5 and 0.1. Tokens exceeding each threshold are excluded from Definition-Learning. Table 1 summarizes the final data statistics.

We use a RoBERTa-base model with a WP tokenizer, fine-tuned on Korean, as the baseline and report downstream results in Table 2. Switching to MeCab tokenizer results in -5.10% on KorSTS, but improves all other tasks: +370.90% on Casename, +157.78% on Casename+, +172.73% on Statute, and +613.74% on Statute+. These improvements persist even with the baseline model's vocabulary size. The KorSTS performance drop is mainly due to the OOV frequency gap, with 19 [UNK] tokens for WP-based model versus 994 for the MeCab-based model. Because STS relies on the similarity score itself, it is more affected by semantic loss from [UNK] token substitutions than by differences in tokenization quality. Notably, after applying Definition-Learning, performance improves by +1.28% compared to the MeCab-based RoBERTa.

We also find that applying a filtering yields significantly better performance. Moreover, increasing the number of filtered tokens produces a consistent performance improvement across most tasks. The filtered model improves by +5.51% on Casename, +2.29% on Casename+, +1.31% on Statute, and +12.52% on Statute+ over the unfiltered model.

This suggests that for high-frequency tokens, it may be more effective to intentionally leave them as [UNK] token rather than assign explicit meanings. When such tokens are replaced with meaningful embeddings, they appear frequently in both inputs and labels, and this pulls sentence representations toward a shared semantic direction. This causes semantic flattening, making embeddings more similar to each other and ultimately reducing their discriminative power. In contrast, the outcome may differ when high-frequency tokens are left as [UNK] token. The [UNK] token is not an empty or meaningless vector. Instead, it serves as a fallback embedding that groups OOV terms together and learns from the contexts where rare or unseen words occur. Although it lacks explicit lexical meaning, [UNK] token functions as a compact abstraction that signals the presence of a specific but unidentified token. Thus, keeping some high-frequency tokens as [UNK] token helps maintain clearer distinctions between sentences. Our findings align with previous research (Li et al., 2020; Chen et al., 2025) showing that high-frequency or overly common tokens can hinder sentence embedding quality and downstream task performance.

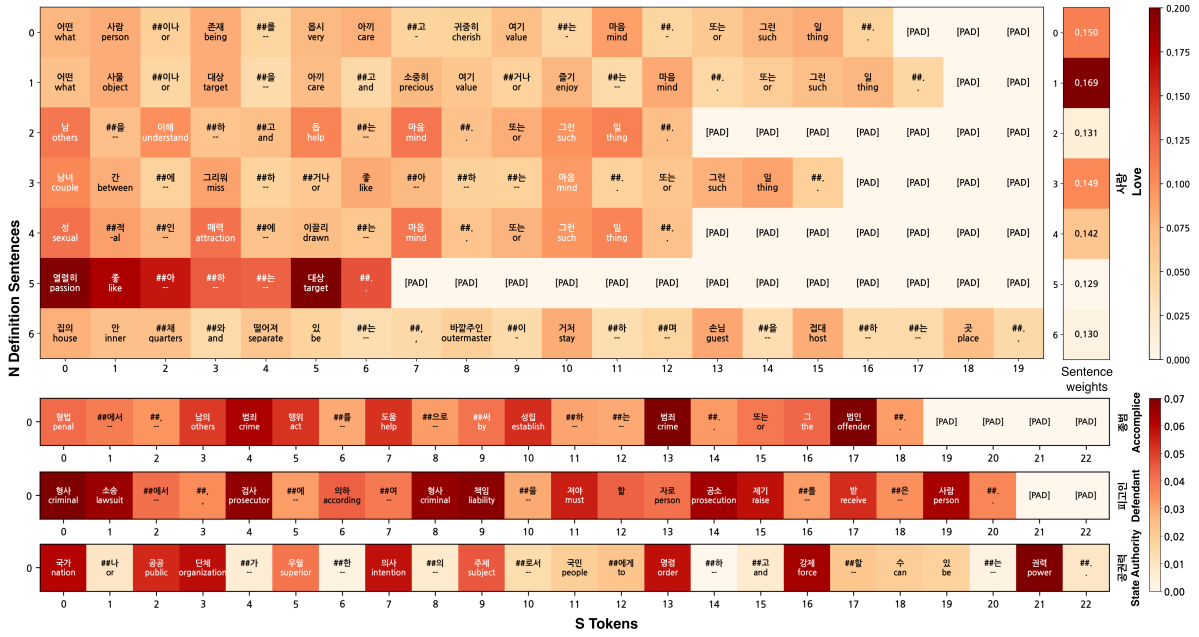

Figure 5: Token- and sentence-level scores from Multi-head Scorer, obtained by summing outputs across all 12 layers and normalizing the scores within each sentence and across the seven definition sentences for '사랑 (Love)'.

4.4 Qualitative Evaluation

Definition-Learning aims to generate vocabulary token embedding by assigning higher weights to semantically important tokens in definition sentences. To validate this architecture, we visualized the scores from the Multi-head Scorer. As shown in Figure 5, key semantic tokens receive higher scores, a pattern observed not only for general nouns such as 사랑 (Love) but also for domain-specific terms like 종범 (Accomplice), 피고인 (Defendant), and 공권력 (State Authority). In the sentence weights of 사랑 (Love), sentences with weaker semantic relevance, such as those describing "이해 (understand) and 돕 (help)" or presenting a homonym definition, are assigned lower weights.

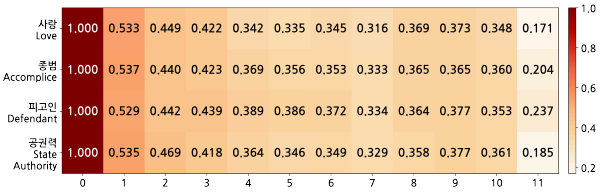

Figure 6: Layer-level scores from Final Scorer. The first score is 1 since there is no prior layer to aggregate from.

Figure 6 shows that the layer-level scores drop in the later layers for all sample words. Prior studies (Vulić et al., 2020; Tenney et al., 2019) have shown that lower layers retain more lexical and surface-form information, whereas upper layers gradually encode increasingly contextual semantics. This pattern aligns with our model's objective of recovering static vocabulary embedding rather than producing fully contextualized representations. Therefore, it is appropriate to emphasize the lower layers and reduce the contribution of the upper layers.

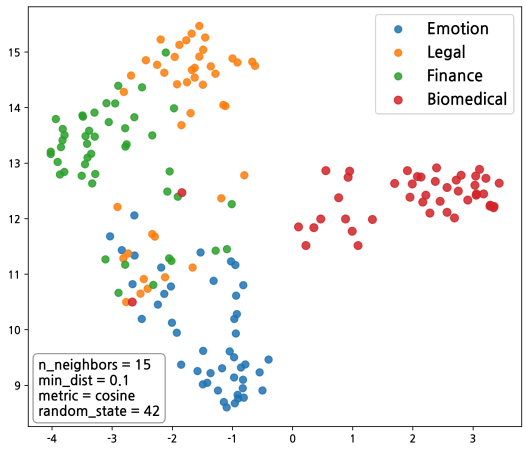

Figure 7: UMAP projection of vocabulary token embeddings using Definition-Learning across four domains.

Figure 7 presents a Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) visualization of new vocabulary token embeddings using Definition-Learning in two dimensions. The tokens are grouped into four domains, Emotion, Legal, Finance, and Biomedical, and each domain forms a distinct cluster. This clustering

pattern demonstrates that Definition-Learning preserves domain-level semantic structure even without explicit supervision.

4.5 Abliation Study

D/R	P/L	R/C	KorSTS	Casename	Casename+	Statute	Statute+
✓	✓	✓	0.4991	0.2931	0.4503	**0.4870**	**0.2105**
✓	✗	✓	**0.5011**	0.2906	0.4516	0.4410	0.1896
✗	✗	✗	0.5007	**0.2981**	**0.4542**	0.4497	0.2009

Table 3: D/R, P/L, and R/C indicate the Reduce Dimension Layer, Projection Loss, and Dimension Reconstruction, respectively. The reported score for each task corresponds to the best-performing filter ratio, no-filter for KorSTS, 0.5 for Casename+, and 0.1 for others.

In Table 3, removing $\mathcal{L}_{\text{projection}}$ slightly improves KorSTS and Casename+, but substantially degrades performance on all other tasks. This shows that the proposed $\mathcal{L}_{\text{projection}}$ effectively preserves the structural and geometric information required for low-dimensional projection. In addition, removing dimensionality reduction shows minor improvements in STS and classification, but clear degradation in retrieval tasks, with Statute scores even falling below the baseline. This indicates that removing dimension reduction weakens robustness, as unnecessary high-dimensional noise (Voita et al., 2019) interferes with downstream performance.

P/N	C/B	KorSTS	Casename	Casename+	Statute	Statute+
Ent	Same	0.4991	**0.2931**	0.4503	0.4870	0.2105
Soft	Same	0.4974	0.2890	0.4504	**0.4907**	**0.2126**
Ent	Expand	**0.4998**	0.2930	**0.4526**	0.4522	0.1864

Table 4: P/N, and C/B, indicate the Probability Normalization function, and Calibrator Layer, respectively. The same task-wise filter ratios as in Table 3 are used.

Table 4 shows no notable differences between Entmax1.5 and Softmax. This implies that performance mainly depends on selecting informative tokens in the Definition Aggregator, whereas P/N has only a minor effect. In addition, the Calibrator Layer with a Transformer-style FFN expansion: $H \rightarrow 4H \rightarrow H$ shows almost no change in STS and classification performance, but leads to clear degradation in retrieval tasks. This demonstrates that the Calibrator Layer is most effective when kept in the same dimension, as its role is to refine alignment with the original vocabulary embedding space rather than expand semantic capacity.

We analyze the performance impact of replacing the MLP-based Scorer with a Set Transformer (Lee et al., 2019)-style attention–based Scorer. The detailed architecture of the attention-based Scorer is provided in the Appendix G. In Table 5, the attention-based Scorer shows lower performance across all tasks compared to the MLP-based Scorer. Dot-attention computes weights based on correlations between input tokens, and the tokens used during aggregation have already learned these correlations via the backbone model's self-attention blocks. As a result, applying dot-attention again during aggregation in the Definition Aggregator introduces redundant computation rather than extracting additional useful information. In the sentence and layer aggregation stages, the objective is not to model input correlations but to assess each input's importance independently. Therefore, the MLP-based Scorer achieves better performance by treating each input as an independent feature.

Scorer	KorSTS	Casename	Casename+	Statute	Statute+
MLP	**0.4991**	**0.2931**	**0.4503**	**0.4870**	**0.2105**
Attention	0.4921	0.2911	0.4483	0.4484	0.1747

Table 5: Comparison of MLP and attention-based Scorer. The same task-wise filter ratios as in Table 3 are used.

5 Conclusion

In this paper, we propose a Definition-Learning embedding framework with the MeCab morphological tokenizer. Through extensive quantitative and qualitative analyses, we demonstrate that token embeddings generated from definition sentences naturally align with the existing vocabulary embedding space and improve performance across multiple embedding downstream tasks. We also verified that the SLERP-based approximation and the Multi-head Scorer play an essential role in token compression and operate effectively in practice. These findings demonstrate that definition-based token embedding inference provides a conceptually clearer alternative to subword processing and is highly effective for neologisms, rare words, and technical terminology across specialized domains. Furthermore, Definition-Learning is independent of tokenizer types and specific languages, and can be applied to various tokenizers and languages. In addition, during Definition-Learning training, the backbone model remains frozen, so the method can be applied directly to any model with a similar attention-based backbone.

Limitations

In this work, we demonstrate the effectiveness of Definition-Learning as a tokenizer- and backbone-agnostic framework, and highlight the importance of adopting linguistically appropriate tokenizers tailored to each language. Building on these observations, an interesting direction for future work is to extend the framework to a wider range of languages by integrating language-specific tokenizers, as well as to explore its behavior across backbone models of different architectures and scales to further broaden empirical understanding.

In addition, the combination of the MeCab tokenizer and Definition-Learning offers a promising direction for handling subword fragments that appear in later positions, such as **##b** and **##c** in A = a + **##b** + **##c**. Such fragments typically lack intrinsic semantic meaning and do not have corresponding definition sentences, which makes them difficult to handle with conventional rule-based tokenizers trained on fixed segmentation rules. In contrast, the MeCab tokenizer allows these fragments to be treated as complete lexical units by enabling users to register compound words and prevent unnecessary segmentation. This design choice suggests a potential pathway for reducing OOV occurrences and improving embedding quality by encouraging inference at the word level rather than over fragmented subword representations.

References

Jean Aitchison. 2012. Words in the mind: An introduction to the mental lexicon. John Wiley & Sons.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. arxiv 2019. arXiv preprint arXiv:1908.10063.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. arXiv preprint arXiv:2010.02559.

Kexin Chen, Dongxia Wang, Yi Liu, Haonan Zhang, and Wenhai Wang. 2025. Sticking to the mean: Detecting sticky tokens in text embedding models. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 28660–28681.

Bernard Comrie. 1989. Language universals and linguistic typology: Syntax and morphology. University of Chicago press.

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.

Jiyeon Ham, Yo Joong Choe, Kyubyong Park, Ilji Choi, and Hyungjoon Soh. 2020. Kornli and korsts: New benchmark datasets for korean natural language understanding. arXiv preprint arXiv:2004.03289.

Dae Yon Hwang, Bilal Taha, and Yaroslav Nechaev. 2023. Embedtextnet: Dimension reduction with weighted reconstruction and correlation losses for efficient text embedding. In Findings of the Association for Computational Linguistics: ACL 2023, pages 9863–9879.

Wonseok Hwang, Dongjun Lee, Kyoungyeon Cho, Hanuhl Lee, and Minjoon Seo. 2022. A multi-task benchmark for korean legal language understanding and judgement prediction. Advances in Neural Information Processing Systems, 35:32537–32551.

Yewon Hwang, Sungbum Jung, Hanwool Lee, and Sara Yu. 2025. Twice: What advantages can low-resource domain-specific embedding model bring?–a case study on korea financial texts. arXiv preprint arXiv:2502.07131.

Md Mofijul Islam, Gustavo Aguilar, Pragaash Ponnusamy, Clint Solomon Mathialagan, Chengyuan Ma, and Chenlei Guo. 2022. A vocabulary-free multilingual neural tokenizer for end-to-end task learning. arXiv preprint arXiv:2204.10815.

Taehee Jeon, Bongseok Yang, Changhwan Kim, and Yoonseob Lim. 2023. Improving korean nlp tasks with linguistically informed subword tokenization and sub-character decomposition. arXiv preprint arXiv:2311.03928.

Kakao. 2019. Khaiii: Kakao hangul analyzer iii. https://github.com/kakao/khaiii. Accessed: 2025-01-XX.

Taku Kudo. 2005. Mecab: Yet another part-of-speech and morphological analyzer. http://mecab.sourceforge. net/.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, 36(4):1234–1240.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In International conference on machine learning, pages 3744–3753. PMLR.

Jungseob Lee, Hyeonseok Moon, Seungjun Lee, Chanjun Park, Sugyeong Eo, Hyunwoong Ko, Jaehyung Seo, Seungyoon Lee, and Heui-Seok Lim. 2024. Length-aware byte pair encoding for mitigating over-segmentation in korean machine translation. In Findings of the Association for Computational Linguistics ACL 2024, pages 2287–2303.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. arXiv preprint arXiv:2011.05864.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

lovit. 2021. Kowikitext: Wikitext format korean corpus (v20200920.v3). GitHub repository. Version v3, CC-BY-SA 3.0 License.

Leland McInnes, John Healy, and James Melville. 2018. Umap: uniform manifold approximation and projection for dimension reduction. arxiv. arXiv preprint arXiv:1802.03426, 10.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and 1 others. 2017. Mixed precision training. arXiv preprint arXiv:1710.03740.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

Multi-Linguality Multi-Functionality Multi-Granularity. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.

JULIAN M PINE. 2005. Tomasello, m., constructing a language: a usage-based theory of language acquisition. cambridge, ma: Harvard university press, 2003. pp. 388. hardback,£ 29.95. isbn 0-674-01030-2. Journal of Child Language, 32(3):697–702.

Elena Sofia Ruzzetti, Leonardo Ranaldi, Michele Mastromattei, Francesca Fallucchi, Noemi Scarpato, and Fabio Massimo Zanzotto. 2022. Lacking the embedding of a word? look it up into a traditional dictionary. In Findings of the Association for Computational Linguistics: ACL 2022, pages 2651–2662.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. 2018. Representation tradeoffs for hyperbolic embeddings. In International conference on machine learning, pages 4460–4469. PMLR.

Ho-Min Sohn. 2001. The korean language. Cambridge University Press.

Yixuan Tang and Yi Yang. 2024. Do we need domain-specific embedding models? an empirical investigation. arXiv preprint arXiv:2409.18511.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. arXiv preprint arXiv:1905.05950.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems, 30.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. arXiv preprint arXiv:1905.09418.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. arXiv preprint arXiv:2010.05731.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. arXiv preprint arXiv:2212.03533.

Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. 2021. Understanding how dimension reduction tools work: an empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. Journal of Machine Learning Research, 22(201):1–73.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual bert? arXiv preprint arXiv:2005.09093.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and 1 others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Zi Yin and Yuanyuan Shen. 2018. On the dimensionality of word embedding. Advances in neural information processing systems, 31.

Algorithm 1 Geodesic Distance Approximation on LayerNorm–Induced Hyperellipsoid

Require: Hidden states $X \in \mathbb{R}^{N\times S\times H}$, scale $\gamma \in \mathbb{R}^H$, shift $\beta \in \mathbb{R}^H$, number of segments n

(1) Map to Unit Hypersphere and Compute SLERP Coefficients

$X_{\text{center}} \leftarrow X - \beta$

$U \leftarrow X_{\text{center}}/\gamma$ ▷ Project onto hypersphere

$\theta_{ij} \leftarrow \arccos\left(\frac{U_i^\top U_j}{\|U_i\|\,\|U_j\|}\right)$

$\{t_k\}_{k=0}^n \leftarrow \{0, \frac{1}{n}, \ldots, 1\}$

$c_1(t_k) \leftarrow \dfrac{\sin((1-t_k)\theta_{ij})}{\sin(\theta_{ij})}, \quad c_2(t_k) \leftarrow \dfrac{\sin(t_k\theta_{ij})}{\sin(\theta_{ij})}$ ▷ SLERP interpolation coefficients

$\Delta c_1^{(k)} \leftarrow c_1(t_{k+1}) - c_1(t_k), \quad \Delta c_2^{(k)} \leftarrow c_2(t_{k+1}) - c_2(t_k)$ ▷ Segment-wise coefficient differences

(2) Prepare Ellipsoid Metric Terms

$s_1 \leftarrow \|X_{\text{center},i}\|^2, \quad s_2 \leftarrow \|X_{\text{center},j}\|^2, \quad s_{12} \leftarrow X_{\text{center},i}^\top X_{\text{center},j}$

(3) Piecewise Ellipsoid Geodesic Approximation

$\Delta p^{(k)} \leftarrow \sqrt{(\Delta c_1^{(k)})^2 s_1 + (\Delta c_2^{(k)})^2 s_2 + 2\,\Delta c_1^{(k)}\Delta c_2^{(k)} s_{12}}$

return $D_{ij} \leftarrow \sum_{k=0}^{n-1} \Delta p^{(k)}$

A Projection Loss

Motivated by the fact that the final outputs of the backbone encoder layers are normalized by Layer Normalization, we propose a loss function based on an approximate geodesic distance. As derived in Equations (6)–(8), Layer Normalization places embedding vectors on a hyper-ellipsoid, which can be interpreted as a shifted hypersphere of radius $\sqrt{H}$ with axis-wise scaling γ and a mean shift β.

$$\hat{x}_i = \text{LayerNorm}(x_i) = \gamma \odot \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta \quad (6)$$

$$\mu_i = \frac{1}{H}\sum_{k=1}^{H} x_{ik}, \quad \sigma_i^2 = \frac{1}{H}\sum_{k=1}^{H}(x_{ik} - \mu_i)^2 \quad (7)$$

$$\left\| \frac{x_i - \mu_i}{\sigma_i} \right\|^2 = \frac{1}{\sigma_i^2}\sum_{k=1}^{H}(x_{ik} - \mu_i)^2 = H \quad (8)$$

where $x_i \in \mathbb{R}^H$ is the input token representation, $\hat{x}_i \in \mathbb{R}^H$ is its normalized output, $\odot$ is element-wise multiplication, $\gamma, \beta \in \mathbb{R}^H$ are the learnable scale and shift parameters of LayerNorm.

Based on this observation, we first map the vectors onto the unit hypersphere, compute the spherical angle θ between them, and then approximate the geodesic distance using a SLERP-based segmented integral method, as summarized in Algorithm 1).

B Projection Loss Relative Error

The relative approximation error between the exact and SLERP-based geodesic distances is derived in Equations (9)–(13). As shown in Figure 8, the error remains small across the entire range $\theta \in [0, \pi]$.

For two points on a hypersphere of radius R with angular separation θ, the exact geodesic distance is

$$d_{\text{geo}} = R\theta \quad (9)$$

Using n-segment SLERP interpolation, the approximated geodesic distance is

$$\hat{d}_{\text{geo}} = 2R \cdot n \sin\left(\frac{\theta}{2n}\right) \quad (10)$$

Thus, the relative approximation error is

$$\epsilon(\theta, n) = \left| 1 - \frac{\hat{d}_{\text{geo}}}{d_{\text{geo}}} \right| = \left| 1 - \frac{2n \sin(\theta/2n)}{\theta} \right| \tag{11}$$

By substituting $x = \frac{\theta}{2n}$, the relative error term can be rewritten in a compact form as

$$\epsilon(\theta, n) = \left| 1 - \frac{2n \sin(\theta/2n)}{\theta} \right| = \left| 1 - \frac{\sin x}{x} \right| \tag{12}$$

Since the geodesic on the scaled ellipsoid is obtained by applying the axis-wise scaling factor γ to the spherical geodesic, the relative approximation error on the ellipsoid can be closely approximated by the spherical case:

$$\epsilon_{\text{ellipsoid}}(\theta, n) \approx \epsilon_{\text{sphere}}(\theta, n) \tag{13}$$

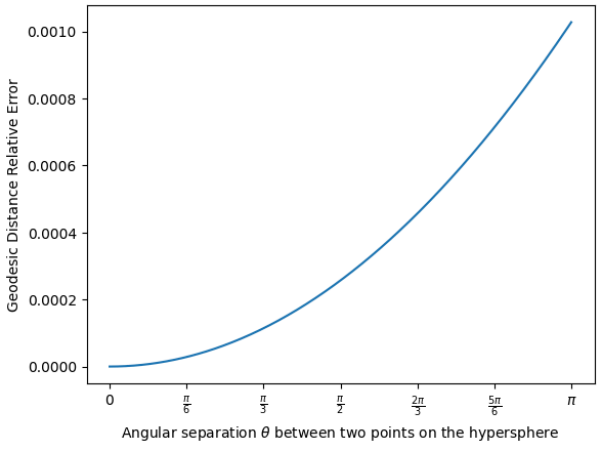

Figure 8: Relative approximation error between the exact geodesic distance and its SLERP-based n-segment interpolation on a hypersphere, expressed as $1 - \frac{\sin x}{x}$ where $x = \frac{\theta}{2n}$. As shown, the error remains very small (below 0.1%) across the full angular range $\theta \in [0, \pi]$.

C Dataset

Purpose	Dataset	Type	Size
Backbone Training	Ko-wiki	corpus	2,877,800
Definition-Learning	Standard Korean Dictionary	definition sentences	25,746
Evaluation	LCUBE	casename train	8,000
		casename test	2,294
		casename+ train	22,494
		casename+ test	4,790
		statute test	814
		statute+ test	2,137
	KorSTS	Sentence pair	1,379

Table 6: Datasets used for backbone training, Definition-Learning, and evaluation.

Table 6 shows the detailed statistics of all datasets. First, we use the Ko-wiki (lovit, 2021) train corpus to learn morpheme-level linguistic representations and contextual structures for the RoBERTa backbone model. We randomly hold out 10% of the data for validation. Second, Definition-Learning is trained on definition sentences from the Standard Korean Dictionary[5]. This contains the largest number of Korean definition sentences and is produced by a national institution, making it more reliable than other sources. Among all definition data, we select data that already exist in the backbone vocabulary if a word does not exist in the vocabulary, we cannot make a label of it. Also, we randomly hold out 10% of the data for validation. Third, to evaluate the downstream task performance of the DefVEx model using Definition-Learning, we use the LCUBE (Hwang et al., 2022) dataset, which includes classification and retrieval tasks in the legal domain. Prior studies (Tang and Yang, 2024; Hwang et al., 2025) have emphasized that evaluating embeddings in specialized domains requires domain-specific datasets written in the original language rather than translated ones. LCUBE satisfies these requirements, as it is built from official Korean legal documents with domain-specific terminology. In addition to domain-specific evaluation, we evaluate the KorSTS (Ham et al., 2020) benchmark, which contains sentence pairs originally written in Korean and is widely used to assess Korean sentence embedding models.

[5] https://stdict.korean.go.kr/main/main.do.

Loss Term	KorSTS	Casename	Casename+	Statute	Statute+
DefVEx	**0.4991**	**0.2931**	**0.4503**	**0.4870**	**0.2105**
$\mathcal{L}_{\text{norm}} = \left(\|\hat{e}\| - \|e\| \right)^2$	0.4912	0.2911	0.4483	0.4484	0.1747
w/o $\mathcal{L}_{\text{InfoNCE}}$	0.4839	0.2861	0.4455	0.4186	0.1736

Table 7: Performance comparison of $\mathcal{L}_{\text{EA}}$, showing the effect of the magnitude term $\mathcal{L}_{\text{norm}}$ and removing $\mathcal{L}_{\text{InfoNCE}}$.

D Training Settings

D.1 Backbone RoBERTa model training setup

We follow the same backbone architecture and training setup as the original RoBERTa-base model. The backbone consists of 12 Transformer encoder layers, a maximum sequence length of 512, and a hidden size of 768. We use the GeLU activation function, LayerNorm with $\epsilon = 1 \times 10^{-12}$, and a dropout rate of 0.1. For optimization, we use the AdamW optimizer with a weight decay of 0.01. The learning rate is set to 1×10^{-4}, using a linear learning-rate schedule with a warm-up ratio of 0.1. To stabilize training, we employ mixed-precision training (AMP) (Micikevicius et al., 2017), apply gradient clipping with a max-norm of 1.0, and initialize all weights from a truncated normal distribution with mean 0.0 and standard deviation 0.02.

For pretraining, we use the dynamic MLM objective with cross-entropy loss. The masking ratio is 15%, following the original RoBERTa setup: 80% of selected tokens are replaced with [MASK], 10% with a random token, and 10% remain unchanged. We train the model for 50 epochs on a single RTX 4090 GPU. Due to VRAM limitations, we use gradient accumulation. Each forward pass processes a batch of 16 samples, and 16 accumulation steps are combined into a single backward pass, resulting in the same effective batch size of 256 as used in RoBERTa.

D.2 Definition-Learning training setup

For Definition-Learning training, we adopt the same optimization setup as in backbone pretraining, except that AMP is disabled. The Definition Aggregator uses four inference heads. The ϵ used in the loss computation is set to 1×10^{-6}. To stabilize training, we reuse the same gradient-accumulation strategy adopted in backbone training and set the accumulation steps to 128, resulting in an effective batch size of 128. Because definition learning uses a base batch size of 1, the number of definition sentences N for a single target word replaces the conventional batch dimension used in RoBERTa pretraining. Training is performed for 20 epochs on a single RTX Pro 6000, and early stopping is applied when the validation loss converges. In addition, the projection components of the Definition Aggregator module are frozen after 2 epochs, as these layers converge sufficiently early.

Epoch	Hard-negatives	Soft-negatives	Positive	τ
0	0	2047		0.15
1	4	2043		0.12
2	8	2039	1	0.10
3	12	2035		0.08
4∼	16	2031		0.07

Table 8: Schedule of hard/soft negatives and temperature τ during Definition-Learning training.

To construct contrastive training batches, we use a total of 2,048 examples per step, consisting of hard negatives, soft negatives, and one positive example. The positive example is the label token embedding corresponding to the input definition sentences. Hard negatives are selected as the top-k most similar vectors to the label embedding based on cosine similarity. In contrast, soft negatives are randomly sampled at each step from the top 10,000 most similar candidates. The number of hard negatives and the temperature parameter τ are scheduled over the epochs to allow the model to begin with easier discrimination and progressively adapt to more challenging contrastive distinctions. The detailed schedule is summarized in Table 8

D.3 Ablation Study on Terms $\mathcal{L}_{\text{EA}}$

We use the following loss weights during training:

$\mathcal{L}_{\text{EA}} = \lambda_{\text{norm}} \cdot \mathcal{L}_{\text{norm}} + \lambda_{\text{cos}} \cdot \mathcal{L}_{\text{cos}} + \lambda_{\text{nce}} \cdot \mathcal{L}_{\text{InfoNCE}}$
($\lambda_{\text{norm}} = 0.05,\ \lambda_{\text{cos}} = 0.5,\ \lambda_{\text{nce}} = 1.0$)

Table 7 reports an ablation study of each component in the $\mathcal{L}_{\text{EA}}$. As discussed earlier, the vocabulary embedding space is not magnitude-normalized, which motivates the inclusion of the magnitude alignment term to prevent inferred embeddings from having mismatched magnitudes. The variant shown in the table penalizes all norm differences, whereas the original formulation only penalizes cases where the absolute difference exceeds 0.05. Including small norm differences below this thresh-

사랑 (Love)		종범 (Accomplice)		피고인 (Defendant)		공권력 (State Authority)	
token	cos-sim	token	cos-sim	token	cos-sim	token	cos-sim
가족 (family)	0.3804	범죄 (crime)	0.4818	피의자 (suspect)	0.4700	권력 (authority)	0.3890
마음 (heart)	0.3730	범인 (offender)	0.4158	항소심 (appellate trial)	0.4045	사법권 (judicial power)	0.3884
청춘 (youth)	0.3620	피의자 (suspect)	0.3956	무죄 (innocence)	0.3965	권리 (right)	0.3718
교제 (dating)	0.3577	공범 (accomplice)	0.3952	증인 (witness)	0.3955	헌법 (constitution)	0.3640
매력 (attractiveness)	0.3560	살인법 (homicide law)	0.3747	##소송 (##litigation)	0.3947	행정권 (administrative power)	0.3633

Table 9: Top-5 nearest tokens and their cosine similarities for each embedding inferred using Definition-Learning.

old degrades performance, as it over-regularizes embeddings that are already well aligned in magnitude and suppresses task-relevant variation. Moreover, performance degrades even when small norm differences are penalized under the already small weight $\lambda_{\text{norm}} = 0.05$, suggesting that increasing this weight would further over-regularize the embedding space and is therefore undesirable. In addition, excluding the contrastive term $\mathcal{L}_{\text{InfoNCE}}$ consistently reduces performance, suggesting that contrastive supervision is crucial to preserve discriminability among semantically similar neighbors in a dense embedding space. First, we fix λ_{nce} to 1.0 and select the weight of $\lambda_{\cos}$ by progressively decreasing it while evaluating downstream task performance. Overall, these results highlight that the combined loss formulation is necessary to achieve stable and well-aligned vocabulary expansion.

E Performance Analysis of DefiNNet, DefBERT and Definition-Learning

		nouns	**verbs**
Dataset	*Model*	*sim*	*sim*
Test_{w2v}	Additive	0.25 (±0.17)	0.29 (±0.19)
	Head	0.26 (±0.21)	0.29 (±0.25)
	DefiNNet	**0.39** (±0.18)	**0.46** (±0.14)
Test_{BERT}	DefBERT_{Head}	**0.46** (±0.13)	**0.41** (±0.14)
	$\text{DefBERT}_{[CLS]}$	0.32 (±0.08)	0.30 (±0.09)
	$\text{BERT}_{Head\text{-}Example}$	0.41 (±0.12)	0.39 (±0.12)
$\text{Test}_{w2v \cap BERT}$	DefBERT_{Head}	**0.47** (±0.13)	**0.42** (±0.15)
	$\text{DefBERT}_{[CLS]}$	0.28 (±0.09)	0.30 (±0.09)
	DefiNNet	0.33 (±0.13)	**0.47** (±0.13)

Table 10: Cosine similarity results between original word embeddings and embeddings inferred from definition sentences, as reported in (Ruzzetti et al., 2022), for DefiNNet and DefBERT. The test sets consist of words from the Word2Vec and BERT vocabularies, with evaluations reported separately for nouns and verbs.

Ruzzetti et al. (2022) evaluated embeddings inferred from definition sentences by measuring cosine similarity with label embeddings, as shown in Table 10. Both DefiNNet and DefBERT infer vocabulary token embeddings by identifying head tokens through structural analysis of definition sentences. For DefiNNet, the inferred embedding is obtained by passing the head token embedding through a two-layer MLP. As baselines, DefiNNet compares its predictions against two variants: (1) Additive, which sums the embeddings of the head token without an MLP, and (2) Head, which directly uses the head token embedding. For DefBERT, the definition sentence is encoded using BERT, and the contextual embedding of the head token is used as the inferred embedding. Its baselines include $\text{DefBERT}_{[CLS]}$, which uses the [CLS] token embedding instead of the head token, and $\text{BERT}_{Head\text{-}Example}$, which uses the head token embedding extracted from an example sentence.

As shown in Table 9, the cosine similarity between embeddings inferred via Definition-Learning and their nearest tokens ranges from $0.38 \sim 0.48$, which is comparable to DefiNNet and DefBERT. This observation indicates that cosine similarity alone is insufficient to determine whether an embedding is truly inferred from the definition of a specific word or merely represents a vector located in a similar region of the embedding space.

F Token Classification Procedure

The token classification procedure consists of three steps: (1) tokenizing each input sentence using MeCab, (2) identifying [UNK] tokens, and (3) checking whether each [UNK] token has a definition in the dictionary. The complete procedure is detailed in Algorithm 2. Based on this procedure, tokens labeled as the $fst\ unk\ token\ in\ dict$ are added to the vocabulary using Definition-Learning.

Algorithm 2 Token type filtering algorithm

Input: Test dataset $\mathcal{D}$, MeCab tokenizer T, Backbone vocabulary $\mathcal{V}$, Definition dictionary $\mathcal{D}_{dict}$

for each sentence $s \in \mathcal{D}$ **do**
  Split s into word-level units $W = \{w_1, w_2, \ldots, w_n\}$ based on whitespace

(1) Tokenizing with MeCab tokenizer
  for each word $w \in W$ **do**
    Tokenize using MeCab: $T(w) = \{t_0, t_1, \ldots, t_k\}$
    Define t_0 as *fst token* and $\{t_1, \ldots, t_k\}$ as *snd token*

(2) [UNK] token filtering
    if $t_0 \notin \mathcal{V}$ **then**
      add t_0 to $fst\ unk\ token$
    end if

    for each $t_j \in \{t_1, \ldots, t_k\}$ **do**
      if $t_j \notin \mathcal{V}$ **then**
        add t_j to $snd\ unk\ token$
      end if
    end for

(3) Dictionary filtering
    for each token $u \in fst\ unk\ token \cup snd\ unk\ token$ **do**
      if $u \in \mathcal{D}_{dict}$ **then**
        label as *in dict*
      else
        label as *no dict*
      end if
    end for
  end for
end for

G Attention-Based Scorer

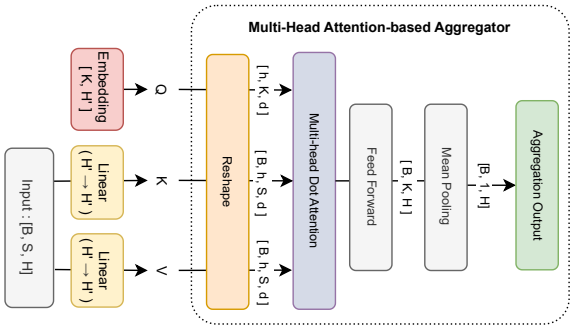

Figure 9: Overview of the token compression mechanism used in Set Transformer.

The Set Transformer (Lee et al., 2019)-style attention–based Scorer is described in 9. In this architecture, the input vectors serve as the Keys and Values of a multi-head attention block, while independently learned Query vectors determine which elements should be emphasized. Depending on the number of Queries, the model can compress the input set into a single vector or multiple vectors The attention outputs are subsequently passed through an MLP layer followed by mean pooling to produce the token- and sentence-aggregated representation. Through this process, a $[H']$-dimensional vector is produced at each of the N layers, and final layer aggregation is generated using a $[1, H']$ Query vector with the same attention-based Scorer before the Dimension Reconstruction stage.