# RECONSTRUCTING WORD EMBEDDINGS VIA SCATTERED $k$-SUB-EMBEDDING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The performance of modern neural language models relies heavily on the diversity of the vocabularies. Unfortunately, the language models tend to cover more vocabularies, the embedding parameters in the language models such as multilingual models used to occupy more than a half of their entire learning parameters. To solve this problem, we aim to devise a novel embedding structure to lighten the network without considerably performance degradation. To reconstruct $N$ embedding vectors, we initialize $k$ bundles of $M(\ll N)$ $k$-sub-embeddings to apply Cartesian product. Furthermore, we assign $k$-sub-embedding using the contextual relationship between tokens from pretrained language models. We adjust our $k$-sub-embedding structure to masked language models to evaluate proposed structure on downstream tasks. Our experimental results show that over 99.9%+ compressed sub-embeddings for the language models performed comparably with the original embedding structure on GLUE and XNLI benchmarks.

## 1 INTRODUCTION

Embedding is useful for mapping high-dimensional vector to a low-dimensional space for various neural network models such as language models and graph neural networks. Especially in modern language models, a contextual embedding is represented respectively for each contextual unit. For instance, some language models like word2vec (Mikolov et al., 2013) which deal with a whole word as one embedding vector, suffer from the Out-of-Vocabulary(OOV) problem due to the variety of words. Later language models (Bojanowski et al., 2017) subdivide words into multiple segmented words. To deal with the subdividing, various tokenizers (Sennrich et al., 2016; Kudo & Richardson, 2018) are proposed to construct common tokens based on word corpuses. However, each token contains a different context, and a good criterion for dividing the token is not clear.

There are some cases that the atomic-level tokens share notable common properties. For example, two tokens $\langle A \rangle$ and $\langle a \rangle$ are assigned to different embeddings, but they both are vowels and the same alphabet. In the case of combined characters(i.e., Hangul of Korean character) or ideograms(i.e., Chinese and Japanese characters), some tokens might include more common meanings than other languages. The tokens of similar meanings tend to locate closely in embedding space when the language model is trained with those tokens.

In this paper, we propose a novel embedding structure which replaces a word embedding with several sub-embeddings. If a token consists of several contextual elements, a sub-embedding can be assigned to each element to constitute the original embedding. An embedding vector shares its learning parameters with other embedding vectors because of the nested allocated sub-embeddings. Various scattered sub-embedding structures can be generated depending on structural topology of the sub-embeddings. Firstly, we sequentially allocate sub-embeddings through module operation, and conduct Cartesian product with sub-embeddings to build embedding vectors(Figure 1). This could robustly map the sub-embeddings, but each sub-embedding does not reflect the context of each token. To address this problem, we modify the scattering algorithm by clustering the tokens in the embedding space using a pretrained network. Notable factors of this paper are as follows:

1. The number of learning parameters in embedding part is dramatically reduced by Cartesian product.
2. The language models are free from OOV problem through sub-embedding.

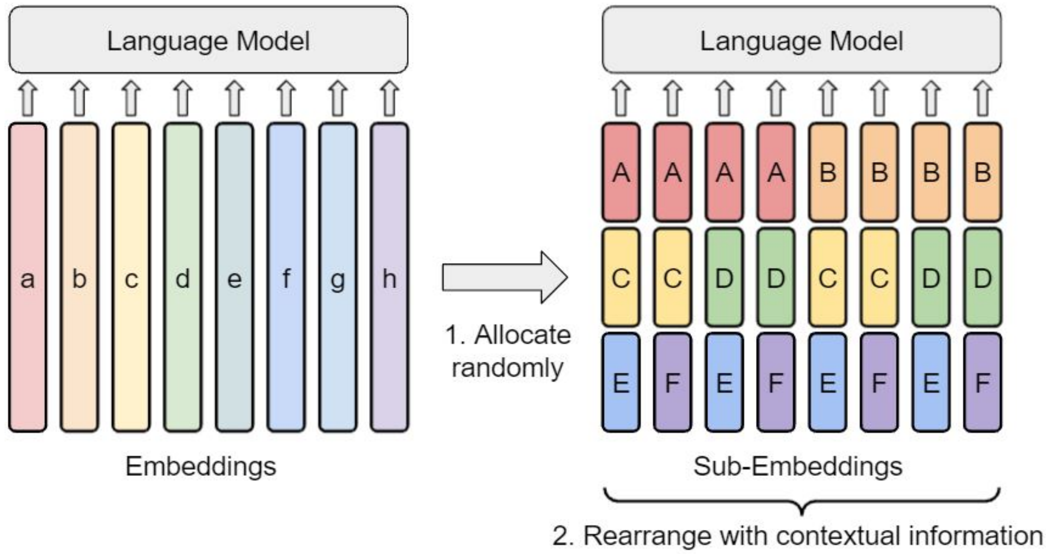



Figure 1: Assuming the language model has 8 vocabularies(embedding vectors), and an embedding vector is ready for separation into three sub-embedding vectors. The sub-embedding blocks denoted in the same letter share learning parameters across the embeddings. As a result, 8 embedding vectors can be reconstructed with only 6 sub-embedding vectors. We suggest two allocating sub-embedding methods, 1) sequentially allocate sub-embeddings(Algorithm 1); 2) rearrange the sub-embeddings using their contextual information from a pretrained network(Algorithm 2).

3. The proposed embedding structure can be applied to most existing language models easily by replacing only the input embedding part.

We evaluate our sub-embedding structure on English and multilingual downstream tasks. We borrow RoBERTa (Liu et al., 2019) structure for the standard English downstream tasks. In addition, XLM-R (Conneau et al., 2020) is used to test the performance of sub-embedding in multilingual tasks. We show that the network replaced by the $k$-sub-embedding compresses the embedding over 99.9% while the accuracy on multilingual benchmark drops only 1.4%p. Finally, we demonstrate how the sub-embedding is trained by visualizing the distribution of $k$-sub-embeddings.

## 2 Related Work

**Distributed Representations of Word Embedding.** Word embedding is commonly used in the language models to represent the word in the latent space. Distributed representation is mainly used instead of a one-hot vector for the efficiency. Despite the difficulty of learning distributed representation, word2vec (Mikolov et al., 2013) is able to map words to embedding space successfully and shows context analogies of each word embedding. In the case of word2vec, the vocabulary is composed with the words in the input corpus, it is easily suffered from OOV problem. To deal with this problem, some language models (Pennington et al., 2014; Bojanowski et al., 2017) split the words into subtokens and learn the co-occurrence of words. Furthermore, ELMo (Peters et al., 2018) gathers hidden states and the embeddings of the bidirectional language models to represent contextual embeddings. Around the same time, language models (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020) based on transformer (Vaswani et al., 2017) are proposed to learn the context of entire input sequence with the attention mechanism. In this paper, we split the contextual embeddings into sub-embeddings that contain common context of the embeddings.

**Relationship Between Tokenizers and Language Models.** Language Models usually split a sequence into tokens through tokenizers. Classical tokenizers are designed to divide a sequence into contextual units such as words, characters, or morphs. Those tokenizers are intuitive to implement, but they are easily suffered from OOV problem and need special knowledge to split into morphs. To alleviate the problem mentioned above, some works propose that the tokenizers learn the input corpus to construct concrete vocabularies. Byte-Pair Encoding (Sennrich et al., 2016) iteratively merge tokens to a larger token, byte-level tokens are used to cover all input cases (Radford et al., 2019). On the other hand, Kudo (2018); Kudo & Richardson (2018) iteratively reduce meaningless tokens from a large token set. We can build up the vocabulary robustly using these tokenizers, but the contextual unit can be different to each token.

Typical embedding scheme maps the token to each embedding vector, CharformerTay et al. (2021) pinpoints this inflexible problem. They propose the gradient-based method to figure out latent subword representation. In contrasts to other static tokenizers, the embedding of token is changed by subword block. In addition, Clark et al. (2021) alleviates the tokenization framework, the tokenizer is changed to character-level without human knowledge. Xue et al. (2021) proposes token-free models which encode input sequence with contextual unit, not tokens. In this paper, we propose the embedding structure regardless of tokenizers. Unlike token-free models (Xue et al., 2021), the our $k$-sub-embedding structure needs a tokenizer, but the networks do not suffered anymore from the number of tokens.

Since the embeddings act a prominent role in the language models, a series of studies come up with simplifying the embeddings. One of the lightening methods is to compress the embedding weight with matrix factorization. Lan et al. (2020) conducts matrix factorization to the embedding part, and the high-dimensional embedding vector could be compressed suceessfully to 128-dimensional vector. Our proposed structure also reconstruct the embedding through lookup operation without additional computations.

# 3 Proposed Methods

We investigate the requirements of each sub-embedding, and suggest (1) randomly scattered $k$-sub-embedding with naive approach; (2) clustered $k$-sub-embedding using contextual knowledge.

## 3.1 Preliminaries

Modern language models suffer from massive parameters of embeddings because the number of embeddings is associated directly with the diversity of output tokens. Some studies tried to compress the embedding part; Lan et al. (2020) lightened the embedding weight via matrix factorization. We horizontally split embedding vectors into sub-embedding vectors which are shared with some other embedding vectors. In the proposed embedding structure, sub-embeddings tend to highly correlate. We suggest the following requirements to verify the modified embeddings perform like the original embedding.

1. (Uniqueness of divided embeddings) Let $\boldsymbol{e}_i, \boldsymbol{e}_j$ be embedding vectors, and $\{\boldsymbol{e}_i^k\}_{k=1}^K$, $\{\boldsymbol{e}_j^k\}_{k=1}^K$ are their sub-embeddings, $\forall i, j \in \{1, \ldots, N\}\ \exists k^* \in \{1 \ldots K\}\ s.t.\ \boldsymbol{e}_i^{k^*} \neq \boldsymbol{e}_j^{k^*}$ and $i \neq j$.
2. (Contextual mapped sub-embedding) If a pair of tokens have similar contextual meanings, their sub-embeddings share more parts than an arbitrary pair.

We try to find the function to map the origin embedding vectors into sub-embedding vectors. In this work, we suggest the bijective function $\mathcal{F} : \mathcal{N} \rightarrow \mathcal{M} \times \cdots \times \mathcal{M}$ for converting the embedding to each sub-embedding where $\mathcal{N} = \{1, \ldots, N\} \subset \mathbb{N}$ is a set of the embedding index and $\mathcal{M} = \{1, \ldots, M\} \subset \mathbb{N}$ is a set of each sub-embedding index. To expand the function to allocate sub-embedding in several ways, we generalize the mapping function using Cartesian product of functions:

$$\mathcal{F}(n) = (f_1 \times f_2 \times \cdots \times f_k)(\underbrace{n, \ldots, n}_{k}), \tag{1}$$

where the function $f_k : \mathcal{N} \rightarrow \mathcal{M}$ denotes $k$-th sub-embedding index. Since the cardinality of $k$-ary Cartesian product is $M^k$, the embeddings can be covered only $N^{\frac{1}{k}}$ sub-embeddings. This makes the number of embedding parameters dramatically reduce with $log$ scale. We suppose that the dimension of the embedding $d$ is distributed equally to sub-embedding $M = d/k$. Then, the number of the embedding parameters $N \times d$ is scaled down to $k \times M \times (d/k) = M \times d$. Finally, the representation of the embedding would be replaced by

$$\boldsymbol{e}_n = \bigoplus_{k=1}^{K} \boldsymbol{e}_{f_k(n)},$$

where $\bigoplus$ is a concatenate operator, and $\boldsymbol{e}_n, \boldsymbol{e}_{f_k}$ are the corresponding embedding vector of the embedding index. We suggest Algorithm 1, 2 by modifying $f_k$ that satisfies the requirements.

### 3.2 Randomly Scattered $k$-Sub-Embedding

We construct naively $f_k$ to combine sub-embeddings through a Cartesian product which can generate up to $M^k$ embeddings. To ensure generated embeddings to be distinct, the number of each sub-embedding $M$ should be larger than $N^{1/k}$. Randomly allocated sub-embedding algorithm(Algorithm 1) shows that whole embeddings are generated by repeatedly applying the modulo operation. We set $M = \lceil N^{\frac{1}{K}} \rceil$ to extract the most compressed sub-embeddings. The modulo operation is derived with the perspective of $M$-base number. Converting to $M$-base number is bijective, and we can easily set the index of each sub-embedding as each digit of the radix.

**Algorithm 1** Randomly allocated Sub-Embedding

1: **Input:** Number of the embeddings $N$, embedding dimension $d$, sub-embedding sets $K$
2: $M \leftarrow \lceil N^{\frac{1}{K}} \rceil$ ▷ number of each sub-embedding
3: Initialize $k$-th $M$ sub-embedding vectors $\{\boldsymbol{e}_m^k \in \mathbb{R}^{\frac{d}{k}}\}_{m=1}^{M}$ for all $k \in \{1, \ldots, K\}$
4: **for** $n = 1, 2, \ldots, N$ **do**
5:     **for** $k = 1, 2, \ldots, K$ **do**
6:         $f_k(n) \equiv (n/M^{k-1}) \bmod M^k$, ▷ $k$-th digit of the $M$-base number
7:     **end for**
8:     $e_n = \bigoplus_{k=1}^{K} e_{f_k(n)}^k$
9: **end for**
10: **Output:** The combined embedding vectors $\{\boldsymbol{e}_n\}_{n=1}^{N}$.

### 3.3 Clustered $k$-Sub-Embedding

The language models based on transformer learn the whole context of the input sequence, and each embedding vector of a token is mapped to the embedding space reflecting its context. Mikolov et al. (2013) recognizes that arbitrary two closely mapped word vectors tend to have similar context. If the context of each token is given, we can enhance the allocation heuristic using the contexts. In the case of tokens that have a similar meaning, we assume that the two tokens can be classified with smaller differences. In this case, we can allocate more sub-embeddings to be shared. To estimate the similarity of each pair of the tokens, a pretrained network is used to get each L2 distance between each embedding vector. Assuming that all sub-embeddings are independently initialized randomly, the tokens that share more sub-embedding are expected to have less L2 distance.

The method of allocating sub-embedding to similar tokens is based on k-means (Arthur & Vassilvitskii, 2007) algorithm. Each embedding vector is treated as an instance of the k-means algorithm, thus the algorithm is adjusted recursively to each sub-embedding space. The recursive clustering algorithm aims to separate the instances which are allocated in some identical sub-embeddings. According to Algorithm 2, the mapped sub-embeddings can satisfy the second requirement due to the k-means algorithm based on L2 space.

## 4 Experiments

### 4.1 Experimental Setup

There are a few variant language modelings such as translation language modeling(TLM), casual language modeling(CLM), and MLM, that depends on the purpose of each modeling. We replace the

**Algorithm 2** Allocating Clustered Sub-Embedding

**Input:** Number of embeddings $N$, number of sub-embeddings $M$, embedding dimension $d$, number of sub-embedding sets $K$, embeddings of pretrained network $\mathcal{P} = \{\rho_n\}_{n=1}^{N}$
Initialize $k$-th $M$ sub-embedding vectors $\{\boldsymbol{e}_m^k \in \mathbb{R}^{d/K}\}_{m=1}^{M}$ for all $k \in \{1, \ldots, K\}$
$f_k(n) \leftarrow 0, \forall k = 1, \ldots, K,\ n = 1, \ldots, N$ ▷ Initialize the labels to zero
**for** $k = 1, 2, \ldots, K$ **do**
    extract unique tuples from $\{\mathcal{F}(n)\}_{n=1}^{N}$ ▷ Equation 1
    **for** unique $\mathcal{F}(n^*)$ in $\{\mathcal{F}(n)\}_{n=1}^{N}$ **do**
        **if** $k \neq K$ **then**
            $\mathcal{P}_{\mathcal{F}(n^*)} \leftarrow \{\rho_n : \mathcal{F}(n) = \mathcal{F}(n^*)\}_{n=1}^{N}$
            Adjust k-means algorithm to $\mathcal{P}_{\mathcal{F}(n^*)}$
            Labeling the results to $f_k(n)$ where $\mathcal{F}(n) = \mathcal{F}(n^*)$
        **else**
            $f_k(n) \leftarrow$ random number among $M$ candidates where $\mathcal{F}(n) = \mathcal{F}(n^*)$.
        **end if**
    **end for**
**end for**
Gather $\boldsymbol{e}_n = \bigoplus_{k=1}^{K} \boldsymbol{e}_{f_k(n)}^k \quad \forall n \in \mathcal{N}$
**Output:** The combined embedding vectors $\{\boldsymbol{e}_n\}_{n=1}^{N}$.

word embeddings of MLM with the sub-embedding to inspect the effect of our proposed embedding structure.

#### 4.1.1 Datasets

Common MLM is trained from a large plain text dataset and then fine-tuned in downstream tasks. The language models are trained primarily in monolingual dataset. Additionally, the multilingual dataset is trained to verify the $k$-sub-embedding also works within large vocabularies. BooksCorpus (Zhu et al., 2015) and English Wikipedia corpuses are used for the monolingual set as following BERT (Devlin et al., 2019). In the case of the multilingual experiments, we extract CommonCrawl (Wenzek et al., 2020) corpuses written in 15 languages which defined in XNLI(Conneau et al., 2018). More details of the dataset are summarized in the appendix.

#### 4.1.2 Configurations of the Language Models

We modify RoBERTa (Liu et al., 2019), XLM-R (Conneau et al., 2020) implementations in HuggingFace (Wolf et al., 2020) framework. RoBERTa is followed BERT approach except for NSP prediction, and optimize hyperparameters such as momentum value and learning rate. We borrow the hyperparameters of RoBERTa, also we adapt MLM prediction which has a 0.15 probability of masking token. In order to simplify the network scheme, we set the base network to be composed of 8 transformer encoder layers and 512-d embeddings in contrast to $\text{BERT}_{\text{BASE}}$. In similar fashion for the multilingual case, XLM-R is modified to 8 layers like predefined RoBERTa. The networks that we modified are denoted as $\text{RoBERTa}_{\text{MEDIUM}}$, and $\text{XLM-R}_{\text{MEDIUM}}$ as shown in Table 1. We present randomly allocated case of $k$-sub-embeddings in the table. The number of embedding parameters in the modified networks is reduced remarkably from $\text{RoBERTa}_{\text{MEDIUM}}$, and $\text{XLM-R}_{\text{MEDIUM}}$. Other training settings(e.g., tokenizers) are followed as (Liu et al., 2019; Conneau et al., 2020).

#### 4.1.3 Other Embeddings in the Language Models

The language models (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Lan et al., 2020) are built on common transformer backbones, but their criterions are slightly different. In this study, we construct the network as MLM structure not using NSP. Token prediction models such as MLM needs a decoder from language models to predict token representation. Many language models tie the last output weights to input embedding weights for robustness, Chung et al. (2021) shows that not sharing between the embeddings and the decoder also can be performed better by regulating the number of features in the decoder. Empirically, the result of decoupling the embedding and

| | $\lvert V \rvert$ | $\lvert \theta \rvert$ | $E$ | $\lvert \theta_{\textbf{emb}} \rvert$ | **#layers** | $d$ |
|---|---|---|---|---|---|---|
| RoBERTa$_{\text{MEDIUM}}$ | 50k | 51M | 50k | 25.7M | 8 | 512 |
| +2-sub-embedding | 50k | 26M | 225 | 115k | 8 | 512 |
| +3-sub-embedding | 50k | 26M | 37 | 18.9k | 8 | 512 |
| +8-sub-embedding | 50k | 26M | 4 | 2k | 8 | 512 |
| XLM-R$_{\text{MEDIUM}}$ | 250k | 154M | 250k | 128M | 8 | 512 |
| +3-sub-embedding | 250k | 26M | 63 | 32k | 8 | 512 |

Table 1: **Overview of the neural language networks.** The configurations of each networks where $E$ is the number of the embeddings, $d$ is embedding vector size, and $|\theta|$, $|\theta_{\text{emb}}|$ are the number of parameters. We assume that the $k$-sub-embedding is allocated randomly by Algorithm 1.

the decoder is not different from coupling weights, we trained the language models with coupling decoders. We replace the embedding part with the previous network into sub-embedding structure. There are additional embeddings to capture external information such as positional embeddings and token type embeddings, but we do not replace those embeddings to the sub-embedding.

#### 4.1.4 Evaluation Benchmarks

The modified neural language models are evaluated by GLUE (Wang et al., 2019) benchmarks which are consisted of single-sentence tasks, similarity and paraphrase tasks, and inference tasks.[1] We also test multilingual language models with XNLI(Conneau et al., 2018) benchmark. XNLI is constituted with 15 monolingual corpuses for Natural Language Interface(NLI). Following Huang et al. (2019), we fine-tune the pretrained XLM-R in two ways, cross-lingual task and multi-language task.

### 4.2 Comparing the Randomly Allocated Sub-Embeddings

We construct the base network as in (Liu et al., 2019), and we replace the input embedding part to randomly scattered $k$-sub-embedding structure(Algorithm 1). Table 2 shows that the results on GLUE benchmarks of base network and $k$-sub-embedding. As the number of sub-embedding is defined by $M = N^{\frac{1}{k}}$, it needs only 4 sub-embeddings when $k$ is 8 for the given $N$ is 50627. The results on GLUE among $k$-sub-embedding networks are similar between each other. However, the results underperform compared to RoBERTa$_{\text{MEDIUM}}$, it seems that randomly allocated sub-embeddings could not overcome the extremely entangled embedding part. The performance might decrease because we do not consider some special tokens(e.g. separate token, and padding token) when the sub-embeddings are allocated. To alleviate this problem, we introduce clustered allocating procedure based on the pretrained knowledge.

| **Model** | $k$ | $E$ | **SST-2** | **MNLI** | **QNLI** | **QQP** | **RTE** | **MRPC** | **CoLA** | **STS-B** |
|---|---|---|---|---|---|---|---|---|---|---|
| RoBERTa$_{\text{MEDIUM}}$(ours) | 1 | 50k | 89.9 | 79.6 | 88.2 | 86.6 | 72.9 | 88.4 | 38.1 | 88.1 |
| +2-Sub-Embedding | 2 | 225 | 88.6 | 74.3 | 84.0 | 84.0 | 66.8 | 88.1 | 35.7 | 79.3 |
| +3-Sub-Embedding | 3 | 37 | 88.0 | 73.2 | 83.5 | 83.0 | 67.9 | 85.6 | 18.4 | 77.4 |
| +4-Sub-Embedding | 4 | 15 | 88.1 | 72.7 | 84.2 | 83.4 | 70.0 | 87.5 | 23.3 | 78.5 |
| +6-Sub-Embedding | 6 | 7 | 87.4 | 73.6 | 84.2 | 82.6 | 67.5 | 85.3 | 25.6 | 79.6 |
| +8-Sub-Embedding | 8 | 4 | 88.1 | 73.1 | 83.1 | 83.1 | 67.9 | 86.4 | 20.1 | 76.5 |

Table 2: **Results of randomly allocating sub-embedding on GLUE.** We compare the performance of RoBERTa$_{\text{MEDIUM}}$ and randomly allocated $k$-sub-embeddings on GLUE benchmark.

[1] single-sentence tasks contain CoLA (Warstadt et al., 2018), SST-2 Socher et al. (2013), similarity and paraphrase tasks contain MRPC (Dolan & Brockett, 2005), STS-B (Cer et al., 2017), QQP (Shankar et al., 2017), and inference tasks contain MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), and RTE (Dagan et al., 2005)

### 4.3 ALLOCATING SUB-EMBEDDING FROM PRETRAINED KNOWLEDGE

Randomly scattered $k$-sub-embeddings compress successfully the original embeddings, but in some cases, the performance is struggled due to highly entangled sub-embeddings. Despite the tokens locate closely in the embedding space, randomly allocating algorithm cannot reflect the context between them. To solve this context mismatching problem, we extract pretrained RoBERTa (Liu et al., 2019) network from HuggingFace[2]. The 768-d embedding vectors of the pretrained network are clustered with Algorithm 2. We set the number of sub-embedding $M$ to 50, 100, and 200 to allocate 3-sub-embeddings. Table 3 shows that advanced results on GLUE benchmarks. The original k-means algorithm deals with the instances regardless of cluster size, i.e. the cluster size can be different across $k$ clusters. We perform both naive k-means and equally assigned k-means due to Algorithm 2. The results show that *sub-embeddings with equal cluster size* outperforms on small embedding set. Surprisingly, even clustered 3-sub-embedding has fewer parameters than 2-sub-embedding network, it outperforms every GLUE benchmark. The proposed method performs comparably with the original embedding structure, and outperforms on SST-2 benchmark.

| | $k$ | $\|\theta_{\textbf{emb}}\|(\downarrow\%)$ | **SST-2** | **MNLI** | **QNLI** | **QQP** | **RTE** | **MRPC** | **CoLA** | **STS-B** |
|---|---|---|---|---|---|---|---|---|---|---|
| RoBERTa$_{\text{MEDIUM}}$(ours) | 1 | 25.7M(-) | 89.9 | 79.6 | 88.2 | 86.6 | 72.9 | 88.4 | 38.1 | 88.1 |
| *randomly allocated sub-embedding* | | | | | | | | | | |
| 2-sub-embedding | 2 | 115k(99.5) | 88.6 | 74.3 | 84.0 | 84.0 | 66.8 | 88.1 | 35.7 | 79.3 |
| 3-sub-embedding | 3 | 18.9k(99.93) | 88.0 | 73.2 | 83.5 | 83.0 | 67.9 | 85.6 | 18.4 | 77.4 |
| *clustered sub-embeddings with k-means algorithm* | | | | | | | | | | |
| 3-sub-embedding, $M = 100$ | 3 | 104k(99.6) | 88.2 | 75.9 | 85.1 | 84.7 | 67.1 | 87.3 | 37.5 | 81.6 |
| 3-sub-embedding, $M = 200$ | 3 | 154k(99.4) | 90.0 | 77.6 | 85.5 | 85.6 | 69.7 | 88.7 | 34.9 | 84.5 |
| *clustered sub-embeddings with equal cluster size* | | | | | | | | | | |
| 3-sub-embedding, $M = 50$ | 3 | 25.6k(99.9) | 89.3 | 75.8 | 83.5 | 84.6 | 67.9 | 87.7 | 33.6 | 80.2 |
| 3-sub-embedding, $M = 100$ | 3 | 51.2k(99.8) | 89.3 | 77.2 | 85.8 | 84.8 | 70.8 | 87.4 | 36.8 | 84.9 |

Table 3: **Clustered $k$-sub-embedding Results on GLUE.** The networks of 3-sub-embedding are enhanced using the pretrained network where $M$ is the number of each sub-embedding.

### 4.4 TOKENS IN SUB-EMBEDDING SPACE

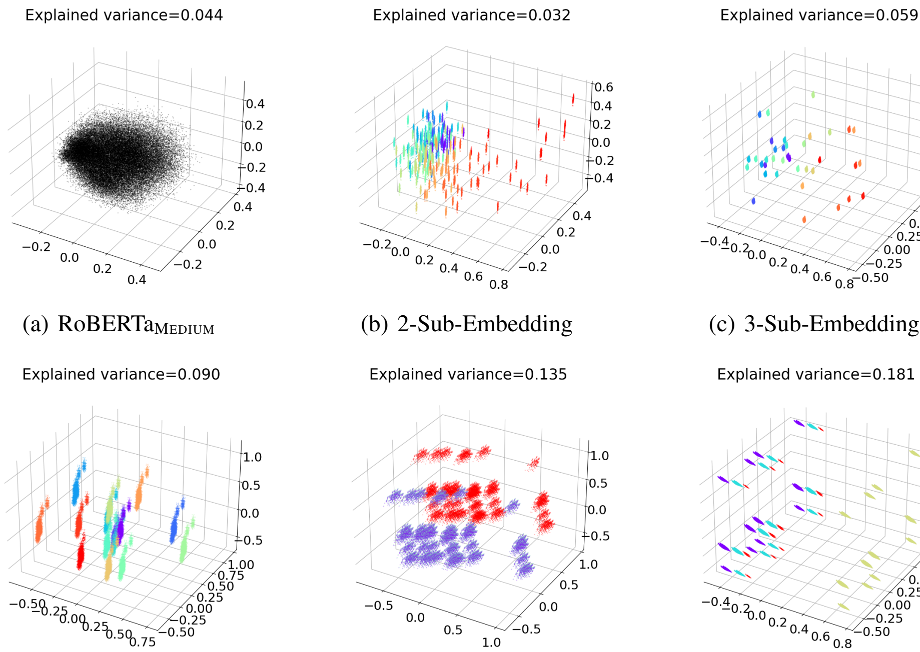
We also perform PCA to visualize $k$-sub-embeddings on the embedding space and derive the variance of the embeddings. The embedding weight $\boldsymbol{E} \in \mathbb{R}^{N\times d}$ would be transfered to $\boldsymbol{E}^p \in \mathbb{R}^{N\times p}$ where $p < N$ is the number of main components according to PCA procedure. The explained variance ratio can be defined with $r_p = \Sigma_{i=1}^{p}\sigma_i/\Sigma_{i=1}^{d}\sigma_i$ where $\sigma_i$ is $i$-th largest eigenvalue of $\boldsymbol{E}$'s covariance matrix. It is used to measure the variance of factorized embedding space. We visualize 3-d mapped $k$-sub-embedding vectors and corresponding explained variance ratio in Figure 2. The sub-embedding vectors are easily factorized due to sharing parameters, so the instances are located in each isolated subspace.

### 4.5 COMPARING INSIDE OF THE EMBEDDING AND THE SUB-EMBEDDING

We found that $k$-sub-embedding is able to be performed comparably on some benchmarks, but it is still ambiguous that how the combined sub-embedding works like the embeddings. Inspired by (Mimno & Thompson, 2017; Ethayarajh, 2019; Cai et al., 2021), we calculate the inter-similarity and intra-similarity among the embedding vectors. Cai et al. (2021) captures all hidden states between each layer of the neural language model to address inter-type and intra-type cosine similarities. Likewise for early studies, we define the inter-similarity $S_{inter}(l) = \mathbb{E}_{i\neq j}\text{cos}(\boldsymbol{h}_i^l, \boldsymbol{h}_j^l)$ and intra-similarity $S_{intra}(l) = \mathbb{E}_i\mathbb{E}_{j_1\neq j_2}\text{cos}(\boldsymbol{h}_{j_1}^l, \boldsymbol{h}_{j_2}^l)$ where $\boldsymbol{h}_*^l$ is one of the $l$-th hidden states. The cosine similarity indicates the isotropy of the embedding space. We report inter-similarity and intra-similarity of each layer in Figure 3.

The inter-similarities of each $k$-sub-embedding network are reported almost 1.0 at the end of the layer. This indicates that each hidden state of token is located in narrow cone in the latent space because the embeddings share their sub-embeddings.

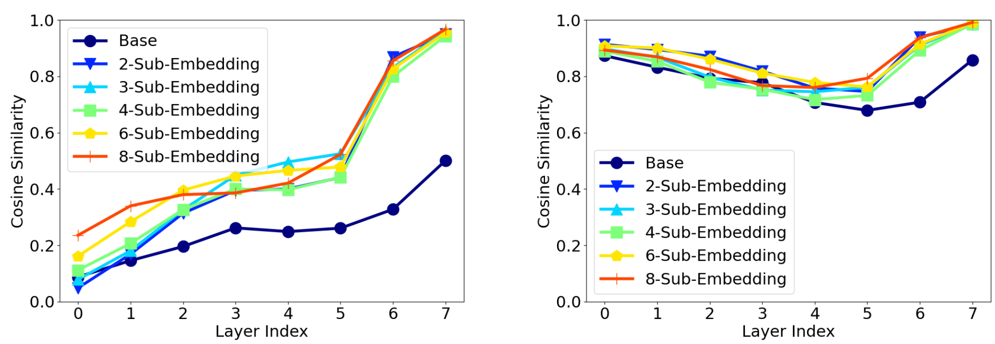
[2]https://huggingface.co/roberta-base




(a) $\text{RoBERTa}_{\text{MEDIUM}}$ (b) 2-Sub-Embedding (c) 3-Sub-Embedding

(d) 4-Sub-Embedding (e) 6-Sub-Embedding (f) 8-Sub-Embedding

Figure 2: **3-d scatter plots of each $k$-sub-embedding.** The embedding vectors are pointed in different colors depending on the last sub-embedding vector.



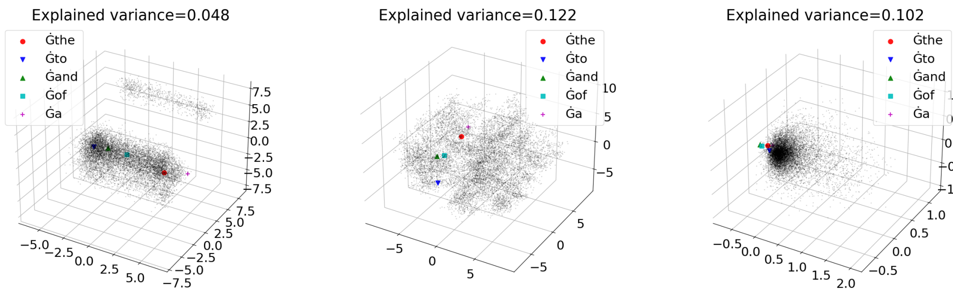

(a) $S_{inter}$ (b) $S_{intra}$

Figure 3: **Inter-similarity and intra-similarity of each hidden state.** $S_{inter}$ and $S_{intra}$ of each $k$-sub-embedding goes almost 1.0 according to highly correlated embeddings.

Figure 4 shows more details of hidden states in each layer. We extract some commonly used tokens(e.g., $\langle the \rangle$, $\langle to \rangle$, $\langle and \rangle$, $\langle of \rangle$, and $\langle a \rangle$) to figure out where they locate in the hidden spaces. The embedding vectors tend to be clustered each other due to the shared sub-embedding as shown in Figure 2. Figure 4(a), 4(b) show that the networks try to distinguish the tokens by spreading out the tokens into the hidden spaces. The last hidden states congregate the tokens for the decoder of the language models.

### 4.6 Experiments on Multilingual Dataset

Training multilingual language models treats as much harder than monolingual case. Some multilingual language models (Lample et al., 2018; Lample & Conneau, 2019) focus on cross-linguity to train the pairs of translation dataset. XLM-R (Conneau et al., 2020) modifies RoBERTa to deal with multilingual embeddings, that leverages on extremely large dataset. We adapt the XLM-R network




(a) Layer 0 (b) Layer 2 (c) Layer 7

Figure 4: **3-d scatter plot of each layer in 3-sub-embedding network.** The hidden states of each token are applied PCA to draw in 3-d space. The tokens gather around to narrow cone at the last layer.

to XLM-R$_{\text{MEDIUM}}$ which is described in Table 1.
Following (Huang et al., 2019), We evaluate the multilingual networks in two ways; cross-lingual transfer and multilingual transfer(translate-train-all). Table 4 shows the accuracy on each of the 15 XNLI languages. The cross-lingual transfer results of Arabic and Urdu are underperformed due to lack of pretraining corpora. While randomly allocated sub-embedding cut down the number of embedding over 99.97%, the performance of each language is also diminished. We allocate sub-embeddings twice as much than the previous case to adjust clustering algorithm using the pretrained network[3]. The results on XNLI are improved 2%p on multilingual transfer task, while the embeddings are still compressed over 99.95%.

| **Model** | $E$ | **en** | **fr** | **es** | **de** | **el** | **bg** | **ru** | **tr** | **ar** | **vi** | **th** | **zh** | **hi** | **sw** | **ur** | **Avg** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Fine-tune multilingual model on English training set (Cross-lingual Transfer)* | | | | | | | | | | | | | | | | | |
| XLM-R$_{\text{MEDIUM}}$(ours) | 250k | 74.0 | 56.6 | 60.5 | 55.1 | 49.4 | 53.5 | 52.0 | 45.0 | 35.1 | 54.5 | 41.5 | 52.4 | 42.0 | 45.1 | 43.3 | 50.7 |
| 3-Sub-Embedding | 63 | 72.6 | 55.4 | 54.7 | 48.4 | 45.0 | 51.0 | 49.8 | 43.4 | 35.1 | 50.8 | 36.4 | 44.2 | 40.2 | 45.9 | 43.2 | 47.7 |
| +clustered | 128 | 72.9 | 55.4 | 54.1 | 51.9 | 49.8 | 51.9 | 51.1 | 45.5 | 35.0 | 49.7 | 44.3 | 45.0 | 42.5 | 44.3 | 40.6 | 48.9 |
| *Fine-tune multilingual model on all training sets (TRANSLATE-TRAIN-ALL)* | | | | | | | | | | | | | | | | | |
| XLM-R$_{\text{MEDIUM}}$(ours) | 250k | 77.0 | 72.4 | 74.3 | 73.0 | 71.7 | 70.9 | 69.0 | 69.0 | 58.7 | 72.3 | 61.2 | 63.8 | 61.0 | 63.5 | 62.8 | 68.0 |
| 3-Sub-Embedding | 63 | 72.7 | 69.6 | 68.6 | 68.1 | 67.0 | 65.1 | 65.6 | 63.2 | 59.7 | 69.6 | 60.0 | 58.3 | 61.7 | 61.9 | 58.2 | 64.6 |
| +clustered | 128 | 74.7 | 69.9 | 71.5 | 69.6 | 68.3 | 67.6 | 67.6 | 66.8 | 60.8 | 71.7 | 62.1 | 62.0 | 63.3 | 62.4 | 60.4 | 66.6 |

Table 4: **Results on cross-lingual classification.** We evaluate base network and $k$-sub-embedding that is fixed to $k = 3$. $E$ denotes the number of embeddings in each network.

# 5 Conclusions and Future Work

In this work, we introduce the $k$-sub-embedding to replace the original embedding. We suggest two ways to allocate shared sub-embedding to the embedding vector; one is to assign sequentially by modulo operation(Algorithm 1), and the other is allocating scattered sub-embedding using contextual information from pretrained network(Algorithm 2). The number of parameters of the scattered sub-embedding is reduced over 99%, because the combined embedding can be generated exponentially as a result of Cartesian product. Although the embeddings are highly compressed, the results on GLUE, XNLI show that $k$-sub-embedding structure also performs similar with the base result.
We conducted the experiments to replace the embeddings to $k$-sub-embedding through MLMs. Other language modelings such as TLM and CLM can be also trained with the replaced embeddings. Although TLM has additional decoder to translate the source language, we expect that the contextual embedding replaced to $k$-sub-embedding would be outperformed according to cross-lingual test results. We can further investigate about the relationship between the embedding dimension $d$ and the number of sub-embedding space $k$. This will help to theoretically understand how the $k$-sub-embedding works.

[3] https://huggingface.co/xlm-roberta-base

### Reproducibility

The embedding part of the language model is usually defined in *Embedding* layer. We design our $k$-sub-embedding layer to be easily replaced with the predefiend embedding layer. The implementations of $k$-sub-embedding are written in PyTorch framework, and it is also compatible with HuggingFace framework. Source code is accessible to the supplementary material.

## References


David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007.

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

Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*, 2021.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017.

Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. Rethinking embedding coupling in pre-trained language models. In *International Conference on Learning Representations*, 2021.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation, 2021.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pp. 177–190. Springer, 2005.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings, 2019.

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining, 2019.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *International Conference on Learning Representations*, 2018.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

David M. Mimno and Laure Thompson. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

Iyer Shankar, Dandekar Nikhil, and Csernai Kornl. First quora dataset release: Question pairs, 2017. URL `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. Charformer: Fast character transformers via gradient-based subword tokenization, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments, 2018.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models, 2021.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, 2019.

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

# A Appendix

## A.1 Pretraining Monolingual Language Models

We adapt RoBERTa case for training standard English corpora. The structure of RoBERTa$_{\text{BASE}}$ is based on BERT$_{\text{BASE}}$ which has 12 layers of transformer encoder. We cut down some layers to build 8 layers, the details of other hyperparameters are described in Table 5. The $k$-sub-embedding networks are also trained with the same environment of RoBERTa$_{\text{MEDIUM}}$, and we accelerate training speed through half-precision floating point.
The fine-tuning tasks are trained with the hyperparameters that are used in pretraining. Some values are modified to fit in GLUE benchmark as Table 6. We conduct the fine-tuning tasks and report the best accuracy among the candidates of hyperparameters. We also fine-tune RTE, STS-B, and MRPC tasks from the network checkpoint of MNLI downstream task.

## A.2 Pretraining Multilingual Language Models

As we deal with the multilingual corpus with XLM-R framework, the pretraining procedure of multilingual case is not different from the monolingual case. To cover the tokens of multiple languages, XLM-R expands the vocabularies to 250k. We also adapt these tokenization method and other hyperparameters as described in Table 5. The hidden size and layers are reduced to 512 and 8 likewise for RoBERTa$_{\text{MEDIUM}}$. We extract corpus of 15 languages from CommomCrawl. The details of each monolingual corpus are described in Table 7. Unlike the MNLI benchmark, we fine-tune XNLI task on fixed learning rate and batch size(Table 6).

## A.3 Layerwise Analysis

In this section we analyze each hidden state of $k$-sub-embedding. The embeddings that are composed with the sub-embeddings tend to be highly correlated. The language models learn to distinguish the entangled embeddings. Figure 5, 6, 7, 8, 9, and 10 show that the tokens are spread out through the hidden space. The tokens in the last layer are located closely each other.

| Hyperparameters | RoBERTa$_{\text{MEDIUM}}$ | XLM-R$_{\text{MEDIUM}}$ |
|---|---|---|
| Number of layers | 8 | 8 |
| Hidden size | 512 | 512 |
| Intermediate hidden size | 2048 | 2048 |
| Attention heads | 8 | 8 |
| Dropout | 0.1 | 0.1 |
| Attention Dropout | 0.1 | 0.1 |
| Warmup Steps | 24k | 24k |
| Peak learning rate | 2e-4 | 2e-4 |
| Batch Size | 128 | 256 |
| Weight Decay | 0.01 | 0.01 |
| Max Steps | 250k | 250k |
| Learning Rate Decay | Linear | Linear |
| Adam $\epsilon$ | 1e-6 | 1e-6 |
| Adam $\beta_1$ | 0.9 | 0.9 |
| Adam $\beta_2$ | 0.98 | 0.98 |
| Gradient Clipping | 0.0 | 0.0 |
| Vocabularies | 50267 | 250002 |
| Max Sequence Length | 512 | 512 |

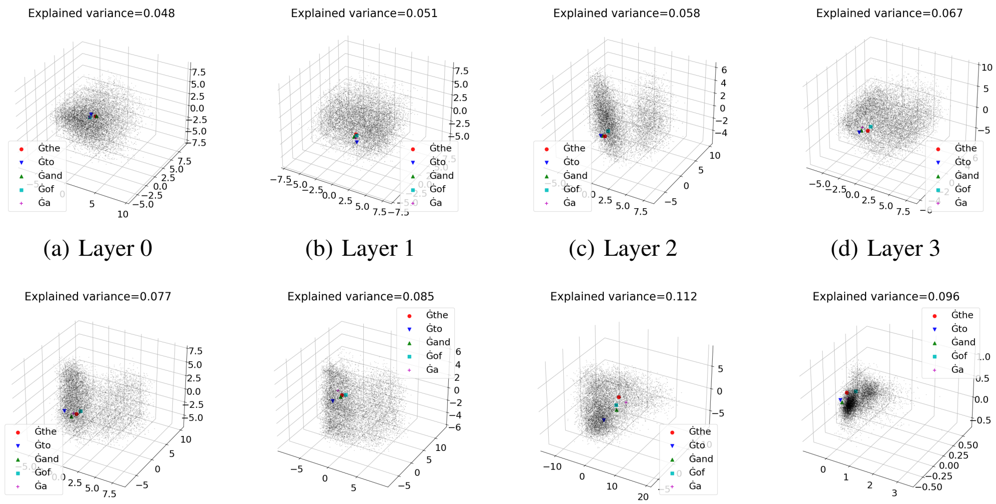
Table 5: **Hyperparameters of RoBERTa$_{\text{MEDIUM}}$ and XLM-R$_{\text{MEDIUM}}$.** The details of pretraining models. Most of the hyperparameters adapt from RoBERTa.

| Hyperparameters | GLUE | XNLI |
|---|---|---|
| Learning rate | 1e-5, 2e-5, 3e-5 | 2e-5 |
| Batch Size | 16, 32 | 32 |
| Max Sequence Length | 128 | 128 |
| Learning Rate Decay | Linear | Linear |
| Warmup Ratio | 0.06 | 0.06 |
| Max Epochs | 10 | 5, 10 |

Table 6: **Hyperparameters to evaluate on GLUE and XNLI.** The details of fine-tuning hyperparameters.

| ISO code | Language | Tokens(M) | Size(GiB) |
|---|---|---|---|
| **ar** | Arabic | 181 | 1.0 |
| **bg** | Bulgarian | 193 | 1.2 |
| **de** | German | 338 | 1.4 |
| **el** | Greek | 170 | 1.0 |
| **en** | English | 802 | 3.1 |
| **es** | Spanish | 273 | 1.1 |
| **fr** | French | 315 | 1.2 |
| **hi** | Hindi | 91 | 0.7 |
| **ru** | Russian | 426 | 2.8 |
| **sw** | Swahili | 232 | 0.8 |
| **th** | Thai | 125 | 1.1 |
| **tr** | Turkish | 178 | 0.7 |
| **ur** | Urdu | 101 | 0.6 |
| **vi** | Vietnamese | 323 | 1.4 |
| **zh** | Chinese | 237 | 1.0 |

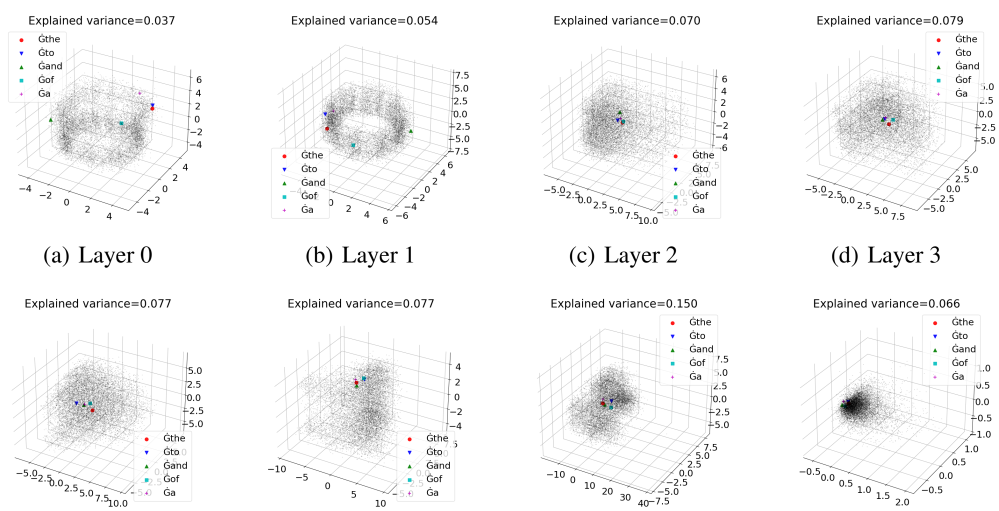
Table 7: **Statistics of each monolingual corpus.** We extract 15 languages from CommonCrawl.




(a) Layer 0 (b) Layer 1 (c) Layer 2 (d) Layer 3

(e) Layer 4 (f) Layer 5 (g) Layer 6 (h) Layer 7

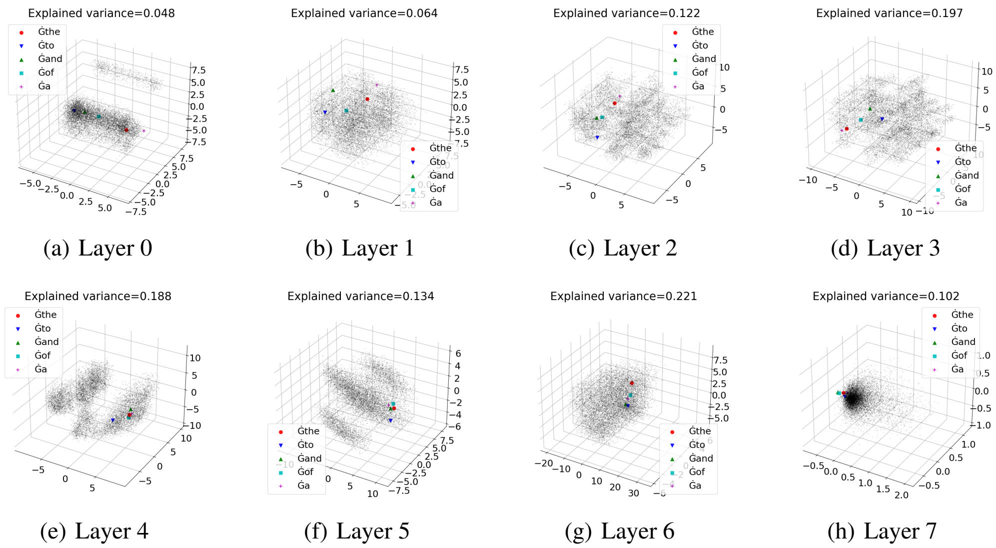
Figure 5: **3-d scatter plot of each layer in RoBERTa$_{\text{MEDIUM}}$(ours).**




(a) Layer 0 (b) Layer 1 (c) Layer 2 (d) Layer 3

(e) Layer 4 (f) Layer 5 (g) Layer 6 (h) Layer 7

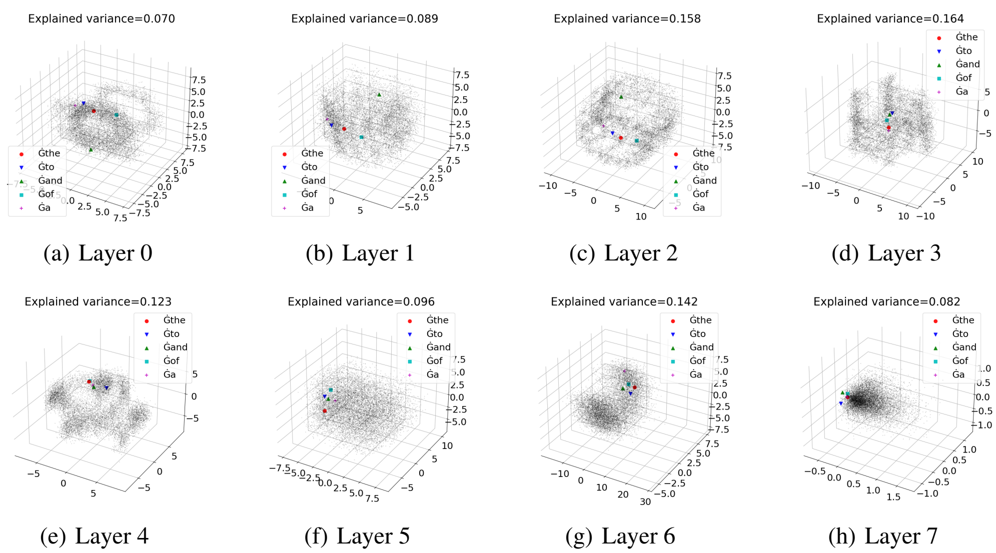
Figure 6: **3-d scatter plot of each layer in 2-sub-embedding network.**




(a) Layer 0 (b) Layer 1 (c) Layer 2 (d) Layer 3

(e) Layer 4 (f) Layer 5 (g) Layer 6 (h) Layer 7

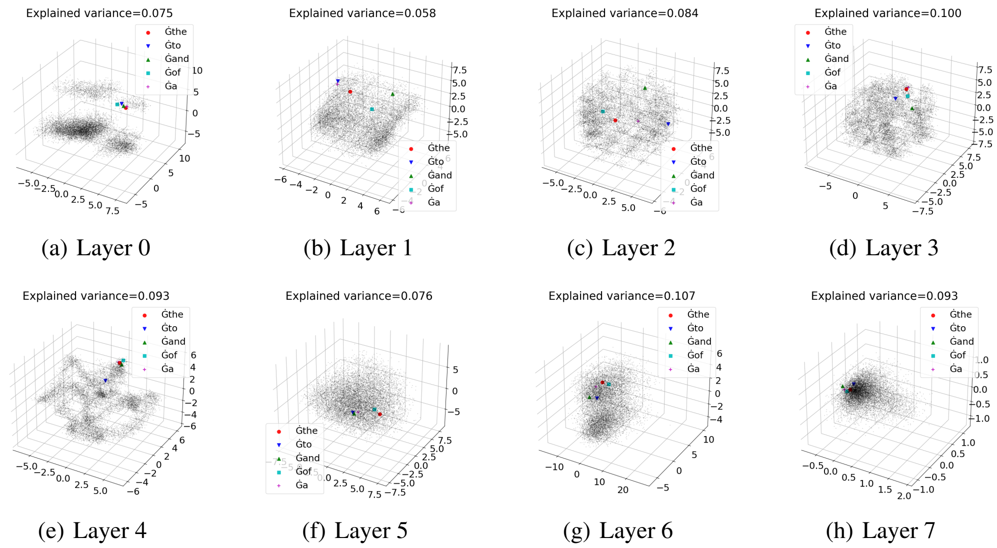
Figure 7: **3-d scatter plot of each layer in 3-sub-embedding network.**




(a) Layer 0 (b) Layer 1 (c) Layer 2 (d) Layer 3

(e) Layer 4 (f) Layer 5 (g) Layer 6 (h) Layer 7

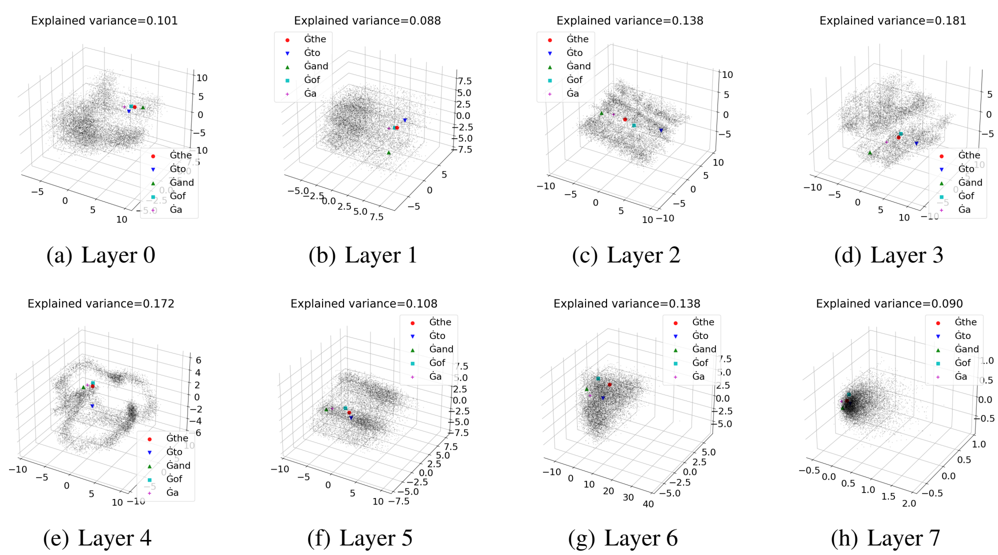
Figure 8: **3-d scatter plot of each layer in 4-sub-embedding network.**




(a) Layer 0 (b) Layer 1 (c) Layer 2 (d) Layer 3

(e) Layer 4 (f) Layer 5 (g) Layer 6 (h) Layer 7

Figure 9: **3-d scatter plot of each layer in 6-sub-embedding network.**




(a) Layer 0 (b) Layer 1 (c) Layer 2 (d) Layer 3

(e) Layer 4 (f) Layer 5 (g) Layer 6 (h) Layer 7

Figure 10: **3-d scatter plot of each layer in 8-sub-embedding network.**