UnClE: Explicitly Leveraging Semantic Similarity to Reduce the Parameters of Word Embeddings

Zhi Li[1], Yuchen Zhai[1], Chengyu Wang[2] , Minghui Qiu[2], Kailiang Li[1], Yin Zhang[1]*
[1]College of Computer Science and Technology, Zhejiang University, China
[2]Alibaba Group
{zhili,zhaiyuchen,likailiang,zhangyin98}@zju.edu.cn
{chengyu.wcy,minghui.qmh}@alibaba-inc.com

Abstract

Natural language processing (NLP) models often require a massive number of parameters for word embeddings, which limits their application on mobile devices. Researchers have employed many approaches, e.g. adaptive inputs, to reduce the parameters of word embeddings. However, existing methods rarely pay attention to semantic information. In this paper, we propose a novel method called **Un**ique and **Cl**ass **E**mbeddings (UnClE), which explicitly leverages semantic similarity with weight sharing to reduce the dimensionality of word embeddings. Inspired by the fact that words with similar semantic can share a part of weights, we divide the embeddings of words into two parts: unique embedding and class embedding. The former is one-to-one mapping like traditional embedding, while the latter is many-to-one mapping and learn the representation of class information. Our method is suitable for both word-level and sub-word level models and can be used to reduce both input and output embeddings. Experimental results on the standard WMT 2014 English-German dataset show that our method is able to reduce the parameters of word embeddings by more than 11x, with about 93% performance retaining in BLEU metrics. For language modeling task, our model can reduce word embeddings by 6x or 11x on PTB/WT2 dataset at the cost of a certain degree of performance degradation.

1 Introduction

Recently, deep learning models like LSTM networks (Hochreiter and Schmidhuber, 1997), and Transformer (Vaswani et al., 2017) based models like BERT (Devlin et al., 2019) have made remarkable progress in the field of natural language processing (NLP). However, the sizes of these models are usually too humongous, making them difficult to deploy on low-resource machines such as edge-computing devices, sensors or mobile phones.

Method	Adaptive	Weight Sharing	Semantic
Baevski and Auli (2019)	✓		
Grave et al. (2017)	✓		
Li et al. (2016)		✓	
Li et al. (2018)		✓	
Our Method		✓	✓

Table 1: Our method differs from previous methods in that it takes advantage of inter-word semantic similarity to reduce vanilla word embeddings significantly.

For a typical neural language model, especially one with a large vocabulary size, the large memory consumption of the model is mostly due to the need of storing the input and output word embedding matrices. The dimension of recurrent layers (e.g., LSTM), which corresponds to the hidden state, is typically small and independent of the vocabulary size. In contrast, the dimensions of the embedding and softmax layers grow with the size of vocabulary, which can easily reach the scale of hundreds of thousands. As a result, parameter matrices of the embedding and softmax layers are often responsible for the majority of memory consumption of a neural language model. For Transformer-based models like Universal Transformer (Dehghani et al., 2019) and Albert(Lan et al., 2020), which share parameters across layers, word embeddings also consume a large amount of memory.

Researchers have sought to reduce word embeddings through many efforts, such as, adaptive method (Grave et al., 2017; Baevski and Auli, 2019) or designing mechanisms for weight sharing (Li et al., 2016, 2018). However, previous methods rarely take semantic information into consideration. Some words are very similar regarding the semantics. For example, "better" and "best" are similar, except that the former is comparative level and the latter is superior level. Also, their suffixes "er" and "est", as subwords, have similar semantics to some extent. We hold the same assumption as Chen et al. (2016) and Shu and Nakayama (2018) that learning independent embeddings of large dimensions

*Corresponding Author: Yin Zhang.

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1815–1828
November 7–11, 2021.

causes more redundancy in the embedding vectors, as the inter-similarity among words is ignored.

In this paper, we propose **UnClE**, an intuitive method for reducing the dimensionality of word embeddings by explicitly leveraging semantic similarity with weight sharing. We divide traditional word embedding processes into two parts: *unique embedding* and *class embedding*. There is a one-to-one mapping between words and unique embeddings like traditional word embedding. For class embeddings, words in the same class will share a single class embedding. Before being fed to the downstream neural network, each word will get its unique embedding and class embedding through its word index and class index. Then, the two embeddings are concatenated to generate the final embedding of the word.

Our contributions are mainly three-fold:

- We propose a novel method to reduce the parameters of word embeddings. We first construct semantic classes and group words into different classes so that the words in the same class can share a part of weights. Since the class size is much smaller than the vocabulary size and dimensions of share part can range from 0 to the size of whole word embedding dimensions, the method theoretically can achieve a big reduction ratio .

- Our method can be used to reduce both input and output embeddings. Meanwhile, our method can share input and output embedding. And our method is suitable to both word-level and subword-level models.

- Experimental results on the standard WMT 2014 English-German dataset show that our proposed approach can achieve 11x or more reduction in the number of both input and output embedding parameters while keeping about 93% (25.92/27.82) BLEU score performance. For language modeling task, our model can reduce the parameters of word embeddings by 6x or 11x on PTB/WT2 dataset at the cost of a certain degree of performance degradation. Our method also outperforms the projective and adaptive methods. Ablation experiments further prove that our method indeed can leverage semantic similarity to reduce redundancy of word embeddings.

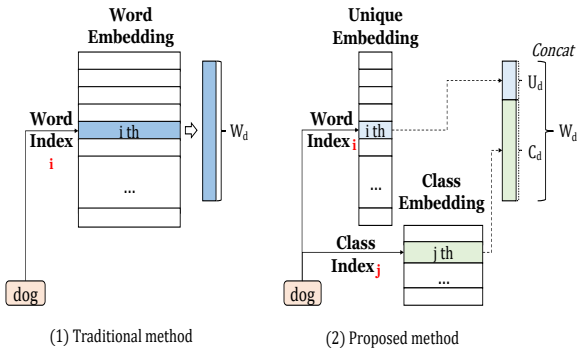

Figure 1: Traditional model vs our proposed model.

2 Related Works

Adaptive Method There are many efforts towards reducing word embeddings. Baevski and Auli (2019) extend the adaptive softmax (Grave et al., 2017) to input word representations and assign more capacity to frequent words and reduce the capacity for less frequent words with the benefit of reducing overfitting to rare words. Mehta et al. (2020) point that projective embedding, including Transformer-xl (Dai et al., 2019) and Albert (Lan et al., 2020) is a special case of adaptive method when the number of clusters is one. Similarly, Chen et al. (2018) group words into blocks based on their frequencies, and then refine the clustering iteratively by constructing the weighted low-rank approximation for each block. Other works like (Goodman, 2001; Morin and Bengio, 2005; Mnih and Hinton, 2008) also propose hierarchical clustering of words, but they mainly focus on training effectively rather than reducing word embeddings.

Weight Sharing Another line of work employs weight sharing to reduce the number of parameters. Li et al. (2016) allocate all the words in the vocabulary into a table. The words in the same row share the row vector and the words in the same column share the column vector. Slim embedding (Li et al., 2018) randomly shares the structured parameters at both the input and output embedding layers of the recurrent neural language models. Press and Wolf (2017) and Inan et al. (2017) share input and output embeddings to improve language model while significantly reducing the number of network parameters. Mehta et al. (2020) use a deeper network with significantly fewer parameters to replace word embedding layer. Zhao et al. (2019) introduce a novel knowledge distillation technique for training a student model with a significantly smaller vocab-

ulary by mixing teacher vocabulary-tokenized with student vocabulary-tokenized words.

Different from the above researches, our approach leverages semantic similarity to reduce word embeddings. Words with similar semantic will be grouped into the same class, and share a part of weights.

Inter-similarity among Words Similar to our method, several earlier studies (Chen et al., 2016; Shu and Nakayama, 2018; Kim et al., 2020; Tissier et al., 2019) are also based on the hypothesis that the inter-similarity among words is ignored in the conventional way of constructing word vectors. Chen et al. (2016) represent infrequent words' embeddings with frequent words' embeddings by sparse linear combinations. Our method is different in that words in the same class share a part of weights, no matter their frequency. Of course, in the same class, there are also infrequent words and frequent words. The training of both frequent and infrequent words will update shared weights. To some extent, frequent words will help infrequent words via shared weights.

Shu and Nakayama (2018) also construct the embeddings with a few basis vectors. For each word, the composition of basis vectors is determined by a hash code rather than precomputation. However, this method assigns the same length of codes to each word without considering the significance of downstream tasks. Kim et al. (2020) further compress word embeddings by adaptively assigning different lengths of codes to each word by considering downstream tasks.Tissier et al. (2019) employ an autoencoder architecture to transform real-valued embeddings into binary embeddings while preserving semantic information. Different from these works above that implicitly leverage semantic similarity, our method explicitly construct semantic classes.

More importantly, Chen et al. (2016); Shu and Nakayama (2018) focus on compressing the pre-trained embeddings and our method is a novel representation of word embeddings. The reduction of parameters comes from the smaller size of dimensionality of unique embeddings rather than from compressing pre-trained embeddings. And our method also don't need to modify the objective function to learn the sparse linear combinations or the code-book. So our method is easier to implement and have a wider range of applications.

3 Unique and Class Embedding

3.1 Constructing Semantic Classes

When we consider constructing semantic classes, WordNet (Miller, 1995) is an optional choice. It contains many nouns, verbs, adjectives, and adverbs that are grouped into sets of cognitive synonyms (synsets). However, tokens in datasets are not always available in WordNet such as punctuations, numbers and subwords. Inspired by (Chen et al., 2016; Inan et al., 2017), word vector (Mikolov et al., 2013; Pennington et al., 2014) provides an alternative to WordNet. In language modeling, there is a well established metric space for the outputs (words in the language) based on word embeddings, with meaningful distances between words. Yaghoobzadeh et al. (2019) also show that semantic classes are recognizable in embedding space. So we use the knowledge of word vectors to measure semantic similarity approximately. After we get the well-trained word vectors, a clustering algorithm will be employed to group words into different classes.

3.2 Method Formulation

Traditional Method Suppose x_i is the i-th word in the vocabulary and $i \in [1, 2, ..., V_s]$, then $W = (W_1, W_2, ..., W_{V_s}) \in \mathbb{R}^{W_d \times V_s}$ is the embedding matrix of the traditional method, where $W_i \in \mathbb{R}^{W_d}$ is the word embedding of x_i, W_d represents the dimension of word embedding and V_s denotes the size of vocabulary.

Our Method $U = (U_1, U_2, ..., U_{V_s}) \in \mathbb{R}^{U_d \times V_s}$ is a unique word embedding matrix, U_i represents the unique word embedding of x_i. We use $C_j, j \in [1, 2, ..., C_s]$ to denote the class embedding of x_i. And $C = (C_1, C_2, ...C_{C_s}) \in \mathbb{R}^{C_d \times C_s}$ is the class embedding matrix. The whole word embedding $W_i^{'}$ of x_i is obtained by concatenating the unique word embedding U_i and the corresponding class embedding C_j.

$$W_i^{'} = concatenate(U_i, C_j) \quad (1)$$

Our method can also share input and output embeddings (Press and Wolf, 2017; Inan et al., 2017). As stated above, input embedding contains unique embedding and class embedding. And output embedding actually is a linear projection matrix added after the output of decoder. In our method, this matrix consists of concatenated embedding (including unique and class embedding) of each word in

vocabulary. We first feed all vocabulary words to input embedding of our method in order and get an embedding matrix $W \in \mathbb{R}^{W_d \times V_s}$. Then we transpose it and get $W^T \in \mathbb{R}^{V_s \times W_d}$. At last, we replace the traditional output embedding with W^T.

When training, the parameters of unique embedding will be optimized when the corresponding word is trained. The parameters of class embedding will be optimized when any word in the class is trained. Except for this, everything else of training is consistent with the base model. The gradients of unique embedding and class embedding come from the concatenated embedding and the gradients of concatenated embedding come from input embedding and output embedding.

In fact, our method can be extended to more fine-grained and hierarchical approach. Each class can be divided into multiple sub-classes. For example, class C_j have two subclass C_{j1} and C_{j2}. Then for x_i belongs to C_j and C_{j1}, Equation (1) can be rewritten as:

$$W_i^{'} = concatenate(U_i, C_j, C_{j1}) \tag{2}$$

In this paper, we just discuss the solution of Equation (1) to validate the effectiveness of our method. We leave exploring the solution of Equation (2) as future work.

3.3 Reduction Ratio

Given that the vocabulary size is V_s, we use W_d to represent the dimension of traditional word embedding. When we use the reduction method, the class size is denoted by C_s, and U_d, C_d represents the dimension of unique embedding and class embedding, respectively.

$$\textit{Reduction Ratio} = \frac{V_s * W_d}{C_s * C_d + V_s * U_d} \tag{3}$$

When we set $C_d = 0$, $U_d = W_d$, *Reduction Ratio* will be equal to 1, which means it turns into the traditional word embedding. Further, we define

$$\textit{Pseudo Reduction Ratio} = \frac{W_d}{U_d} \tag{4}$$

In practice, if we set the number of classes 10x less than the number of words, we have $V_s * U_d \gg C_s * C_d$ and then

$$RR \approx PRR \tag{5}$$

where RR and PRR represent *Reduction Ratio* and *Pseudo Reduction Ratio* respectively.

4 Experiments

We demonstrate the performance of our method on two sequence-to-sequence modeling tasks: language modeling and machine translation. For language modeling, we compare our method with AWD-LSTM (Merity et al., 2018). For machine translation, we use Transformer (Vaswani et al., 2017) as base model. We also compare our method with projective method and adaptive method. Following the same setting as in (Merity et al., 2018; Vaswani et al., 2017), we share input and output embeddings for all models in this paper.

Task	DataSet	Type	Vocab.
Language Model	WikiText-2	Word	33278
	PTB	Word	10000
MT	WMT14 en-de	SubWord	40724

Table 2: Statistics of Datasets. MT stands for machine translation.

4.1 Dataset and Hyperparameter Setting

Language Model. To evaluate the impact of our method, we perform language modeling over the preprocessed versions of the Penn Treebank (PTB) (Mikolov et al., 2010) and the WikiText-2 (WT2) dataset (Merity et al., 2017). There is a detailed introduction to these two datasets in the appendix. We use the same hyper-parameters and PyTorch version as the original AWD-LSTM. For all language modeling experiments, each model is trained with one 1080Ti GPU. We use perplexity (PPL) as the measure to evaluate the performance of models (the lower, the better).

Machine Translation. We further evaluate our approach on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Following Vaswani et al. (2017), we encode sentences using byte-pair encoding (Britz et al., 2017) and get a shared source-target vocabulary of about 40000 tokens.The input embedding is shared by encoder and decoder. We use newstest2014 and newstest2017 as validation and test sets, respectively. BLEU is employed to evaluate the performance of models. The higher the score, the better the model. Our implementation is based on Ott et al. (2019). For all models in machine translation without additional declaration, we use the same parameter settings described as follows.

Model	PRR	UniqueDim	ClassDim	EmbedParams(10^6)	RR	BLEU
Transformer(Vaswani et al., 2017)	1	512	0	20.9	1.00	27.30
Transformer(Our Implementation)	1	512	0	20.9	1.00	**27.82**
UnClE(Our Method)	2	256	256	10.7	1.95	27.73
	4	128	384	5.59	3.73	27.35
	8	64	448	3.05	6.83	26.83
	16	32	480	1.78	11.69	25.92

Table 3: The effect of reduction ratio on Tranformer for machine transaltion. RR, PRR represent *Reduction Ratio* and *Pseudo Reduction Ratio* of embeddings respectively. *EmbedParams* represents the number of parameters of embedding layer. All results shown in the paper are averaged over three runs with different random seeds without additional declaration.

Parameter Settings. We use a machine with 8 NVIDIA V100 GPUs. Each GPU has up to 4096 tokens. In practice, a training batch contains a set of sentence pairs, which have approximately 30000 source tokens and 30000 target tokens. We train all models for a total of 100,000 steps and use the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We also employ a warmup mechanism and increase the learning rate linearly for the first 4000 training steps, and decrease it thereafter proportionally to the inverse square root of the step number. We apply dropout=0.1 (Srivastava et al., 2014) to the output of each sub-layer before it is added to the sub-layer input and normalize the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. We set label smoothing (Szegedy et al., 2016) value $\epsilon_{ls} = 0.1$.

4.2 Constructing Semantic Classes

After preprocessing, we get the word-level (PTB, WT2) and subword-level (WMT 2014 English-German dataset) datasets. Then a word representation model (SkipGram) (Mikolov et al., 2013) will be trained for each dataset. The training process is much faster than language modeling and machine translation. For example, the training of word vector on PTB dataset takes several minutes while its experiment of language modeling needs a couple of hours even without taking finetuning into consideration. Through these models, we obtain a set of vectors for each word in the vocabularies. These word vectors contain meaningful distances between words. Then we explicitly construct semantic classes via a clustering algorithm. In our experiments, k-means (MacQueen et al., 1967) implementation in NLTK (Xue, 2011) is employed. After clustering, each word gets its class index.

Figure 2 shows the histogram of the number of words in different classes on the standard WMT 2014 English-German dataset. We set class size to 1000. It's not too difficult to spot that the distribution is unbalanced and similar to long-tailed distribution if we reorder the class index (abscissa) according to the number of words in each class (ordinate). This way of constructing semantic classes may not be subtle and exact, but an approximate distribution is enough to test our hypothesis. It should be noted that only class index will be used in downstream tasks and we do not use the trained vanilla word vector to initialize word embeddings of Transformer-based or LSTM-based models.

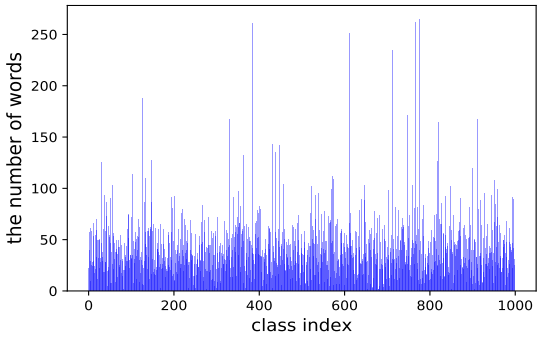

Figure 2: The histogram of the number of words in different classes on the standard WMT 2014 English-German dataset.

4.3 Effect of Reduction Ratio on Model Capability

In this section, we study the effects of different reduction ratios on model capability. Following Vaswani et al. (2017), we set the word embedding dimension W_d to 512 for the baseline model on machine translation. In our method, dimension of whole word embedding $W_d^{'} = U_d + C_d = 512$ is consistent with word embedding W_d of baseline model. The unique dimension U_d is initially set to

DataSet	PRR	UniqueDim	ClassDim	EmbedParams(10^6)	RR	PPL(dev)	PPL(test)
PTB	1	400	0	4.0	1.00	**61.51**	**59.08**
PTB	2	200	200	2.20	1.82	**63.50**	**60.90**
PTB	4	100	300	1.30	3.08	66.78	64.16
PTB	8	50	350	0.85	4.71	71.33	68.75
PTB	16	25	375	0.625	6.40	78.30	75.58
WT2	1	400	0	13.31	1.00	**68.70**	**65.60**
WT2	2	200	200	6.86	1.94	**71.40**	**68.07**
WT2	4	100	300	3.63	3.67	76.82	72.73
WT2	8	50	350	2.01	6.61	85.27	80.54
WT2	16	25	375	1.21	11.03	98.26	92.48

Table 4: Effect of Reduction Ratio on LSTM(PTB/WT2) for language modeling. RR, PRR represent *Reduction Ratio* and *Pseudo Reduction Ratio* of embeddings respectively. *EmbedParams* represents the number of parameters of embedding layer.

512 and then divided by the power of 2. We have

$$U_d^i = \frac{512}{2^i} \tag{6}$$

$$C_d^i = 512 - U_d^i \tag{7}$$

where i represents the i-th model and ranges from 0 to 4 for machine translation. For language modeling, the settings are the same except that the word embedding $W_d = 400$ and i ranges from 0 to 4.

The corresponding reduction ratio is calculated according to Equation (3). To compare the results fairly, we set the class size to 1000 for all models in this section. Refer to Appendix for the choice of class size. Moreover, in Section 4.5, we will further study the impacts of different settings of class size.

Table 3 shows the BLEU scores of models. It is worth noting that when we set the dimension of unique embedding to 256 or 128, the BLEU score just drops within 0.5. When $U_d = 32, RR = 11.69$, the model still achieves about 93% performance. In appendix, we sample some specific cases to study the model performance. Table 4 shows the PPL scores of LSTM models and indicates that our method is also suitable for LSTM models.

Table 5: Fine-tuning, continuous cache pointer on LSTM for language modeling.

Model	PRR	RR	PPL
Baseline	1	1.00	59.08
Baseline + ft	1	1.00	56.52
UnClE + ft	2	1.95	56.60
Baseline + pt	1	1.00	52.60
UnClE + pt	2	1.95	52.63

Table 5 shows the results of our proposed model with fine-tuning and continuous cache pointer (Merity et al., 2018) on PTB dataset. We choose $RR = 1$ model in Table 4 as baseline. For our method, we choose the setting: $U_d = 200, C_d = 200, RR = 1.95$. The results indicate that these two important strategies can also be employed in our method.

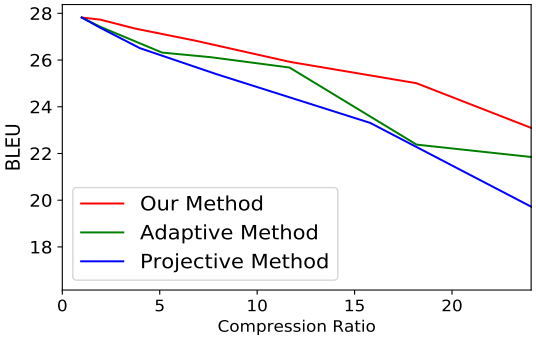

Figure 3: The BLEU scores of projective method, adaptive method and our method (the higher, the better).

4.4 Comparison with Related Works

Tradeoff between the model performance and the number of word embedding parameters is studied in section 4.3. In order to further validate the effectiveness of our method, we compare our method with the projective method and adaptive method on WMT 2014 English-German dataset.

Projective Method The projective embedding method factorizes the embedding matrix $W \in \mathbb{R}^{V_s \times W_d}$ into $W_f \times L_{in}$. $W_f \in \mathbb{R}^{V_s \times W_f}$, $L_{in} \in \mathbb{R}^{W_f \times W_d}$. It's easy to implement and adopted by several works (Dai et al., 2019; Lan et al., 2020). In our experiments, we also share input and output embeddings and W_f is initially set to 512 and then

Table 6: Effect of class size on the performance of Transformer-based models.

Model	PRR	Class Size	BLEU
Transformer	16	100	25.24
	16	1000	25.92
	16	10000	**26.93**

Table 7: Effect of class size on the performance of LSTM-based models.

Model	PRR	Class Size	PPL
LSTM	2	10	62.50
	2	100	61.69
	2	1000	**60.90**

divided by the power of 2 according to reduction ratio (Refer to Appendix for more details). Other hyperparameters are the same as parameter settings in section 4.1.

Adaptive Method Adaptive input embedding (Baevski and Auli, 2019) defines a number of bands that partition the frequency ordered vocabulary $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2, \ldots, \mathcal{V}_{n-1} \cup \mathcal{V}_n$ and then reduces the capacity for each band by a factor of k. If words in $\mathcal{V}_1$ have dimension d, then words in $\mathcal{V}_n$ have dimension $\frac{d}{k^{n-1}}$. Next, linear projections $W_1 \in \mathbb{R}^{d \times d}, \ldots, W_n \in \mathbb{R}^{d/k^{n-1} \times d}$ will be added in order to map the embeddings of each band to dimension d. Then the output of the adaptive input embedding layer can be easily used by the subsequent model. Adaptive softmax (Grave et al., 2017) is similar to adaptive input by exploiting the unbalanced word distribution to form clusters except that it applys to output embedding rather than input embedding.

In our experiments, the adaptive method uses both adaptive input word representations (Baevski and Auli, 2019) and an adaptive softmax (Grave et al., 2017). We share the weight of adaptive input and adaptive softmax to keep the same setting as in Baevski and Auli (2019). For the sake of fairness, we directly employ the author's implementation [1] and we choose different bands' sizes and factors k according to the reduction ratio (Refer to Appendix for more details). Other hyperparameters are the same as parameter settings in Section 4.1.

Figure 3 shows the experimental results. Although the adaptive method has a little fluctuation due to different bands' sizes and factors, our method outperforms both projective method and adaptive method consistently when the reduction ratio remains the same.

[1] https://github.com/pytorch/fairseq

4.5 Impact of Class Size

In order to compare the impacts of different class sizes, we keep the unique embedding dimension and class embedding dimension unchanged. For Transformer-based models, we choose $100, 1000, 10000$ as class size. For LSTM-based model, $10, 100, 1000$ are used. Table 6 and 7 indicate that the larger class size will achieve better results. In an extreme case, the reduction effect will disappear when class size equals vocabulary size. So there is a trade-off between the model performance and the word embedding reduction ratio. There is also another explanation. Although frequent words will help infrequent words via shared weights in our method, the words far away from any common class will have to be assigned to one class anyways, which possibly hurts the long tail performance. When we increase the class size, the words far from any common class can be assigned to a separate category and the effect of long tail problem can be alleviated.

4.6 Ablation Experiments

To further validate the effectiveness of our method, we conduct ablation experiments on both language modeling and machine translation tasks. Class index and class embedding are two key components different from the traditional word embedding method and play important roles in our method. When removing them, the model degenerates back to the traditional word embedding method.

Table 8: BLEU scores of random index/our method on Transformer-based machine translation.

UniqueDim	Random Index	Our Method
256	27.39	**27.73**
128	26.9	**27.35**
64	26.43	**26.83**
32	25.74	**25.92**
16	**25.17**	25.01
8	**23.77**	22.75

4.6.1 Ablation of Class Index

For the ablation of class index, we assign words to random classes on Transformer-based machine

translation. We employ exactly the same setting in Section 4.3 except that the indexes are drawn randomly from the uniform distribution over the range 0 to class size 1000.

Table 8 shows that our methods outperform the random ones consistently when the dimension of unique embedding U_d ranges from 32 to 256. It indicates that leveraging the class index following semantic similarity is more effective in reducing word embeddings.

However, for extreme setting $U_d = 16, 8$, our method performs worse. In fact, when unique embedding is too small and underfitting, our assumption that learning unique embeddings of large dimensions for words causes more redundancy in the embedding vectors no longer holds. And unique embedding of small dimensions lacks capacity to learn rich representations to distinguish one word from other words in the same class. Meanwhile, the class index distribution of our method will be more concentrated than uniform distribution, which will undoubtedly increase the difficulty of distinguishing one word from other words in the same class. So our method is lagging behind the random one. And it also reminds us to pay attention to the assumption when using the methods.

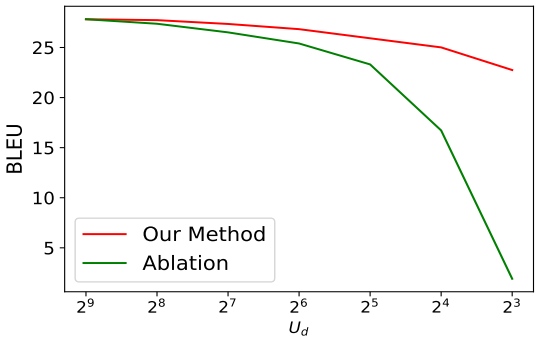

Figure 4: Different BLEU scores of our method and ablation model (the higher, the better). U_d represents the dimension of unique embedding.

4.6.2 Ablation of Class Embedding

We further conduct the ablation of class embedding on both the Transformer-based and LSTM-based models.

Following the similar setting in Section 4.3, the unique dimension U_d is initially set to $W_d = 512$ and then divided by the power of 2. Class embedding is set to 0. However, for a standard Transformer, the dimension of the encoder and decoder need to be equivalent with the word embedding dimension. When we reduce the unique embedding dimension, the number of encoder and decoder parameters will also be reduced, which obviously hurts the model's performance. So in order to compare models fairly, we add a linear projection $L_{in} \in \mathbb{R}^{U_d \times W_d}$ to map the embeddings of each word to dimension W_d so that the dimensions of encoder and decoder do not change. We also add a linear projection $L_{out} \in \mathbb{R}^{W_d \times U_d}$ before output to ensure the sharing of input and output embeddings. We find that the method is the same as projective embedding method, introduced in Section 4.4. For LSTM-based models, the dimension of hidden states is independent to the input embeddings. So we just remove the class embedding and leave the unique embedding alone.

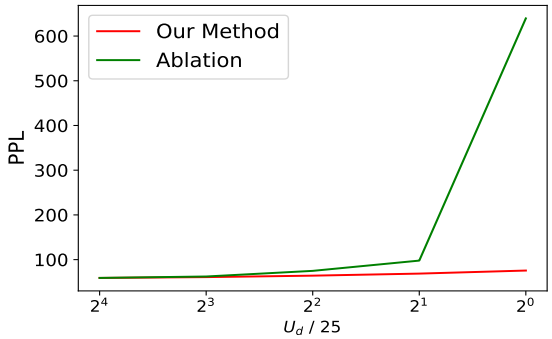

Figure 5: Different PPL scores of our method and ablation model(the lower, the better). U_d represents the dimension of unique embedding.

Experimental results illustrate that class embedding is essential to model performance. Figure 4 compares the different BLEU scores of our method and ablation model. Our method still achieves acceptable BLEU scores at larger reduction ratios, while the scores of the ablation models drop rapidly. For language modeling, we get similar results with the Transformer. Figure 5 compares the different PPL scores of our method and ablation models on PTB dataset. Our method significantly outperforms ablation models when the dimension of unique embedding is small on both PTB and WT-2 datasets. Refer to Appendix for more details.

4.6.3 Analysis of Unique and Class Embeddings

We use C, U and W to denote the class, unique and traditional word embeddings. CS means ***Cosine-Similarity***. (1) Cosine similarities of unique em-

beddings (even in the same class) are usually low, while class embeddings guarantee the overall similarity. (2) In the PTB-LSTM-based experiment, ***year*** and ***month*** belong to the same class. When U_d decreases from 400 to 25 (with $U_d + C_d = W_d = 400$ unchanged), $CS(U_{year}, U_{month})$ becomes smaller. It indicates unique embeddings are more independent and informative. For $W_d = 400, U_d = 100, C_d = 300$, we have:

$$CS(C_{year}, C_{month}) = 1.0$$
$$CS(U_{year}, U_{month}) = 0.067$$
$$CS(W_{year}, W_{month}) = 0.734$$

$$CS(cat(C_{year}, U_{year}), cat(C_{month}, U_{month})) = 0.782$$

In terms of the relationship between class embeddings, we find that cosine similarities of different class embeddings also reveal semantic information. In the following examples, ***year***, ***dog*** and ***Monday*** belong to different classes.

$$CS(C_{year}, C_{dog}) = 0.085$$
$$CS(C_{year}, C_{monday}) = 0.299$$
$$CS(C_{dog}, C_{monday}) = -0.098$$

The similarity of class embeddings between ***year*** and ***Monday*** is significantly larger than that between ***dog*** and ***Monday***. This result conforms to semantics, because ***year*** and ***Monday*** are both related to date/time.

5 Conclusion

In this paper, we propose a novel and intuitive method, UnClE, to explicitly construct semantic classes to reduce the parameters of word embeddings. We divide the traditional word embedding into unique embedding and class embedding. Class embedding is shared by all words belonging to the same class. So we can reduce traditional word embedding at a larger ratio. In the future, we plan to apply our method to other tasks, such as question answering and sentiment analysis. In addition, we will try to apply UnClE on even larger corpora, such as the One Billion Word benchmark, which contains $768M$ word tokens and has a vocabulary of about $800K$ word types.

Acknowledgments

We would like to thank anonymous reviewers for their valuable comments. This work is supported by National Key R&D Program of China (No. 2018AAA0101900), the NSFC projects (No. 62072399, No. U19B2042), Chinese Knowledge Center for Engineering Sciences and Technology, MoE Engineering Research Center of Digital Library, and the Fundamental Research Funds for the Central Universities.

References

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. 2017. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906.

Patrick H. Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh. 2018. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 11011–11021.

Yunchuan Chen, Lili Mou, Yan Xu, Ge Li, and Zhi Jin. 2016. Compressing neural language models by sparse word representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001, 7-11 May, 2001, Salt Palace Convention Center, Salt Lake City, Utah, USA, Proceedings*, pages 561–564. IEEE.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for gpus. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1302–1310. PMLR.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Yeachan Kim, Kang-Min Kim, and SangKeun Lee. 2020. Adaptive compression of word embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3950–3959. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Xiang Li, Tao Qin, Jian Yang, and Tie-Yan Liu. 2016. Lightrnn: Memory and computation-efficient recurrent neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4385–4393.

Zhongliang Li, Raymond Kulhanek, Shaojun Wang, Yunxin Zhao, and Shuang Wu. 2018. Slim embedding layers for recurrent neural language models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5220–5228. AAAI Press.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Sachin Mehta, Rik Koncel-Kedziorski, Mohammad Rastegari, and Hannaneh Hajishirzi. 2020. Define: Deep factorized input token embeddings for neural sequence modeling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048. ISCA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1081–1088. Curran Associates, Inc.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*. Society for Artificial Intelligence and Statistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 157–163. Association for Computational Linguistics.

Raphael Shu and Hideki Nakayama. 2018. Compressing word embeddings via deep compositional code learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2019. Near-lossless binarization of word embeddings. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7104–7111. AAAI Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Nianwen Xue. 2011. Steven bird, evan klein and edward loper. *Natural Language Processing with Python*. o'reilly media, inc 2009. ISBN: 978-0-596-51649-9. *Nat. Lang. Eng.*, 17(3):419–424.

Yadollah Yaghoobzadeh, Katharina Kann, Timothy J Hazen, Eneko Agirre, and Hinrich Schütze. 2019. Probing for semantic classes: Diagnosing the meaning content of word embeddings. *arXiv preprint arXiv:1906.03608*.

Sanqiang Zhao, Raghav Gupta, Yang Song, and Denny Zhou. 2019. Extreme language model compression with optimal subwords and shared projections. *CoRR*, abs/1909.11687.

Model	UniqueDim	EmbedParams(10^6)	TotalParams(10^6)	EmbedMemory	TotalMemory	BLEU
Transformer(Vaswani)	512	20.9	65.0	79.5M	247.9M	27.30
Transformer(Our Implementation)	512	20.9	65.0	79.5M	247.9M	**27.82**
UnClE(Our Method)	256	10.7	54.8	40.7M	209.1M	27.73
	128	5.59	49.7	21.3M	189.7M	27.35
	64	3.05	47.2	11.6M	180.0M	26.83
	32	1.78	45.9	6.8M	175.2M	25.92

Table 9: The number of parameters and the memory sizes for Transformer-based models, when the $UniqueDim$ ranges from 512 to 32. *EmbedParams* and *TotalParams* represent the parameters of embedding layer and whole model respectively. *EmbedMemory* and *TotalMemory* represent the memory sizes of embedding layer and whole model respectively.

DataSet	UniqueDim	EmbedParams(10^6)	TotalParams(10^6)	EmbedMemory	TotalMemory	PPL(test)
PTB	400	4.0	24.2	15.26	92.32M	**59.08**
PTB	200	2.20	22.4	8.39	85.45M	**60.90**
PTB	100	1.30	21.5	4.96	82.02M	64.16
PTB	50	0.85	21.07	3.24	80.38M	68.75
PTB	25	0.625	20.85	2.38	79.54M	75.58
WT2	400	13.31	33.56	50.78	128.03M	**65.60**
WT2	200	6.86	27.01	26.15	102.97M	**68.07**
WT2	100	3.63	23.87	13.84	91.00M	72.73
WT2	50	2.01	22.26	7.68	85.08M	80.54
WT2	25	1.21	21.45	4.60	81.62 M	92.48

Table 10: The number of parameters and the memory sizes for LSTM-based models, when the $UniqueDim$ ranges from 400 to 25. *EmbedParams* and *TotalParams* represent the parameters of embedding layer and whole model respectively. *EmbedMemory* and *TotalMemory* represent the memory sizes of embedding layer and whole model respectively.

A Appendix

A.1 PTB and WT2 Datasets

The Penn Treebank dataset contains about 929K/74K/82K tokens in its train, validation, and test sets respectively. The dataset does not contain capital letters, numbers, or punctuation. There are about 10K different words in its vocabulary.

WikiText-2 contains 2,088k training, 217k validation, 245k test tokens, and a vocabulary of 33,278 words. The text is tokenized and processed using the Moses tokenizer. Capitalization, punctuation, and numbers are retained in this dataset.

A.2 Memory Size

Table 9 and Table 10 show the number of parameters and the memory sizes for Transformer-based and LSTM-based models. We believe our method will be more impressive when the size of vocabulary is very large, such as the One Billion Word benchmark (800K), or working with Universal Transformer or Albert models that share parameters across layers.

A.3 Related Works

A.3.1 Projective Method

Word Dimension	RR	BLEU
256	1.975	27.37
128	3.98	26.51
64	7.9	25.40
32	15.8	23.31
16	31	16.71
8	63.2	1.91

Table 11: Hyperparameter setting and the corresponding results of the projective method.

We use V_s and W_d to represent the vocabulary size and the dimension of traditional word embedding, respectively. When we use the projective reduction method, let W_f denote another dimension of linear projection, then we have the reduction ratio as follows:

$$RR = \frac{V_s * W_d}{V_s * W_f + W_f * W_d} \tag{8}$$

A.3.2 Adaptive Method

For adaptive method, we use $V_1, V_2..., V_n$ to denote size of bands $\mathcal{V}_1 \cup \mathcal{V}_2, \ldots, \cup \mathcal{V}_n$. If words in

Source : He was accompanied on his visit to Russia by two German journalists .
Target : Er wurde bei seinem Besuch in Russland von zwei deutschen Journalisten begleitet .
Baseline : Er wurde bei seinem Besuch in Russland von zwei deutschen Journalisten begleitet .
Our method: Er wurde bei seinem Besuch in Russland von zwei deutschen Journalisten begleitet .
Ablation : Er wurde von zwei deutschen Journalisten auf seinen Besuch in Russland begleitet .

Source : The darker the meat , the higher the pH value .
Target : Je dunkler das Fleisch , desto höher der ph Wert .
Baseline : Je dunkler das Fleisch , desto höher der pH Wert .
Our method: Je dunkler das Fleisch , desto höher der pH Wert .
Ablation : Je neu das Fleisch ist , desto höher ist der pH Wert .

Figure 6: We sample two cases from the test set of WMT 2014 English-German. The words underlined in red are wrong. In the first case, the ablation model's result has a different word order, which is written in blue and green. In the second case, there are several wrong words in the ablation result.

k	V_1	V_2	V_3	RR	BLEU
	15000	21000	4728	1.96	27.42
4	4000	8000	28728	5.12	26.32
	2000	4000	34728	7.64	26.12
	600	1200	38928	11.66	25.68
	1000	5000	34728	18.18	22.38
8	500	4500	35228	24.06	21.85

Table 12: Hyperparameter setting and the corresponding results of the adaptive method.

$\mathcal{V}_1$ have dimension W_{f_1}, then words in $\mathcal{V}_n$ have dimension $W_{f_n} = \frac{W_{f_1}}{k^{n-1}}$. In our experiments, we choose $W_{f_1} = 512$ and the factor $k = 4$ or 8. We have the reduction ratio:

$$RR = \frac{V_s * W_d}{[V_1 * W_{f_1} + W_{f_1} * W_d] + ... + [V_n * W_{f_n} + W_{f_n} * W_d]} \quad (9)$$

A.4 Choice of the Number of Classes

We suggest that the number of classes should be a bit less than one tenth of vocabulary size. Linguistically speaking, dozens of words may have similar semantics in most cases. Figure 2 also supports this point. Experimentally, we also find that the result of 1000 class size is better than those of 100 and 10000 cases under the same reduction ratio. However, the number of classes indeed is the hyperparameter and doing grid search is always necessary for the best one. Here we just give a suggestion that is worth trying.

A.5 Out-of-vocabulary Words

Here we give a way to handle Out-of-vocabulary words in our method. We use the publicly trained word vectors such as Glove, GoogleNews-vectors or the embedding of BERT, in which the out-of-vocabulary words can be found, to compute the similarities between the center word of each class and out-of-vocabulary words. The class embedding of out-of-vocabulary words will be the same as that of the closest center words. As for the unique embedding, we compute the similarities between the given out-of-vocabulary word and other words in the same class by word vectors. Then we normalize the similarities distribution, and obtain the unique embedding of the out-of-vocabulary word by weighted linear combination of the unique embeddings of other words in the same class. The weight is the normalized distance.

A.6 Ablation Experiment

UniqueDim	BLEU(A)	BLEU(U)
256	27.37	27.73
128	26.51	27.35
64	25.40	26.83
32	23.31	25.92
16	16.71	25.01
8	1.91	22.75

Table 13: Results of ablation studies of UnClE along with Transformer for machine transaltion. We denote the BLEU score of ablation models and our method as BLEU(A) and BLEU(U) respectively. The ablation model here is equivalent to the predictive method.

Table 13 and Table 14 summarize the results of ablation studies of UnClE along with Transformer models and LSTM-based models. For machine translation, our method still achieves acceptable BLEU scores at larger reduction ratios, while the score of ablation models drops rapidly. Especially,

DataSet	UniqueDim	PPL(A)	PPL(U)
PTB	200	62.27	60.90
	100	74.87	64.16
	50	97.84	68.75
	25	639.47	75.58
WikiText-2	200	71.15	68.07
	100	84.96	72.73
	50	157.12	80.54
	25	708.02	92.48

Table 14: Results of ablation studies of UnClE along with LSTM for language modeling. We denote the PPL of ablation models and our method as PPL(A) and PPL(U) respectively. The ablation model here is equivalent to the predictive method.

when unique embedding size decreases to 8, BLEU score of ablation method drops to 1.91 and our method achieves 22.75 BLEU score.

A.7 Case Study on Translation Quality

We sample several specific examples to evaluate the translation quality of our method (See Figure 6). We choose the model with $U_d = 32$, $C_d = 480$, $RR = 11.69$ to represent our method. We compare our method with baseline model ($W_d = 512$) and ablation model ($U_d = 32$). Sampled results indicate that the baseline model and our model are consistent with the target while the ablation model has a poor performance.

A.8 Polysemous Words

Refer to Equation (2). Our method can be extended to more fine-grained and hierarchical approach. Each class can be divided into multiple sub-classes. This will alleviate the polyseme issue. Also, the Transformer-based model itself can alleviate this problem by learning contextual representations.