A Simple and Effective Usage of Word Clusters for CBOW Model

Yukun Feng[1], **Chenlong Hu**[1], **Hidetaka Kamigaito**[1], **Hiroya Takamura**[1,2] **and Manabu Okumura**[1]
[1]Tokyo Institute of Technology
[2]National Institute of Advanced Industrial Science and Technology (AIST)
{yukun,huchenlong,kamigaito,oku}@lr.pi.titech.ac.jp
takamura@pi.titech.ac.jp

Abstract

We propose a simple and effective method for incorporating word clusters into the Continuous Bag-of-Words (CBOW) model. Specifically, we propose to replace infrequent input and output words in CBOW model with their clusters. The resulting cluster-incorporated CBOW model produces embeddings of frequent words and a small amount of cluster embeddings, which will be fine-tuned in downstream tasks. We empirically show our replacing method works well on several downstream tasks. Through our analysis, we show that our method might be also useful for other similar models which produce word embeddings.

1 Introduction

Word embeddings have been widely applied to various natural language processing (NLP) tasks. These embeddings can be pretrained on a large corpus and carry useful semantic information. One of the most well-known methods for obtaining word embeddings is based on Continuous Bag-of-Words (CBOW) (Mikolov et al., 2013a) and there have been many research efforts to extend it.

In this paper, we focus on incorporating word clusters into CBOW model. Each word cluster consists of words that function similarly. By aggregating such words, we can alleviate data sparsity, even though each of those words is infrequent. In the past few years, word clusters have been applied to various tasks, such as named-entity recognition (Ritter et al., 2011), machine translation (Wuebker et al., 2013) and parsing (Kong et al., 2014). Many word clustering algorithms can be applied to a raw corpus with different languages and help us obtain word clusters easily without additional language resources.

In our method, we keep only very frequent words and replace the other words with their clusters for both input and output words in the CBOW model. This is motivated by the fact that word clusters are more reliable than infrequent words. Thus, only very frequent word embeddings and a small amount of cluster embeddings are produced as the output. When fine-tuning the trained embeddings on downstream tasks, the embeddings of infrequent words within one cluster are initialized by the embedding of their cluster to increase the coverage of pretrained word embeddings.

Since word embeddings are usually trained on the large-scale dataset. For making clusters on the large-scale dataset, we choose bidirectional, interpolated, refining, and alternating (BIRA) predictive exchange algorithm (Dehdari et al., 2016)[1] as our clustering method. Because BIRA was reported to be faster than many other methods. Notably, it can produce 800 clusters on 1 billion English tokens in 1.4 hours.

We evaluate our cluster-incorporated word embeddings[2] on downstream tasks, in which *fine-tuning* of word embeddings is involved. The evaluation for frequent words, for which our method also works well, on word similarity tasks can be found in appendix A. For the downstream tasks, we choose language modeling (LM) tasks, which are a fundamental task in NLP, as well as two machine translation (MT) tasks. To verify the effect of word clusters across different languages, 8 typologically diverse languages are further selected for the LM task. Finally, an analysis is provided for our method. In summary, our replacing method can be used to improve the embeddings of frequent and infrequent words, to reduce the number of word embeddings and to make training more effective.

[1]We used ClusterCat (https://github.com/jonsafari/clustercat) as the implementation.
[2]https://github.com/yukunfeng/cluster-cbow

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 80–86

2 Related Work

A number of related research efforts have been done to help to learn better word embeddings aiming at different aspects. For example, Neelakantan et al. (2014) proposed an extension that learns multiple embeddings per word type. Ammar et al. (2016) proposed methods for estimating embeddings for different languages in a single shared embedding space. There is also a lot of work that incorporates internal information of words, such as character-level information (Chen et al., 2015; Bojanowski et al., 2017) and morpheme information (Luong et al., 2013; Qiu et al., 2014). Our research aims at another aspect and focuses on incorporating word clusters into the CBOW model, which has not been studied before.

There have also been some previous researches that utilized word clusters for reducing the number of word embeddings. Botha et al. (2017) used word clusters to reduce the network size for the part-of-speech tagging task. Shu and Nakayama (2018) attempted to compress word embeddings without losing performance by constructing the embeddings with a few basic vectors. Our goal is different from the previous work in that we attempt to learn better word embeddings and do not aim at reducing the parameters when our embeddings are fine-tuned in downstream tasks. Nonetheless, the reduction of the number of word embeddings from the CBOW model before fine-tuning is still one of our goals as we can save space to store these embeddings and save time to download them. For example, Google News Vectors have around 3 million words, and we need only 2% of the number of the word embeddings if we choose 100K most frequent words and 10K word clusters in our method.

3 Our Method

3.1 CBOW Model

Let w_t denote the t-th word in a given text. We adopt the basic CBOW model architecture for learning word embeddings. The CBOW model predicts the output word w_t given the input words in the window which precede or follow the output word. When the window size is 2, as an example, the input words are $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$. We denote the input and output embeddings of word w_i respectively as $\vec{x}_i$ and $\vec{o}_i$. The CBOW model computes the hidden representation as follows:

$$\vec{h} = \frac{1}{2c} \sum_{i=-c, i\neq 0}^{c} \vec{x}_{t+i}, \quad (1)$$

where c is the window size. We use negative sampling (Mikolov et al., 2013b) to train the CBOW model by maximizing the following objective function:

$$\log\sigma(\vec{h}^T \vec{o}_t) + \sum_{j=1}^{k} \log\sigma(-\vec{h}^T \vec{o}^j), \quad (2)$$

where k is the size of the negative sample, $\vec{o}^j$ is the j-th noise word embedding and σ is the sigmoid function. Each word in the negative sample is drawn from the unigram distribution.

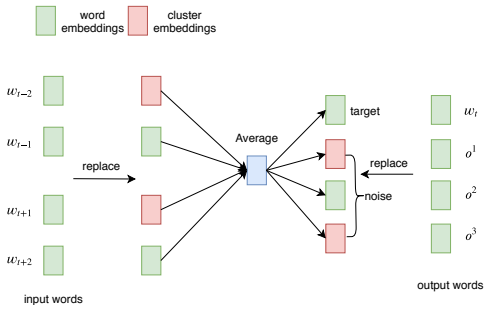

Figure 1: CBOW architecture with our replacing method for input and output words trained with negative sampling. Suppose that w_{t-2}, w_{t+1}, o^1 and o^3 are infrequent words.

3.2 Replacing Methods

As a method for incorporating word clusters, we propose to replace infrequent words with their clusters for the input and output. The architecture is shown in Figure 1. This is motivated by the intuition that the embeddings of clusters should be more reliable than those of infrequent words. We denote the embedding of the cluster for word w_{t+i} as $\vec{d}_{t+i}$. We present the following two replacing methods:

- **ReIn**: In the input, $\vec{x}_{t+i}$ in Eq. (1) will be replaced with $\vec{d}_{t+i}$ if the frequency of w_{t+i} is less than threshold f_{in}.
- **ReOut**: In the output, output words whose frequency is less than f_{out} are replaced with their clusters. Thus, in negative sampling, a noise word will be sampled from clusters and frequent words.

As with the standard CBOW model, we use the input word embeddings and input cluster embeddings for downstream tasks. Thresholds f_{in} and f_{out} are set to 100 in all experiments. Due to this large value, each cluster contains many infrequent words, which share the same embedding. We use two methods together, which is referred to as **ReIn+ReOut** in the following experiments.

3.2.1 Motivation of ReIn and ReOut

Since the embeddings of clusters are learned by aggregating many infrequent words, they are more robust than the embeddings of the infrequent words. During the fine-tuning process for a downstream task, the embeddings of infrequent words are first initialized with the embeddings of their clusters. As most of these infrequent words appear only a few times, these embeddings will not be updated far away from each other within one cluster. The visualization of these embeddings before and after fine-tuning can be found in the appendix B. As a result, these embeddings for infrequent words become more reliable since originally most infrequent word embeddings are updated only several times and are not far away from where they were randomly initialized. Since the context of frequent words becomes less noisy by replacing all the infrequent words with their clusters, the learned frequent word embeddings are also better, as shown later in our experiments.

The standard CBOW model is usually trained with negative sampling, which is designed for speeding up the training process. By using ReOut, infrequent noise words will be replaced with their clusters, which contain more noise words than the original CBOW model. As a result, ReOut makes the training of the CBOW model more effective, as shown later in our experiments.

4 Experiments on LM and MT

We applied our embeddings to downstream tasks: language modeling (LM) and low-resource machine translation (MT). When applying to the downstream tasks, we only used the training data of the specific task to obtain word clusters and embeddings without any extra data. We then used the learned embeddings to initialize the lookup table of word embeddings for the task. In this paper, we limit the applications of our model to relatively small datasets to demonstrate the usefulness of our method. We plan to conduct larger-scale experiments on more downstream tasks in future work. In the following tables, CBOW and ReIn+ReOut indicate that they are initialization methods for specific downstream tasks.

4.1 Hyper-parameter Settings

In this section, we describe the hyper-parameters for producing word clusters and word embeddings. As we mentioned before, we obtained word clusters through the ClusterCat software. For most hyper-parameters, we used its default values. We set the number of clusters to 600 in all our experiments. Since our work involves many tasks in total, it is hard to choose the optimal number of word clusters for each task. We experimented with several values (600, 800 and 1000) and observed the same trend. Thus, we simply chose 600, for convenience, for all tasks. For producing word embeddings, our implementation was based on the fasttext [3]. Our cluster-incorporated CBOW model and the standard CBOW model were trained under the same hyper-parameters. We set most hyper-parameters as its default values. Namely, we set the training epoch to 5, the number of negative examples to 5, the window size to 5, and the minimum count of word occurrence to 5[4].

4.2 LM on Standard English Datasets

We test ReIn+ReOut based on the recent state-of-the-art awd-lstm-lm codebase[5](Merity et al., 2018) using two standard language modeling datasets: Penn Treebank (PTB) and WikiText-2 (Wiki2). We followed exactly the same setting in the source code. The results are shown in Table 1, and we found that our ReIn+ReOut is effective even with the strong baseline.

	PTB	Wiki2
AWD-LSTM w/o fine-tuning (Merity et al., 2018)	58.80	66.00
CBOW	58.39	65.48
ReIn+ReOut	**57.85**	**63.93**

Table 1: Perplexity results on PTB and Wiki2.

4.3 Low-resource NMT

We applied our method to the standard long-short term memory networks (LSTMs) based sequence-to-sequence (seq2seq) model on two datasets: German-English (de-en) with 153K sentence pairs

[3]https://github.com/facebookresearch/fastText

[4]When we set the minimum count of word occurrence to 1, the standard CBOW does not perform well.

[5]https://github.com/salesforce/awd-lstm-lm

from IWSLT 2014 (Cettolo et al., 2014), English-Vietnamese (en-vi) with 133K sentence pairs from IWSLT 2015 (Cettolo et al., 2012). The detailed data statistics of two low-resource NMT datasets is in Table 2. We used the opennmt-py toolkit[6] with a 2-layer bidirectional LSTM with hidden size of 500 and set the training epoch to 30. The word embedding size is set to 500 and the batch size is 64. We trained the seq2seq models by the SGD optimizer with start learning rate being 1.0, which will be decayed by 0.5 if perplexity does not decrease on the validation set. Other hyper-parameters were kept default. We also include some published results based on LSTM-based seq2seq models to gauge the result of our baseline. As shown in Table 3, without any extra language pair resources, the ReIn+ReOut initialization improves the BLEU score over the baseline by 1.29 and 0.51 points on de-en, en-vi respectively.

	de-en	en-vi
#Training pairs	153,348	133,317
#Test pairs	6,750	1,268
#Valid pairs	6,970	1,553
Train Vocab (source)	103,796	54,169
Train Vocab (target)	50,045	25,615

Table 2: Data statistics of two low-resource NMT datasets.

	de-en	en-vi
seq2seq with attention (Luong and Manning, 2015)	-	23.3
AC+LL (Bahdanau et al., 2017)	28.53	-
NPMT (Huang et al., 2018)	29.92	27.69
Our seq2seq with attention	28.95	28.16
CBOW	29.25	28.24
Our ReIn+ReOut	**30.24**	**28.67**

Table 3: BLEU scores on two low-resource MT datasets. NPMT in Huang et al. (2018) used a neural phrase-based machine translation model and AC+LL in Bahdanau et al. (2017) used a one-layer GRU encoder and decoder with attention.

4.4 LM in Diverse Languages

To verify the effect of word clusters on different languages, we selected 8 datasets containing typologically diverse languages from LM datasets released by Gerz et al. (2018). The data statistics of 8 LM datasets is in Table 5. We basically used standard LSTMs instead of AWD-LSTM-LM to save time. We chose the available standard LSTM-LM code[7]. Hyper-parameters of our standard LSTM model on language modeling tasks is in Table 4. The results are shown in Table 6. Our LSTM-LM obtained better results than the one from Gerz et al. (2018) on all datasets. As we see, ReIn+ReOut is effective for typologically diverse languages and also requires a smaller input vocabulary. For example, the input vocabulary of ReIn+ReOut for en dataset contains 1.3K words while the full vocabulary 50K.

Embedding size	200
Epochs	40
LSTM layers	2
Optimizer	SGD
LSTM sequence length	35
Learning rate	20
LSTM hidden unit	200
Learning rate decay	4
Param. init: rand uniform	[-0.1,0.1]
Gradient clipping	0.25
Dropout	0.2
Batch size	20

Table 4: Hyper-parameters of our standard LSTM model on language modeling task.

5 Analysis

In this section, we analyse ReIn+ReOut on the basis of LM experiments with en and de datasets.

5.1 Targeted Perplexity Results

To show the gain for frequent and infrequent words, we measured the perplexity for frequent and infrequent words in the test data separately. Specifically, we calculated the perplexity of the next word, when an infrequent word is given as the current word. A similar analysis on language models can be found in Vania and Lopez (2017). Our analysis do not contain new words in the test dataset. The results are shown in Table 7. As we see, ReIn+ReOut is more effective than CBOW in learning both the embeddings of frequent and infrequent words, as we explained in Sec. 3.2.1.

5.2 Ablation Study

The results of ablation study are in Table 8. Comparing the methods ReIn and CBOW, we found replacing only input infrequent words in CBOW also works better than the original CBOW. We can also conclude that replacing only output infrequent words in CBOW works better than the original CBOW, by comparing ReOut and CBOW. Both ReIn and ReOut work well even when they are used alone. As mentioned in the motivation of ReOut, it makes the training more effective. To verify

[6] https://github.com/OpenNMT/OpenNMT-py

[7] https://github.com/pytorch/examples/tree/master/word_language_model

	Typology	Train vocab	#Train tokens	#Test tokens	#Valid tokens	#Input vocab of ReIn+ReOut
zh (Chinese)	Isolating	43674	746K	56.8K	56.9K	1661
vi (Vietnamese)	Isolating	32065	754K	61.9K	64.8K	1716
de (German)	Fusional	80743	682K	51.3K	52.6K	1163
en (English)	Fusiona	55522	783K	59.5K	57.3K	1381
ar (Arabic)	Introflexive	89091	723K	54.7K	55.2K	1431
he (Hebrew)	Introflexive	83223	719K	54.7K	52.9K	1345
et (Estonian)	Agglutinative	94184	556K	38.6K	40.0K	1285
tr (Turkish)	Agglutinative	90847	627K	45.2K	47.4K	1241

Table 5: Data statistics of 8 language modeling datasets and size of input vocabulary of our ReIn+ReOut.

Dataset	Random	CBOW	ReIn+ReOut
zh	555	527	**494**
vi	153	145	**138**
de	609	542	**484**
en	365	317	**289**
ar	1647	1447	**1305**
he	1482	1236	**1175**
et	1451	1157	**1004**
tr	1379	1220	**1148**

Table 6: Perplexity results of standard LSTM LM on 8 datasets with different initialization methods.

		Freq.	Infreq.	All
en	CBOW	340	198	283
	ReIn+ReOut	316	184	264
de	CBOW	591	352	489
	ReIn+ReOut	564	318	458

Table 7: Targeted perplexity results of standard LSTM LM with different initializations.

this, we increased the number of negative samples for ReIn and CBOW. The training will be more effective if we increase the number of negative samples, while training the model will also take longer time. As we increased the size of negative samples, we obtained better results for both ReIn and CBOW. We increased it only to 30 because we did not observe improvements when we made it further larger. This result indicates that we can use word clusters to obtain better results with a small amount of negative samples. In reality, we can also use off-the-shelf word clusters to avoid spending time for producing word clusters.

	en	de		en	de
ReIn	300	528	CBOW	317	542
ReIn neg+10	293	499	CBOW neg+10	309	523
ReIn neg+30	300	494	CBOW neg+30	312	554
ReIn+ReOut	289	484	ReOut	312	515

Table 8: Perplexity results of LSTM LM by changing the number of negative samples. '+neg' represents the number of negative samples, which is 5 at default.

5.3 LM Results on Off-the-shelf Vectors

To gauge the improvements, we used off-the-shelf pretrained word vectors in English: GloVe vectors (Pennington et al., 2014) and Google News Vectors[8]. We obtained 258, 290 and 289 perplexity scores on en with Google News Vectors, Glove vectors and ReIn+ReOut respectively. Although ReIn+ReOut underperforms Google News Vectors, which were trained on 100 billion tokens, it obtained the results comparable to Glove Vectors, trained on 6 billion tokens. This indicates that our ReIn+ReOut is effective even without extra training data (only 783K training tokens in en).

	en
Google News Vectors	258
GloVe Vectors	290
ReIn+ReOut	289

Table 9: Perplexity results of standard LSTM compared with off-the-shelf vectors.

6 Conclusion

We proposed a simple and effective method to incorporate word clusters into the CBOW model. Our method is effective on several downstream tasks. For future work, we will test our methods on larger corpora and also add more downstream tasks. We will also study how to combine word clusters and subword information.

Acknowledgments

We would like to thank anonymous reviewers for their constructive comments and Hu also thanks his support from China Scholarship Council.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith.

[8]https://code.google.com/archive/p/word2vec/

2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*.

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

Jan A. Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan T. McDonald, and Slav Petrov. 2017. Natural language processing with small feed-forward networks. In *EMNLP*.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, page 57.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Jon Dehdari, Liling Tan, and Josef van Genabith. 2016. BIRA: Improved predictive exchange word clustering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1169–1174, San Diego, CA, USA. Association for Computational Linguistics.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association of Computational Linguistics*, 6:451–465.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *International Conference on Learning Representations*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. In *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 141–150.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1524–1534. Association for Computational Linguistics.

Raphael Shu and Hideki Nakayama. 2018. Compressing word embeddings via deep compositional code learning. In *International Conference on Learning Representations*.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2016–2027, Vancouver, Canada. Association for Computational Linguistics.

Joern Wuebker, Stephan Peitz, Felix Rietig, and Hermann Ney. 2013. Improving statistical machine translation with word class models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1377–1381.

A Experiments on Word Similarity Task

The word similarity task is not necessarily suitable for our replacing method due to many infrequent words sharing the same embedding within one cluster. Thus, we report the results of the task for three different groups of test word pairs: *frequent-word-pair* consisting of frequent word pairs, *infrequent-word-pair* consisting of word pairs that share a cluster embedding with other words, and *all-word-pair* consisting of all test word pairs. We used the publicly available enwik8[9] corpus as the training data to obtain both word embeddings and word clusters. Note that we use this data only for the word similarity task, not for downstream tasks such has language modelling and machine translation. We preprocessed the corpus by lowercasing all words, removing words that contain non-alphabetical characters, and removing words whose frequency is less than 5. The final corpus contains approximately 12 million tokens and 60K word types. We chose MEN, MTurk287, MTurk771, RW and WS353 as our datasets. Then, we evaluated the quality of these representations by computing Spearman's rank correlation coefficient. One straightforward method to incorporate word cluster into CBOW model is to average the embeddings of word and its cluster referred as to AvgIn.

	Dataset (#word pairs)	CBOW	ReIn+ ReOut	AvgIn	AvgIn+ ReOut
Frequent word pair	MTurk287 (198)	65.12	66.03	64.22	**66.56**
	MEN (1296)	65.09	**68.74**	61.03	63.34
	WS353 (244)	69.51	**70.36**	63.20	62.00
	RW (169)	49.13	**51.57**	45.59	48.83
	MTurk771 (530)	54.10	**56.83**	48.37	51.34
Infrequent word pair	MTurk287 (86)	**49.50**	34.28	43.83	36.89
	MEN (1686)	**46.71**	23.58	23.01	26.61
	WS353 (89)	**52.12**	33.68	32.35	31.08
	RW (828)	**31.42**	21.49	20.75	20.68
	MTurk771 (237)	**50.88**	25.55	31.08	31.35
All word pair	MTurk287 (284)	**60.98**	58.14	58.49	58.42
	MEN (2982)	**54.79**	44.65	40.55	43.70
	WS353 (333)	**64.41**	58.61	53.26	52.96
	RW (997)	**35.15**	25.12	23.92	24.78
	MTurk771 (767)	**52.73**	46.93	42.50	45.10

Table 10: Spearman's rank correlation coefficient on word similarity datasets for different groups. The best scores in each group are in bold.

We first applied ClusterCat to the preprocessed corpus to obtain word clusters and then produced cluster-incorporated word embeddings with ReIn+ReOut. The results are shown in Table 10. In ReIn+ReOut, the number of input words is 10,203, which is the sum of 9,603 frequent words and 600 clusters. This is only 16.9% of the number of input words for the original CBOW, which maintains 60K input words. In all word pair group, CBOW outperformed ReIn+ReOut on all datasets. This is because ReIn+ReOut does not perform well in infrequent-word-pair group as many infrequent words share exactly the same embedding in one cluster. In this experiment, each cluster had 82 words on average. However, ReIn+ReOut outperformed CBOW on frequent-word-pair group in all datasets. This result suggests that ReIn+ReOut is effective in learning embeddings for frequent words with much fewer parameters. AvgIn underperformed CBOW in all-word-pair group, which suggests that this straightforward way to incorporate word clusters is not effective. We also found that AvgIn+ReOut can improve the performance on 3 datasets in all-word-pairs group compared with AvgIn. However, AvgIn+ReOut still underperformed CBOW on all datasets.

[9] http://mattmahoney.net/dc/enwik8.zip

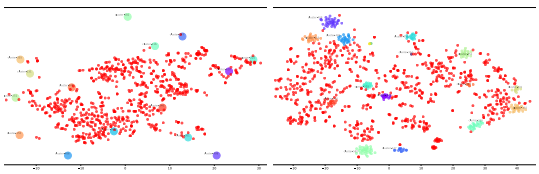

Figure 2: Visualization of the embeddings of frequent words and clusters before fine-tuning (left) and the embeddings of frequent and infrequent words after fine-tuning (right). The red circle represents frequent words. The color of infrequent words within different clusters are different (right), and the big circle represents word clusters (left).

B Visualization of Word Embeddings

We visualize word embeddings using t-SNE projections. Specifically, we randomly chose 15 clusters and all frequent words from en and visualize frequent and infrequent word embeddings in these 15 clusters in Figure 2. The embeddings of infrequent words within one cluster are located close together after being fine-tuned. Some infrequent word embeddings are updated only several times and are not far away from where they were randomly initialized, and now they become more reliable.