FRAGE: Frequency-Agnostic Word Representation

Chengyue Gong[1]
cygong@pku.edu.cn

Di He[2]
di_he@pku.edu.cn

Xu Tan[3]
xu.tan@microsoft.com

Tao Qin[3]
taoqin@microsoft.com

Liwei Wang[2,4]
wanglw@cis.pku.edu.cn

Tie-Yan Liu[3]
tie-yan.liu@microsoft.com

[1]Peking University
[2]Key Laboratory of Machine Perception, MOE, School of EECS, Peking University
[3]Microsoft Research Asia
[4]Center for Data Science, Peking University, Beijing Institute of Big Data Research

Abstract

Continuous word representation (aka word embedding) is a basic building block in many neural network-based models used in natural language processing tasks. Although it is widely accepted that words with similar semantics should be close to each other in the embedding space, we find that word embeddings learned in several tasks are biased towards word frequency: the embeddings of high-frequency and low-frequency words lie in different subregions of the embedding space, and the embedding of a rare word and a popular word can be far from each other even if they are semantically similar. This makes learned word embeddings ineffective, especially for rare words, and consequently limits the performance of these neural network models. In this paper, we develop a neat, simple yet effective way to learn *FRequency-AGnostic word Embedding* (FRAGE) using adversarial training. We conducted comprehensive studies on ten datasets across four natural language processing tasks, including word similarity, language modeling, machine translation and text classification. Results show that with FRAGE, we achieve higher performance than the baselines in all tasks.

1 Introduction

Word embeddings, which are distributed and continuous vector representations for word tokens, have been one of the basic building blocks for many neural network-based models used in natural language processing (NLP) tasks, such as language modeling [20, 18], text classification [26, 8] and machine translation [5, 6, 44, 42, 12]. Different from classic one-hot representation, the learned word embeddings contain semantic information which can measure the semantic similarity between words [30], and can also be transferred into other learning tasks [31, 3].

In deep learning approaches for NLP tasks, word embeddings act as the inputs of the neural network and are usually trained together with neural network parameters. As the inputs of the neural network, word embeddings carry all the information of words that will be further processed by the network, and the quality of embeddings is critical and highly impacts the final performance of the learning task [16]. Unfortunately, we find the word embeddings learned by many deep learning approaches are far from perfect. As shown in Figure 1(a) and 1(b), in the embedding space learned by word2vec model, the nearest neighbors of word "Peking" includes "quickest", "multicellular", and "epigenetic", which

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

are not semantically similar, while semantically related words such as "Beijing" and "China" are far from it. Similar phenomena are observed from the word embeddings learned from translation tasks.

With a careful study, we find a more general problem which is rooted in low-frequency words in the text corpus. Without any confusion, we also call high-frequency words as popular words and call low-frequency words as rare words. As is well known [25], the frequency distribution of words roughly follows a simple mathematical form known as Zipf's law. When the size of a text corpus grows, the frequency of rare words is much smaller than popular words while the number of unique rare words is much larger than popular words. Interestingly, the learned embeddings of rare words and popular words behave differently. (1) In the embedding space, a popular word usually has semantically related neighbors, while a rare word usually does not. Moreover, the nearest neighbors of more than 85% rare words are rare words. (2) Word embeddings $encode$ frequency information. As shown in Figure 1(a) and 1(b), the embeddings of rare words and popular words actually lie in different subregions of the space. Such a phenomenon is also observed in [31].

We argue that the different behaviors of the embeddings of popular words and rare words are problematic. First, such embeddings will affect the semantic understanding of words. We observe more than half of the rare words are nouns or variants of popular words. Those rare words should have similar meanings or share the same topics with popular words. Second, the neighbors of a large number of rare words are semantically unrelated rare words. To some extent, those word embeddings encode more frequency information than semantic information which is not good from the view of semantic understanding. It will consequently limit the performance of down-stream tasks using the embeddings. For example, in text classification, it cannot be well guaranteed that the label of a sentence does not change when you replace one popular/rare word in the sentence by its rare/popular alternatives.

To address this problem, in this paper, we propose an adversarial training method to learn *FRequency-AGnostic word Embedding* (FRAGE). For a given NLP task, in addition to minimize the task-specific loss by optimizing the task-specific parameters together with word embeddings, we introduce another discriminator, which takes a word embedding as input and classifies whether it is a popular/rare word. The discriminator optimizes its parameters to maximize its classification accuracy, while word embeddings are optimized towards a low task-dependent loss as well as fooling the discriminator to mis-classify the popular and rare words. When the whole training process converges and the system achieves an equilibrium, the discriminator cannot well differentiate popular words from rare words. Consequently, rare words lie in the same region as and are mixed with popular words in the embedding space. Then FRAGE will catch better semantic information and help the task-specific model to perform better.

We conduct experiments on four types of NLP tasks, including three word similarity tasks, two language modeling tasks, three sentiment classification tasks and two machine translation tasks to test our method. In all tasks, FRAGE outperforms the baselines. Specifically, in language modeling and machine translation, we achieve better performance than the state-of-the-art results on PTB, WT2 and WMT14 English-German datasets.

2 Background

2.1 Word Representation

Words are the basic units of natural languages, and distributed word representations (i.e., word embeddings) are the basic units of many models in NLP tasks including language modeling [20, 18] and machine translation [5, 6, 44, 42, 12]. It has been demonstrated that word representations learned from one task can be transferred to other tasks and achieve competitive performance [3].

While word embeddings play an important role in neural network-based models in NLP and achieve great success, one technical challenge is that the embeddings of rare words are difficult to train due to their low frequency of occurrences. [39] develops a novel way to split word into sub-word units which is widely used in neural machine translation. However, the low-frequency sub-word units are still difficult to train: [33] provides a comprehensive study which shows that the rare (sub)words are usually under-estimated in neural machine translation: during inference step, the model tends to choose popular words over their rare alternatives.

2.2 Adversarial Training

The basic idea of our work to address the above problem is adversarial training, in which two or more models learn together by pursuing competing goals. A representative example of adversarial training is Generative Adversarial Networks (GANs) [13, 38] for image generation [36, 46, 2], in which a discriminator and a generator compete with each other: the generator aims to generate images similar to the natural ones, and the discriminator aims to detect the generated ones from the natural ones. Recently, adversarial training has been successfully applied to NLP tasks [7, 24, 23]. [7, 24] introduce an additional discriminator to differentiate the semantics learned from different languages in non-parallel bilingual data. [23] develops a discriminator to classify whether a sentence is created by human or generated by a model.

Our proposed method is under the adversarial training framework but not exactly the conventional generator-discriminator approach since there is no generator in our scenario. For an NLP task and its neural network model (including word embeddings), we introduce a discriminator to differentiate embeddings of popular words and rare words; while the NN model aims to fool the discriminator and minimize the task-specific loss simultaneously.

Our work is also weakly related to adversarial domain adaptation which attempts to mitigate the negative effects of domain shift between training and testing [10, 40]. The difference between this work and adversarial domain adaptation is that we do not target at the mismatch between training and testing; instead, we aim to improve the effectiveness of word embeddings and consequently improve the performance of end-to-end NLP tasks.

3 Empirical Study

In this section, we study the embeddings of popular words and rare words based on the models trained from Google News corpora using word2vec [1] and trained from WMT14 English-German translation task using Transformer [42]. The implementation details can be found in the supplementary material (part A).

Experimental Design In both tasks, we simply set the top 20% frequent words in vocabulary as popular words and denote the rest as rare words (roughly speaking, we set a word as a rare word if its relative frequency is lower than 10^{-6} in WMT14 dataset and 10^{-7} in Google News dataset). We have tried other thresholds such as 10% or 25% and found the observations are similar.

We study whether the semantic relationship between two words is reasonable. To achieve this, we randomly sampled some rare/popular words and checked the embeddings trained from different tasks. For each sampled word, we determined its nearest neighbors based on the cosine similarity between its embeddings and others'.[2] We also manually chose words which are semantically similar to it. For simplicity, for each word, we call the nearest words predicted from the embeddings as *model-predicted neighbors*, and call our chosen words as *semantic neighbors*.

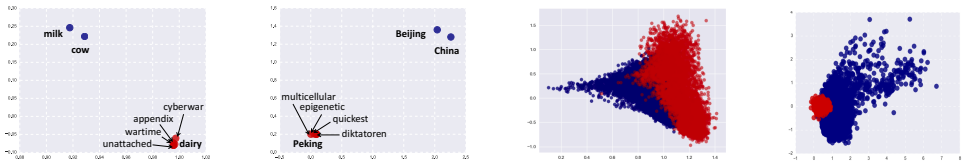
Observation To visualize word embeddings, we reduce their dimensionalities by SVD and plot two cases in Figure 1. More cases and other studies without dimensionality reduction can be found in the supplementary material (part C).

We find that the embeddings trained from different tasks share some common patterns. For both tasks, more than 90% of model-predicted neighbors of rare words are rare words. For each rare word, the model-predicted neighbor is usually not semantically related to this word, and semantic neighbors we chose are far away from it in the embedding space. In contrast, the model-predicted neighbors of popular words are very reasonable.

As the patterns in rare words are different from that of popular words, we further check the whole embedding matrix to make a general understanding. We also visualize the word embeddings using SVD by keeping the two directions with top-2 largest eigenvalues as in [30, 32] and plot them in Figure 1(c) and 1(d). From the figure, we can see that the embeddings actually *encode* frequencies to

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

[2] Cosine distance is the most popularly used metric in literature to measure semantic similarity [30, 35, 31]. We also have tried other metrics, e.g., Euclid distance, and the phenomena still exist.



(a) WMT En→De Case (b) Word2vec Case (c) WMT En→De (d) Word2vec

Figure 1: Case study of the embeddings trained from WMT14 translation task using Transformer and trained from Google News dataset using word2vec is shown in (a) and (b). (c) and (d) show the visualization of embeddings trained from WMT14 translation task using Transformer and trained from Google News dataset using word2vec. Red points represent rare words and blue points represent popular words. Red points represent rare words and blue points represent popular words. In (a) and (b), we highlight the semantic neighbors in bold.

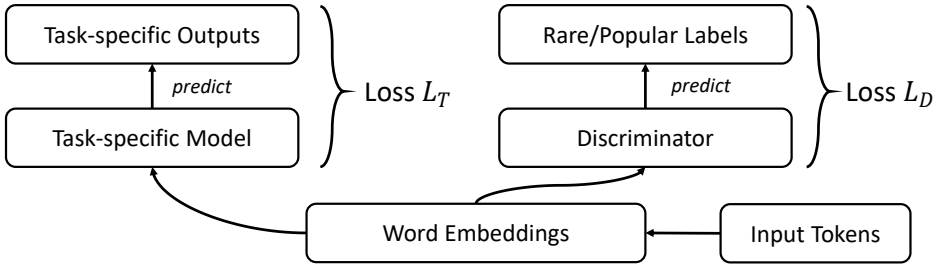


Figure 2: The proposed learning framework includes a task-specific predictor and a discriminator, whose function is to classify rare and popular words. Both modules use word embeddings as the input.

a certain degree: the rare words and popular words lie in different regions after this linear projection, and thus they occupy different regions in the original embedding space. This strange phenomenon is also observed in other learned embeddings (e.g.CBOW and GLOVE) and mentioned in [32].

Explanation From the empirical study above, we can see that the occupied spaces of popular words and rare words are different and here we intuitively explain a possible reason. We simply take word2vec as an example which is trained by stochastic gradient descent. During training, the sample rate of a popular word is high and the embedding of a popular word updates frequently. For a rare word, the sample rate is low and its embedding rarely updates. According to our study, on average, the moving distance of the embedding for a popular word is twice longer than that of a rare word during training. As all word embeddings are usually initialized around the origin with a small variance, we observe in the final model, the embeddings of rare words are still around the origin and the popular words have moved far away.

Discussion We have strong evidence that the current phenomena are problematic. First, according to our study,[3] in both tasks, more than half of the rare words are nouns, e.g., company names, city names. They may share some similar topics to popular entities, e.g., big companies and cities; around 10% percent of rare words include a hyphen (which is usually used to join popular words), and over 30% rare words are different PoSs of popular words. These words should have mixed or similar semantics to some popular words. These facts show that rare words and popular words should lie in the same region of the embedding space, which is different from what we observed. Second, as we can see from the cases, for rare words, model-predicted neighbors are usually not semantically related words but frequency-related words (rare words). This shows, for rare words, the embeddings encode more frequency information than semantic information. It is not good to use such word embeddings into semantic understanding tasks, e.g., text classification, language modeling, language understanding and translation.

[3]We use the POS tagger from Natural Language Toolkit, https://github.com/nltk.

4 Our Method

In this section, we present our method to improve word representations. As we have a strong prior that many rare words should share the same region in the embedding space as popular words, the basic idea of our algorithm is to train the word embeddings in an adversarial framework: We introduce a discriminator to categorize word embeddings into two classes: popular ones or rare ones. We hope the learned word embeddings not only minimize the task-specific training loss but also fool the discriminator. By doing so, the frequency information is removed from the embedding and we call our method frequency-agnostic word embedding (FRAGE).

We first define some notations and then introduce our algorithm. We develop three types of notations: embeddings, task-specific parameters/loss, and discriminator parameters/loss.

Denote $\theta^{emb} \in R^{d \times |V|}$ as the word embedding matrix to be learned, where d is the dimension of the embedding vectors and $|V|$ is the vocabulary size. Let V_{pop} denote the set of popular words and $V_{rare} = V \setminus V_{pop}$ denote the set of rare words. Then the embedding matrix θ^{emb} can be divided into two parts: θ^{emb}_{pop} for popular words and θ^{emb}_{rare} for rare words. Let θ^{emb}_w denote the embedding of word w. Let θ^{model} denote all the other task-specific parameters except word embeddings. For instance, for language modeling, θ^{model} is the parameters of the RNN or LSTM; for neural machine translation, θ^{model} is the parameters of the encoder, attention module and decoder.

Let $L_T(S; \theta^{model}, \theta^{emb})$ denote the task-specific loss over a dataset S. Taking language modeling as an example, the loss $L_T(S; \theta^{model}, \theta^{emb})$ is defined as the negative log likelihood of the data:

$$L_T(S; \theta^{model}, \theta^{emb}) = -\frac{1}{|S|} \sum_{y \in S} \log P(y; \theta^{model}, \theta^{emb}), \tag{1}$$

where y is a sentence.

Let f_{θ^D} denote a discriminator with parameters θ^D , which takes a word embedding as input and outputs a confidence score between 0 and 1 indicating how likely the word is a rare word. Let $L_D(V; \theta^D, \theta^{emb})$ denote the loss of the discriminator:

$$L_D(V; \theta^D, \theta^{emb}) = \frac{1}{|V_{pop}|} \sum_{w \in V_{pop}} \log f_{\theta^D}(\theta^{emb}_w) + \frac{1}{|V_{rare}|} \sum_{w \in V_{rare}} \log(1 - f_{\theta^D}(\theta^{emb}_w)). \tag{2}$$

Following the principle of adversarial training, we develop a minimax objective to train the task-specific model (θ^{model} and θ^{emb}) and the discriminator (θ^D) as below:

$$\min_{\theta^{model}, \theta^{emb}} \max_{\theta^D} L_T(S; \theta^{model}, \theta^{emb}) - \lambda L_D(V; \theta^D, \theta^{emb}), \tag{3}$$

where λ is a coefficient to trade off the two loss terms. We can see that when the model parameter θ^{model} and the embedding θ^{emb} are fixed, the optimization of the discriminator θ^D becomes

$$\max_{\theta^D} -\lambda L_D(V; \theta^D, \theta^{emb}), \tag{4}$$

which is to minimize the classification error of popular and rare words. When the discriminator θ^D is fixed, the optimization of θ^{model} and θ^{emb} becomes

$$\min_{\theta^{model}, \theta^{emb}} L_T(S; \theta^{model}, \theta^{emb}) - \lambda L_D(V; \theta^D, \theta^{emb}), \tag{5}$$

i.e., to optimize the task performance as well as fooling the discriminator. We train θ^{model}, θ^{emb} and θ^D iteratively by stochastic gradient descent or its variants. The general training process is shown in Algorithm 1.

5 Experiment

We test our method on a wide range of tasks, including word similarity, language modeling, machine translation and text classification. For each task, we choose the state-of-the-art architecture together with the state-of-the-art training method as our baseline [4].

[4] Code for our implement is available at https://github.com/ChengyueGongR/FrequencyAgnostic

Algorithm 1 Proposed Algorithm

Input: Dataset S, vocabulary $V = V_{pop} \cup V_{rare}$, θ^{model}, θ^{emb}, θ^{D}.
repeat
  Sample a minibatch $\hat{S}$ from S.
  Sample a minibatch $\hat{V} = \hat{V}_{pop} \cup \hat{V}_{rare}$ from V.
  Update θ^{model}, θ^{emb} by gradient descent according to Eqn. (5) with data $\hat{S}$.
  Update θ^{D} by gradient ascent according to Eqn. (4) with vocabulary $\hat{V}$.
until Converge
Output: θ^{model}, θ^{emb}, θ^{D}.

For fair comparisons, for each task, our method shares the same model architecture as the baseline. The only difference is that we use the original task-specific loss function with an additional adversarial loss as in Eqn. (3). Due to space limitations, we put dataset description, model description, hyper-parameter configuration into supplementary material (part A).

5.1 Settings

We conduct experiments on the following tasks.

Word Similarity evaluates the performance of the learned word embeddings by calculating the word similarity: it evaluates whether the most similar words of a given word in the embedding space are consistent with the ground-truth, in terms of Spearman's rank correlation. We use the skip-gram model as our baseline model [30][5], and train the embeddings using Enwik9[6]. We test the baseline and our method on three datasets: RG65, WS and RW. The RW dataset is a dataset for the evaluation of rare words. Following common practice [30, 1, 35, 31], we use cosine distance while computing the similarity between two word embeddings.

Language Modeling is a basic task in natural language processing. The goal is to predict the next word conditioned on previous words and the task is evaluated by perplexity. We do experiments on two widely used datasets [27, 28, 45], Penn Treebank (PTB) [29] and WikiText-2 (WT2) [28]. We choose two recent works as our baselines: the AWD-LSTM model[7] [27] and the AWD-LSTM-MoS model,[8] [45] which achieves state-of-the-art performance.

Machine Translation is a popular task in both deep learning and natural language processing. We choose two datasets: WMT14 English-German and IWSLT14 German-English datasets, which are evaluated in terms of BLEU score[9]. We use Transformer [42] as the baseline model, which achieves state-of-the-art accuracy on multiple translation datasets. We use *transformer_base* and *transformer_big* configurations following `tensor2tensor` [41][10].

Text Classification is a conventional machine learning task and is evaluated by accuracy. Following the setting in [22], we implement a Recurrent CNN-based model[11] and test it on AG's news corpus (AGs), IMDB movie review dataset (IMDB) and 20 Newsgroups (20NG).

In all tasks, we simply set the top 20% frequent words in vocabulary as popular words and denote the rest as rare words, which is the same as our empirical study. For all the tasks except word embedding, we use full-batch gradient descent to update the discriminator. For word embedding, mini-batch stochastic gradient descent is used to update the discriminator with a batch size 3000, since the vocabulary size is large. For language modeling and machine translation tasks, we use logistic regression as the discriminator. For other tasks, we find using a shallow neural network with

[5] https://github.com/tensorflow/models/blob/master/tutorials/embedding
[6] http://mattmahoney.net/dc/textdata.html
[7] https://github.com/salesforce/awd-lstm-lm
[8] https://github.com/zihangdai/mos
[9] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl
[10] To improve the training for imbalanced labeled data, a common method is to adjust loss function by reweighting the training samples; To regularize the parameter space, a common method is to use l_2 regularization. We tested these methods in machine translation and found the performance is not good. Detailed analysis is provided in the supplementary material (part B).
[11] https://github.com/brightmart/text_classification

one hidden layer is more efficient and we set the number of nodes in the hidden layer as 1.5 times embedding size. In all tasks, we set the hyper-parameter λ to 0.1. We list other hyper-parameters related to different task-specific models in the supplementary material (part A).

5.2 Results

RG65		WS		RW	
Orig.	**with FRAGE**	**Orig.**	**with FRAGE**	**Orig.**	**with FRAGE**
75.63	**78.78**	66.74	**69.35**	52.67	**58.12**

Table 1: Results on three word similarity datasets.

In this subsection, we provide the experimental results of all tasks. For simplicity, we use "with FRAGE" as our proposed method in the tables.

Word Similarity The results on three word similarity tasks are listed in Table 1. From the table, we can see that our method consistently outperforms the baseline on all datasets. In particular, we outperform the baseline for about 5.4 points on the rare word dataset RW. This result shows that our method improves the representation of words, especially the rare words.

		Paras	**Orig.**		**with FRAGE**	
			Validation	Test	Validation	Test
PTB	AWD-LSTM w/o finetune[27]	24M	60.7	58.8	60.2	58.0
	AWD-LSTM[27]	24M	60.0	57.3	58.1	56.1
	AWD-LSTM + continuous cache pointer[27]	24M	53.9	52.8	**52.3**	**51.8**
	AWD-LSTM-MoS w/o finetune[45]	24M	58.08	55.97	57.55	55.23
	AWD-LSTM-MoS[45]	24M	56.54	54.44	55.52	53.31
	AWD-LSTM-MoS + dynamic evaluation[45]	24M	48.33	47.69	**47.38**	**46.54**
WT2	AWD-LSTM w/o finetune[27]	33M	69.1	67.1	67.9	64.8
	AWD-LSTM[27]	33M	68.6	65.8	66.5	63.4
	AWD-LSTM + continuous cache pointer[27]	33M	53.8	52.0	**51.0**	**49.3**
	AWD-LSTM-MoS w/o finetune[45]	35M	66.01	63.33	64.86	62.12
	AWD-LSTM-MoS[45]	35M	63.88	61.45	62.68	59.73
	AWD-LSTM-MoS + dynamic evaluation[45]	35M	42.41	40.68	**40.85**	**39.14**

Table 2: Perplexity on validation and test sets on Penn Treebank and WikiText2. Smaller the perplexity, better the result. Baseline results are obtained from [27, 45]. "Paras" denotes the number of model parameters.

Language Modeling The results of language modeling on PTB and WT2 datasets are presented in Table 2. We test our model and the baselines at several checkpoints used in the baseline papers: without finetune, with finetune, with post-process (continuous cache pointer [14] or dynamic evaluation [21]). In all these settings, our method outperforms the two baselines. On PTB dataset, our method improves the AWD-LSTM and AWD-LSTM-MoS baseline by 0.8/1.2/1.0 and 0.76/1.13/1.15 points in test set at different checkpoints. On WT2 dataset, which contains more rare words, our method achieves larger improvements. We improve the results of AWD-LSTM and AWD-LSTM-MoS by 2.3/2.4/2.7 and 1.15/1.72/1.54 in terms of test perplexity, respectively.

Machine Translation The results of neural machine translation on WMT14 English-German and IWSLT14 German-English tasks are shown in Table 3. We outperform the baselines for 1.06/0.71 in the term of BLEU in *transformer_base* and *transformer_big* settings in WMT14 English-German

WMT En→De		IWSLT De→En	
Method	**BLEU**	**Method**	**BLEU**
ByteNet[19]	23.75	DeepConv[11]	30.04
ConvS2S[12]	25.16	Dual transfer learning [43]	32.35
Transformer Base[42]	27.30	ConvS2S+SeqNLL [9]	32.68
Transformer Base with FRAGE	**28.36**	ConvS2S+Risk [9]	32.93
Transformer Big[42]	28.40	Transformer	33.12
Transformer Big with FRAGE	**29.11**	Transformer with FRAGE	**33.97**

Table 3: BLEU scores on test set on WMT2014 English-German and IWSLT German-English tasks.

task, respectively. The model learned from adversarial training also outperforms original one in IWSLT14 German-English task by 0.85. These results show improving word embeddings can achieve better results in more complicated tasks and larger datasets.

Text Classification The results are listed in Table 4. Our method outperforms the baseline method for 1.26%/0.66%/0.44% on three different datasets.

AG's		IMDB		20NG	
Orig.	**with FRAGE**	**Orig.**	**with FRAGE**	**Orig.**	**with FRAGE**
90.47%	**91.73%**	92.41%	**93.07%**	96.49%[22]	**96.93%**

Table 4: Accuracy on test sets of AG's news corpus (AG's), IMDB movie review dataset (IMDB) and 20 Newsgroups (20NG) for text classification.

As a summary, our experiments on four different tasks with 10 datasets verify the effectiveness of our method. We provide some case studies and visualizations of our method in the supplementary material (part C), which show that the semantic similarities are reasonable and the popular/rare words are better mixed together in the embedding space.

6 Conclusion

In this paper, we find that word embeddings learned in several tasks are biased towards word frequency: the embeddings of high-frequency and low-frequency words lie in different subregions of the embedding space. This makes learned word embeddings ineffective, especially for rare words, and consequently limits the performance of these neural network models. We propose a neat, simple yet effective adversarial training method to improve the model performance which is verified in a wide range of tasks.

We will explore several directions in the future. First, we will investigate the theoretical aspects of word embedding learning and our adversarial training method. Second, we will study more applications which have the similar problem even beyond NLP.

References

[1] R. Al-Rfou, B. Perozzi, and S. Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[4] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.

[8] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[9] S. Edunov, M. Ott, M. Auli, D. Grangier, and M. Ranzato. Classical structured prediction losses for sequence to sequence learning. *arXiv preprint arXiv:1711.04956*, 2017.

[10] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[11] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*, 2016.

[12] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[14] E. Grave, A. Joulin, and N. Usunier. Improving neural language models with a continuous cache. *CoRR*, abs/1612.04426, 2016.

[15] S. Hingmire, S. Chougule, G. K. Palshikar, and S. Chakraborti. Document classification by topic labeling. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pages 877–880, 2013.

[16] E. Hoffer, I. Hubara, and D. Soudry. Fix your classifier: the marginal value of training the last weight layer. *ICLR*, 2018.

[17] P.-S. Huang, C. Wang, D. Zhou, and L. Deng. Neural phrase-based machine translation. *arXiv preprint arXiv:1706.05565*, 2017.

[18] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[19] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

[20] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

[21] B. Krause, E. Kahembwe, I. Murray, and S. Renals. Dynamic evaluation of neural sequence models. *CoRR*, abs/1709.07432, 2017.

[22] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.

[23] A. M. Lamb, A. G. A. P. GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[24] G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017.

[25] R. R. Larson. Introduction to information retrieval. *Journal of the American Society for Information Science and Technology*, 61(4):852–853, 2010.

[26] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[27] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182, 2017.

[28] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016.

[29] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. 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, 2010.

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[31] J. Mu, S. Bhat, and P. Viswanath. All-but-the-top: simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*, 2017.

[32] J. Mu, S. Bhat, and P. Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *CoRR*, abs/1702.01417, 2017.

[33] M. Ott, M. Auli, D. Granger, and M. Ranzato. Analyzing uncertainty in neural machine translation. *arXiv preprint arXiv:1803.00047*, 2018.

[34] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[35] J. Pennington, R. Socher, and C. D. Manning. 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, 2014.

[36] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[37] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[38] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[39] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[40] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2962–2971, 2017.

[41] A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, L. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, and J. Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[43] Y. Wang, Y. Xia, L. Zhao, J. Bian, T. Qin, G. Liu, and L. Tie-Yan. Dual transfer learning for neural machine translation with marginal distribution regularization.

[44] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[45] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. *CoRR*, abs/1711.03953, 2017.

[46] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

A Experimental settings

A.1 Dataset Description

For word similarity, we use three test datasets. WordSim-353 (WS) dataset consists of 353 pairs of commonly used verbs and nouns; The rare-words (RW) dataset contains rarely used words; The RG65 dataset contains 65 word pairs, and the similarity values in the dataset are the means of judgments made by 51 subjects.

For language modeling tasks, we use Penn Treebank dataset and WikiText-2 dataset. The details of the datasets are provided in Table 5.

	Penn Treebank			**WikiText-2**		
	Train	**Valid**	**Test**	**Train**	**Valid**	**Test**
Tokens	887,521	70,390	78,669	2,088,628	217,646	245,569
Vocab	10,000			33,278		
OOV	4.8%			2.6%		

Table 5: Statistics of the Penn Treebank, and WikiText-2 dataset used in language modeling. The out of vocabulary (OOV) words will be replaced by <unk> during training and testing.

For machine translation, we use WMT14 English-German and IWSLT14 German-English datasets. The training set of WMT14 English-German task consists of 4.5M sentence pairs. Source and target tokens are processed into 37K shared sub-word units based on byte-pair encoding (BPE) [39]. We use the concatenation of newstest2012 and newstest2013 as the validation set and use newstest2014 as the test set following all previous works. IWSLT14 German-English dataset contains 160K training sentence pairs and 7K validation sentence pairs. Tokens are processed using BPE and eventually

we obtain a shared vocabulary of about 32K tokens. We use the concatenation of dev2010, tst2010, tst2011 and tst2011 as the test set, which is widely adopted in [37, 17, 4].

For text classification tasks, we use three datasets: AG's News, IMDB and 20NG. AG's news corpus is a news article corpus with categorized articles from more than 2,000 news. IMDB movie review dataset is a sentiment classification dataset. It consists of movie review comments with binary sentiment labels. 20 Newsgroups is an email collection dataset, in which the emails are categorized into 20 different groups. We use the bydate version and select 4 major categories (comp, politics, rec, and religion) following [15]. Table 6 shows detailed information.

Dataset	Ave. Len	Max Len	#Classes	#Train : #Text
AG's News	34	211	4	120000 : 7600
IMDB	281	2956	2	25000 : 25000
20NG	429	11924	4	7250 : 5563

Table 6: Detailed statistics about text classification datasets.

A.2 Hyper-parameter configurations

The hyper-parameters used for AWD-LSTM with/without MoS in language modeling experiment is shown in Table 7.

For machine translation tasks, we choose Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\varepsilon = 10^{-9}$, and follow the learning rate schedule in [42]. For evaluation, we use the case-sensitive tokenized BLEU score [34] for WMT14 English-German and case-insensitive tokenized BLEU score [34] for IWSLT14 German-English. The hyper-parameters used in machine translation task are summarized in Table 8. The hyper-parameters used in word embedding task are summarized in Table 9.

For all text classification tasks, we use convolutional kernel with size 2, 3, 5. We implement batch normalization and shortcut connection, and use Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\varepsilon = 10^{-8}$.

	AWD-LSTM		**+ MoS**	
	PTB	**WT2**	**PTB**	**WT2**
Learning rate	30	30	20	15
Batch size	20	80	12	15
Embedding size	400	400	280	300
Number of mixture components	-	-	15	15
Word-level V-dropout	0.10	0.10	0.10	0.10
Embedding V-dropout	0.65	0.65	0.55	0.40
Hidden state V-dropout	0.30	0.30	0.20	0.225
Recurrent weight dropout	0.50	0.50	0.50	0.50
Context vector V-dropout	0.30	0.30	0.30	0.30
λ_{sm}	0.09	0.16	-	-
θ	0.75	1.4	-	-
Window size	700	4200	-	-
λ	-	-	0.04	-
ϵ	-	-	0.018	-

Table 7: Hyper-parameter used for AWD-LSTM and AWD-LSTM-MoS on PTB and WT2.

	IWSLT De→En	WMT En→De	
		Transformer Base	Transformer Big
Hidden Layer	5	6	6
Hidden size	256	512	1024
Embedding size	256	512	1024
Head number	4	8	16
Learning rate	0.2	0.2	0.2
Learning rate Warmup step	8000	8000	8000
Batch size(word)	4096	7168	7168
Attention dropout	0.1	0.1	0.1
Relu dropout	0.1	0.1	0.1
Preprocess dropout	0.1	0.1	0.3
Warm-start step	50000	50000	50000
Beam size	4	8	10
Length penalty	0.85	1.1	1.1

Table 8: Hyper-parameter used for neural machine translation on IWSLT14 German-English and WMT14 English-German.

hyper-parameter	Skip-gram
Learning rate	0.20
Embedding size	300
Negative Samples	100
Window size	5
Min count	5

Table 9: Hyper-parameter used for word embedding training.

A.3 Models Description

We use task-specific baseline models. In language modeling, AWD-LSTM [27] is a weight-dropped LSTM which uses Drop Connect on hidden-to-hidden weights as a means of recurrent regularization. The model is trained by NT-ASGD, which is a variant of the averaged stochastic gradient method. The training process has two steps, in the second step, the model is finetuned using another configuration of NT-ASGD. AWD-LSTM-MoS [45] uses the *Mixture of Softmaxes* structure to the vanilla AWD-LSTM and achieves the state-of-the-art result on PTB and WT2.

For machine translation, Transformer [42] is a recently developed architecture in which the *self-attention network* is used during encoding and decoding step. It achieves the best performances on several machine translation tasks, e.g. WMT14 English-German, WMT14 English-French datasets.

Word2vec [30] is one of the pioneer works on using deep learning to NLP tasks. Based on the co-occurrence of words, it produces distributed representations of words (word embeddings).

RCNN [22] contains both recurrent and convolutional layers to catch the key components in texts, and is widely used in text classification tasks.

B Additional Comparisons

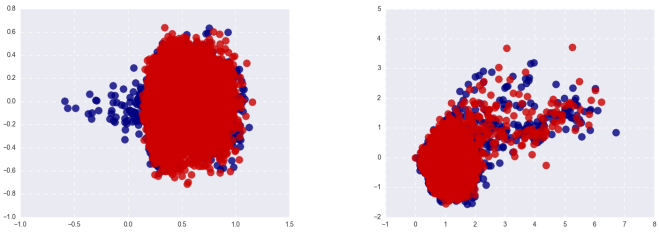
We compare some other simple methods with ours on machine translation tasks, which include reweighting method and l_2 regularization (weight decay). Results are listed in Table 10. We notice that those simple methods do not work for the tasks, even have negative effects.

WMT En→De		IWSLT De→En	
Method	**BLEU**	**Method**	**BLEU**
Transformer Base [42]	27.30	Transformer	33.12
Transformer Base + Reweighting	26.04	Transformer + Reweighting	31.04
Transformer Base + Weight Decay	26.76	Transformer + Weight Decay	32.52
Transformer Base with FRAGE	**28.36**	Transformer with FRAGE	**33.97**

Table 10: BLEU scores on test set of the WMT14 English-German task and IWSLT14 German-English task. Our method is denoted as "FRAGE", "Reweighting" denotes reweighting the loss of each word by reciprocal of its frequency, and "Weight Decay" denotes putting weight decay rate (0.2) on embeddings.

C Case Study on Original Models and Qualitative Analysis of Our Method

We provide more word similarity cases in Table 11 to justify our statement in Section 3. We also present the effectiveness of our method by showcase and embedding visualizations. From the cases and visualizations in Table 12 and Figure 3, we find the word similarities are improved and popular/rare words are better mixed together.



(a) (b)

Figure 3: These figures show that, in different tasks, the embeddings of rare and popular words are better mixed together after applying our method.

internet	quite	always	energy
Model-predicted neighbor			
web	pretty	usually	fuel
iphone	almost	constantly	power
software	truly	often	radiation
finances*	utterly*	definitely	water
Semantic neighbor + Model-predicted Ranking			
web:2	pretty:1	often:3	power:2
computer:14	fairly:8	frequently:93	strength:52

citizens	citizenship*	accepts*	bacterial*
Model-predicted neighbor			
clinicians*	bliss*	announces*	multicellular*
astronomers*	pakistanis*	digs*	epigenetic*
westliche	dismiss*	externally*	isotopic*
adults	reinforces*	empowers*	conformational*
Semantic neighbor + Model-predicted Ranking			
citizen*:771	citizen*:10745	accepted*:21109	bacteria*:116
citizenship*:832	citizens:11706	accept:30612	chemical:233

dairy*	android*	surveys*	peking*
Model-predicted neighbor			
unattached*	1955*	schwangerschaften*	multicellular*
wartime*	1926*	insurgent*	epigenetic*
cyberwar*	doctoral*	pregnancies*	quickest*
appendix*	championships*	biotechnology*	diktatoren*
Semantic neighbor + Model-predicted Ranking			
milk*:10165	java:13498	investigate*:16926	beijing:30938
cow:14351	google:14513	survey:3397	china:31704

Table 11: Case study for the original models. Rare words are marked by "*". For each word, we list its model-predicted neighbors. Moreover, we also show the ranking positions of the semantic neighbors based on cosine similarity. As we can see, the ranking positions of the semantic neighbors are very low.

Orig. with FRAGE		Orig. with FRAGE	
Word: citizens	**Word: citizenship***	**Word: accepts***	**Word: bacterial***
Model-predicted neighbor			
homes	population	registered	myeloproliferative*
citizen*	städtischen*	tolerate*	metabolic*
bürger	dignity	recognizing*	bacteria*
population	bürger	accepting*	apoptotic*
Semantic neighbor + Model-predicted Ranking			
citizen*:2	citizen*:79	accepted*:26	bacteria* : 3
citizenship*:40	citizens:7	accept:29	chemical: 8

Table 12: Case study for our method. The word similarities are improved