Self-Knowledge Distillation in Natural Language Processing

Sangchul Hahn
Handong Global University
Pohang, South Korea
schahn21@gmail.com

Heeyoul Choi
Handong Global University
Pohang, South Korea
heeyoul@gmail.com

Abstract

Since deep learning became a key player in natural language processing (NLP), many deep learning models have been showing remarkable performances in a variety of NLP tasks, and in some cases, they are even outperforming humans. Such high performance can be explained by efficient knowledge representation of deep learning models. While many methods have been proposed to learn more efficient representation, knowledge distillation from pre-trained deep networks suggest that we can use more information from the soft target probability to train other neural networks. In this paper, we propose a new knowledge distillation method *self-knowledge distillation*, based on the soft target probabilities of the training model itself, where multimode information is distilled from the word embedding space right below the softmax layer. Due to the time complexity, our method approximates the soft target probabilities. In experiments, we applied the proposed method to two different and fundamental NLP tasks: language model and neural machine translation. The experiment results show that our proposed method improves performance on the tasks.

1 Introduction

Deep learning has achieved the state-of-the-art performance on many machine learning tasks, such as image classification, object recognition, and neural machine translation (He et al., 2016; Redmon and Farhadi, 2017; Vaswani et al., 2017) and outperformed humans on some tasks. In deep learning, one of the critical points for success is to learn better representation of data with many layers (Bengio et al., 2013) than other machine learning algorithms. In other words, if we make a model to learn better representation of data, the model can show better performance.

In natural language processing (NLP) tasks like language modeling (LM) (Bengio et al., 2003; Mikolov et al., 2013) and neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015), when the models are trained, they are to generate many words in sentence, which is a sequence of classification steps, for each of which they choose a target word among the whole words in the dictionary. That is why LM and NMT are usually trained with the sum of cross-entropies over the target sentence. Thus, although language related tasks are more of generation rather than classification, the models estimate target probabilities with the softmax operation on the previous neural network layers and the target distributions are provided as one-hot representations. As data representation in NLP models, word symbols should also be represented as vectors.

In this paper, we focus on the word embedding and the estimation of the target distribution. In NLP, word embedding is a step to translate word symbols (indices in the vocabulary) to vectors in a continuous vector space and is considered as a standard approach to handle symbols in neural networks. When two words have semantically or syntactically similar meanings, the words are represented closely to each other in a word embedding space. Thus, even when the prediction is not exactly correct, the predicted word might not be so bad, if the estimated word is very close to the target word in the embedding space like 'programming' and 'coding'. That is, to check how wrong the prediction is, the word embedding can be used. There are several methods to obtain word embedding matrices (Mikolov et al., 2013; Pennington et al., 2014), in addition to neural language models (Bengio et al.,

2003; Mikolov et al., 2010). Recently, several approaches have been proposed to make more efficient word embedding matrices, usually based on contextual information (Søgaard et al., 2017; Choi et al., 2017).

On the other hand, knowledge distillation was proposed by (Hinton et al., 2015) to train new and usually shallow networks using hidden knowledge in the probabilities produced by the pretrained networks. It shows that there is knowledge not only in the target probability corresponding to the target class but also in the other class probabilities in the estimation of the trained model. In other words, the other class probabilities can contain additional information describing the input data samples differently even when the samples are in the same class. Also, samples from different classes could produce similar distributions to each other.

In this paper, we propose a new knowledge distillation method, *self-knowledge distillation* (SKD) based on the word embedding of the training model itself. That is, self-knowledge is distilled from the predicted probabilities produced by the training model, expecting the model has more information as it is more trained. In the conventional knowledge distillation, the knowledge is distilled from the estimated probabilities of pretrained (or teacher) models. Contrary, in the proposed SKD, knowledge is distilled from the current model in the training process, and the knowledge is hidden in the word embedding. During the training process, the word embedding reflects the relationship between words in the vector space. A word close to the target word in the vector space is expected to have similar distribution after softmax, and such information can be used to approximate the soft target probability as in knowledge distillation. We apply our proposed method to two popular NLP tasks: LM and NMT. The experiment results show that our proposed method improves the performance of the tasks. Moreover, SKD reduces overfitting problems which we believe is because SKD uses more information.

The paper is organized as follows. Background is reviewed in Section 2. In Section 3, we describe our proposed method, SKD. Experiment results are presented and analyzed in Section 4, followed by Section 5 with conclusion.

2 Background

In this section, we briefly review the cross-entropy and knowledge distillation. Also, since our proposed method is based on word embedding, the layer right before the softmax operation, word embedding process is summarized.

2.1 Cross Entropy

For classification with C classes, neural networks produce class probabilities p_i, $i \in \{0, 1, ...C\}$ by using a softmax output layer which calculates class probabilities from the logit, z_i considering the other logits as follows.

$$p_i = \frac{\exp(z_i)}{\sum_k \exp(z_k)}. \quad (1)$$

In most classification problems, the objective function for a single sample is defined by the cross-entropy as follows.

$$J(\theta) = -\sum_k y_k \log p_k, \quad (2)$$

where y_k and p_k are the target and predicted probabilities. The cross-entropy can be simply calculated by

$$J(\theta) = -\log p_t, \quad (3)$$

when the target probability $\boldsymbol{y}$ is a one-hot vector defined as

$$y_k = \begin{cases} 1, & \text{if } k = t(\text{target class}) \\ 0, & \text{otherwise} \end{cases}. \quad (4)$$

Note that the cross-entropy objective function says only how likely input samples belong to the corresponding target class, and it does not provide any other information about the input samples.

2.2 Knowledge Distillation

A well trained deep network model contains meaningful information (or knowledge) extracted from training datasets for a specific task. Once a deep model is trained for a task, the trained model can be used to train new smaller (shallower or thinner) networks as shown in (Hinton et al., 2015; Romero et al., 2014). This approach is referred to as *knowledge distillation*.

Basically, knowledge distillation provides more information to new models for training and improves the new model's performance. Thus, when a new model which is usually smaller is trained

with the distilled knowledge from the trained deep model, it can achieve a similar (or sometimes even better) performance compared to the pretrained deep model.

In the pretrained model, knowledge lies in the class probabilities produced by softmax of the model as in Eq. (1). All probability values including the target class probability describe relevant information about the input data. Thus, instead of one-hot representation of the target label where only the target class is considered in cross-entropy, all probabilities over the whole classes from the pretrained model can provide more information about the input data in cross-entropy, and can teach new models more efficiently. All probabilities from the pretrained model are considered as *soft target probabilities*.

In a photo tagging task, depending on the other class probabilities, we understand the input image better than just target class. When a class 'mouse' has the highest probability, if 'mascot' has a relatively high probability, then the image would be probably 'mickey mouse'. If 'button' or 'pad' has a high probability, the image would be a mouse as a computer device. The other class probabilities have some extra information and such knowledge in the pretrained model can be transferred to a new model by using a soft target distribution of the training set.

When the target labels are available, the objective function is a weighted sum of the conventional cross-entropy with the correct labels and the cross-entropy with the soft target distribution, given by

$$J(\theta) = -(1-\lambda)\log p_t - \lambda \sum_k q_k \log p_k, \quad (5)$$

where p_k is probability for class k produced by current model with parameter θ, and q_k is the soft target probability from the pretrained model. λ controls the amount of knowledge from the trained model. Note that the conventional knowledge distillation extracts knowledge from a pretrained model, and in this paper, we propose to extract knowledge from the current model itself without any pretrained model.

Furthermore, in a recently proposed paper by (Furlanello et al., 2018), they proved that knowledge distillation can be useful to train a new model which has the same size and the same architecture as the pretrained model. They trained a teacher model first, then they trained a student model with distilled knowledge from the teacher model. Their experiment results show that the student models outperform the teacher model. Also, even though when the teacher model has a less powerful architecture, the knowledge from the trained teacher model can boost student models which have more powerful (or bigger) architectures. It means that even the knowledge is distilled from a relatively weak model, it can be useful to train a bigger model.

2.3 Word Embedding

Word embedding is to convert symbolic representation of words to vector representation with semantic and syntactic meanings, which reflects the relations between words. Including CBOW, Skip-gram (Mikolov et al., 2013), and GloVe (Pennington et al., 2014), various word embedding methods have been proposed to learn a word embedding matrix. The trained embedding matrix can be transferred to other models like LM or NMT (Ahn et al., 2016).

CBOW predicts a word given its neighbor words, and Skip-gram predicts the neighbor words given a word. They use feedforward layers, and the last layer of CBOW includes the word embedding matrix, $\boldsymbol{W}$, as follows.

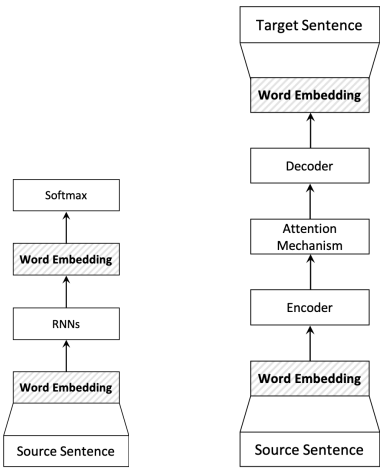
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{h} + \boldsymbol{b}, \quad (6)$$

where $\boldsymbol{b}$ is a bias, $\boldsymbol{h}$ is hidden layer, and $\boldsymbol{z}$ is logits for the softmax operation.

Words in the embedding space have semantic and syntactic similarities, such that two similar words are close in the space. Thus, when the classification is not correct, the error can be interpreted differently depending on the similarity between the predicted word and the target word. For example, when the target word is 'learning', if the predicted word is 'training', then it is less wrong than other words like 'flower' or 'internet'. In this paper, we utilize such hidden information (or knowledge) in the word embedding space, while training. Fig. 1 shows where the word embedding is located in LM and NMT, respectively.

3 Self-Knowledge Distillation

We propose a new learning method *self-knowledge distillation* (SKD) which distills knowledge from a currently training model, following the conventional knowledge distillation. In this section, we describe an algorithm for SKD and its application to language model and neural machine translation.



(a) Language Model (b) NMT Model

Figure 1: Network architectures of LM and NMT. Word embedding is presented as gray boxes in the models.

3.1 SKD Equations

In order to apply knowledge distillation on a current training model, we need to obtain soft target probabilities as q_k in Eq. (5) for all classes, but they are not available explicitly. However, when the model is trained enough, then the word embedding has such information implicitly. If a word $\boldsymbol{w}_i$ is close to $\boldsymbol{w}_j$ in the embedding space, the probability p_i would be close to p_j for a given input sample.

When t is the target class, we calculate the soft target probabilities q_k based on the word embedding. First, we assume that q_t should be high, and if $\boldsymbol{w}_k$ is close to $\boldsymbol{w}_t$ in the embedding space, q_k should be also high. That is, the Euclidean distance between words is used to estimate the soft target probability. The other class probabilities (or soft target probabilities) q_k can be obtained by

$$q_k = \frac{1}{Z} \exp\{-\sigma \|\boldsymbol{w}_t - \boldsymbol{w}_k\|_2\}, \qquad (7)$$

where $\|\cdot\|_2$ is $l2$-norm, and Z is a normalization term. σ is a scale parameter and its value depends on the average distance to the corresponding nearest neighbors in the word embedding space. However, due to the expensive computational cost, we do not calculate q_k for all classes, and we choose just one of the other classes, which is the predicted class of the current model.

Assuming that the model predicts a class n for a given input sample, only q_t and q_n are used as distilled knowledge. We clip the q_n value with 0.5, meaning that the class n cannot be more correct than the real target t, so Eq. (7) becomes

$$\begin{aligned} q_n &= \min\{\exp\{-\sigma \|\boldsymbol{w}_t - \boldsymbol{w}_n\|_2\}, 0.5\}, \\ q_t &= 1 - q_n, \end{aligned} \qquad (8)$$

where $q_n + q_t = 1$. That is, we consider only two soft target probabilities as shown in Fig. 2. Note that we use Euclidean distance between $\boldsymbol{w}_t$ and $\boldsymbol{w}_n$ to calculate q_n, but other approaches like inner product would be possible.

Now, the objective function of SKD becomes similar to Eq. (5), and is defined by

$$\begin{aligned} J(\theta) &= -(1-\lambda)\log p_t \\ &\quad -\lambda(q_t \log p_t + q_n \log p_n), \end{aligned} \qquad (9)$$

where the second term of Eq. (5) is approximated by $\lambda(q_t \log p_t + q_n \log p_n)$, ignoring the other class probabilities. Eq. (9) can be rewritten simply as follows.

$$J(\theta) = -(1 - \lambda q_n)\log p_t - \lambda q_n \log p_n. \qquad (10)$$

Eqs. (9) or (10) can be understood in three cases. First, if the prediction is correct ($n = t$), then Eq. (9) is the same as the conventional cross-entropy objective. Second, if $\boldsymbol{w}_n$ is far from $\boldsymbol{w}_t$ in the word embedding space, then q_n is close to zero and Eq. (9) becomes close to the conventional cross-entropy objective. Finally, if $\boldsymbol{w}_n$ is close to $\boldsymbol{w}_t$ (e.g. q_n = 0.4), it approximates the soft target probability with only two classes t and n, and the model is trained to produce probabilities for class t and n as close as q_t and q_n. This approach trains the model with different targets for different input samples.

Fig. 2 presents how SKD obtains simplified soft target distribution based on the distance of target and estimated vectors in the word embedding space.

3.2 SKD Algorithm

Since SKD distills knowledge from the current training model, at the beginning of the training process, the model does not contain relevant information. That is, we cannot extract any knowledge from the training model at the beginning. Thus, we start training process without knowledge distillation at first and gradually increase the amount of knowledge distillation as the training iteration goes. So, our algorithm starts with the conventional cross-entropy objective function in Eq. (3), and after training the model for a while, it gradually

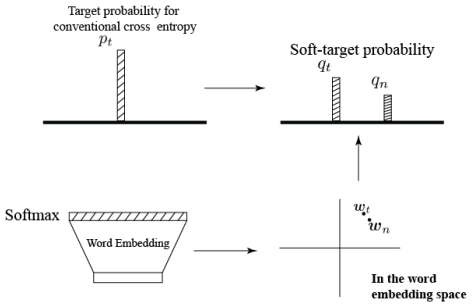


Figure 2: Given a target class t, a soft target probabilities are obtained based on the distance in the word embedding space. However, only the target class and the predicted class have soft target probabilities in SKD.

transits to Eq. (10). To implement the transition, another parameter α is introduced to Eq. (10), leading to the final objective function as follows.

$$J(\theta) = -(1 - \alpha\lambda q_n) \log p_t - \alpha\lambda q_n \log p_n, \quad (11)$$

α starts from 0 with which Eq. (11) becomes the conventional cross-entropy. After K iterations, α increases by η per iteration and eventually goes up to 1 with which Eq. (11) becomes the same as Eq. (9). In our experiments, we used a simplified equation as in Eq. (12) without λ so that the objective function relies gradually more on the soft target probabilities as training goes.

$$J(\theta) = -(1 - \alpha q_n) \log p_t - \alpha q_n \log p_n. \quad (12)$$

Table 1 summarizes the proposed SKD algorithm.

Table 1: Self-Knowledge Distillation Algorithm

Algorithm 1: SKD Algorithm
Initialize the model parameters θ
Initialize $\alpha = 0$ and σ
(See the experiments for σ values.)
Repeat K times:
Train the network based on the cross-entropy in Eq. (3)
Repeat until convergence:
Train the network based on the SKD objective function in Eq. (12)
Update α with $\alpha + \eta$
(See the experiments for η values.)

3.3 NLP Tasks

SKD is applied to two different NLP tasks: language modeling (LM), and neural machine translation (NMT). Although LM and NMT are actually sentence generation rather than classification, they have classification steps to generate words for the target sentence. Also, the sum of cross-entropies over the words in the sentence is adapted as an objective function for them.

In addition, to check if SKD is robust against errors in the word embedding space, we also evaluate SKD when we add Gaussian noise in the word embedding space for target words in the decoder.

4 Experiments

To evaluate self-knowledge distillation, we compare it to the baseline models for language modeling and neural machine translation.

4.1 Dataset

For language modeling, we use two different datasets: Penn TreeBank (PTB) and WiKi-2. PTB was made by (Marcus et al., 1993), and we use the pre-processed version by (Mikolov et al., 2010). In the PTB dataset, the train, valid and test sets have about 887K, 70K, and 78K tokens, respectively, where the vocabulary size is 10K. The WiKi-2 dataset introduced by (Merity et al., 2016) consists of sentences that are extracted from Wikipedia. It has about 2M, 217K, and 245K tokens for train, valid, and test sets. Its vocabulary size is about 33K. We did not apply additional pre-processing for the PTB dataset. The WiKi-2 dataset is pre-tokenized data, therefore we only added an end-of-sentence token (<EOS>) to every sentence.

For machine translation, we evaluated models on three different translation tasks (En-Fi, Fi-En, and En-De) with the available corpora from WMT'15 [1]. The dictionary size is 10K for En-fi and Fi-En translation task, and 30K for the En-De translation task.

4.2 Language Modeling

Language modeling (LM) has been used in many different NLP tasks like automatic speech recognition (ASR), and machine translation (MT) to capture syntactic and semantic structure of a natural language. The neural network-based language models (NNLM) and recurrent neural network language model (RNNLM) catch the syntactic and semantic regularities of an input language (Bengio et al., 2003; Mikolov et al., 2013). RNNLM is our baseline, which consists of a single LSTM layer and

[1] http://www.statmt.org/wmt15/translation-task.html

single feed forward layer with ReLU (Le et al., 2015).

We evaluate four models: Baseline, Noise (with Gaussian noise on the word embedding), SKD, and Noise+SKD. To show that the information by SKD is more knowledgeable than random noise, we tested a noise injected model which injects only Gaussian noise to the word embedding space. The word dimension is set to 500 and the number of hidden nodes is 400 for all models. We set the σ and η in the SKD algorithm in Table 1 0.1 (both PTB and WiKi-2 dataset) and 0.0002 (PTB), 0.00011 (WiKi-2), respectively. We applied the SKD object function after 500 batches for PTB and 900 batches for WiKi-2. Note that Wiki-2 data is larger than PTB.

The evaluation metric is the negative log-likelihood (NLL) for each sentence (the lower is the better). Table 2 presents NLLs for the test data of two datasets with different models. It shows that our proposed methods (both noise injection and self-distillation knowledge) improve the results in the LM task. Note that SKD provides more knowledgeable information than Gaussian noise.

Table 2: NLLs for LM with different models on PTB and Wiki-2.

Model	PTB	Wiki-2
Baseline	101.40	119.49
+Noise	101.28	118.70
+SKD	99.38	116.85
+Noise+SKD	**97.41**	**116.60**

4.3 Neural Machine Translation

NMT has been widely used in machine translation research, because of its powerful performance and end-to-end training (Sutskever et al., 2014; Bahdanau et al., 2015; Johnson et al., 2017). Attention-based NMT models consist of an encoder, a decoder, and the attention mechanism (Bahdanau et al., 2015), which is our baseline in this paper except for replacing GRU with LSTM and using BPE (Sennrich et al., 2016). The encoder takes the sequence of source words in the word embedding form. The decoder works in a similar way to LM, except the attention mechanism. See (Bahdanau et al., 2015) for NMT and the attention mechanism in detail.

In the experiments, we check how much SKD can improve model's performance using the simple baseline architecture. Since SKD modifies only the objective function, we believe that the improvement by SKD is regardless of model architectures.

Table 3 shows that our proposed method improves NMT performance by around 1 BLEU score. For qualitative comparison, some translation results are presented below. The overall quality of translation of the SKD model looks better than baseline model's. In other words, when the BLEU scores are similar, the sentences translated by the SKD model look better.

- (src) Hallituslähteen mukaan tämä on yksi monista ehdotuksista, joita harkitaan.
  (trg) A governmental source says this is one of the many proposals to be considered.
  (baseline) *According to government revenue, this is one of the many proposals that are being considered to be considered.*
  (SKD) *According to the government, this is one of the many proposals that are being considered.*

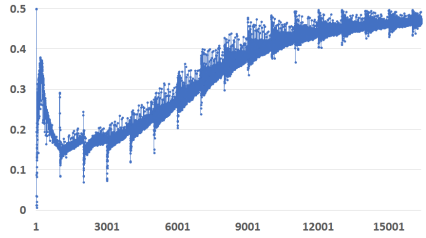
- (src) Meillä on hyvä tunne tuotantoketjunvahvuudesta.
  (trg) We feel very good about supply chain capability.
  (baseline) *We have good knowledge of the strength of the production chain.*
  (SKD) *We have a good sense of the strength of the production chain.*

- (src) En ole oikein tajunnut, että olen näin vanha.
  (trg) I haven't really realized that I'm this old.
  (baseline) *I have not been right to realise that I am so old.*
  (SKD) *I am not quite aware that I am so old.*

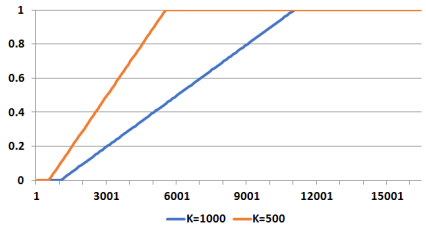
- (src) Ne vaikuttavat vasta tulevaisuudessa.
  (trg) They'll have an impact in the future only.
  (baseline) *They will only be affected in the future.*
  (SKD) *They will only affect in the future.*

Fig. 3 shows a trajectory of the q_n values and scheduling of the α value during training the En-Fi NMT model described in Eq. (12), respectively. As expected, the q_n value becomes larger than 0.5 which means that w_n (the predicted word vector) is close enough to the w_t (the target word vector). Fig. 3(b) shows the scheduled value of α in Eq. (12). The α value starts from 0 and increases up to 1 while training. The model is trained with only the cross-entropy for K iterations, and then when the model captures enough knowledge to be distilled, α increases to utilize knowledge from the model.

Also, as in Fig. 4, the SKD models are not (or more slowly) overfitted to the training data. We believe that SKD provides more information distilled by the training model itself to prevent overfitting. Note that there is no significant difference in the improvements by SKD and Noise, but Noise+SKD



(a) q_n value during NMT model training



(b) Scheduling of α value of NMT training

Figure 3: (a) Change of q_n value during NMT model training for En-Fi translation task, and (b) scheduling of α value in Eq. (12) of NMT training for En-Fi translation task. (a) shows that when the model is trained more, the q_n value become more close to the target.

improves further. It implies that SKD provides different kinds of information from noise, while the synergy effect between SKD and noise needs more research.

Table 3: BLEU scores on the test sets for En-Fi, Fi-En and En-De with two different beam widths. The scores on the development sets are in the parentheses.

Model	Beam width	
	1	12
En-Fi		
Baseline	7.29(8.28)	9.01(9.85)
+Noise	7.68(8.50)	9.35(9.53)
+SKD	8.36(**9.43**)	9.87(10.30)
+Noise+SKD	**8.81**(8.95)	**10.13(10.47)**
Fi-En		
Baseline	10.42(11.39)	11.89(12.78)
+Noise	10.74(11.80)	12.39(13.35)
+SKD	10.70(12.52)	12.43(13.82)
+Noise+SKD	**11.87(12.92)**	**13.16(14.13)**
En-De		
Baseline	19.72(19.28)	22.25(20.91)
+Noise	20.69(19.68)	22.40(20.92)
+SKD	20.29(20.41)	22.59(21.75)
+Noise+SKD	**21.16(20.34)**	**23.07(21.64)**

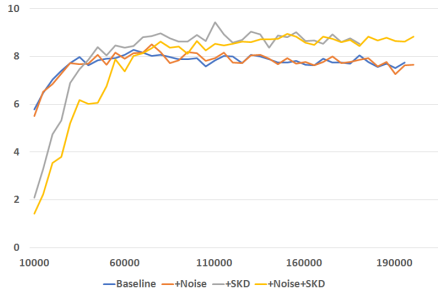


Figure 4: BLEU scores of validation data while training on En-Fi corpus with four different models: Baseline, +Noise, +SKD, and +Noise+SKD. The vertical axis indicates BLEU score and the horizontal axis the number of training iteration.

5 Conclusion

We proposed a new knowledge distillation method, self-knowledge distillation, from the probabilities of the currently training model itself. The method uses only two soft target probabilities that are obtained based on the word embedding space. The experiment results with language modeling and neural machine translation show that our method improves the performance. This method can be straightforwardly applied to other tasks where the cross-entropy is used.

As future works, we want to apply SKD to other applications with different model architectures, to show that SKD does not depend on tasks nor the model architectures. For image classification tasks, if we abuse the term 'word embedding' to refer to the layer right before the softmax operation, it may be possible to apply SKD in a similar way, although it is not guaranteed that comparable image classes are closely located in the word embedding space for image related tasks. Also, we can develop an automatic way for the parameters like α in Eq. (12), and generalize the equation for q_n in Eq. (8).

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (2017R1D1A1B03033341), and by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No. 2018-0-00749, Development of virtual network management technology based on artificial intelligence).

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *CoRR* abs/1608.00318:1–10.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proc. Int'l Conf. on Learning Representations (ICLR)*.

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(8):1798–1828.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2003. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research* 3:1137–1155.

Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2017. Context-dependent word representation for neural machine translation. *Computer Speech and Language* 45:149–160.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born-again neural networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. pages 1602–1611.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. pages 770–778.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL* 5:339–351.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR* abs/1504.00941.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR* abs/1609.07843.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proc. Int'l Conf. on Learning Representations (ICLR)*.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*. pages 1045–1048.

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.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. pages 6517–6525.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *CoRR* abs/1412.6550.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association for Computational Linguistics*. pages 1715–1725.

Anders Søgaard, Yoav Goldberg, and Omer Levy. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. 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 1: Long Papers*. pages 765–774.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*.

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 6000–6010.