Distilling Word Embeddings: An Encoding Approach

Lili Mou, Ran Jia, Yan Xu, Ge Li,* Lu Zhang, Zhi Jin*
Key Laboratory of High Condence Software Technologies (Peking University), MoE, China
Institute of Software, Peking University, China *Corresponding authors
{doublepower.mou, jiaran1994}@gmail.com
{xuyan14, lige, zhanglu, zhijin}@sei.pku.edu.cn

ABSTRACT

Distilling knowledge from a well-trained cumbersome network to a small one has recently become a new research topic, as lightweight neural networks with high performance are particularly in need in various resource-restricted systems. This paper addresses the problem of distilling word embeddings for NLP tasks. We propose an encoding approach to distill task-specific knowledge from a set of high-dimensional embeddings, so that we can reduce model complexity by a large margin as well as retain high accuracy, achieving a good compromise between efficiency and performance. Experiments reveal the phenomenon that distilling knowledge from cumbersome embeddings is better than directly training neural networks with small embeddings.

Keywords

Model compression; neural networks; word embeddings

1. INTRODUCTION

Distilling knowledge from a neural network—that is, transferring valuable knowledge from a cumbersome network to a lightweight one—is pioneered by Buciluǎ et al. [3]; it has attracted increasing attention over the last two years [7, 5].

As addressed by Hinton et al. [7], the objective of training networks is probably different from deploying networks: during training we focus on extracting as much knowledge as possible from a large dataset, whereas deploying networks takes into consideration multiple aspects, including accuracy, memory, time, and energy consumption. It would be appealing if we can first well train a cumbersome network offline, and then distill its knowledge to a small one for deployment. The aim of knowledge distillation is thus to reduce model complexity as well as to retain high performance, which is particularly important to neural networks' applications in resource-restricted scenarios, e.g., real-time systems, mobile devices, and large ensembles of models.

Much evidence in the literature shows the feasibility of transferring knowledge from one neural network to another, for instance, from shallow networks to deep ones [11], from feed-forward networks to recurrent ones [12], or vice versa [1, 4]. The main idea of the above studies is to train a *teacher model* first, and then use the teacher model's output (estimated probabilities by softmax, say, in a classification problem) to guide a *student model*. Several variants of training objectives include applying regression over the input of softmax [1] and softening the teacher model's probabilities [7]. We call such approaches "matching softmax" (Figure 1a). It is also argued that the estimated probabilities by a teacher model convey more information than one-hot represented ground truth; hence knowledge distillation is feasible and beneficial [7].

Despite the above generic approach, this paper focuses on distilling word embeddings in NLP applications. Particularly, we find the specificity of embeddings brings new opportunities for knowledge distillation.

As word embeddings map discrete words to distributed, real-valued vectors, it can be viewed that a word is first represented as a one-hot vector and then the vector is multiplied by a large embedding matrix, known as a look-up table. During the matrix-vector multiplication, one and only one column in the look-up table is verbatim retrieved for a particular word. Thus, we may build an interlayer—sandwiched between the high-dimensional embeddings and the ensuing network—to squash embeddings to a low-dimensional space (Figure 1b). The standard cross-entropy loss can then be applied to train the encoding layer and other parameters in the network. In such a supervised manner, task-specific knowledge in the original cumbersome embeddings can be distilled to low-dimensional ones.

In summary, the main contributions of this paper are three-fold: (1) We address the problem of distilling word embeddings in NLP applications. (2) We propose a supervised encoding approach to distill task-specific knowledge from cumbersome word embeddings. (3) Our experimental results in sentiment analysis and relation classification tasks reveal a phenomenon that distilling low-dimensional embeddings from large ones is better than directly training a network with small embeddings. It should also be noticed that the proposed encoding approach does not rely on a teacher model; our method is complementary to existing matching softmax for knowledge distillation.

2. BACKGROUND OF MATCHING SOFTMAX

As said, existing approaches to knowledge transfer between two neural networks mainly follow a two-step strat-

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA

DOI: http://dx.doi.org/10.1145/2983323.2983888

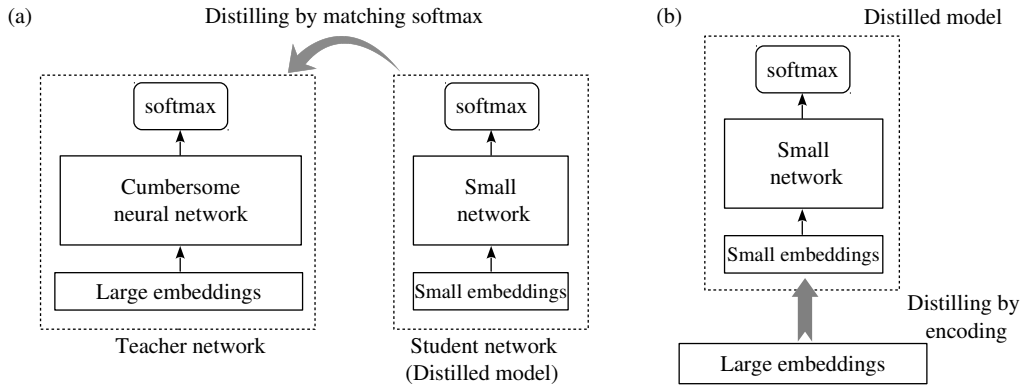

Figure 1: Distilling knowledge by (a) matching softmax, and (b) encoding embeddings.

egy: first training a teacher network; then using the teacher network to guide a student model by matching softmax, depicted in Figure 1a.

For a classification problem, softmax is typically used as the output layer's activation function. Let $\boldsymbol{z} \in \mathbb{R}^{n_c}$ be the input of softmax. (n_c is the number of classes.) The output of softmax is

$$y_i = \frac{e^{z_i/T}}{\sum_{j=1}^{n_c} e^{z_j/T}}$$

where T is a relaxation variable (used later), called *temperature*. $T = 1$ for standard softmax.

Take a 3-way classification problem as an example. If a teacher model estimates $\boldsymbol{y} = (0.95, 0.04, 0.01)^\top$ for three classes, it is valuable information to the student model that Class 2 is more similar to Class 1 than Class 3 to Class 1.

However, directly imposing constraints on the output of softmax may be ineffective: the difference between 0.04 and 0.01 is too small. Ba et al. [1] match the input of softmax, $\boldsymbol{z}$, rather than $\boldsymbol{y}$. Hinton et al. [7] raise the temperature T during training, which makes the estimated probabilities softer over different classes. The temperature of 3, for instance, softens the above $\boldsymbol{y}$ to $(0.64, 0.22, 0.14)^\top$. Matching softmax can also be applied along with standard cross-entropy loss (with one-hot ground truth), or more elaborately, the teacher model's effect declines in an annealing fashion when the student model is more aware of data [11].

3. THE PROPOSED ENCODING APPROACH FOR DISTILLING EMBEDDINGS

This section introduces in detail our proposed method for word embedding distillation (Figure 1b). We also analyze the neural network's model capacity with distilled embeddings, and discuss the rationale for distilling small embeddings from large ones in a supervised manner, instead of directly training with small embeddings.

Word embeddings are a standard component for neural natural language processing. As feeding word indexes directly to neural networks is somewhat nonsensical, words are mapped to a real-valued vector, called *embeddings*, where each dimension captures a certain aspect of underlying word semantics. Usually, they are trained in an unsupervised fashion, e.g., maximizing the probability of a large corpus [2, 9], or maximizing a scoring function [6, 8]. The learned embeddings can be fed to standard neural networks for supervised learning, e.g., POS tagging, named entity recognition, and semantic role labeling [6].

To formalize word embeddings in algebraic notations, we let $\boldsymbol{x}_i \in \mathbb{R}^{|\mathscr{V}|}$ be one-hot representation of the i-th word x_i in the vocabulary $\mathscr{V}$; the i-th element in the vector $\boldsymbol{x}_i$ is on, with other elements being 0. Let $\Phi_c \in \mathbb{R}^{n_{\text{embed}} \times |\mathscr{V}|}$ be a (cumbersome) embedding matrix (look-up table). Then the vector representation of the word is exactly the i-th column of the matrix, given by $\Phi_c \cdot \boldsymbol{x}_i$.

Now we consider distilling, from cumbersome embeddings $\Phi_c \cdot \boldsymbol{x}_i$, an n_{distill}-dimensional vector for the word, where n_{distill} is smaller than n_{embed}. It is accomplished by encoding with a non-linear neural layer, i.e.,

$$\text{vec}(x_i) = f(W_{\text{encode}} \cdot \Phi_c \boldsymbol{x}_i + \boldsymbol{b}_{\text{encode}}) \tag{1}$$

where $W_{\text{encode}} \in \mathbb{R}^{n_{\text{distill}} \times n_{\text{embed}}}$ and $\boldsymbol{b}_{\text{encode}} \in \mathbb{R}^{n_{\text{distill}}}$ are parameters of the encoding layer; $\text{vec}(\cdot)$ denotes the distilled vector representation of a word.

These distilled embeddings can then be fed to a neural network (with parameters Θ) for further processing. Let m be the number of data samples and n_c be the number of target classes; suppose further $\boldsymbol{y}^{(j)}$ is the output of softmax for the j-th data sample and $\boldsymbol{t}^{(j)}$ the one-hot represented ground truth. Our training objective is the standard cross-entropy loss, given by

$$\underset{W_{\text{encode}}, \boldsymbol{b}_{\text{encode}}, \Theta, \Phi}{\text{minimize}} \; -\sum_{j=1}^{m} \sum_{i=1}^{n_c} t_i^{(j)} \log y_i^{(j)}$$

We would like to point out that distilling embeddings does not increase, or in fact may reduce, model capacity vis-à-vis directly training with small embeddings, despite the large number of parameters in cumbersome embeddings and the coding layer's weights.

Theorem 1. *The model capacity of a neural network with distilled embeddings is less than or equal to that of a neural network trained directly with small embeddings.*

Proof. The intuition is straightforward: small embeddings are free parameters which are not constrained, whereas the distilled embeddings are subject to the form in Equation 1. Formally, let $\mathcal{H}_d, \mathcal{H}_s$ be the hypothesis classes of networks with distilled/small embeddings, respectively. For each $h_d \in \mathcal{H}_d$ with cumbersome embeddings Φ_c and encoding parame-

ters $W_{\text{encode}}, \boldsymbol{b}_{\text{encode}}$, there exists a hypothesis $h_s \in \mathcal{H}_s$, satisfying that $h_s = h_d$ with small embeddings Φ_s, whose i^{th} column (the small embedding for i^{th} word) is $f(W_{\text{encode}}\Phi_c\boldsymbol{x}_i + \boldsymbol{b}_{\text{encode}})$. Hence, $\mathcal{H}_d \subseteq \mathcal{H}_s$. ◄

A curious question is then why distilling embeddings may help, compared with directly training the neural network with small embeddings. We provide an intuitive explanation as follows.

Since word embeddings are typically learned from a large corpus in an unsupervised manner, the knowledge in embeddings is restrained by dimensionality. For example, the sentiment of a word is of secondary importance compared with its syntactic functionality in a sentence. Hence, sentiment information might be lost in low-dimensional embeddings, which is unfavorable in a sentiment analysis task. On the contrary, large embeddings have the capacity to capture different aspects of word semantics. The proposed supervised encoding approach may then distill task-specific (e.g., sentiment) knowledge to a small space, while eliminating irrelevant information. Therefore, we may reasonably expect that distilling embeddings would outperform direct use of small ones.

Deployment Issues

Before deploying the model, we shall precompute the distilled embeddings, $\text{vec}(\cdot)$, according to Equation 1 after training all parameters. The original embeddings $\Phi_c\boldsymbol{x}$ and encoding parameters (W_{encode} and b_{encode}) can then be safely discarded, and we obtain a small model (dashed rectangle in Figure 1b) with a set of small embeddings, which are distilled from large ones.

As we shall see in the experiments, the small model will be very computational efficient because we have reduced a large number of parameters.

4. EVALUATION

In this section, we present our experimental results. We first describe the testbed and protocol of our experiments in Subsection 4.1. Then we analyze in Subsection 4.2 the performance of our approach regarding several aspects, namely accuracy, memory, and time consumption.

4.1 Tasks, Models, and Protocols

We tested our distilling approach in two tasks: sentiment analysis and relation classification.

The sentiment analysis task aims to classify a sentence into 5 categories according to its sentiment: strongly/weakly positive/negative and neural. We used Stanford Sentiment Treebank[1] as our dataset, which contains 8544/1101/221 sentences for training, validation, and testing. Phrases (subsentences) in the training set are also labeled with sentiment, enriching the training set to more than 150k samples. For validation and testing, only the sentiment of a whole sentence was considered.

The second task is to classify the relation between two tagged entities in a sentence. The SemEval 2010 dataset,[2] we used, comprises 8000 training samples, from which we split 10% for validation; there are additional 3000 samples for testing. Target labels include 9 directed relations (e.g. `Component-Whole`) plus a default `Other`; in total, we have 19 classes. The official F_1-score was applied as our measurement.

[1]http://nlp.stanford.edu/sentiment/

[2]http://semeval2.fbk.eu/semeval2.php?location=data

Task	Method	Acc.	#Param	Time
Sentiment analysis by TBCNN	Cumbersome embed.	51.6	6.9M	1x
	Small embed.	46.4	0.94M (0.14x)	0.04x
	Distilled embed.	47.5		
	Matching softmax	45.8		

Table 1: Comparison between cumbersome embeddings, small embeddings, matching softmax, and distilled embeddings. The official measure is accuracy (acc.) in percentage.

Task	Method	F_1	#Param	Time
Relation classification by SDP-LSTM	Cumbersome embed.	82.1	8.8M	1x
	Small embed.	79.0	1.3M (0.15x)	0.04x
	Distilled embed.	79.4		
	Matching softmax	80.1		
	Hybrid	80.2		

Table 2: Comparison between cumbersome embeddings, small embeddings, matching softmax, and distilled embeddings. We further made an attempt to combine matching softmax and distilling embeddings (denoted as "Hybrid"). The official measure for relation classification is the F_1-score.

To set up our experiments, we leveraged two state-of-the-art neural models: a tree-based convolutional neural network (TBCNN) for sentiment analysis [10], and a long short term memory-based recurrent network along shortest dependency path (SDP-LSTM)[3] between two entities for relation classification [13].

For each task, we evaluated our proposed methods by distilling 300-dimensional embeddings to 50 dimensions, further processed by a thin network (also 50d). In comparison, we trained the 50d network directly with small 50d embeddings. All models were trained by mini-batch gradient descent with back-propagation. For both settings of distilling and non-distilling, we tried extensive configurations of hyperparameters, mainly following the original papers.[4] After choosing the setting with the highest validation accuracy, we ran each model 5 times for smoothing with different random initializations, and report the average test accuracy or F_1-score.

4.2 Results

Tables 1 and 2 presents the results of our proposed model as well as two competing settings: training a wider network with cumbersome embeddings, and directly training a thin network with small embeddings.

In both experiments, cumbersome embeddings yield the highest performance, the distilled embeddings rank second, and small embeddings are worst. Basically, our method outperforms direct training of a small network by a margin of approximately one standard deviation (std = 1.1 and 0.6, re-

[3]We only used word embeddings, and ignored other features like hyponymy, dependency types, which were used in [13]. In this way, we focus on the problem of embedding distillation itself.

[4]Due to the limitation of space, we list candidate configurations on our website:
https://sites.google.com/site/distillembeddings/

spectively). As the results were obtained by averaging over 5 initializations, we deem the improvement is fair.

Regarding model complexity, our distilled embeddings reduce memory and time to a large extent to 14–15% and 4%, respectively (C++ implementation on a single CPU). Therefore, the resulting network is significantly more lightweight, which is helpful to deployment in neural networks' applications.

To further test our method under extreme conditions, we distilled word embeddings to 10d and 30d. We chose to conduct the experiments in the second task, because it is of lower variance. As demonstrated in Figure 2, our method consistently outperforms direct training with small embeddings in all scenarios; moreover, the margin increases when the dimension becomes small. Such result is consistent with our human intuition, and verifies the conjecture in Section 3—small embeddings contain less knowledge specific to the task of interest; the proposed supervised encoding approach can distill task-specific knowledge from large embeddings.

We also notice that our approach is, in fact, complementary to existing matching softmax methods: the encoding layer distills task-specific knowledge from large embeddings in a bottom-up fashion, whereas matching softmax distills generic knowledge in a top-down fashion.

In both tasks, we also tried the matching softmax approach, whose settings and hyperparameters are mainly derived from [7], i.e., $T = 2$ and a $1:1$ mixture of ground truth and the teacher model's output. Its performance is not consistent: in the sentiment analysis task, matching softmax hurts the performance by 0.6%, whereas it improves the relation classification task by 1.1%. (See Tables 1 and 2 again.) One plausible explanation is that the teacher model itself has not achieved remarkable accuracy (only about 50%) in the 5-way sentiment classification task. Using a teacher model introduces additional knowledge as well as errors. If the latter dominates, matching softmax may hurt the student model. However, our encoding approach to embedding distillation does not reply on a teacher model. Combined with matching softmax, it improves another 0.1% (although may not be large) in the second experiment, showing that the two methods can be potentially combined, as they are complementary to each other.

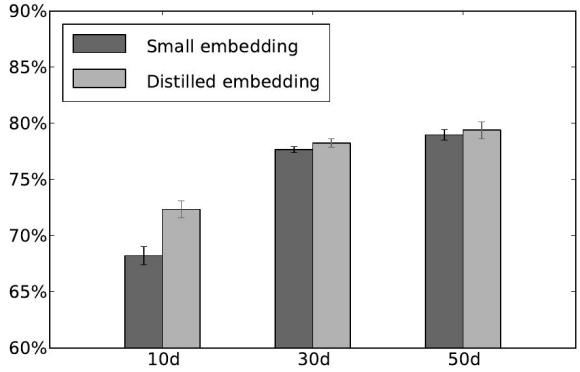

Figure 2: Accuracy versus dimension in the experiment of relation classification.

5. CONCLUSION

In this paper, we addressed the problem of distilling embeddings for NLP, which is important when deploying a neural network in resource-restricted scenarios. We proposed an encoding approach that distills cumbersome word embeddings to a low dimensional space. Experimental results have shown the superiority of our proposed distilling method to training neural networks directly with small embeddings; that the performance gain increases significantly especially when the dimension becomes small. Moreover, our approach does not reply on a teacher model, which is complementary to matching softmax; these two methods of knowledge distillation could also be combined.

6. ACKNOWLEDGMENTS

We thank all reviewers for their insightful comments and Rui Yan for discussion of the manuscript. This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201 and the National Natural Science Foundation of China under Grant Nos. 61232015, 91318301, 61421091, and 61502014.

7. REFERENCES

[1] J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, pages 2654–2662, 2014.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *JMLR*, 3:1137–1155, 2003.

[3] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *SIGKDD*, pages 535–541, 2006.

[4] W. Chan, N. R. Ke, and I. Lane. Transferring knowledge from a RNN to a DNN. *arXiv:1504.01483*, 2015.

[5] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *ICML*, pages 2285–2294, 2015.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[7] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2014.

[8] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[9] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *AISTAT*, pages 246–252, 2005.

[10] L. Mou, H. Peng, G. Li, Y. Xu, L. Zhang, and Z. Jin. Discriminative neural sentence modeling by tree-based convolution. In *EMNLP*, pages 2315–2325, 2015.

[11] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: Hints for thin deep nets. In *ICLR*, 2014.

[12] D. Wang, C. Liu, Z. Tang, Z. Zhang, and M. Zhao. Recurrent neural network training with dark knowledge transfer. *arXiv:1505.04630*, 2015.

[13] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*, pages 1785–1794, 2015.