Knowledge Injection to Neural Networks with Progressive Learning Strategy

Ha Thanh Nguyen(✉), Trung Kien Vu, Teeradaj Racharak, Le Minh Nguyen, and Satoshi Tojo

Japan Advanced Institute of Science and Technology, Ishikawa, Japan
{nguyenhathanh,kienvu,racharak,nguyenml,tojo}@jaist.ac.jp

Abstract. Nowadays, deep learning has become the most modern and practical approach to solve a wide range of problems. With the ability to automatically extract the hierarchy of semantic level from the data, neural networks often outperform other techniques in complex issues. However, to perform well, the models need a vast amount of data, which is not always available. To overcome that problem, we propose an approach of injecting knowledge into the neural network instead of letting it struggles by itself. Our proposed policy for the training process is guiding the model to learn the label from a similarity distribution. Finally, we conduct experiments in the chord modeling problem to show the effectiveness of our method.

Keywords: Knowledge injection · Simlarity distribution · Chord2Vec · Neural networks · Progressive training

1 Introduction

Our knowledge about the world is a model itself. We percept nature and sociality by our senses, process the information by our brain, and remember the conclusion as experience. When a new situation comes, we compare it with our experience and conduct an appropriate judgment for the case. The artificial neural network is much more straightforward but has the same mechanism. Given input/output pairs in the training data, the model tries to predict the answer, correct the prediction from feedbacks, and remember the hidden rules in the data.

Different architectures of neural networks share the same mechanism. Specifically, given input/output pairs, the model during the training process needs to learn to minimize the designed loss function to achieve better and better predictions. Theoretically, after being trained, the parameters of the model make it able to output the correct label similar to what it has learned from the data. However, in practice, there are often situations that the data is not enough for the model to formulate an excellent interpretation to make the right prediction for unseen samples.

A. P. Rocha et al. (Eds.): ICAART 2020, LNAI 12613, pp. 280–290, 2021.
https://doi.org/10.1007/978-3-030-71158-0_13

Many authors came up with different methods to overcome this challenge. Most of them are about adding more information to the input. Pretrained embeddings (*cf.* [12,14,16]) are the methods that enhance the information about relationship between the tokens via learned vectors. Transfer learning methods use pre-trained models like BERT [4] and fine-tune them later according to downstream tasks.

Those approaches are suitable for the problems in a domain containing a large amount of material like problems in Natural language processing or Computer vision. Nevertheless, in a field in which the training data is limited even for training pre-trained models, the transfer learning technique is useless. In such situations, to improve the learning performance of the model, there must be a way to inject the domain knowledge into its learning progress. Song et al. [18] uses an ontology to train parameters of the model selectively. In the work of [2], instead of using a single target vector, the authors let the model guided by multiple vectors.

The deep learning models often update their parameters directly from the label of training data. This optimization has a potential drawback. The feedback for the model is only whether right or wrong and the model need to look up a different value set of parameter randomly. This *try-and-error* progress takes a lot of resource for training and possibly lead to overfitting when the model only tries to remember the training data. Based on such observation, we propose a progressive training method that injects the domain knowledge to the neural network by similarity metric. For that purpose, we implement a training method that gradually turns the similarity target, which is obtained from the domain knowledge, into the original distribution, which is annotated in the training data.

In this lecture note, besides the contributions presented at ICAART 2020 [22], we study and convey the theoretical basis of using similarity as an element of knowledge. We also introduce an approach to extract the knowledge when the corpus is given automatically. Our experiment shows that our method is effective and general.

2 Backgrounds

2.1 Similarity in Cognitive Science

The similarity is a field of research in cognitive science. There are lots of research approaches to the concept of similarity. They are strongly related to knowledge representation; each of them focuses on a set of assumptions.

Shepard et al. [17] bases on the assumption that points within the space can represent the concepts and the distances between the points represent the similarity between concepts. This approach is a vital basis for machine learning techniques like clustering or latent semantic analysis.

The distance approach is straightforward to understand, but it has the limitation as the distance is symmetric. In the featural approach, Tversky et al. [20] define a function $s(a, b)$ as a metric of similarity from a to b. We need to notice that this function is asymmetry and comparable. $s(a, b) > s(b, a)$ is valid

following this theory. In detail, the authors propose this approach based on two assumptions: Matching and Monotonicity. Let A and B be the set of features of a and b. The matching assumption is that similarity of a to b is a function accepting three arguments: $A \cap B$, $A - B$, and $B - A$. Figure 1 illustrates how these subsets affect the similarity function. Monotonicity assumption states that $s(a, b) \geq s(a, c)$ whenever: $A \cap C \subset A \cap B$, $A - B \subset A - C$, and $B - A \subset C - A$.

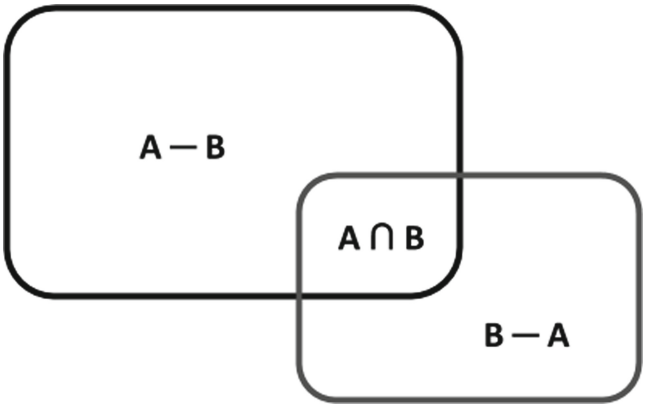

Fig. 1. Illustration of arguments of similarity function.

Another theory of similarity, Gentner et al. [5] develops structural mapping. In this theory, there are two kinds of differences which are the non-alignable difference and the alignable difference. The alignable difference comes from the commonality, for example, both dogs and chickens have legs, but dogs have four legs while chickens have two legs. Non-alignable differences are the features that do not belong to both entities. Another approach, Hahn et al. [7] evaluate the similarity based on the minimum number of transformation steps to make one representation into another.

2.2 Neural Network Models

The most classical deep learning model is multiple layer perceptron (MLP) or feedforward neural network. An MLP consists of at least three layers known as the input layer, hidden layer(s), and output layer. This simple model can approximate a wide range of functions and classify nonlinear separable data. Each node in this neural network is assigned with a weight value. Learning is about adjusting these weights to fit the data. When the model makes a wrong prediction, a signal in the form of the loss function is calculated. Based on that, the model updates the weights to improve the predictions.

The limitation of MLP is that it cannot capture the relation of entities in an input sequence because each input is feed through the network separately.

The relation information is essential to obtain a good prediction. As a result, the vanilla version of the neural network as MLP does not perform well on sequential data.

A different well-known neural network class is recurrent neural networks. The precursor of this class is the Hopfield Networks [11]. This architecture makes use of the information from the previous step in the prediction of the current phase. Consequently, this neural network is able to handle consecutive inputs. LSTM [9] and GRU [3] are the better versions of RNN.

Although the recurrent neural network can handle the sequential input, it faces a challenge, namely "vanishing gradients". This architecture maintains the relationship between entities in a long sequence. As a result, the feedback signals have to pass through a dense bundle of layers, which leads to the loss value becomes zero at the first layers. Overcome such challenge, LSTM is the neural network that can skip some unnecessary input in the sequence and focus only on the critical parts. This architecture contains an input gate, an output gate, and a forget gate, as shown in Fig. 2. This model can choose which input to remember and ignore others. GRU is another architecture, which has a similar principle of design but contains less number of parameter than LSTM.

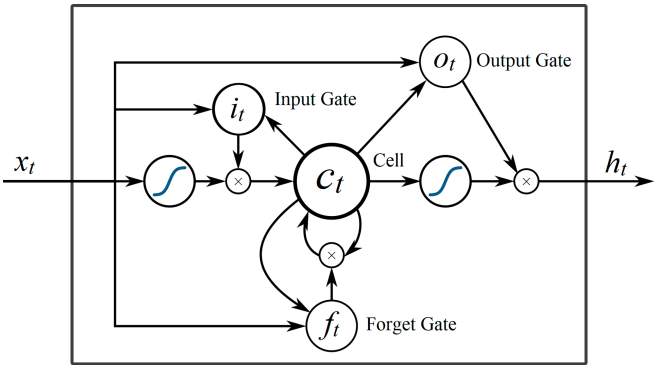

Fig. 2. LSTM architecture can capture longer input sequence [22].

LSTM models are widely used and bring good performance in many tasks like sequence tagging [10], speech recognition [6], or language modeling [19].

The lastest well-known architecture of deep learning is the transformer models [21]. They were invented with the idea of attention mechanism. This mechanism helps the model to choose the most relevant part in the sequence to pay attention to. The advantage of this approach is that the distance between the word does not limit the relationship probability.

2.3 Related Works

This work is not the first research to try to enhance learning efficiency by exploiting knowledge at training progress. We can mention some of the well-known

approaches, such as multi-task learning, transfer learning, curriculum learning, and knowledge compilation.

The aim of multi-task learning is leveraging the relationship between classes at the training time. For example, Vural et al. [23] proposed a classification model to extract interclass relationships. Wu et al. [24] built a classifier that can learn the relationship features using deep learning architecture in the video classification task.

The idea behind transfer learning is to make use of learned parameters from one domain to another domain. Compared to the target domain, the source domain often contains more resources and is more feasible for deep learning models to exploit knowledge. This approach is natural as the way humans learn. For example, if a person has experience in learning a foreign language, such experience can help to learn other foreign languages faster. Pre-trained models like BERT [4] or GPT-2 [15] are good examples of transfer learning.

Curriculum learning approaches try to use a metric for measuring the difficulty of each sample. The model then can choose to learn the sorted example based on that metric. Gradually increasing the complexity of the learning problem makes the model learn and build the hierarchy of concepts instead of trying to remember precisely the samples. [2] makes use of curriculum learning to make their model achieve better performance without increasing training time.

Along with transfer learning, we can distill the knowledge from a model to another model. For instance, [8] proposes a training procedure in which there are two models. This procedure allows the student model to learn from the output distribution of the teacher model. This technique is useful for model compression for creating lightweight but robust models. Besides, knowledge distillation can help to avoid overfitting (*cf.* [1]).

3 Knowledge Injection via Simlarity Metric

In a multi-class classification problem, there are K classes of the instances, and the model needs to predict which class each instance belongs to. At first, the features of each sentence are fed into the model as a vectorized representation. The model then needs to predict the output by producing a K-dimension vector $\hat{\mathbf{y}} \coloneqq [\hat{y}_1, \ldots, \hat{y}_K]$. This vector represents the probability distribution of the confidence over K classes. If the prediction is far from the gold distribution, the neural network needs to update its weights θ to reduces the loss calculated by KL divergence between the predicted distribution $\hat{\mathbf{y}}$ and the gold distribution $\mathbf{y}$ (*cf.* Eq. 1 [22]).

$$D_{\mathrm{KL}}(\mathbf{y}, \hat{\mathbf{y}}) \coloneqq \sum_{i=1}^{K} y_i \log \frac{y_i}{\hat{y}_i} \tag{1}$$

In general, one-hot representation is the format for the target distribution $\mathbf{y} \coloneqq [y_1, \ldots, y_K]$ *i.e.*, the true distribution. However, this distribution is not always straightforward for the model to learn. The feedback signal is simply whether right and wrong, which does not reflect the fact that among wrong

predictions, some of them are better than the others. As mentioned in Sect. 1, the feedback signal for each class should be unequal. The loss value between two similar classes should be smaller than the very different ones. At that point, the knowledge of similarity between classes is essential for improving training performance. We propose the training process containing two steps as following:

1. Calculate the similarity between classes and form a distribution based on that information.
2. Apply a progressive learning start from the similarity distribution and gradually turn to the true distribution.

The similarity score between two classes is calculated following the Eq. 2 [22]. In which, $s(a, b)$ is the similarity score between two classes a and b, $w(p)$ is the weighted common properties of the classes.

$$s(a, b) \coloneqq \sum_{p \in \mathcal{P}_a \cap \mathcal{P}_b} w(p) \tag{2}$$

Continuing the previous work on chord data [22], in this paper, we propose a knowledge extraction method named Chord2Vec. Chord2Vec is built on statistics of the occurrence of chords with contexts. Initially, each chord is represented by a random vector. After the training process, the updated vector can reflect the relationship and distance between chords. Given the context, which are previous chords in a sequence, the model needs to predict the next one. Let $\boldsymbol{v_a}$ and $\boldsymbol{v_b}$ are the chord vectors of chord a and chord b, the similarity function $s(a, b)$ is calculated by the Eq. 3.

$$s(a, b) = 1 + cos(\boldsymbol{v_a}, \boldsymbol{v_b}) \tag{3}$$

In the formula, we add 1 into the cosine similarity to make sure the outputs of the function is greater or equal to 0.

After calculating the pairwise similarity score, we construct the similarity vector representation, in which each unit calculated following the Eq. 4 [22].

$$s_{ak} \coloneqq \frac{s(i, k)}{\sum_{j=1}^{K} s(i, j)} \tag{4}$$

Finally, the mass distribution is calculated from the similarity vectors following the Eq. 5 [22]. To make use of the knowledge of similarity in the training period, we apply a progressive learning process.

$$t_{ak} \coloneqq \text{softmax}(s_{ak}) \coloneqq \frac{\exp(s_{ak}/T)}{\sum_{j=1}^{K} \exp(s_{aj}/T)} \tag{5}$$

We use a variable T to control the learning temperature. At the beginning of the process, we use a high value of T and gradually decrease it following Eq. 6 [22]. As a result, the model is forced to learn from the similarity distribution and slowly turns to the actual distribution.

$$T_t \coloneqq \frac{T_{t-1}}{1 + \lambda t} \tag{6}$$

4 Experiments

This section has two main purposes. First, we verify the effectiveness of knowledge injection in a training process when similarity knowledge is given by the user. Second, we aim at showing that similarity knowledge can be automatically learnt from a dataset. For that, we introduce an approach called Chord2Vec. Note that this idea of automatic knowledge extraction is new and has not well investigated in [22].

4.1 Experimental Settings

To verify the effectiveness of our method, we choose the chord modeling problem. We use the ABC dataset [13] which are from 17 Beethoven string quartets. For the experimental setup purpose, we split all pieces into phrases. And because we do not model the duration of each chord, repetitions of phrases are removed. Table 1 represents some important properties of the final dataset.

Table 1. Properties of the final dataset.

Property	Value
Number of phrases	968
Average length of phrases	21
Number of keys	21
Number of chord types	13

We used 5-fold cross-validation and two metrics as perplexity and accuracy to assess the model. We use 200 phrases as the evaluation data in each fold, and the rest to train the model. We use the data augmentation technique for the data in the training set. Each phrase is transposed into 12 keys. At last, we got 8000 samples in the training set.

The deep learning model in our experiments is LSTM. The embedding size, hidden size is set to 128. We optimized the model with Adam Optimizer. Learning rate is 10^{-3}. The early stopping patience is 10 and maximum number of epochs is 200.

As described in Sect. 3, we conduct experiments to confirm whether it is feasible to extract knowledge directly from the corpus to improve model efficiency in specific tasks. This has two meanings. Firstly, we can cross-evaluate the effectiveness of knowledge injection in our method. Secondly, it gives a hint about how to guide the model using knowledge injection in domains where people do not have much experience.

The idea is borrowed from Word2Vec [12]. The musician places the chords in a music sheet intentionally. If the placement is random, the creation is not called music, it is just a sequence of noise. By such observation, we have a hypothesis

that there must be a kind of grammar in the music, and the machine learning model can learn such knowledge.

We sort the chord by descending frequency and the chord appearing less than 5 times in the corpus is eliminated. We train the Chord2Vec with the starting learning rate is 0.025 and is linearly reduced to zero. We stop the training process as soon as the loss stop decreasing.

4.2 Experimental Results

As reported in our previous paper [22], training model with our proposed knowledge injection overperformed using the one-hot vector as the target. The chord properties which we use to calculate the similarity score includes: *token name*, *key name*, *key number*, *triad form*, *figured bass*, and *note pair*. Besides, the weights for properties make the performance of the model better than using uniform weights. The optimal weight setting for this approach is $(16, 32, 16, 4, 4, 32)$. Interestingly, our novel approach using Chord2Vec achieved state-of-the-art results with hyperparameter value $T_0 = 0.025$ and $\lambda = 0.01$. Although this value does not create a big gap with the performance of knowledge injection with chord properties, this result still can confirm that injecting knowledge with chord property is reasonable and the knowledge can be learned automatically from the corpus.

Table 2. Performance of different approaches.

Approach	Perplexity
One-hot vector	3.48
Uniform weight set	2.49
Optimal weight set	2.39
Chord2Vec	**2.37**

Figure 3 shows similarity matrices calculated by chord properties with weight sets (1, 1, 1, 1, 1, 1) and (16, 32, 16, 4, 4, 32) as well as calculated by cosine similarity of Chord2Vec vectors according to Eq. 3. Axis X and Y of the matrices are the ids of the chords. In all three matrices, high similarity points often gather in the diagonal (Table 2).

The figure of the similar pattern calculated by cosine has more diverse similarity points. However, there are rows and columns with all null values because of some chords which do not exist in the corpus used to train Chord2Vec embedding. It is one of the limitations of this approach compared to human knowledge.

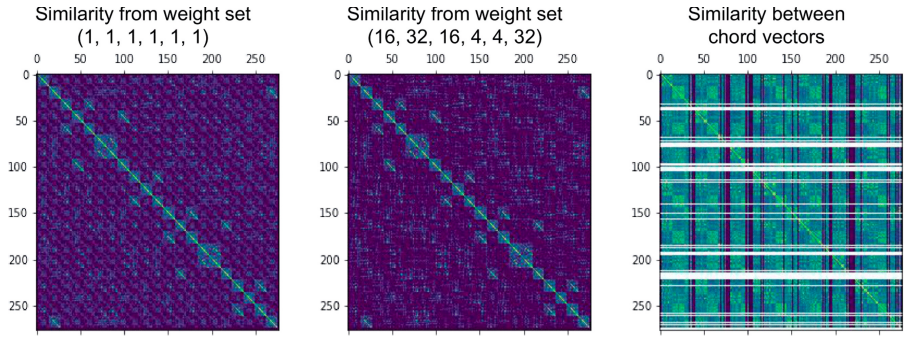

Fig. 3. Similarity matrices are calculated base on weighted properties and chord vectors.

5 Conclusion

This work leverages the proposal in our previous paper [22]. Human knowledge is a model about the world, and such information can guide the neural network to learn better. In this paper, we have introduced the theory basis for knowledge encoding as a similarity metric and propose an approach to make use of knowledge during the training time of the model.

To experiment with our idea, we defined a similarity encoding based on the common weighted properties of chords to deal with the problem of chord prediction. There is an interesting finding in our experiment that turning similarity target to one-hot target using appropriate temperature and decay can significantly improve the performance of the model.

In this paper, we also make an additional experiment to extract the knowledge from the corpus, then compare its performance with expert knowledge. The result confirms the effectiveness of our method and gives us a hint to use this method as a transfer learning approach in a suitable domain. The model can learn to construct the similarity encoding if there is a large enough corpus. The proposed method in this paper can be applied to a wide range of works using neural networks.

References

1. Asami, T., Masumura, R., Yamaguchi, Y., Masataki, H., Aono, Y.: Domain adaptation of dnn acoustic models using knowledge distillation. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5185–5189, March 2017. https://doi.org/10.1109/ICASSP.2017.7953145
2. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS 2015, vol. 1. pp. 1171–1179. MIT Press, Cambridge (2015). http://dl.acm.org/citation.cfm?id=2969239.2969370

3. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
5. Gentner, D., Markman, A.B.: Structure mapping in analogy and similarity. Am. Psychol. **52**(1), 45 (1997)
6. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 6645–6649. IEEE (2013)
7. Hahn, U., Chater, N., Richardson, L.B.: Similarity as transformation. Cognition **87**(1), 1–32 (2003)
8. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
11. John, J.H.: Neural network and physical systems with emergent collective computational abilities. Proc. Nat. Acad. Sci. U.S.A. **79**, 2554–2558 (1982)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
13. Neuwirth, M., Harasim, D., Moss, F.C., Rohrmeier, M.: The Annotated Beethoven Corpus (ABC): a dataset of harmonic analyses of all Beethoven string quartets. Front. Digit. Hum. **5**, 16 (2018). https://doi.org/10.3389/fdigh.2018.00016. https://www.frontiersin.org/article/11.3389/fdigh.2018.00016
14. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1532–1543. Association for Computational Linguistics, October 2014. https://doi.org/10.3115/v1/D14-1162
15. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8), 9 (2019)
16. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8), 9 (2019)
17. Shepard, R.N.: The analysis of proximities: multidimensional scaling with an unknown distance function. i. Psychometrika **27**(2), 125–140 (1962)
18. Song, L., Cheong, C.W., Yin, K., Cheung, W.K., Fung, B.C.M., Poon, J.: Medical concept embedding with multiple ontological representations. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 4613–4619. International Joint Conferences on Artificial Intelligence Organization (7 2019)
19. Sundermeyer, M., Schlüter, R., Ney, H.: LSTM neural networks for language modeling. In: Thirteenth Annual Conference of the International Speech Communication Association (2012)
20. Tversky, A.: Features of similarity. Psychol. Rev. **84**(4), 327 (1977)
21. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
22. Vu, T.K., Racharak, T., Tojo, S., Nguyen, H.T., Nguyen, L.M.: Progressive training in recurrent neural networks for chord progression modeling. In: Proceedings of the 12th International Conference on Agents and Artificial Intelligence (2020)

23. Vural, V., Fung, G., Rosales, R., Dy, J.G.: Multi-class classifiers and their underlying shared structure. In: IJCAI (2009)
24. Wu, Z., Jiang, Y.G., Wang, J., Pu, J., Xue, X.: Exploring inter-feature and inter-class relationships with deep neural networks for video classification. In: Proceedings of the 22Nd ACM International Conference on Multimedia, MM 2014, pp. 167–176. ACM, New York (2014). https://doi.org/10.1145/2647868.2654931