Few Shot and Meta Learning for NLP

Mehrnoosh Mirtaheri
USC Information Sciences Institute
mehrnoom@usc.edu

Abstract

Meta learning and few shot learning approaches have shown promising results in computer vision, with low-resouce tasks. Recently they have gained attention in natural language processing tasks such as machine translation and text classification. In this lecture we cover how meta learning approaches such as MAML and metric learning approaches such as matching and prototypical networks are used, combined with episodic training to improve the performance on low resource NLP tasks.

1 Introduction

Many studies address the low resource problem in NLP by transfer learning or combining multiple data sources and tasks (multi task learning.) However, the main focus of this lecture is to explain how few shot and meta learning approaches are adopted for in various natural language processing tasks. Some methods use metric learning by learning a good similarity metric between input examples; some other methods adapt a meta-learning framework, and train the model to quickly adapt to new tasks with gradients on small samples. The lecture is structures as follows: Section 2 describes two meta learning approaches for sequence modeling. Section 3 explains how prototypical networks are used for text classification and finally 4 focuses on unsupervised tasks such as word and graph embeddings.

2 Sequence Labeling

The task of sequence modeling is to assign a label sequence $Y = \{y_1, y_2, \ldots, y_T\}$ to a text sequence $X = \{x_1, x_2, \ldots, x_{T'}\}$. A special case would be text classification where Y is a single label.

2.1 Neural Machine Translation (NMT)

NMT is one example of sequence modeling which is known to easily over-fit and result in an inferior performance when the training data is limited. In Gu et al. (2018), the MAML applied to low resource NMT by viewing language pairs as separate tasks. The underlying idea of MAML is to use a set of source tasks $T_1, \ldots, T_k$ to find the initialization of parameters θ^0 from which learning a target task T_0 would require only a small number of training examples. In the context of machine translation, this amounts to using many high-resource language pairs to find good initial parameters and training a new translation model on a low-resource language starting from the found initial parameters. In other words:

$$\theta^* = \text{Learn}(T^0; \text{MetaLearn}(T^1, \ldots, T^k))$$

Given a dataset of tasks (language pairs) D_T, we can formulate the language-specific learning process $\text{Learn}(D_T; \theta^0)$ as:

$$\text{Learn}(D_t; \theta^0) = \text{argmax}_\theta \mathcal{L}^{D_T}(\theta) =$$

Preprint. Under review.

$$\text{argmax}_\theta \sum_{(X,Y)\in D_T} \log p(Y|X,\theta) - \beta\|\theta - \theta^0\|^2$$

The first part is a regular NMT objective, and the second part is to force the parameters to be close to the initialization. The initialization θ^0 is found by repeatedly simulating low-resource translation scenarios using auxiliary, high-resource language pairs. The objective function is defined as:

$$\mathcal{L}(\theta) = \mathbf{E}_k \mathbf{E}_{D_{T^k}, D'_{T^k}} \left[\sum_{(X,Y)\in D'_{T^k}} \log p(Y|X; \text{Learn}(D_{T^k}; \theta)) \right]$$

where $k \approx \mathcal{U}(\{1, \ldots, K\})$ is one meta-learning episode, D_T and D'_T are meta-train and meta-test, drawn from the same distribution. The training process and gradient update is simillar to what proposed in Finn et al. (2017). They also use first order gradient approximation.

Universal Lexical Representation (URL)

One major challenge here is the space mismatch between input and output languages. So they follow the approach proposed in . Let say each language has a $\epsilon^k_{query} = \mathbf{R}^{|V_k|\times d}$ where $|V_k|$ is the size of vocabulary for language k. They define a universal embedding matrix ϵ_u and a key matrix ϵ_{key} both with size $\mathbf{R}^{m\times d}$ and the embedding of token x is defined as:

$$\epsilon^0[x] = \sum_{i=1}^{M} \alpha_i \epsilon_u[x], \alpha_i \propto \exp\{\frac{1}{t}\epsilon_{key}[i] A \epsilon^k_{query}[x]\}$$

fine tuning on a small corpus which contains a limited set of unique tokens in the target language could adversely influence the other tokens' embedding vectors. They thus estimate the change to each embedding vector induced by language specific learning by a separate parameter $\Delta\epsilon^k[x]$

$$\epsilon^k[x] = \epsilon^0[x] + \Delta\epsilon^k[x]$$

2.2 Meta Multi Task Learning

In Chen et al. (2018), authors propose a new sharing scheme of semantic composition function across multiple tasks. Their focus however, is not on few shot training but on how to use meta learning as a parameter sharing method. Their proposed method for multi-task sequence learning combines a meta-LSTM (as they call it in their paper), shared between different tasks, and a separate LSTM, for each task.

Basic-LSTM Different from the standard LSTM, the parameter of the basic LSTM is controlled by a meta vector z_t, generated by meta-LSTM

$$\begin{bmatrix} g_t \\ o_t \\ i_t \\ f_t \end{bmatrix} = \begin{bmatrix} tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} (W(z_t) \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + b(z_t))$$

Considering a matrix for each time step t would make the parameter space very big, so they define a low-rank matrix factorization for $W(z_t)$ and $b(z_t)$.

Meta-LSTM The meta-LSTM depends on x_t and the previous hidden state h_{t-1} of the basic-LSTM

$$\begin{bmatrix} g'_t \\ o'_t \\ i'_t \\ f'_t \end{bmatrix} = \begin{bmatrix} tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} (W_m \begin{bmatrix} x_t \\ h'_{t-1} \\ h_{t-1} \end{bmatrix} + b_m)$$

and $z_t = W_z h'_t$. Figure 1 shows the architecture of Meta Multi-task Learning.

More precisely, we can describe the update of the units of the Meta-LSTMs as follows:

$$[h'_t, z_t] = \text{Meta-LSTM}(x_t, h'_{t-1}, h_{t-1}; \theta_m)$$
$$h_t = \text{Basic-LSTM}(x_t, h_{t-1}; z_t, \theta_b)$$

For converting this from a single task learning, to a mult-task learning framework, we can assign a basic network to each task, while sharing a meta network among tasks.

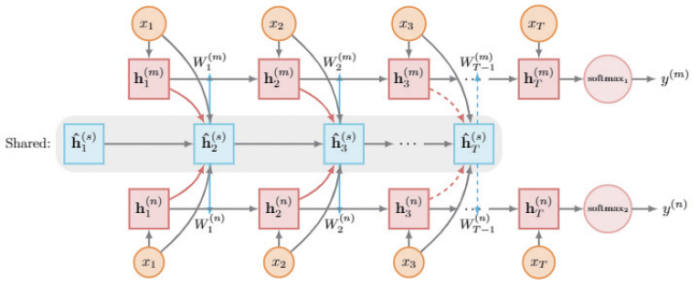


Figure 1: Architecture of Meta Multi-task Learning. The blue modules are shared between different tasks, which control the parameters of private layers

3 Text Classification

As an example of few shot learning for text classification, we describe how Tan et al. (2019) adapts prototypical networks with episodic training for Out-of-Domain Detection. Out-of-Domain (OOD) detection refers to assigning a tag to the input when it doesn't belong to any of the labels in the training dataset (In-Domain).

In this paper, they target solving the zero-shot OOD detection problem for a few-shot meta-test dataset $D = (D_{train}, D_{test})$ by training a transferable prototypical network model from large-scale independent source datasets $T = T_1, T_2, \ldots, T_N$ for dynamic construction of the meta-train set. Each task T_i contains labeled training examples

General Framework

1. Sample a training task T_i from T and another task T_j from $T - T_i$.
2. Sample an ID training example x_i^{in} from T_i and a simulated OOD example x_j^{out} from T_j.
3. Sample N labels from T_i and for the ground truth, sample k training example for each label, we call it support set $\mathcal{S}^{in} = \{S_l^{in}\}_{l=1}^{N}$
4. An encoder $E(.)$ encodes x_i^{in}, x_j^{out} and S_l^{in} using a deep network.
5. Build the prototypical vector representation for each label in the support set.

Similar to the training objective is a cross entropy loss defined as:

$$\mathcal{L}_{in} = -\log \frac{\log \alpha F(x_i^{in}, S_{l_i}^{in})}{\sum_{l'} \alpha F(x_i^{in}, S_{l'}^{in})}$$

However, unlike prototypical networks, they want the training examples from OOD to be far from a prototype and the ID training examples be close to the prototype. So they define two more hinge loss:

$$\mathcal{L}_{ood} = max[0, max_l(F(x_j^{out}, S_l^{in}) - \mathcal{M}_1)]$$

$$\mathcal{L}_{gt} = max[0, \mathcal{M}_2 - F(x_j^{in}, S_l^{in}))$$

and the final loss is defines as:

$$\mathcal{L} = \mathcal{L}_{in} + \lambda \mathcal{L}_{ood} + \beta \mathcal{L}_{gt}$$

4 Unsupervised Learning

In this section, we explain how few shot learning is used for Out-Of-Vocabulary word embeddings (Section 4.1) and predicting unseen relations in knowledge graphs (Section 4.2) .

4.1 Few-Shot Representation Learning for Out-Of-Vocabulary Words

In Hu et al. (2019) the problem of learning OOV embeddings is formulated as a few-shot regression problem. Consider a training corpus D_T and a pretrained word embedding (e.g. Word2Vec). The goal is to infer word embeddings for OOV words with just a few examples, demonstrating its usage, on a new Testing corpus D_N. Note that D_N could be relatively small and fine-tuning over it could result in over fitting. The neural regression function $F(.)$ is trained on D_T, from which N words $\{w_t\}_{t=1}^N$ are picked with sufficient number of sentences (S_t for word w_t), and their pretrained embeddings $\{T_{w_t}\}_{t=1}^N$. Each episode is built as follows:

1. Select w_t
2. Randomly select K sentences from S_t, mask out w_t from them and build $\mathbf{S}_t^K = s_{t,k}{}_{k=1}^K$
3. As well as $\mathbf{S}_t^K$, the character sequence of w_t is also utilized, denoted by C_t

The objective function is defined as:

$$\hat{\theta} = argmax_\theta \sum_{w_t} \sum_{\mathbf{S}_t^K \sim \mathbf{S}} cos(F_\theta(\mathbf{S}_t^K, C_t), T_{w_t})$$

While $F(.)$ could be any neural network, in this paper they use self attention networks for context encoding.

Adaptation with MAML

Applying the learned regression function $F(\theta)$ could be problematic when there exists some linguistic and semantic gaps between D_T and D_N. They address this issue by using the learned parameter θ as an initialization and conduct one step gradient update using MAML, at each training episode. More specifically:

1. At each episode they first conduct gradient descent $\theta* = \theta - \alpha \nabla_\theta \mathcal{L}_{D_T}(\theta)$
2. Then use θ^* as an initialized weight to optimize θ on D_N

$$\theta' = \theta - \beta \nabla_\theta \mathcal{L}_{D_N}(\theta*)$$

4.2 One-Shot Relational Learning For Knowledge Graphs

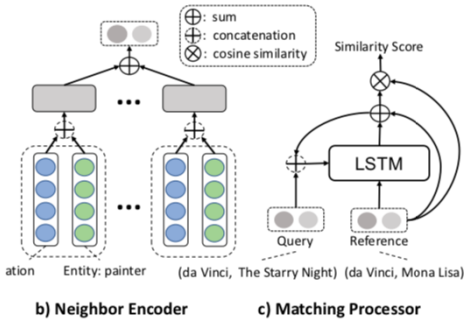
Knowledge graphs G are represented as a collection of triples $\{(h, r, t)\} \subset E \times R \times E$, where E and R are the entity set and relation set. The authors in Xiong et al. (2018) aim at predicting new facts under a challenging setting where only one training instance is available. In other words the task is to predict the tail entity t given the head entity and the query relation: $(h, r, ?)$, given only one training instance from each relation. The model proposed in this paper is a similarity-based meta-learning approach. It learns a similarity function $\mathcal{M}((s, o), (s_0, o_0)))$. So for any query relation r, if there is one known triple (s_0, r, o_0), the model can predict the likelihood of a query triple (s, r, o), based on the matching score between (s_0, o_0) and (s, o). The model consists of two main part:

Setup.

In this context, each training task corresponds to a KG relations $r \in R$, and has its own training/testing triples: $T_r = D_{train}, D_{test}$. This task set is often denoted as the meta-training set, $T_{meta-train}$. To imitate the one-shot prediction at evaluation time, there is only one triple (h_0, r, t_0) in each D_{train}^r. The $D_{test} = (h_i, r, t_i, C_{h_i,r})$ consists of the testing triples of r with ground-truth tail entities t_i for each query (h_i, r), and the corresponding tail entity candidates $C_{h_i,r} = \{t_{ij}\}$ where each t_{ij} is an entity in G. The metric model can thus be tested on this set by ranking the candidate set $C_{h_i,r}$ given the test query (h_i, r) and the labeled triple in D_{train}.

Neighborhood Encoder

For each entity pair, it aggregates the neighborhood information of a node by a neural network as d dimensional vector. There are two ways to initialize the node and relation embeddings:



Algorithm 1 One-shot Training

Input:
a) Meta-training task set $\mathbb{T}_{meta-training}$;
b) Pre-trained KG embeddings (excluding relation in $\mathbb{T}_{meta-training}$);
c) Initial parameters θ of the metric model;
for epoch = 0:M-1 **do**
  Shuffle the tasks in $\mathcal{T}_{meta-learning}$
  for T_r in $\mathcal{T}_{meta-learning}$ **do**
    Sample one triple as the reference
    Sample a batch B^+ of query triples
    Pollute the tail entity of query triples to get B^-
    Calculate the matching scores for triple in B^+ and B^-
    Calculate the batch loss $\mathcal{L} = \sum_B \ell$
    Update θ using gradient $g \propto \nabla\mathcal{L}$
  end for
end for

Figure 2: The right Figure is the meta-learning algorithm and the left Figure is the model architecture for one-shot learning

1. Random initialization
2. Pick a set of relations, make graph G' and train an existing model to obtain the embeddings, and use the rest of relations to make meta-train graph.

Matching Network

Proposed in Vinyals et al. (2016). It takes the vector representations of any two entity pairs from the neighbor encoder; then performs multi-step matching between two entity-pairs and outputs a scalar as the similarity score. The architecture of the model is shown in the left side in Figure 2

The training process is explained in the right side in Figure 2, The objective function is a triple hinge loss defined as follow:

$$l_\theta = max(0, \lambda + score_\theta^- - score_\theta^+)$$

References

References

Gu, J.; Wang, Y.; Chen, Y.; Cho, K.; Li, V. O. *arXiv preprint arXiv:1808.08437* **2018**,

Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. Proceedings of the 34th International Conference on Machine Learning-Volume 70. 2017; pp 1126–1135.

Chen, J.; Qiu, X.; Liu, P.; Huang, X. Meta multi-task learning for sequence modeling. Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

Tan, M.; Yu, Y.; Wang, H.; Wang, D.; Potdar, S.; Chang, S.; Yu, M. *arXiv preprint arXiv:1909.05357* **2019**,

Hu, Z.; Chen, T.; Chang, K.-W.; Sun, Y. *arXiv preprint arXiv:1907.00505* **2019**,

Xiong, W.; Yu, M.; Chang, S.; Guo, X.; Wang, W. Y. *arXiv preprint arXiv:1808.09040* **2018**,

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D., et al. Matching networks for one shot learning. Advances in neural information processing systems. 2016; pp 3630–3638.