Improving Syntactic Parsing with Consistency Learning

Anonymous ACL submission

Abstract

In this paper, we propose using *consistency learning* to improve constituency and dependency parsing performances on a multi-task setting. It utilizes a consistent constraint between the predictions. While multi-task learning implicitly learns shared representations for multiple sub-tasks, our method introduces an explicit consistency objective, which encourages shared representations that result in consistent predictions. Our intuition is that correct predictions are more likely consistent ones. To introduce consistent constraints, we propose a general method for introducing consistency objectives, as well as other prior knowledge, into existing neural models. This method only requires a boolean function that tells whether or not the multiple predictions are consistent, which does not need to be differentiable. We demonstrate the efficacy of our method by showing that it out-performs a state-of-the-art joint dependency and constituency parser on CTB.

1 Introduction

Multi-task learning (Caruana, 1998) uses a shared representation to learn sub-tasks of different objectives. While it has been shown to improve both learning efficiency and efficacy in applications of various domains, there is no guarantee that the shared representation in multi-task learning is more meaningful than those in separate tasks. In this paper, we propose *consistency learning*, which introduces a consistency objective for the shared representation that encourages consistent (non-conflicting) predictions. As an intuition, if the two predictions are consistent with each other, we can, to some extend, explain each prediction by using the other prediction as its necessary condition.

The only assumption in the proposed *consistency learning* method is a user-defined *consistency function*, which returns a *consistency label* indicating whether the given predictions are consistent or not. This function can be as simple as a few lines of code, expressing a prior knowledge about the data, and it is not required to be differentiable. Therefore, our method can also be applied to many objectives other than consistency. Specifically, in this paper, we aim at improving the consistency between each pair of constituency and dependency parses, and our consistency function simply checks if there are more than one dependency edge originated from each constituency span.

We have two intuitions about why consistency improves learning with multiple predictions. Firstly, correct predictions are likely to be consistent and incorrect ones are likely to be conflicting. Consistency objectives therefore effectively penalize the noisy training samples, which are pervasive in practice (Marcus et al., 1993a)). Secondly, the consistency label introduce a simple auxiliary task that is related to the original main task with multiple predictions.

To sum up, we make the following contributions in this paper:

- We propose the *consistency learning* method to improve the performance of multiple-prediction tasks.
- We applied the proposed method on a joint dependency and constituency parser and evaluate its efficacy over a state-of-the-art baseline (Mrini et al., 2020) .
- We obtain 0.43 F1 improvement on constituency parsing and a 0.36 UAS improvement on dependency parsing over the state-of-the-art joint parser on the CTB (Xue et al., 2005) dataset.

2 Consistency learning

In the section, we will first describe the proposed learning method in a simple multi-prediction setting that involves two predictions and one consistency objective. Then, we will extrapolate the

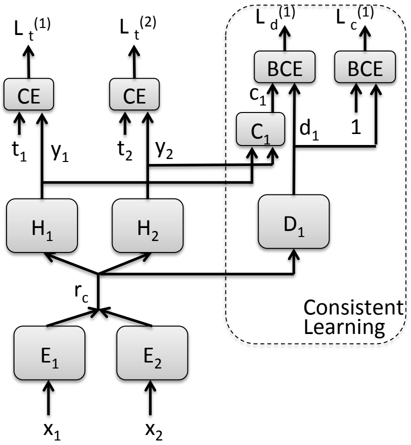


Figure 1: Illustration of adding consistency objective. The modules introduced for the objective are surrounded by the box in dotted line, and the assumed consistency function is C_i. A discriminator D_1 is first trained to imitate C_1 with loss $L_d^{(1)}$, and then provides feedback to r_c via loss $L_c^{(1)}$ to penalized inconsistent shared representation.

method to a more general setting with arbitrary number of inputs, outputs, and consistency objectives.

2.1 Consistency learning with two predictions

We base our consistency learning model on a multi-prediction task learning network as illustrated in Figure 1 and described in the following. Suppose we have two prediction tasks, and $L_t^{(1)}$ and $L_t^{(2)}$ are the training losses of the two sub-tasks, respectively. Inputs x_1 and x_2 of the two sub-tasks are encoded by their encoder neural networks, E_1 and E_2, which project their inputs, x_1 and x_2, into a shared representation, r_c. Here, if the two sub-tasks have the same type of input, they can also share the same encoder, i.e. $E_1 = E_2$. With the shared representation r_c, the two sub-tasks then use their own decoders heads, H_1 and H_2, to obtain their outputs, y_1 and y_2.

Previous succeed in multi-task learning has proven that a shared representation, r_c, that encodes shared knowledge of different sub-tasks can improve both learning efficiency and efficacy. For better model generalization, it is desired that, for each r_c from the same x_1 or x_2, the predictions from H_1 and H_2 should not contradict each other, since otherwise one of the predictions is incorrect. However, conventional multi-task learning does not explicitly define an objective to encourage such consistency, and it is questionable if such a consistency objective can alway be automatically induced from any training data.

Algorithm 1: Training with a consistency objective

Input: A sample in training dataset: (x_1, t_1, t_2)
Output: Gradients ΔE_1, ΔH_1, ΔH_2, ΔD_1.
$r_c \leftarrow E_1(x_1)$
foreach $1 \leq j \leq 2$ **do**
  $y_j \leftarrow H_j(r_c)$
  $L_t^{(j)} \leftarrow CE(y_j, t_j)$ // multi-task loss
  $\Delta H_j \leftarrow \frac{\partial L_t^{(j)}}{\partial H_j}$ // decoder gradient
$d_1 \leftarrow D_1(r_c)$
$c_1 \leftarrow C_1(y_1, y_2)$
$L_d \leftarrow BCE(d_1, c_1)$ // discriminator loss
$\Delta D_1 \leftarrow \frac{\partial L_d}{\partial D_1}$ // discriminator gradient
if $c_1 = 1$ **then**
  // consistency loss
  $L_c \leftarrow BCE(d_1, c_1 = 1)$
else
  $L_c \leftarrow 0$
$\Delta E_1 \leftarrow \frac{\partial(L_t + \alpha_c L_c)}{\partial E_i}$ // encoder gradient

Aiming at better generalization performance, we propose to explicitly introduce consistency learning objectives, which are missing in conventional multi-tasks learning. In the following, we will illustrate this in the above example. Our method only assumes simple user-defined *consistency functions*, e.g. $c_1 = C_1(y_1, y_2)$, where c_1 is a boolean indicating whether y_1 and y_2 are inconsistent.

In Figure 1, E_1, E_2, H_1, H_1 belong to the original model of the multiple prediction tasks. In order to introduce a consistent objective, we add a discriminator D_1, a user-defined consistency function C_1, and two loss functions, as shown in the right hand side of Figure 1. D_1 is first trained to find the distribution of r_c that will lead to consistent outputs. D_1 is then used to correct the distribution of r_c by moving it away from the wrong distribution via $L_c^{(1)}$. This approach followed the usage of the discriminator in GANs (Goodfellow et al., 2014).

2.2 Training with a consistency objective

For simplicity, we assume a single input x_1, and ignore x_2 in Figure 1. As an example, in joint constituency and dependency parsing, x_1 is an input sentence, t_1 is a constituency parse, t_2 is a dependency parse, and c_1 return whether there are conflicts between t_1 and t_2. We define three types of losses in total that are trained in parallel: *multi-task loss* L_t, the *discriminator loss* L_d, and the *consistency loss* L_c. Algorithm 1 shows how to obtain these losses and their derivatives for each sample of training data.

Assume the original tasks are classification problems. The *multi-task loss* $L_t = CE(y_1, t_1) + CE(y_2, t_2)$. We use the *binary cross entropy (BCE)* between the prediction of D_1 and $c_1 = C_1(y_1, y_2)$ to train D_1. Other loss functions rather than BCE can be used, similar to those defined in the GAN variations. Since the accuracy and consistency between y_1 and y_2 increase during training, the ground-truth c_1 in the above loss increases from 0 to 1 during training. To balance labels for L_d during training, we randomly select an equal amount of samples with $c_1 = 1$ and $c_1 = 0$ to obtain L_d. L_c is used to train E_1 while D_1 is fixed. Let B be a batch of training data, B' a subset of B where y_1 and y_2 are inconsistent, i.e. $c_1 = 1$ and r_c is inconsistent in each sample of B'. L_c is obtained on B' also with the BCE loss.

2.3 Extrapolation

The above simple consistency learning illustration with two predictions can be extrapolation to a general multi-prediction scenarios with multiple input, multiple output, and multiple functions of diferent prior knowledge can be added to the network to make r_c more meaningful.

3 Joint Parsing Experiment

Our implementation is based on the open source project [1] of the state-of-the-art joint dependency and constituency parser (Mrini et al., 2020) at the time of writing.

3.1 Baseline parser model

The joint parser model is illustrated in Figure 2 and briefly explained below. Please refer to the original paper for the detailed model, hype-parameter, and training settings. For English parsing experiments, we use the large cased pre-trained XL-Net (Yang et al., 2019). For Chinese parsing experiments, we use the bert-base-chinese BERT (Devlin et al., 2018) model. The output of the pre-trained model is input into the transformer encoder layers, and the output of the latter is the shared representation

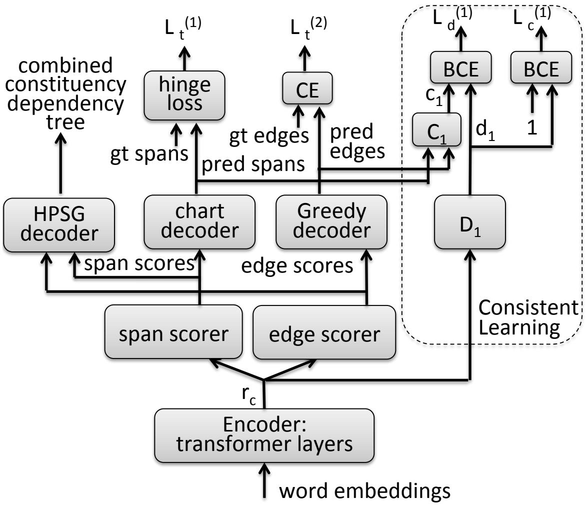


Figure 2: The neural network for the joint parsing experiment.

r_c. r_c is then sent to both the span and the edge scorers (Dozat and Manning, 2017; Stern et al., 2017). During training, the span scores are sent to the chart decoder to predict the best constituency tree with the maximum total span scores, and it is sent to the greedy decoder to predict the best dependency tree. The loss $L_t^{(1)}$ of the first sub-task is a hinge loss between the total span scores of the predicted constituency tree and that of its ground-truth. The loss $L_t^{(2)}$ of the second sub-task is the mean cross-entropy between each dependency edge and its ground-truth. At inference time, the span scores and the edge scores are used by the HPSG decoder (Zhou and Zhao, 2019) to decode a simplified HPSG tree, which is a combined constituency and dependency tree.

3.2 Experiment settings

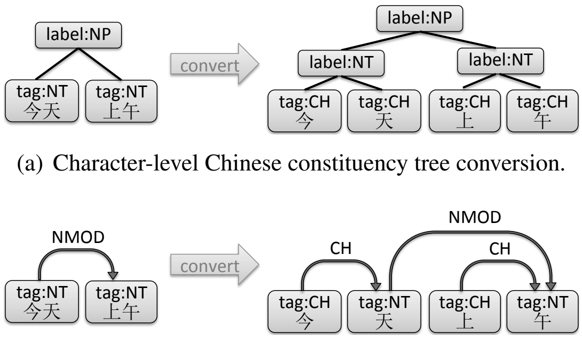
The constituency parsing dataset that we use include the *English Penn Treebank (PTB)* (Marcus et al., 1993b) and the *Chinese Treebank (CTB)* (Xue et al., 2005). We follow the standard data splits, and use the *EVALB* program (Sekine and Collins, 1997) to report the constituency parsing results. The English dependency trees are obtained by converting constituent trees with the Stanford Parser [2]. The Chinese dependency trees are converted from constituent trees with Penn2Malt [3].

In addition, we perform character-level Chinese constituency and dependency parsing (Zheng et al., 2015; Li et al., 2018; Yan et al., 2019). We use sim-

[1] https://github.com/KhalilMrini/LAL-Parser

[2] http://nlp.stanford.edu/software/lex-parser.html

[3] https://cl.lingfil.uu.se/ nivre/research/Penn2Malt.html



(a) Character-level Chinese constituency tree conversion.

(b) Character-level Chinese dependency tree conversion.

Figure 3: Illustration of character-level trees conversion.

		Recall	Precision	FScore
CTB	baseline	93.22	92.98	93.10
CTB	ours	**93.46**	**93.59**	**93.53**
CTB-C	baseline	95.69	95.39	95.51
CTB-C	ours	**95.78**	**95.43**	**95.57**
CTB-W	baseline	92.88	93.11	92.99
CTB-CW	baseline	93.04	93.19	93.11
CTB-CW	ours	**93.81**	**93.70**	**93.75**
PTB	baseline	96.08	96.23	96.16
PTB	ours	95.91	96.17	96.04

Table 1: Constituency parsing results

		UAS	LAS	UCM	LCM
CTB	baseline	94.53	93.02	62.36	53.74
CTB	ours	**94.89**	**93.38**	**65.23**	**56.90**
CTB-C	baseline	96.10	94.69	60.06	49.71
CTB-C	ours	**96.15**	**94.85**	**60.63**	**50.86**
CTB-W	baseline	94.90	92.91	67.92	58.70
CTB-CW	baseline	95.07	93.69	66.21	57.68
CTB-CW	ours	**95.08**	93.44	**68.26**	**60.07**
PTB	baseline	97.30	96.21	72.52	62.87
PTB	ours	97.20	96.10	71.32	62.33

Table 2: Dependency parsing results (w/o punct.)

		UAS	LAS	UCM	LCM
CTB	baseline	94.19	92.87	62.07	53.45
CTB	ours	**94.71**	**93.38**	**64.66**	**56.61**
CTB-C	baseline	95.78	94.57	60.06	49.71
CTB-C	ours	**95.80**	94.50	60.06	**50.29**
CTB-W	baseline	94.80	93.06	67.92	58.70
CTB-CW	baseline	94.85	93.63	66.21	57.68
CTB-CW	ours	**94.86**	93.42	**67.58**	**59.39**
PTB	baseline	96.94	95.98	68.87	60.06
PTB	ours	96.81	95.85	67.76	59.56

Table 3: Dependency parsing results (with punct.)

ple conversion methods to obtain character-level constituency and the dependency trees as explained in Figure 3, and we perform character-level experiments using the same model and hype-parameters as the word-level Chinese parsing experiment.

3.3 Consistency function

As shown in Figure 2, the consistency label c_1 is obtained from the predicted consistency spans, y_s, and dependency edges, y_e. $c_1 = C_1(y_s, y_e) = 1$ if the consistency tree derived from y_s and the dependency tree derived from y_e can be successfully combined into a simplified HPSG tree (Zhou and Zhao, 2019). In the simplified HPSG tree, for each constituent span $s \in y_s$, a word $h \in s$ is assigned the *head* and another word $p \notin s$ is assigned the *parent* if there exists a dependency edge $(h, p) \in y_e$. If any span $s \in y_s$ fails to find its unique h or p, y_s fails to combine with y_e, and $c_1 = 1$. Simply put, $c_1 = 0$ if all non-root constituency span have exactly one out-going dependency edge.

Our discriminator D_1 consists of three transformer layers with factored content and position information (Kitaev and Klein, 2018), one layer of label attention layer (Mrini et al., 2020), a two-layer feed-forward network with output size 1,024, a summation over the word index dimension, and a two-layer feed-forward network with a hidden size 64, a ReLU activation function, and an output linear layer of size 1.

3.4 Results

As shown in Tables 1 to 3, the results on CTB are obtained with a consistency loss weight $\alpha_c = 0.5$. Our consistency learning method achieves a 0.43 increment over the 93.10 F1, and a 0.36 improvement over the 94.19 UAS (without punctuations) of the state-of-the-art baseline.

For character-level Chinese parsing, we set $\alpha_c = 0.1$, and report results on the character-level trees (CTB-C) and on the back-converted word-level trees (CTB-CW). For the latter, 14% of the predicted parses have different word-segmentations than the CTB testing set. We therefore compare CTB-CW with this subset of the CTB testing set (CTB-W). Consistency learning shows improvement in all evaluation metrics except for LAS, probably due to the fact that our consistency objective focuses only on structures rather than labels.

On PTB, we failed to find an α to obtain improvement. The results reported are with $\alpha_c = 0.1$. It is probably because the consistency sub-task wastes extra model capacity while its improvement is not dominating in PTB whose size if twice that of CTB and its labels are less noisy and more consistent.

References

R. Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative Adversarial Networks. In *Proceedings of the 27th International Conferenceon Neural Information Processing Systems*, page 2672–2680.

Tsuyoshi Idé, Dzung T. Phan, and Jayant Kalagnanam. 2017. Multi-task multi-modal models for collective anomaly detection. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 177–186.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *CoRR*, abs/1705.07115.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database.

Haonan Li, Zhisong Zhang, Yuqi Ju, , and Hai Zhao. 2018. Neural character-level dependency parsing for chinese. In *AAAI*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993a. Building a large annotated corpus of english: The penn treebank.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993b. Building a large annotated corpus of english: The penn treebank.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Satoshi Sekine and Michael Collins. 1997. Evalb bracket scoring program. http://www. cs. nyu. edu/cs/projects/proteus/evalb.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2018. Kdgan: Knowledge distillation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 775–786.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. Natural language engineering.

Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2019. A unified model for joint chinese word segmentation and dependency parsing. In *arXiv preprint arXiv:1904.04697*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Xiaoqing Zheng, Haoyuan Peng, Yi Chen, Pengjing Zhang, and Wenqiang Zhang. 2015. Character-based parsing with convolutional neural network. In *IJCAI*.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on penn treebank. In *ACL*.