Injecting output constraints into neural NLP models

Jay Yoon Lee

May 2020

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
William Cohen, Co-Chair
Jaime Carbonell, Co-Chair
Graham Neubig
Yulia Tsvetkov
Dan Roth (University of Pennsylvania)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Structured Prediction, Arbitrary Knowledge Injection, Natural Language Processing, Multi-task, Transfer Learning, Domain adaptation, Semantic Role Labeling, Span-based models.

For my parents and Jaime.

Abstract

The goal of this thesis is injecting prior knowledge and constraints into neural models, primarily for natural language processing (NLP) tasks. While neural models have set new state of the art performance in many tasks from vision to NLP, they often fail to learn to consistently produce well-formed structures unless there is immense amount of training data. This thesis argues that not all the aspects of the model have to be learned from the data itself, and shows that injecting simple knowledge and constraints into the neural models can help low-resource, out-of-domain settings, as well as improve state-of-the-art models.

This thesis focuses on structural knowledge of the output space and injects knowledge of correct or preferred structures as an objective to the model without any modification to the model structure, in a model-agnostic way. The first benefit in focusing on knowledge on the output space is that it is *intuitive* as we can *directly enforce* output to satisfy logical/linguistic constraints. Another advantage of structural knowledge is that it often *does not require* a labeled dataset.

Focusing on deterministic constraints on the output values, this thesis first applies output constraints at inference time via the gradient-based inference (GBI) method. In the spirit of gradient-based training, GBI enforces constraints for each input at test-time by optimizing continuous model weights until the network's inference procedure generates an output that satisfies the constraints.

Then, this thesis shows that constraint injection on inference-time can be extended to training time: from instance-based optimization at test time to generalization to multiple instances at training time. In training with structural constraints, this thesis presents (1) a structural constraint loss, (2) a joint objective of structural loss and supervised loss on a training set and, (3) a joint objective in a semi-supervised setting. All the loss functions show improvements and among them, the semi-supervised approach shows the largest improvement and is particularly effective in a low-resource setting. The analysis shows that the efforts at training time and at inference time are complementary rather than exclusive: the performance is best when efforts on train-time and inference-time methods are combined.

Finally, this thesis presents an agreement constraint on a multi-view learning that can utilize the semi-supervised approach with the constraint. The presented agreement constraint in multi-view learning is general in that it can be applied to any sequence-labeling problem with multiple views, while other constraints in this thesis consider prior knowledge about specific tasks. This semi-supervised approach again shows large gains in low-resource settings and shows effectiveness on high-resource as well.

Acknowledgments

.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The importance of constraints in learning

With recent advances in artificial neural network models and computing power, the neural models have set new state-of-the-art performances in many tasks across different applications such as vision and Natural Language Processing (NLP). However, even the best-performing models often fail to learn to consistently produce constraint-satisfying outputs unless there is an immense amount of training data.

To this end, this thesis asks whether ancillary knowledge or constraint injection into the neural models can benefit the learning process. Machine Learning, especially neural network, is a great tool for modeling and learning a relation between an input X and output Y as a black-box function. The common approach is maximizing the conditional likelihood of Y given X on training data. While this black-box modeling approach can be beneficial in learning complex functions that we do not know how to model, this thesis shows that not all the aspects of the model have to be learned solely from the training data itself when there is prior knowledge for the task. This thesis studies whether the injection of ancillary knowledge or constraints to the neural models leads to not only better satisfaction of constraints but also to a better performance, especially in, but not limited to, low-resource settings.

NLP often requires learning to perform multiple tasks and this multi-task setting can be expressed as: primary task and ancillary tasks where ancillary tasks can sometimes be expressed as simple constraints to satisfy. Furthermore, in an automated NLP system, model's output is often used as an input to the downstream tasks such as a question answering or recommendation system, where simple constraint violation can lead to unrecoverable errors for the downstream tasks.

In the pre-neural era, in order to satisfy constraints, a cascaded system was often employed. For example, in semantic role labeling (SRL), the model has to identify the relationship between a verb predicate and a word span. This problem could be decomposed into two tasks: chunking a sentence into spans and labeling each of the spans given the verb predicate. A typical SRL system would first output a set of possible spans using a syntactic parser and then sequentially find the most likely labels on the provided spans. Nonetheless, thanks to the representation power of neural network, end-to-end learning showed the best efficacy with a system simultaneously

outputting the spans and labels in case of SRL.

Most current state-of-the-art models in NLP are learned end-to-end. However, as ironic as it may sound, the state-of-the-art models often violate simple constraints as opposed to the old cascaded models which have zero violation. Returning to the SRL example, note that there is a constraint on SRL spans that they must all be parsing constituents. While the unconstrained end-to-end model gives better overall performance, the limited, weaker performing, cascaded system did not have any constraint violation on the output unless the parser made errors as the system only considered valid spans from the syntactic parser.

Enforcing constraints may be executed by simple post processing such as filtering or re-ranking outputs using a constraint-aware cost function. This thesis shows that we can actually learn more from the fact that the neural representation fails to satisfy a constraint. The thesis further asks whether a simple measure of constraint violation can be a useful signal of model quality and whether this signal can be leveraged to learn a better model.

To address this question, the thesis focuses on structural knowledge of the output space as a constraint, and uses the degree of constraint violation as a learning signal for the model in a model-agnostic way, i.e. without modification to the model structure (Lee et al., 2017; Mehta et al., 2018). The first benefit in focusing on knowledge of the output space is that it is intuitive, as we can directly enforce logical and linguistic constraints. Another advantage of structural knowledge of the output space is that it often does not require labeled data to evaluate. Leveraging these advantages, I enforce structural constraints to improve inference procedure using the gradient-based inference (GBI) method, to improve on low-resource scenarios via semi-supervised learning, and also to improve on cases such as domain adaptation where the resource is only available on the source side and there is no labeled dataset on the target domain.

1.2 Organization of the thesis

The organization of the thesis is provided as follows.

Chapter 2 discusses the previous work relevant to constraint injection. While each chapter has its own section for literature review, Chapter 2 serves as a short survey of the constraint injection methods relevant to more than one of the subsequent chapters.

Chapter 3 introduces the gradient-based inference (GBI), a framework for structured prediction that can utilize a provided constraint function at inference time. We present a novel algorithm to convert the discrete combinatorial problem into search in continuous space using gradients from a constraint violation signal. This chapter introduces a *constraint-loss* function, which is used throughout this thesis, that can be defined whenever constraint violation can be expressed as a positive scalar score function. The rest of this chapter shows the experimental results on GBI across different applications and diverse set ups. On all the experiments, GBI increased the main metric (F1 or % accuracy) of various tasks in the process of enforcing constraints. The material presented in this chapter is based on Lee et al. (2019).

Chapter 4 proposes a semi-supervised learning framework that can leverage known constraints. The key idea comes from noticing that an evaluation of constraint satisfaction does not require any labeled data. This idea led to extending the *constraint-loss* defined for inference in the previous chapter into unsupervised loss for training. We present three different loss functions

(unsupervised, joint loss and semi-supervised loss) and examine the effect of each loss function in terms of F1 score and mean constraint violation score. In the context of semantic role labeling, this chapter shows that semi-supervised learning can significantly help low-resource models as well as improving high-resource models on the popular OntoNotes v5.0 (Pradhan et al., 2013) dataset. This chapter is an exposition of Mehta et al. (2018)[1].

Chapter 5 proposes CO-META, a semi-supervised learning framework that promotes coherence between multi-view models to further improve them. While other chapters focus on constraint functions based on a knowledge of specific applications, this chapter proposes a general agreement constraint that can be defined in multi-view setup. Motivated by Co-Training (Blum and Mitchell, 1998), CO-META encourages multi-view models to learn from each other. However, rather than approaching the problem as data augmentation as Co-Training did, CO-META weighs learning signals by comparing with other models in multi-view setup. To use the comparison score in the learning process, a *constraint-loss* is again applied in this chapter. Similar to Chapter 4, but this time in the context of dependency parsing, the experiments on CO-META show significant improvement on low-resource models. In addition, the presented method improves performance in some of the relatively high-resource tasks such as Chinese dependency parsing. This chapter is based on Lim et al. (2020)[2]

Lastly, Chapter 6 concludes the findings of this dissertation and suggests future directions for research.

[1]For Mehta et al. (2018), Lee and Mehta are the co-first authors with equal contributions.
[2]For Lim et al. (2020), Lee and Lim are the co-first authors with equal contributions.

Chapter 2

Related Work

In this related work section, I will survey existing methods that inject constraints into the model inference and learning steps and their relation to the methods in this thesis.

2.1 Inference with output constraints

The process of inference is the process of producing an output with maximum score given the trained model and input. In other words, given the trained model weight θ, input $\mathbf{x}$, and a score function $\Psi(\mathbf{x}, \mathbf{y}, \theta)$, the goal is to obtain $\hat{y} = \arg\max_y \Psi(\mathbf{x}, \mathbf{y}, \theta)$. In particular, this thesis focuses on the inference of sequence output $\mathbf{y}$ with variable length t (i.e., $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2 \dots \mathbf{y}_t$) that has a constraint defined over $\mathbf{y}$. If one wants to enforce constraint in the inference procedure with output $\mathbf{y}$ to belong in the constraint set $\mathcal{L}^{\mathbf{x}}$, one can solve the following optimization problem:

$$\begin{aligned} \max_{\mathbf{y}} \quad & \Psi(\mathbf{x}, \mathbf{y}, \theta) \\ \text{s. t.} \quad & \mathbf{y} \in \mathcal{L}^{\mathbf{x}} \end{aligned} \tag{2.1}$$

For problem (2.1), standard inference methods, built without consideration of constraints, cannot be used blindly since the resulting output might violate the constraints (i.e.,$\mathbf{y} \notin \mathcal{L}^{\mathbf{x}}$). To use standard inference methods as is, one would need to apply post processing on the constraint-violating output to modify the output to the closest constraint-satisfying form or to get multiple candidates and filter out ones with a constraint violation. However, this process might not bring the optimal solution for the problem (2.1). In order to bring the constraint satisfaction into optimization, one can define constraint evaluation function $g(\mathbf{y}, \mathcal{L}^{\mathbf{x}}) \rightarrow \mathbb{R}_0^+$ that measures a loss between output $\mathbf{y}$ and a constraint set $\mathcal{L}^{\mathbf{x}}$ such that $g(\mathbf{y}, \mathcal{L}^{\mathbf{x}}) = 0$ if and only if there are no constraint violations in $\mathbf{y}$. There are two important lines of work that incorporate a constraint function $g(\mathbf{y}, \mathcal{L}^{\mathbf{x}})$ in the inference step: the dual decomposition method (Rush et al., 2010) and the constrained conditional model (CCM) (Chang et al., 2012).

Dual decomposition (Sontag et al., 2010) is the classic approach to solving the optimization problem with constraints and also has been widely used in NLP community (Rush et al., 2010; Riedel and McCallum, 2011; Das et al., 2012) as well. Lagrangian relaxation is the most widely adopted technique of dual decomposition and converts the original *primal* problem (2.2)

$$\begin{aligned} \max_{\mathbf{y}} \quad & \Psi(\mathbf{x}, \mathbf{y}, \theta) \\ \text{s.t.} \quad & g(\mathbf{y}, \mathcal{L}) = 0, \end{aligned} \tag{2.2}$$

to *dual* form (2.3) as

$$\min_{\lambda} \max_{\mathbf{y}} \Psi(\mathbf{x}, \mathbf{y}, \theta) + \lambda g(\mathbf{y}, \mathcal{L}). \tag{2.3}$$

Note that the primal problem (2.1) is equivalent to Eq. (2.2) as $g(\mathbf{y}, \mathcal{L}^{\mathbf{x}}) = 0 \leftrightarrow \mathbf{y} \in \mathcal{L}^{\mathbf{x}}$. Many NLP applications(Rush et al., 2010; Riedel and McCallum, 2011; Das et al., 2012) solved (2.3) via alternating the minimizing and maximizing optimization steps. The most popular setting in these applications were where the model and constraints are both linear:

$$\min_{\lambda} \max_{\mathbf{y}} \ \mathbf{y}\theta + \lambda(A\mathbf{y} - b). \tag{2.4}$$

With vectors θ, λ, and matrix A, $\mathbf{y}\theta$ and $\lambda(A\mathbf{y} - b)$ in Eq. (2.4) corresponds to linear version of $\Psi(\mathbf{x}, \mathbf{y}, \theta)$ and $\lambda g(\mathbf{y}, \mathcal{L})$ in Eq. (2.3). In such a case, the maximization step with constraint reduces to the same complexity as the unconstrained inference step as one can solve (2.4) by solving $\max_{\mathbf{y}} \mathbf{y}\theta'$, which has the same form as the unconstrained inference step where $\theta' = \theta + A^T\lambda$. However, when the constraint function is non-linear or global, this maximization step becomes very expensive as the maximization becomes a combinatorial search problem, which has exponential complexity $O(V^t)$ with V being the output space for each token y_i.

CCM (Chang et al., 2008, 2012) is another popular model that incorporates constraints in the inference process. The formulation of CCM(2.5) is similar to the (2.3) dual decomposition in that CCM introduces a penalty term with constraint violation, as the Lagrangian relaxation technique does.

$$\max_{\mathbf{y}} \Psi(\mathbf{x}, \mathbf{y}, \theta) + \alpha g(\mathbf{y}, \mathcal{L}) \tag{2.5}$$

However, the two methods take very different perspective: note that $\min_y$ operation is subtracted from Eq. (2.3) to Eq. (2.5) and that λ has been modified to α. CCM *permanently expands* its model to incorporate a constraint penalty term and learns the α[1] to do so. In contrast, dual decomposition is introducing a *temporary auxiliary* variable, the dual variable λ, per new input to perform efficient search of the optimal output $\hat{y}$. As a result, CCM applies the same penalty weight for every instance it visits while dual decomposition optimizes per each instance. Expanding the model with a constraint penalty term makes CCM's inference more efficient than dual decomposition since it does not take alternating minimization and maximization operation steps, nevertheless, the expansion comes with one caveat: CCM has to be exposed to the constraint-violating examples in the training process.

While there are some differences between dual decomposition and CCM, the two methods share many strengths and weaknesses. Both approaches have the strength that the method can be applied in a model-agnostic way to general constraints. However, both approaches show a weakness for the global constraint. The maximization step of Eq. (2.3) and Eq. (2.5), which shares the same form in the maximization step, requires combinatorial search in presence of

[1]CCM learns α to be the ratio of probability ratio of constraint violation happening over not happening. The probabilities are estimated through counting how many instance violate constraint in the training set.

global constraint, which leads to exponential complexity $O(V^t)$ respect to the output length t. Furthermore, requiring combinatorial search for a global constraint is even more problematic for the the sequence-to-sequence model (seq2seq), a widely used neural sequence model, as combinatorial search on seq2seq is extremely expensive.[2]. Thus there has been no previous research that has applied CCM or dual decomposition method to neural sequence models. No inference method with constraints has been applied to the seq2seq model for inference with global constraint, to the best of my knowledge.

To summarize, the central problem with prior approaches is that the maximization step becomes too expensive given the *fixed* model weights and global constraint penalty term. To avoid this, this thesis explores a new approach: "*Can we search for a constraint-satisfying output by nudging model weights θ with gradients from constraint violation loss?*". By changing the problem of searching instance to nudging the model weights, GBI reduces the search to back propagation, the complexity[3] of which is often much smaller than combinatorial search complexity of $O(V^t)$ for NLP applications. Through experiments on synthetic transducer problem and syntactic parsing, this thesis shows that GBI can efficiently improve the performance of the seq2seq framework given global constraints. Furthermore, with experiments on SRL, this thesis also shows that GBI is both faster and better performing than the combinatorial search approach, by comparing to Astar-search for the sequence tagging model with global constraints.

2.2 Learning with output constraints

Learning with explicit constraints was a popular direction of research and various approaches which share a similar framework were proposed around the same era: Constraint-Driven Learning (CoDL) (Chang et al., 2012), Posterior Regularization (PR) (Ganchev et al., 2010), and Unified EM (UEM) (Samdani et al., 2012). Most of the approaches were based on the Expectation Maximization (EM) algorithm (Dempster et al., 1977) framework. To compare and contrast each of the methods in detail, the section begins by explaining the common framework: the EM algorithm.

The goal of the EM algorithm is to maximize the marginal likelihood $\mathcal{L}(\theta) = \log P(Y|X;\theta)$ by maximizing the following variational lower bound $F(q,\theta)$ of the original loss function $\mathcal{L}(\theta)$ [4]. The lower bound $F(q,\theta)$ is obtained by applying Jensen's inequality to the $\mathcal{L}(\theta)$ as

$$\begin{aligned}\mathcal{L}(\theta) = \log P(Y|X;\theta) = \log \sum_{Y\in\mathcal{Y}_x} q(Y)\frac{P(Y|X;\theta)}{q(Y)} &\geq \sum_{Y\in\mathcal{Y}_x} q(Y)\log\frac{P(Y|X;\theta)}{q(Y)} \\ &= F(q,\theta) = -\mathbf{KL}\left(q(Y)||P(Y|X;\theta)\right)\end{aligned}$$

[2] Seq2seq models the probability y_t at time t to be different if its prefix $y_1, y_2, \ldots, y_{t-1}$ is different which means that every possible prefix have to be evaluated separately for the combinatorial search. The exponential complexity of combinatorial search inflates by complexity of forward propagation of RNN. Typical complexity of RNN forward propagation is $O(tLh^2)$ for length t, number of layers L, and representation embedding size h.

[3] Which is the same as the aforementioned forward propagation complexity, i.e. $O(tLh^2)$.

[4] While common description of EM algorithm maximizes the distribution of $P(X;\theta)$ as the thesis mainly study direct modeling of conditional models $P(Y|X;\theta)$, the description will also focus on the conditional models.

where q, θ stands for posterior distribution of the output space and model parameter respectively. Typically, the EM algorithm maximizes $F(q, \theta)$, by employing block-coordinate ascent following the style of (Neal and Hinton, 1998) where Expectation (E), Maximization (M) steps are defined as,

$$\begin{aligned} \mathrm{E}: q^{t+1} &= \arg\max_{q \in \mathcal{Q}} F(q, \theta^t) = \arg\min_{q \in \mathcal{Q}} \mathbf{KL}\left(q(Y) || P(Y|X;\theta)\right), \quad (2.6) \\ \mathbf{M}: \theta^{t+1} &= \arg\max_{\theta} F(q^{t+1}, \theta) = \arg\min_{\theta} \mathbf{KL}\left(q^{t+1}(Y) || P(Y|X;\theta)\right) \\ &= \arg\max_{\theta} \mathbf{E}_{q^{t+1}}\left[\log P(Y|X;\theta)\right], \quad (2.7) \end{aligned}$$

where $\mathcal{Q}$ is defined as the space of all possible posterior distribution. This EM block-coordinate ascent will alternate over E-steps and M-steps until convergence criterion is met. The EM alternation guarantees to increase the $J(\theta) = \arg\max_q F(q, \theta)$ through the following simple reasoning (Neal and Hinton, 1998) :

$$J(\theta^{t+1}) = F(q^{t+2}, \theta^{t+1}) \geq F(q^{t+1}, \theta^{t+1}) \geq F(q^{t+1}, \theta^t) = J(\theta^t).$$

The key idea for incorporating constraints, which were generally shared across previous approaches(Ganchev et al., 2010; Chang et al., 2012; Samdani et al., 2012), was to modify the E steps with constraint information. More specifically, the previous approaches have incorporated provided constraints by restricting the possible posterior distribution space $\mathcal{Q}$ considered on the E steps using the constraint information. When one wants to have a constraint-satisfying output, one way of doing it is by limiting the output space considered to be the feasible set $\mathcal{L}^{\mathbf{x}}$, by defining $\mathcal{L}^{\mathbf{x}}$ to include all constraint-satisfying outputs in the entire output space. For example, a linear constraint $\phi(x, y) \leq b$ with constant vector b and constraint feature function $\phi(x, y)$ can define a feasible set $\mathcal{L}^{\mathbf{x}}$ as $\{y | \phi(x, y) \leq b\}$. Similarly, we can define the feasible space for posterior distribution $\mathcal{Q}_{\mathcal{L}^{\mathbf{x}}}$ as the space of all the distributions which will satisfy constraint in expectation as $\mathcal{Q}_{\mathcal{L}^{\mathbf{x}}} = \{q | \mathbf{E}_q\left[\phi(x, y)\right] \leq b\}$ given original constraint $\phi(x, y) \leq b$. In summary, while general EM considered all possible posterior space $\mathcal{Q}$ on the E steps, the previous methods incorporated constraint information by limiting the search space of E steps to $\mathcal{Q}_{\mathcal{L}^{\mathbf{x}}}$ so the constraint can guide block-coordinate ascent of EM algorithm. In this section, we denote this approach as *constraint-modified EM*.

One standard way of solving EM is by applying stochastic gradient descent on the M step with the samples $y \sim q$ drawn from the q distribution obtained from the E step. However, there is a variant called hard EM which only takes gradient with y' that satisfies the mode of q or in other words, hard EM defines q as Kronecker-Delta distribution $\delta(y = y')$ centered at y'[5]. In some cases, this hard EM is known to work better than regular EM (Spitkovsky et al., 2010).

The same analogy between standard EM and hard EM applies to distinguish the approaches PR and CoDL takes in solving constraint-modified EM. PR solves the constraint-modified EM with standard EM approach and CoDL solves it with hard EM. Unified EM (UEM) approach tried to express the different variants of constraint learning methods in a unified way (Samdani

[5]Kronecker-Delta distribution $\delta(y = y')$ puts a probability of 1 at the center and 0 elsewhere.

et al., 2012). UEM unified the previous methods by adding a parameter γ to the original objective function of E step (2.6) as followings:

$$E : q^{t+1} = \arg\min_{q \in \mathcal{Q}_{\mathcal{L}^{\mathbf{x}}}} \mathbf{KL}\left(q(Y)||P(Y|X;\theta);\gamma\right) \tag{2.8}$$

$$= \arg\min_{q \in \mathcal{Q}_{\mathcal{L}^{\mathbf{x}}}} \sum_{y} \gamma q(y)\log q(y) - q(y)\log P(y|x). \tag{2.9}$$

The new parameter γ controls the weight of the negative entropy term of q. $\gamma = 1$ represents PR method as it makes Eq. (2.8) the same as the standard EM formulation Eq. (2.6). $\gamma = -\inf$ corresponds to CoDL. Taking $\gamma = -\inf$ in minimization problem leads to ignoring the distribution $P(y|x)$ and thus this leads to the same formulation of hard EM.

To summarize, PR and CoDL differs by the mechanics they take in obtaining a sample from $q \in \mathcal{Q}_{\mathcal{L}^{\mathbf{x}}}$, to take M step in the constraint-modified EM. PR samples $y \sim q$ and takes gradient steps with this y. This step requires computation of denominator for q distribution and will have complexity of $O(V^t)$ for sequence y with length t. CoDL, instead, requires finding the max-scoring output of unnormalized q. The unnormalized q basically has the penalty term for constraint violation on top of $P(y)$ and takes the same form as CCM (2.5)[6]. While finding max-scoring output does not require computation of normalization factor, in the face of global constraint, this step too will require $O(V^t)$ as discussed in the previous survey of constrained inference.

All of the approaches reviewed in this section focus on modifying the E steps in order to incorporate constraint information in the learning process. However, to the best of my knowledge, no studies have explored adjusting the M steps to inject constraint function into learning. This thesis attempts to bring in this new approach in Part II by modifying the optimization step, which is similar to the adjusting the M step[7], to be influenced by the constraint function. The idea in the thesis is to guide the gradient's direction and magnitude by introducing the constraint satisfaction score $g(y, \mathcal{L}^{\mathbf{x}})$, such that $g(y, \mathcal{L}^{\mathbf{x}}) = 0 \iff y \in \mathcal{L}^{\mathbf{x}}$, on the current performance of the model much like the REINFORCE (Williams, 1992) algorithm.

2.3 Modeling with prior knowledge

In the earlier subsections, we mainly looked at injecting constraints via modification of objectives in model-agnostic way without changing the model structure. However, there are other works that tried to tie the constraints directly to the model formulation as well. These types of work vary by how explicit the constraint was expressed and to what degree the model structure embodies the constraint. We will discuss these variants of constraint incorporation in model structure with a focus on application of our interest: syntactic parsing and SRL.

[6]The only difference between the formulations of CoDL and CCM (from the previous section, 2.1) is that CCM only learns the model parameter and penalty weights from the labeled training set whereas CoDL utilizes the unlabeled data.

[7]While there are no EM steps on the proposed work in Part II To make even more concrete comparisons with the previous work which uses the modified EM framework, one could regard the thesis simply regards q obtained on the E-step as a conditional model distribution $P(Y|X;\theta)$.

Introducing constraint-related features to a model was a popular approach before the neural models became prevalent as now. For example, in syntactic parsing, Collins (2003) have added distance features to reflect the importance of *distance to the head* in syntactic parsing and also added gap features in order to reflect knowledges from generalized phrase structure grammar (Gazdar et al., 1985). With these features, they added a prior into the model so that a parse tree with long distance dependency (to the head of the tree) will get lower probability. While not explicitly enforcing the output constraints, this approach expands the input features to enable probabilistic modeling of the exterior knowledge.

With the recent neural models, explicit probabilistic modeling with extra features are difficult to implement. Most models are trained in end-to-end fashion which means that just input and output pairs are presented to the black-box neural network. In this black-box case, returning to the example of (Collins, 2003), even if we add extra distance feature, it is hard to add the long distance dependency prior as discussed, as we do not know specific roles each features play in the black-box model.

However, the black-box approach of neural network opened up many more ways to incorporate constraints albeit sometimes not explicitly. The most extreme case is just knowing that some set of tasks are relevant to each other and jointly training the related tasks rather than explicitly modeling the constraint. The multi-task training made the task-specific features unnecessary by processing the same input with a shared infrastructure. A common infrastructure across tasks made it easy to train multiple task objectives simultaneously with a shared parameters. Perhaps the first attempt of this kind is Collobert et al. (2011) which learned neural network model that jointly models the Part-Of-Speech tagging, Chunking, Named Entity Recognition, and SRL together. Collobert et al. (2011) also showed that pre-training neural networks simply with a language model objective and fine-tuning the specific tasks improves the model performance, leveraging the relation of language modeling to many other NLP tasks. Recently, BERT (Devlin et al., 2018) essentially took the similar approach, albeit on much larger scale, and surpassed many state-of-the art results in numerous tasks.

The naive multi-task approach of simply training multiple objectives together with shared parameters is useful even when a explicit constraint is given. Using the constraint between SRL output and parse-tree output as introduced in the section 3.2.1, Swayamdipta et al. (2018) jointly trained SRL model and shallow parser together and showed improvement in SRL's performance. While acknowledging that parse-tree span information is important for the SRL task, this approach does not explicitly enforce constraint, e.g, the fact that any SRL span has to be an element of the set of parse-tree spans.

More direct injection of relation between SRL and parse tree in neural model via modification of models structure was proposed concurrently by Strubell et al. (2018). This work injected the knowledge of dependency parse tree, rather than the knowledge of constituency parse tree, in the self-attention (Vaswani et al., 2017) level. Dependency parse tree provides the dependency information between the word pairs, and this work uses the dependency weights between a word pair to directly model self-attention weights between the same word pair.

Injecting constraints in a model-agnostic way, as in section 2.2 differs from incorporating constraint in model structure such as feature expansion and multi-task learning. The model-agnostic way is mainly comprised of changing the objective function to reflect the constraint information. In order to do this, one has to formulate exact constraint information in mathematical

expression whereas the multi-task approach can be taken even when one is not exactly sure about the relation between the output of multiple tasks. However, when you know the exact expression it often can be more powerful to incorporate the constraint in a model-agnostic way. Designing the model structure to exactly incorporate an explicit constraint is complex and this is why many work simply uses multi-task approach of simply jointly training multiple task objective without carefully modeling the relation between the tasks even when they know the relation explicitly. Furthermore, in neural models, state-of-the-art models often rely on many engineered factors, from structure to training schemes, and modifying one of the aforementioned factors often hurts the model. Thus, incorporating the constraint in the model-agnostic way often enhances the model in a more stable manner, across domains and across applications, as demonstrated in this thesis.

Part I

Inference with Output Constraints

Practitioners apply neural networks to increasingly complex problems in natural language processing, such as syntactic parsing and semantic role labeling that have rich output structures. Many such structured-prediction problems require deterministic constraints on the output values; for example, in sequence-to-sequence syntactic parsing, we require that the sequential outputs encode valid trees. While hidden units might capture such properties, the network is not always able to learn such constraints from the training data alone, and practitioners must then resort to post-processing. In this chapter, we present an inference method for neural networks that enforces deterministic constraints on outputs without performing rule-based post-processing or expensive discrete search. Instead, in the spirit of gradient-based training, we enforce constraints with gradient-based *inference* (GBI): for each input at test-time, we nudge continuous model weights until the network's unconstrained inference procedure generates an output that satisfies the constraints. We study the efficacy of GBI on three tasks with hard constraints: semantic role labeling, syntactic parsing, and sequence transduction. In each case, the algorithm not only satisfies constraints, but improves accuracy, even when the underlying network is state-of-the-art.

Chapter 3

Inference with Output Constraints: Gradient-Based Inference (GBI)

Suppose we have trained a sequence-to-sequence (seq2seq) network (Cho et al., 2014; Sutskever et al., 2014; Kumar et al., 2016) to perform a structured prediction task such as syntactic constituency parsing (Vinyals et al., 2015). We would like to apply this trained network to novel, unseen examples, but still require that the network's outputs obey an appropriate set of problem specific hard constraints; for example, that the output sequence encodes a valid parse tree. Enforcing these constraints is important because downstream tasks, such as relation extraction or coreference resolution, typically assume that the constraints hold. Moreover, the constraints impart informative hypothesis-limiting restrictions about joint assignments to multiple output units, and thus enforcing them holistically might cause a correct prediction for one subset of the outputs to beneficially influence another.

Unfortunately, there is no guarantee that the neural network will learn these constraints from the training data alone, especially if the training data volume is limited. Although in some cases, the outputs of state-of-the-art systems mostly obey the constraints for the test-set of the data on which they are tuned, in other cases they do not. In practice, the quality of neural networks are much lower when run on data in the wild (e.g., because small shifts in domain or genre change the underlying data distribution). In such cases, the problem of constraint violations becomes more significant.

This raises the question: how should we enforce hard constraints on the outputs of a neural network? We could perform expensive combinatorial discrete search over a large output space, or manually construct a list of post-processing rules for the particular problem domain of interest. However, we might do even better if we continue to "train" the neural network at test-time to learn how to satisfy the constraints on each input. Such a learning procedure is applicable at test-time because learning constraints requires no labeled data: rather, we only require a function that measures the extent to which a predicted output violates a constraint.

In this chapter, we present *gradient-based inference* (GBI), an inference method for neural networks that strongly favors respecting output constraints by adjusting the network's weights at test-time, for each input. Given an appropriate function that measures the extent of a constraint violation, we can express the hard constraints as an optimization problem over the continuous weights and apply back-propagation to tune them. That is, we iteratively adjust the weights so

that the neural network becomes increasingly likely to produce an output configuration that obeys the desired constraints. Much like scoped learning, the algorithm customizes the weights for each example at test-time (Blei et al., 2002), but does so in a way to satisfy the constraints.

We study GBI on three tasks, semantic role labeling (SRL), syntactic constituency parsing and a synthetic sequence transduction problem, and find that the algorithm performs favorably on all three tasks. In summary, our contributions are that we:

1. Propose a novel Gradient-Based Inference framework.
2. Verify that GBI performs well on various applications, thus providing strong evidence for the generality of the method.
3. Examine GBI across wide range of reference model performances and report its consistency.
4. Show that GBI also perform well on out-of-domain data.

For all the tasks, we find that GBI satisfies a large percentage of the constraints (up to 98%) and that in almost every case (out-of-domain data, state-of-the art networks, and even for the lower-quality networks), enforcing the constraints improves the accuracy. On SRL, for example, the method successfully injects truth-conveying side-information via constraints, improving state-of-the-art network[1] by 1.03 F1 (Peters et al., 2018b). This improvement surpass a $\mathtt{A}^*$-search algorithm for incorporating constraints while also being robust, in a way that $\mathtt{A}^*$is not, to cases for which the side constraints are inconsistent with the labeled ground truth.

3.1 Constraint-aware inference in neural networks

We present below an *approximate* optimization algorithm that is similar in spirit to Lagrangian relaxation in that we replace a complex constrained decoding objective with a simpler unconstrained objective that we can optimize with gradient descent (Koo et al., 2010; Rush et al., 2010; Rush and Collins, 2012), but is better suited for non-linear non-convex optimization with global constraints that do not factorize over the outputs. Although the exposition in this section uses the motivation of Lagrangian relaxation, we emphasize that the purpose is merely to provide intuition and motivate design choices.

3.1.1 Problem definition and motivation

Typically, a neural network parameterized by weights θ is a function from an input $\mathbf{x}$ to an output $\mathbf{y}$. The network has an associated compatibility function $\Psi(\mathbf{y}; \mathbf{x}, \theta) \rightarrow \mathbb{R}_+$ that measures how likely an output $\mathbf{y}$ is given an input $\mathbf{x}$ under weights θ. The goal of inference is to find an output that maximizes the compatibility function, and this is usually accomplished efficiently with feed-forward greedy-decoding. In this work, we want to additionally enforce that the output values belong to a feasible set or grammar $\mathcal{L}^{\mathbf{x}}$ that in general depends on the input. We are thus

[1]Since our submission of Lee et al. (2019) which this chapter is based on, the state-of-the-art Peters et al. (2018b) in SRL, on which we apply our technique, has been advanced by 1.7 F1 points (Ouchi et al., 2018). However, this is a training time improvement which is orthogonal to our work.

interested in the following optimization problem:

$$\max_{\mathbf{y}} \quad \Psi(\mathbf{x}, \mathbf{y}, \theta) \text{ s. t. } \mathbf{y} \in \mathcal{L}^{\mathbf{x}} \tag{3.1}$$

Simple greedy inference is no longer sufficient since the outputs might violate the global constraints (i.e., $\mathbf{y} \notin \mathcal{L}^{\mathbf{x}}$). Suppose instead that we had a constraint evaluation function $g(\mathbf{y}, \mathcal{L}) \rightarrow \mathbb{R}_0^+$ that measures a degree of violation of an output $\mathbf{y}$ respect to a constraint $\mathcal{L}$ such that $g(\mathbf{y}, \mathcal{L}) = 0$ if and only if there is no constraint violation in $\mathbf{y}$. That is, $g(\mathbf{y}, \mathcal{L}) = 0$ for the feasible region and is strictly positive everywhere else. For example, if the feasible region is a CFL, g could be the *least errors count* function (Lyon, 1974). We could then express the constraints as an equality constraint and minimize the Lagrangian:

$$\min_{\lambda} \max_{\mathbf{y}} \Psi(\mathbf{x}, \mathbf{y}, \theta) + \lambda g(\mathbf{y}, \mathcal{L}) \tag{3.2}$$

However, this leads to optimization difficulties because there is just a single dual variable for our global constraint, resulting in an intractable problem and thus leading to brute-force trial and error search for λ.

Instead, we might circumvent these issues if we optimize over the model parameters rather than a single dual variable. Intuitively, the purpose of the dual variables is to simply penalize the score of *infeasible* outputs that otherwise have a high score in the network, but happen to violate constraints. Similarly, the network's weights can control the compatibility of the output configurations with the input. By properly adjusting the weights, we can affect the outcome of inference by removing mass from invalid outputs—in much the same way a dual variable affects the outcome of inference. Unlike a single dual variable however, the network expresses a *different* penalty weight for each output. And, because the weights are typically tied across space (e.g., CNNs) or time (e.g., RNNs) the weights are likely to generalize across related outputs. As a result, lowering the compatibility function for a single invalid output has the potential effect of lowering the compatibility for an entire family of related, invalid outputs; enabling faster search.

With this in mind, it is tempting to replace the single dual-variable with a "dual neural-network" that is parameterized by a set of "dual weights." This is powerful because we have effectively introduced an exponential number of implicit "dual variables" (via the compatibility function, which scores each output) that we can easily control via the weights; although similar, the new optimization is no longer equivalent to the original:

$$\min_{\theta_\lambda} \max_{\mathbf{y}} \Psi(\mathbf{x}, \mathbf{y}, \theta) + \Psi_\lambda(\mathbf{x}, \mathbf{y}, \theta_\lambda) g(\mathbf{y}, \mathcal{L}) \tag{3.3}$$

While a step in the right direction, the objective still requires combinatorial search because (1) the maximization involves two non-linear functions (2) the constraints might be global. In contrast, the functions involved in classic Lagrangian relaxation methods for NLP have multipliers for each output variable that can be combined with linear models to form a single unified decoding problem for which efficient inference exists (Koo et al., 2010; Rush et al., 2010; Rush and Collins, 2012). In the next subsection, we propose a novel approach that utilizes the amount of constraint violation as part of the objective function so that we can adjust the model parameters to search for a constraint-satisfying output efficiently.

3.1.2 Algorithm

To avoid combinatorial search problem, we modify the optimization problem one more time by (1) removing the compatibility function that involves the original weights θ and compensate with a regularizer that attempts to keep the dual weights θ_λ as close to these weights as possible, and (2) maximizing exclusively over the network parameterized by θ_λ while ignoring the constraint term during the maximization. This results in the following optimization problem:

$$\begin{aligned} \min_{\theta_\lambda} \quad & \Psi(\mathbf{x}, \hat{\mathbf{y}}, \theta_\lambda) g(\hat{\mathbf{y}}, \mathcal{L}) + \alpha \|\theta - \theta_\lambda\|_2 \\ \text{where} \quad & \hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{x}, \mathbf{y}, \theta_\lambda) \end{aligned} \tag{3.4}$$

The Lagrangian formulations in Eq. (3.2), (3.3) requires combinatorial search that considers both compatibility function and constraint information on the maximization step, and the minimization step only requires constraint information. In contrast, if we solve Eq. 3.4 in alternating maximization, minimization steps, we can take an unconstrained maximization step to obtain $\hat{\mathbf{y}}$. The burden of reflecting both constraint information and original parameter has been pushed to minimization step which is much cheaper than the combinatorial search.

Although this objective deviates from the original optimization problem, it is reasonable because by definition of the constraint function $g(\cdot)$, the global minima must correspond to outputs that satisfy all constraints. Further, we expect to find high-probability optima if we initialize $\theta_\lambda = \theta$. Moreover, the objective is intuitive: if there is a constraint violation in $\hat{\mathbf{y}}$ then $g(\cdot) > 0$ and the gradient will lower the compatibility of $\hat{\mathbf{y}}$ to make it less likely. Otherwise, $g(\cdot) = 0$ and the gradient of the energy is zero and we leave the compatibility of $\hat{\mathbf{y}}$ unchanged. We name this $\Psi(\mathbf{x}, \hat{\mathbf{y}}, \theta_\lambda) g(\hat{\mathbf{y}}, \mathcal{L})$ as *constraint loss*. Crucially, the optimization problem yields computationally efficient subroutines that we exploit in the optimization algorithm.

Algorithm 1 Gradient-Based Inference (GBI) for neural nets

Inputs: test instance $\mathbf{x}$, input specific CFL $\mathcal{L}^{\mathbf{x}}$, max-iteration M, pretrained weights θ
$\theta_\lambda \leftarrow \theta$ #reset instance-specific weights
while $g(\mathbf{y}, \mathcal{L}^{\mathbf{x}}) > 0$ and iteration$< M$ **do**
  $\mathbf{y} \leftarrow f(\mathbf{x}; \theta_\lambda)$ #perform inference using weights θ_λ
  $\nabla \leftarrow \ g(\mathbf{y}, \mathcal{L}^{\mathbf{x}}) \frac{\partial}{\partial \theta_\lambda} \Psi(\mathbf{x}, \mathbf{y}, \theta_\lambda) + \alpha \frac{\theta - \theta_\lambda}{\|\theta - \theta_\lambda\|_2}$ #compute constraint loss
  $\theta_\lambda \leftarrow \theta_\lambda - \eta \nabla$ #update instance-specific weights with SGD or a variant thereof
end while

To optimize the objective, the *Gradient-Based Inference (GBI)* algorithm alternates maximization to find $\hat{\mathbf{y}}$ and minimization w.r.t. θ_λ (Algorithm 1). In particular, we first approximate the maximization step by employing the neural network's inference procedure (e.g., greedy decoding, beam-search, or Viterbi decoding) to find the $\hat{\mathbf{y}}$ that approximately maximizes Ψ, which ignores the constraint function g. Then, given a fixed $\hat{\mathbf{y}}$, we minimize the objective with respect to the θ_λ by performing stochastic gradient descent (SGD). In the alternating minimization step, by fixing $\hat{\mathbf{y}}$, we regard constraint function term as a constant in the gradient; thus, making it easier to employ external black-box constraint evaluation functions (such as those based on compilers) that may not be differentiable. As a remark, note the similarity to REINFORCE (Williams, 1992): the

decoder outputs as actions and the constraint loss as a negative reward. However, GBI does not try to reduce *expected* reward and terminates upon discovery of an output that satisfies all constraints. We also introduce max-iteration M, in addition to the condition of satisfying constraints, to avoid pathological cases as GBI could run indefinitely without finding constraint-satisfying output.

As another remark, note the similarity to the domain adaptation approach with a regularizer (Chelba and Acero, 2006) as our approach tries to anchor to original model weight while optimizing constraint loss on a new instance. We set the weight for L2 regularizer λ to 0.005 throughout our experiments. Usually, inference step does not update model parameter as there is no objective to update it. In this Gradient-Based Inference (GBI), constraint loss acts a guiding signal to further optimize the model parameter soley geared to the test instance $\mathbf{x}$. This is another reason that GBI resets as it encounters new test instance.

3.2 Applications

There are multiple applications that involve hard-constraints and we provide two illustrative examples that we later employ as case-studies in our experiments: SRL and syntactic parsing. The former exemplifies a case in which external knowledge encoded as hard constraints conveys beneficial side information to the original task of interest while the latter studies a case in which hard constraints are inherent to the task of interest. Finally, we briefly mention sequence transduction as framework in which constraints may arise. Of course, constraints may in general arise for a variety of different reasons, depending on the situation. We provide example-based case studies for each application.

3.2.1 Semantic Role Labeling

As a first illustrative example, consider SRL. SRL focuses on identifying shallow semantic information about phrases. For example, in the sentence "John broke the window with a hammer" the goal is to tag the three-argument verb "broke" as the predicate, "John" as its first argument, the noun phrase "the window" as its second argument, and the prepositional phrase"with a hammer" as the last argument. It is traditional to address SRL as a sequence labeling problem, in which the input is the sequence of tokens and the output are BIO-encoded class labels. For argument span with semantic role ARG$_i$, the BIO notation of B-ARG$_i$ tag indicates that the corresponding token marks the beginning of the argument span, the I-ARG$_i$ tag indicates that the corresponding token is inside of the argument span, and the O tag indicates that token is outside of all argument spans. Thus, BIO represents both the regimentation of tokens into contiguous segments and the semantic role of the those segments.

Note that the parse tree for the sentence might provide constraints that could assist with the SRL task. In particular, each node of the parse tree represents a contiguous segment of tokens that could be a candidate for a semantic role, and no other contiguous segment can become a semantic role (Punyakanok et al., 2008). Therefore, we can include as side-information constraints that force the BIO-encoded class labeling to produce segments of text that each agree with some segment of text expressed by a node in the parse tree.

Formally defining this constraint $\mathcal{L}^x$, for input sentence $\mathbf{x} = x_1, \ldots, x_n$, let the SRL model outputs k spans $s_1, \ldots, s_k$ with corresponding length $n_1, \ldots, n_k$ where n_i sums up to n. Further, let *parse-spans*$(\mathbf{x})$ denotes the set of all parse constituents. Now, we can define per-span constraint function $g_i(s_i, \mathcal{L}^x) = 0$,

$$g_i(s_i, \mathcal{L}^x) = \begin{cases} 0 & \text{if } s_i \in \textit{parse-spans,} \\ \frac{1}{n_i} & \text{otherwise.} \end{cases} \tag{3.5}$$

Now we further define that our objective function $\Psi(\mathbf{x}, \hat{\mathbf{y}}, \theta_\lambda) g(\hat{\mathbf{y}}, \mathcal{L}^x)$ in (3.4) for SRL case to denote

$$\Psi(\mathbf{x}, \hat{\mathbf{y}}, \theta_\lambda) g(\hat{\mathbf{y}}, \mathcal{L}^x) = \sum_{i=1}^{k} g(s_i, \mathcal{L}^x) \Psi(\mathbf{x}, s_i, W_\lambda). \tag{3.6}$$

To continue with our example, the original SRL sequence-labeling might incorrectly label "the window with" as the second argument rather than "the window". Since according to the parse tree "with" is part of the next prepositional phrase, thus while the tree contains the spans "the window with a haamer" and "the window" it does not contain the span "the window with." The hope is that enforcing the BIO labeling to agree with the actual parse spans would benefit SRL. We notice that this is indeed the case, and present a real data-case from our experiments later in section 3.2.6.

While the previous work (Punyakanok et al., 2008) looked into the importance of syntactic parse tree in SRL, the constraint of aligning SRL labeling spans with actual parse spans has not been studied before. The traditional architecture for SRL assumed parse-spans as an input already and only considered span set within the parse-spans to be labeled with some role. As the parse-span information was already utilized as an input, the traditional systems had no need to consider the suggested alignment constraint at all. However, recent neural models use end-to-end model meaning that there is only relationship learning between input *sentence* and output *spans* and these end-to-end models do violate the alignment constraint violation. Thus, this work suggests additional way of using syntactic parse tree as a constraint on the end-to-end model on top of traditional constraints suggested by Punyakanok et al. (2008)[2].

3.2.2 Syntactic parsing

As a second illustrative example, consider the structured prediction problem of syntactic parsing, in which the goal is to input a sentence comprising a sequence of tokens and output a tree describing the grammatical parse of the sentence. Syntactic parsing is a separate but complementary task to SRL. While SRL focuses on semantic information, syntactic parsing focuses on identifying relatively deep syntax tree structures. One way to model the problem with neural networks is to linearize the representation of the parse tree and then employ the familiar sequence-to-sequence (seq2seq) model (Vinyals et al., 2015). Let us suppose we linearize the tree using a sequence of shift (`s`) and reduce (`r,!`) commands that control an implicit shift reduce parser. Intuitively,

[2]Similar motivation was presented by Swayamdipta et al. (2018) at the same conference where the work in this thesis was presented. However, they do not explicitly enforce parse-span information on SRL spans. The work of Swayamdipta et al. (2018) is discussed in section 2.3.

these commands describe the exact instructions for converting the input sentence into a complete parse tree: the interpretation of the symbol `s` is that we shift an input token onto the stack, the interpretation of the symbol `r` is that we start (or continue) reducing (popping) the top elements of the stack, and the interpretation of a third symbol `!` is that we stop reducing and push the reduced result back onto the stack. Thus, given an input sentence and an output sequence of shift-reduce commands, we can deterministically recover the tree by simulating a shift reduce parser. For example, the sequence `ssrr!ssr!rr!rr!` encodes a linearized version of the sequence `(S (NP The ball) (VP is (NP red)))` for the input sentence "The ball is red" where full tree is shown in Figure 3.1. It is easy to recover the tree structure from the input sentence and the output commands by simulating the shift reduce parser. Of course in practice, reduce commands include the standard parts of speech as types (NP, VP, etc).

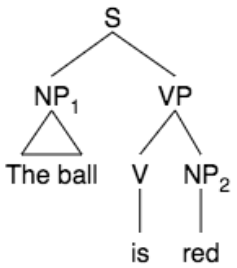


Figure 3.1: The syntactic constituency parse tree for our example "The ball is red".

Note that for output sequences to form a valid tree over the input, the sequence must satisfy a number of constraints. First, the number of shifts must equal the number of input tokens, otherwise either the tree would not cover the entire input sentence or the tree would contain spurious terminal symbols. Second, the parser cannot issue a reduce command if there are no items left on the stack. Third, the number of reduces must be sufficient to leave just a single item, the root node, on the stack. The constraint function $g(\mathbf{y}, \mathcal{L}^x)$ for this task simply counts the errors of each of the three types.

To formally define $g(\mathbf{y}, \mathcal{L}^x)$, let $m_{\mathbf{x}}$, n be the number of input and output tokens, respectively, $\text{ct}_{i=1}^{n}(b(i))$ be the function that counts the number of times proposition $b(i)$ is true for $i = 1, \ldots, n$. Now, we define the following constraint function

$$g(\mathbf{y}, \mathcal{L}^x) = \frac{1}{m_{\mathbf{x}} + n} \left\{ |m_{\mathbf{x}} - \text{ct}_{i=0}^{n}(y_i = s)| + \sum_{i}^{n} \max\left(0, \text{ct}_{j=0}^{i}(y_j = r) - \text{ct}_{j=0}^{i}(y_j \in \{s, !\})\right) \right\}.$$

Parsing this constraint function, the first term provides loss when the number or shifts is not equal to the number of input tokens. The second term provides loss when attempting to reduce an empty stack. The last term provides loss when the number of reduces is not sufficient to attach every lexical item to the tree.

As a minor remark, note that other encodings of trees, such as bracketing (of which the Penn Tree Bank's S-expressions are an example), are more commonly used as output representations for seq2seq parsing (*ibid.*). However, the shift-reduce representation described in the foregoing

paragraphs is isomorphic to the bracketing representations and as we get similar model performance to single seq2seq model (Vinyals et al., 2015) on the same data, we chose the former representation to facilitate constraint analysis. Although output representations sometimes matter, for example, BIO vs BILOU encoding of sequence labelings, the difference is usually minor (Ratinov and Roth, 2009) and breakthroughs in sequence labeling have been perennially advanced under both representations. Thus, for now, we embrace the shift reduce representation as a legitimate alternative to bracketing, *pari passu*.

3.2.3 Synthetic sequence transduction

Finally, although not a specific application per se, we also consider sequence transduction as it provides a framework conducive to studying synthetic tasks with appropriately designed properties. A sequence transducer $T : \mathcal{L}_S \rightarrow \mathcal{L}_T$ is a function from a source sequence to a target sequence. As done in previous work, we consider a known T to generate input/output training examples and train a seq2seq network to learn T on that data (Grefenstette et al., 2015). In our case, we can choose T, $\mathcal{L}_S$ and $\mathcal{L}_T$ to ensure that there are hard constraints on the output, viz. the output must belong to $\mathcal{L}_T$ and also respect any problem-specific constraints that may arise from the application of T on the input sentence.

For our task, we choose a simple transducer, similar to those studied in recent work (Grefenstette et al., 2015). The source language $\mathcal{L}_S$ is `(az|bz)`* and the target language $\mathcal{L}_T$ is `(aaa|zb)`*. The transducer is defined to map occurrences of `az` in the source string to `aaa` in the target string, and occurrences of `bz` in the source string to `zb` in the target string. For example, $T($`bzazbz`$) \mapsto$ `zbaaazb`.

Note that the number of `a`'s in the target sequence is a multiple of three for the given task. This informally defines the constraint rule (or grammar) $\mathcal{L}^x$. If the network is unable to learn such constraints $\mathcal{L}^x$ from the training data, then our method GBI is applicable and we can evaluate its performance for such simple cases. To express this constraint, we define following constraint evaluation function g, a length-normalized quadratic

$$g(\mathbf{y}, \mathcal{L}^x) = (3x_a - y_a)^2/(m+n) \tag{3.7}$$

that is zero when the number of `a`'s in the output (y_a) is exactly three times the number in the input (x_a) with m,n denoting input length, output length, respectively.

In this section, we study our algorithm on three different tasks: SRL, syntactic constituency parsing, and a synthetic sequence transduction task. As introduced in the Chapter3.2, all three tasks consider hard constraints, but they play a different role in each. In the sequence transduction task they force the output to belong to a particular input-dependent regular expression; in SRL, constraints provide side-information about possible true-spans; and in syntactic parsing, constraints ensure that the outputs encode valid trees. While the SRL task involves more traditional recurrent neural networks that have exactly one output per input token, the parsing and transduction tasks provide an opportunity to study the algorithm on various seq2seq networks .

3.2.4 Research questions and metrics for experiments.

We are interested in answering the following questions through experiments.

- Q1. How well does the neural network learn the constraints from the data alone?
- Q2. Can GBI enforce the constraints for cases in which the network is unable to learn the constraints perfectly?
- Q3. Does GBI enforce constraints without compromising the quality of the network's output?

To more thoroughly investigate Q2 and Q3, we also consider:

- Q4. Is the behavior of the method sensitive to the reference network performance?
- Q5. Does GBI also work on an out-of-domain dataset?

Q3 is particularly important because we adjust the weights of the network at test-time and this may lead to unexpected behavior. Q5 deals with our original motivation of using structured prediction to enhance performance on the out-of-domain data.

To address these various questions, we first define some terminology to measure how well the model is doing in terms of constraints. To address Q1, we measure the *failure-rate* (i.e., the ratio of test sentences for which the network infers an output that fails to fully satisfy the constraints). To address Q2, we evaluate our method on the *failure set* (i.e., the set of output sentences for which the original network produces constraint-violating outputs) and measure our method's *conversion rate*; that is, the percentage of failures for which our method is able to completely satisfy the constraints (or "convert"). Finally, to address Q3, we evaluate the quality (e.g., accuracy or F1) of the output predictions on the network's *failure set* both before and after applying our method.

3.2.5 Toy Transduction Experiment

As a first experiment on this chapter, we experiment GBI on a toy transduction task in order to examine whether GBI can work with artificial constraint. We focus on a simple sequence transduction task in which we find that despite learning the training data perfectly, a seq2seq model fails to learn the constraint in a way that generalizes to the test set, while GBI can generalize well to the test set.

As described in section 3.2, we choose a simple transducer where the transducer is defined to map occurrences of `az` in the source string to `aaa` in the target string, and occurrences of `bz` in the source string to `zb` in the target string. comprises 1934 sequences of length 2–20 and the test set contain sentences of lengths 21-24. We employ shorter sentences for training and test for generalization to longer sentences at test time.

We employ a thirty-two hidden unit single-layered, attention-less, seq2seq LSTM in which the decoder LSTM inputs the final encoder state at each decoder time-step. We train the network for 1000 epochs using RMSProp to maximize the likelihood of the output (decoder) sequences in the training set. The network achieves perfect train accuracy while learning the rules of the target grammar $\mathcal{L}_T$ perfectly, even on the test set. However, the network fails to learn the input-specific constraint that the number of `a`s in the output should be three times the number of `a`s in the input. This illustrates how a network might rote-memorize constraints rather than learn the rule in a way that generalizes. Thus, enforcing constraints at test-time is important. To satisfy constraints, we employ GBI with a constraint loss g, a length-normalized quadratic $(3x_a - y_a)^2/(m+n)$ that is zero when the number of `a`s in the output (y_a) is exactly three times the number in the input (x_a) with m,n denoting the length of input, output, respectively. GBI achieves a conversion rate of

65.2% after 100 iterations, while also improving the accuracy on the failure set from 75.2% to 82.4%. This synthetic experiment provides additional evidence in support of Q2 and Q3, on a simpler small-capacity network.

Additionally, we present case study in Table 3.1 where GBI demonstrates fixing the constraint error for transducer given a real example. As can be seen in the example, as the probability of the error sequence becomes less, other sequences get higher probability and in the end the constraint-satisfying output was decoded.

azazbzazbzbzazbzbzbzbzbz ⟶ aaaaaazbaaazbzbaaazbzbzbzbzb

iteration	output	loss	accuracy
0	aaaaaazbaaazbaaazbzbzbzbaaazb	0.2472	66.7
1	aaaaaazbaaazbaaazbzbzbzbaaazb	0.2467	66.7
2	aaaaaazbaaazbaaazbzbzbzbaaazb	0.2462	66.7
3	aaaaaazbaaazbzbaaazbzbzbzbzb	0.0	100.0

Table 3.1: A sequence transduction example for which enforcing the constraints improves accuracy. The loss column indicates $\Psi(\mathbf{x}, \hat{\mathbf{y}}, \theta_\lambda) g(\hat{\mathbf{y}}, \mathcal{L}^x)$ for our toy transduction problem. Red indicates errors.

3.2.6 Semantic Role Labeling

We employ the AllenNLP (Gardner et al., 2017) SRL network with ELMo embeddings, which is essentially a multi-layer highway bi-LSTM that produces BIO output predictions for each input token (Peters et al., 2018b). For data we use OntoNotes v5.0, which has ground-truth for both SRL and syntactic parsing (Pradhan et al., 2013). We evaluate GBI on the test set (25.6k examples from train/dev/test split of 278k/38.3k/25.6k), for which consistent parse information is available for 81.25% examples (we only include side-information in terms of constraints for this subset).

In order to study our algorithm on a wide range of accuracy regimes (section 3.2.8), we train many networks SRL-X, with varying amounts of training dataset where X denotes a % of dataset used for training. We optimize our network for at most 10 iterations at test time applying SGD with step size $\eta = 0.01$. From Table 3.2, we see that GBI is able to convert 42.25 % of failure set, and this boosts the overall F1 measure by 1.23 point over the state-of-the-art network (SRL-100)[3].

To address Q1 we measure the sentence-level *failure rate* as well as span-level *disagreement rate* (i.e., the ratio of predicted spans in a sentence that disagree with the spans implied by the true syntactic parse of the sentence). To address Q2 we evaluate our method on the *failure set* (i.e., the set of sentences for which disagreement rate is nonzero) and measure our method's average disagreement rate. Finally, to address Q3, we evaluate the quality (F1 and exact match) of the output predictions on the network's *failure set* both before and after applying our method. From Table 3.2, we can see that by applying GBI on SRL-100, the average disagreement rate on the

[3] The state-of-the-art in SRL (Peters et al., 2018b) performed 84.6 F1 at the time of submitting Lee et al. (2019) which this chapter is based on. For a fair comparison, we obtain a similar 84.4 F1 baseline with their provided network architecture(Gardner et al., 2017), and achieve 85.63 after enforcing constraints with our inference.

Network	Inference	**Whole test set**			
		Failure rate(%)	Conversion rate(%)	F1	
				before	after
SRL-100	GBI	9.82	42.25	84.40	85.63 (+1.23)
	A*		40.40		84.51 (+0.11)
SRL-70	GBI	10.54	46.22	83.55	84.83 (+1.28)
	A*		44.42		83.90 (+0.35)
SRL-40	GBI	11.06	47.89	82.57	84.03 (+1.46)
	A*		44.74		82.98 (+0.41)
SRL-10	GBI	14.15	44.28	78.56	80.18 (+1.62)
	A*		43.66		78.87 (+0.31)
SRL-1	GBI	21.90	52.85	67.28	69.97 (+2.69)
	A*		48.96		67.97 (+0.69)

Network	Inference	**Failure set**					
		Average (%) Disagreement		Exact Match (%)		F1	
		before	after	before	after	before	after
SRL-100	GBI	44.85	24.92	0.0	19.90	48.00	59.70 (+11.7)
	A*		33.91		13.79		48.83 (+0.83)
SRL-70	GBI	45.54	23.02	0.0	19.57	47.81	59.37 (+11.56)
	A*		32.32		16.12		50.49 (+2.68)
SRL-40	GBI	45.71	22.42	0.0	19.45	46.53	58.83 (+12.3)
	A*		32.17		15.15		46.53 (+2.88)
SRL-10	GBI	47.14	24.88	0.0	15.28	44.19	54.78 (+10.59)
	A*		32.80		12.28		45.93 (+1.74)
SRL-1	GBI	50.38	21.45	0.0	12.83	37.90	49.00 (+11.10)
	A*		30.28		11.25		41.59 (+3.69)

Table 3.2: Comparison of the GBI vs. A* inference procedure for SRL. We first report the F1-score and failure rate (%) over whole test set in the table above. Then, focusing on the *failure set*, we report more detailed analysis with average disagreement rate, exact match, and F1-scores and exact match in table below. Also, we report performances on a wide range of reference models SRL-X, where X denotes % of dataset used for training. We employ Viterbi decoding as a base inference strategy (before) and apply GBI (after) in combination with Viterbi.

name	F1		hyper-parameters			data(%)
	Beam-search (beam size: 9)	greedy	hidden	layers	dropout	
Net1	87.58	87.31	128	3	0.5	100%
Net2	86.63	86.54	128	3	0.2	100%
Net3	81.26	78.32	172	3	no	100%
Net4	78.14	74.53	128	3	no	75%
Net5	71.54	67.80	128	3	no	25%

Table 3.4: Parsing Networks with various performances. Net1,2 are GNMT seq2seq models whereas Net3-5 are trained on lower-resource and simpler seq2seq models, providing a wide range of model performances on which to test GBI.

failure set goes down from 44.85% to 24.92% which results in an improvement of 11.7 F1 and 19.90% in terms of exact match on the same set. These improvements answer Q1-3 favorably.

⟨“ it is really like this , just look at the bus signs . ”⟩
Gold SRL: B-ARG1 B-V B-ARGM-ADV B-ARG2 I-ARG2 O O ··· O

iteration	output	loss	accuracy
0	B-ARG1 B-V B-ARG2 I-ARG2 I-ARG2 O O O O O O O O	0.32	50.0%
6	B-ARG1 B-V B-ARG2 I-ARG2 I-ARG2 O O O O O O O O	0.29	50.0%
7	B-ARG1 B-V B-ARGM-ADV B-ARG2 I-ARG2 O O O O O O O O	0.00	100.0%

Table 3.3: A semantic role labeling example for which the method successfully enforces syntactic constraints with GBI. The initial output has an inconsistent span (which are marked red) for token "really like this". Enforcing the constraint not only corrects the number of agreeing spans, but also changes the semantic role "B-ARG2" to "B-ARGM-ADV" and "I-ARG2" to "B-ARG2".

To enforce constraints during inference, He et al. proposed to employ constrained-A* decoding. For the sake of a fair comparison with GBI, we consider A* decoding as used in He et al. (2017)[4] and report results for SRL-X networks. We see from Table 3.2, that the GBI procedure consistently out-performs A* decoding on all evaluation metrics, thus, demonstrating superiority of our approach. Additionally, we present case study in Table 3.2.6 where GBI demonstrates fixing the constraint error given a real example. In subsection 3.3.2, we conduct an ablation study on GBI by introducing noisy constraints where GBI shows its superiority compared to A* again.

3.2.7 Syntactic parsing

We now turn to a different task and network: syntactic constituency parsing. We investigate the behavior of the constraint inference algorithm on the shift-reduce parsing task described in

[4]Our A* experiment show very similar tendency with He et al. (2017). They report 84.6 F1 baseline with product of experts and 84.8 F1 for constrained-A* decoding. We obtain 84.4 F1, 84.51 F1 for baseline and A*-decoding, respectively, using ELMo embedding.

Net	Failure (/2415)	Conversion rate	F1 (Failure set)		F1 (Whole test set)	
			before	after	before	after
Net1	187	93.58	71.49	77.04	87.31	87.93
Net2	287	89.20	73.54	79.68	86.54	87.57

Table 3.5: Evaluation of GBI on syntactic parsing using GNMT seq2seq. Note that GBI without beam search performs higher than beam search performance in Table 3.4.

Section 3.2.2. We transform the Wall Street Journal (WSJ) portion of the Penn Tree Bank (PTB) into shift-reduce commands in which each reduce command has a phrase-type (e.g., noun-phrase or verb-phrase) (Mitchell et al., 1999). We employ the traditional split of the data with section 22 for dev, section 23 for test, and remaining sections 01-21 for training. We evaluate on the test set with evalb[5] F1. In each experiment, we learn a seq2seq network on a training set and then evaluate the network directly on the test set using a traditional inference algorithm to perform the decoding (either greedy decoding or beam-search).

We repeat our same experimental procedure as before, but for this task. Again, to test GBI on a wide range of accuracy regimes (section 3.2.8), we train many networks with different hyper-parameters producing models of various quality, from high to low, using the standard split of the WSJ portion of the PTB. In total, we train five networks `Net1-5` for this study, that we describe below. We train our two best baseline models (`Net`1,2) using a highly competitive seq2seq architecture for machine translation, GNMT (Wu et al., 2016)[6], with F1 scores, 86.78 and 87.33, respectively. To study a wider range of accuracies, we train a simpler architecture[7] with different hyper parameters and obtain nets (Net3-5). For all models, we employ Glorot initialization, and basic attention (Bahdanau et al., 2014). See Table 3.4 for a summary of the networks, hyper-parameters, and their performance.

We report the behavior of the constraint-satisfaction method in Table 3.5 for Net1-2, and in Table 3.6 for Net3-5. Across all the experimental conditions (Table 3.5, 3.6), the conversion rates are high, often above 80% and sometimes above 90% supporting Q2. Note that beam search alone can also increase constraint satisfaction with conversion rates reaching as high as 51.74% (164/317) in the case of Net3 with beam size 9. However, as the quality of the model increases, the conversion rate becomes minuscule; in the case of Net1,2 the conversion rate is less than 14% with beam 9; in Net1 converting 26 out of 187 and in Net2 converting just 1 out of 287 instances from failure set.

In order to address question Q3—the ability of our approach to satisfy constraints without negatively affecting output quality—we measure the F1 scores on the failure sets both before and after applying the constraint satisfaction algorithm. Since F1 measure is only defined on valid trees, we employ heuristic post-processing, as described earlier, to ensure all outputs are valid.

Note that an improvement on the failure set guarantees an improvement on the entire test set since our method produces the exact same outputs as the baseline for examples that do not initially violate any constraints. Consequently, for example, the GNMT network improves (Net2) on the

[5] `http://nlp.cs.nyu.edu/evalb/`

[6] An encoder with a bidirectional first layer, unidirectional second and third layer and a decoder with unidirectional layers.

[7] Uni-directional encoder/decoder.

Net	Infer method	Failure (/2415)	Conv rate	F1 (Failure set)	
				before	after
Net3	Greedy	317	79.81	65.62	68.79 (+3.14)
	Beam 2	206	87.38	66.61	71.15 (+4.54)
	Beam 5	160	87.50	67.5	71.38 (+3.88)
	Beam 9	153	91.50	68.66	71.69 (+3.03)
Net4	Greedy	611	88.05	62.17	64.49 (+2.32)
	Beam 2	419	94.27	65.40	66.65 (+1.25)
	Beam 5	368	92.66	67.18	69.4 (+2.22)
	Beam 9	360	93.89	67.83	70.64 (+2.81)
Net5	Greedy	886	69.86	58.47	60.41 (+1.94)
	Beam 2	602	82.89	60.45	61.35 (+0.90)
	Beam 5	546	81.50	61.43	63.25 (+1.82)
	Beam 9	552	80.62	61.64	62.98 (+1.34)

Table 3.6: Evaluation of GBI on simpler seq2seq networks Net4, 5 are trained on a lower resource (75% and 25%, respectively). Here, we also evaluate whether GBI can be used in combination with different inference techniques: greedy and beam search of various widths.

failure set from 73.54 to 79.68 F1, resulting in an overall improvement from 86.54 to 87.57 F1 (on the entire test set). These improvements are similar to those we observe in the SRL task, and provide additional evidence for answering Q1-3 favorably.

We also measure how many iterations of our algorithm it takes to convert the examples that have constraint-violations. Across all conditions, it takes 5–7 steps to convert 25% of the outputs, 6–20 steps to convert 50%, 15–57 steps to convert 80%, and 55–84 steps to convert 95%.

In Table 3.7 we provide an example data-case that shows how our algorithm solves the initially violated shift-reduce parse output. For simplicity we omit the phrase-types and display only on the shift (`s`), reduce (`t`) and stop reducing commands (`!`), and color them red if there is an error. The algorithm satisfies the constraint in just 12 iterations, and this results in a perfectly correct parse. What is interesting about this example is that the original network commits a parsing mistake early in the output sequence. This type of error is problematic for a naive decoder that greedily enforces constraints at each time-step. The reason is that the early mistake does not create a constraint violation until it is too late, at which point errors have already propagated to future time-steps and the greedy decoder must shift and reduce the last token into the current tree, creating additional spurious parse structures. In contrast, our method treats the constraints holistically, and uses it to correct the error made at the beginning of the parse. See the second table in Table 3.7 for a comparison.

3.2.8 GBI on wide range of reference models

The foregoing experimental results provide evidence that GBI is a viable method for enforcing constraints. However, we hitherto studied GBI on high quality baseline networks such as SRL-100. To further bolster our conclusions, we now direct our investigation towards lower quality networks, to understand GBI's viability under a broader quality spectrum. We ask, how sensitive is GBI

⟨“ So it ’s a very mixed bag . ”⟩ ⟶ sssr!ssssrr!srrr!rr!ssrrrrrr!

iteration	output	loss	accuracy
0	ssr!sr!ssssrrr!rr!ssrrrrrr!	0.0857	33.3%
11	ssr!sr!ssssrrr!rr!ssrrrrrr!	0.0855	33.3%
12	sssr!ssssrr!srrr!rr!ssrrrrrr!	0.0000	100.0%

inference method	output
unconstrained-decoder	ssr!sr!ssssrrr! rr!ssrrrrrr!
constrained-decoder	ssr!sr!ssssrrr! rr!ssrrrrrr!srr!
our method	sssr!ssssrr!srrr!rr!ssrrrrrr!
true parse	sssr!ssssrr!srrr!rr!ssrrrrrr!

Table 3.7: A shift-reduce example for which the method successfully enforces constraints with GBI. The first table illustrates the steps that GBI takes and the second table compares unconstrained decoder, constrained decoder, and true output. The initial output has only nine shifts, but there are ten tokens in the input. Enforcing the constraint not only corrects the number of shifts to ten, but changes the implied tree structure to the correct tree. For a little more detail, observe on the 2nd table which compares the result with the constrained decoder which constrains the output only in myopic fashion. The initial unconstrained decoder prematurely reduces “So it” into a phrase, missing the contracted verb “is.” Errors then propagate through the sequence culminating in the final token missing from the tree (a constraint violation). The constrained decoder is only able to deal with this at the end of the sequence, while our method is able to harness the constraint to correct the early errors.

to the reference network’s performance (Q4)? To this end, we train poorer quality networks by restricting the amount of available training data or employing simpler architectures.

For *SRL*, we simulate low-resource models by limiting the training data portion to 1%, 10%, 40%, and 70% resulting in F1 score range of 67.28-83.55 (Table 3.2). Similarly, for *syntactic parsing*, we train additional low-quality models Net3-5 with a simpler uni-directional encoders/decoders, and on different training data portions of 25%, 75%, and 100% (Table 3.4). We evaluate GBI on each of them in Table 3.2, 3.6 and find further evidence in support of favorable answers to Q2 (satisfying constraints) and Q3 (improving F1 accuracy) by favorably answering Q4. Moreover, while omitting for brevity, we examined both tasks with over 30 experiments (Lee et al., 2017) with different baseline networks in combination with different inference strategies, and we found GBI favorable in all but one case (but by just 0.04 comparing without GBI).

We also study whether GBI is compatible with better underlying discrete search algorithms, in particular beam search for seq2seq. As we seen in column 2 of Table 3.6, that although beam-search improves the F1 score and reduces the percentage of violating constraints, GBI further improves over beam-search when using the latter in the inner-loop as the decoding procedure. In conclusion, improving the underlying inference procedure has the effect of decreasing the number of violating outputs, but GBI is still very much effective on this increasingly small set, despite it intuitively representing more difficult cases that even eludes constraint satisfaction with beam search.

Genre / Task	Failure rate (%)	Conversion rate (%)	F1 on failure set: before before	F1 on failure set: after after	F1 on whole test set: before before	F1 on whole test set: after after
	SRL					
Source genre (NW)	18.10		-		77.45	
Broadcast Conversation (BC)	26.86	53.88	39.72	52.40 (+12.68)	71.2	73.63 (+3.29)
Broadcast News (BN)	18.51	55.19	39.28	50.58 (+11.3)	73.22	75.64 (+2.42)
Pivot Corpus (PT)	10.01	62.34	47.19	63.69 (+16.5)	83.95	85.34 (+1.39)
Telephone Conversation (TC)	19.09	54.62	47.7	58.04 (+10.34)	73.07	75.38 (+2.31)
Weblogs (WB)	20.32	44.13	47.6	57.39 (+9.39)	77.10	79.04 (+1.94)
Genre / Task	***Syntactic Parsing***					
Source genre (NW/WSJ)	13.90		-		84.41	
Broadcast Conversation (BC)	19.32	99.83	56.38	58.79 (+2.41)	66.93	67.68 (+0.75)
Broadcast News (BN)	11.67	99.70	63.17	67.44 (+4.27)	78.77	79.66 (+0.89)
Pivot Corpus (PT)	9.87	98.85	71.42	76.22 (+4.80)	85.99	86.66 (+0.67)
Telephone Conversation (TC)	10.15	93.33	56.86	58.00 (+1.14)	65.99	66.29 (+0.30)
Weblogs (WB)	17.62	98.58	62.03	62.73 (+0.70)	74.55	74.76 (+0.21)

Table 3.8: Evaluation of GBI method on out-of-domain data on of SRL and syntactic parsing. F1 scores are reported on the *failure set*. SRL model was trained on NW and the syntactic parser was trained on WSJ set which is a subsection of NW on OntoNote v5.0. The table shows that GBI can be successfully applied to reduce performance degradation on out-of-domain data. In average, +12.2 F1 point increased for SRL and +2.7 F1 point for parsing on failure set. Furthermore, in both cases, the gap between source and out-of-domain test results reduces.

3.3 Further analysis

In this section, we want to further analyze GBI in additional aspects. To do so, we run experiments on out-of-domain dataset and analyze the run times of each experiment.

3.3.1 Experiments on out-of-domain data

In the previous subsection we saw how GBI performs well even when the underlying network is of lower quality. We now investigate GBI on out-of-domain data.For SRL, we train a network with ELMo embedding on the NewsWire (NW) section of the OntoNotes v5.0 English PropBank corpus and then test on the other genres provided in the corpus: BC, BN, PT, TC, WB. The failure rate on the within genre data (test set of NW) is 18.10%. We can see from Table 3.8, the failure rate for the NW-trained SRL network (SRL-NW) is higher for most out-of-genre data with the highest being 26.86% for BC (vs. 18.10% NW). Further, by enforcing constraints, we see significant gains in terms of F1 score across all genres (ranging from 9.39-16.5 F1 on Failure set and 1.39-3.29 on whole test set). We observe that the gap between source (NW) section and other sections decreases with the WB genre surpassing (77.10 to 79.04 F1) source test result (77.45 F1). The overall performance difference of source and out-of-domain reduces by 1.1 F1 score, providing additional evidence for answering Q5 favorably.

Similar to SRL, we train a GMNT seq2seq model on the Wall Street Journal of NW section in OntoNotes v5.0 Treebank[8] which shares the same genre classification with PropBank. Following the same experimental protocol as on the PTB data, we report the results in Table 3.8. The F1 on the within-genre data (test set of WSJ) is 84.41, but the F1 on out-of-domain genres are much lower, ranging from 66.93-78.77 F1 score except PT which has 85.99. We observe in some cases, like WB and BC, the failure rate is much higher (17.62%, 19.32% for WB *vs.* 13.90% for WSJ). We see that across all genres, the algorithm has high conversion rates (sometimes close to 100%), and that in each case, enforcing the constraints improves the F1. Similar to SRL case, the overall performance difference of source and out-of-domain reduces (by 0.54 F1 score). Again, we find support for Q2, Q3 and Q5.

3.3.2 Robustness of GBI

We perform additional experiments to analyze the robustness and runtime of GBI. First, to measure robustness, we consider a variant of the SRL task in which we include noisy constraints. OntoNotes v5.0 English PropBank has about 10% of the spans mismatched between the annotated SRL output (Propbank) and the parse tree (Treebank). Here, we define *noisy* scenario as using all the treebank as it is and define *noise-free* scenario as using filtered treebank[9].

Table 3.9 reports the performance of GBI and $\mathtt{A}^{\star}$in the presence of noisy constraints. We can see that the overall performance (F1-score) for $\mathtt{A}^{\star}$drops drastically (-6.92) in the presence of noisy constraints while we still see gains with GBI ($+0.47$). We further analyze the improvement

[8]The WSJ section of OntoNotes Tree bank is slightly different from that of PennTree bank WSJ section. While PennTree bank has 40k instances on WSJ section, OntoNote WSJ section has 30k instances.

[9]We simply assume we do not have syntactic parse tree information, setting $g(s_i, \mathcal{L}^x) = 0$, for the instances that has span-mismatch between propbank and treebank

Decoding	Precision	Recall	F1-score	Exact Match (%)
Viterbi	84.03	84.78	84.40	69.37
	Noisy constraints			
A*	78.13 (-5.90)	76.85 (-7.93)	77.48 (-6.92)	58.30 (-11.70)
GBI	85.51 (+1.48)	84.25 (-0.53)	**84.87** (**+0.47**)	68.45 (-0.92)
	Noise-free constraints			
A*	84.19 (+0.16)	84.83 (+0.05)	84.51 (+0.11)	70.52 (+1.15)
GBI	85.39 (+1.36)	85.88 (+1.10)	**85.63** (**+1.23**)	71.04 (+1.67)

Table 3.9: Comparison of different inference procedures: Viterbi, A*(He et al., 2017) and GBI with noisy and noise-free constraints. Note that the (+/-) F1 are reported w.r.t Viterbi decoding on the same column.

of GBI by looking at the precision and recall scores individually. We see that recall drops slightly for GBI which suggests that noisy constraints do inhibit predicting actual argument spans. On the other hand, we see that precision increases significantly. After analyzing predicted argument spans, we noticed that GBI prefers to predict no argument spans instead of incorrect spans in the presence of noisy constraints, which leads to an increase in precision. Thus, GBI provides flexibility in terms of strictness with enforcing constraints, which makes it robust to noisy constraints. On the other hand, the constrained-A*decoding algorithm is too strict when it comes to enforcing noisy constraints resulting in a significant drop of both precision and recall.

3.3.3 Runtime analysis

Network	Genre(s)	No. of examples	Failure rate (%)	Inference time (approx. mins)		
				Viterbi	GBI	A*
SRL-100	All	25.6k	9.82	109	288	377
SRL-NW	BC	4.9k	26.86	23	110	117
	BN	3.9K	18.51	18	64	100
	PT	2.8k	10.01	8	19	15
	TC	2.2k	19.01	5	23	20
	WB	2.3k	20.32	12	49	69

Table 3.10: Comparison of runtime for difference inference procedures in the noise-free constraint setting: Viterbi, A*(He et al., 2017) and GBI. For SRL-100 refer Table 3.2 and SRL-NW is a model trained on NW genre.

In terms of runtime, GBI is generally faster than A*, though, the difference is less clear on smaller evaluation sets. Table 3.10 reports the runtime for different inference procedures with varying dataset sizes. In general, we observe that GBI tends to be faster than A*, especially when the dataset is large enough. One exception is the BC domain where GBI is just slightly faster than A*. We hypothesize it might be due to the difficulty of the domain as its failure rate is higher than others. GBI will spend more time searching for the correct output (more iterations) if it is harder

to find the solution. Similarly, in the case study with noisy constraints, the runtimes are similar; however, GBI has much better accuracy, showing similar gains as the noise-free setting.

3.3.4 Discussion on max-iteration M

In using GBI, we are required to explicitly set the max-iteration M for GBI after which it will stop iterating to avoid pathological cases. In all our SRL experiments, we have set the M to be 10. To study its scalability, we ran GBI with max-iteration M set to 30. The runtime increases to 556 mins for $M = 30$ as opposed to 288 mins of $M = 10$. However, if we set $M = 30$, the performance of GBI improves significantly in all accuracy metrics compared to that of $M = 10$: overall F1 (+0.34), F1 on failure set (+3.4), exact match (+4.35%), and conversion rate (+11.24%). As can be seen from the demonstration, there is a clear tradeoff between runtime and accuracy as controlled by the maximum number of epochs M. The user can control M by the runtime constraint the system has: lower M when the serving time is most important and larger M when accuracy is more important than the serving time.

3.4 Related work

Recent work has considered applying neural networks to structured prediction; for example, structured prediction energy networks (SPENs) (Belanger and McCallum, 2016). SPENs incorporate soft-constraints via back-propagating an energy function into "relaxed" output variables. In contrast, we focus on hard-constraints and back-propagate into the weights that subsequently control the original non-relaxed output variables via inference. Separately, there has been interest in employing hard constraints to harness unlabeled data in training-time for simple classifications (Hu et al., 2016). Our work instead focuses enforcing constraints at inference-time. More specifically, for SRL, previous work for enforcing syntactic and SRL specific constraints have focused on constrained A* decoding (He et al., 2017) or integer linear programming (Punyakanok et al., 2008). For parsing, previous work in enforcing hard constraints has focused on post-processing (Vinyals et al., 2015) or building them into the decoder via sampling (Dyer et al., 2016) or search constraints (Wiseman and Rush, 2016). Additionally, there was a very recent work that tried to inject constraints using automaton (Deutsch et al., 2019) where automaton will reshape the probability distribution at each decoding step. While this can avoid producing constraint failure output, this approach is different with GBI in that automaton will not modify earlier decoded results as we have discussed in Table 3.7 comparing GBI to a constrained decoder.

Finally, as previously mentioned, our method highly resembles dual decomposition and more generally Lagrangian relaxation for structured prediction (Koo et al., 2010; Rush et al., 2010; Rush and Collins, 2012). In such techniques, it is assumed that a computationally efficient inference algorithm can maximize over a superset of the feasible region (this assumption parallels our case because unconstrained inference in the neural network is efficient, but might violate constraints). Then, the method employs gradient descent to concentrate this superset onto the feasible region. However, these techniques are not directly applicable to our non-linear problem with global constraints.

3.5 Conclusion

We presented an algorithm for satisfying constraints in neural networks that avoids combinatorial search, but employs the network's efficient unconstrained procedure as a black box to coax weights towards well-formed outputs. We evaluated the algorithm on three tasks including SRL and seq2seq parsing and found that GBI can successfully convert failure sets while also boosting the task performance. Accuracy in each of the three tasks was improved by respecting constraints.

Additionally, we extend the constraint-enforcing loss of GBI's to another form in Part II in order to inject constraints on training time. As we will see shortly in Chapter 4, for SRL, we can employ GBI on top of a model trained with similar loss as GBI's. We observe that the additional test-time optimization of GBI on top of constraint-injected model still significantly improves the model output whereas A*does not. We believe this is because GBI searches in the proximity of the provided model weights; however, theoretical analysis of this hypothesis is left as a future work.

Part II

Semi-Supervised Learning with Output Constraints

Neural networks have improved the state-of-the-art on many NLP applications over the years. However, they tend to show poor performance in low-resource and out-of-domain settings. In Chapter 3, we observed that injecting output constraints can be helpful in these settings. In Part II, we study whether the output constraints can aid the learning process, as it did for inference procedures in Part I.

Chapter 4 presents a semi-supervised learning framework that extends the test-time constraint injection to learning time: from per-instance optimization to learning generally on multiple-instances. While typical inference steps are performed on a fixed, learned model, in Chapter 3 we optimized the model parameter with a *constraint loss* for structured inference.

Noting that the *constraint loss* is unsupervised, as it does not require a labeled dataset to evaluate a constraint violation, Chapter 4 presents three loss functions: (1) structural constraint loss similar to that of Chapter 3, (2) a joint objective of structural loss and supervised loss on training set, and lastly (3) a joint objective on semi-supervised setting. Among these objectives, the semi-supervised setting displays the most improvement on task performance and largest reduction of constraint violation. This semi-supervised learning framework shows significant improvements in low-resource settings, and positive results on high-resource setting as well in the case of semantic role labeling.

Chapter 5 applies the semi-supervised framework introduced in Chapter 4 to a multi-view learning setting to inject an *agreement constraint*. This constraint is different to other constraints presented in this thesis, in that it is not based on a prior knowledge of a specific task, but rather can be applied in any multi-view learning setting. Hence, we believe the approaches to be introduced in Chapter 5 can be applied to other sequence-labeling tasks. First establishing recently proposed multi-view learner (META-BILSTM (Bohnet et al., 2018)) as a baseline, and then applying the semi-supervised learning framework on top of it, this thesis presents extensive experiments on low-resource settings. Additionally, Chapter 5 shows that it does not hurt to use *agreement constraint* together with a pre-trained language model and demonstrates state-of-the-art result for Chinese dependency parsing on the CoNLL2018 shared task.

Chapter 4

Semi-Supervised Learning with Syntactic Constraints

Several neural models have shown state-of-the-art performances on Semantic Role Labeling (SRL) (He et al., 2017; Peters et al., 2018c; Tan et al., 2018; He et al., 2018; Ouchi et al., 2018). However, the neural models require an immense amount of labeled data and are thus not well suited for low-resource languages or domains. This chapter proposes a semi-supervised semantic role labeling method that outperforms the state-of-the-art in limited SRL training data. The proposed method is based on explicitly enforcing syntactic constraints by applying *syntactic-inconsistency loss*, a loss function defined in a similar manner to the *constraint loss* in Chapter 3. Whereas *constraint loss* for GBI (Chapter 3) was applied to the single instance, the *syntactic-inconsistency loss* is defined on a batch of unlabeled instances. This unsupervised loss combined with supervised loss from training data forms a semi-supervised learning loss.

The proposed semi-supervised learning is examined on CoNLL-2012 English section. The experiment comprises of low-resource setting, which considers 1%, 10% of available SRL-labeled data, as well as high-resource setting, using 100% of available SRL-labeled data[1], with varying amounts of *parse trees* on SRL-unlabeled data. For example, the semi-supervised learning with injection of *gold parse tree* on 1% of available SRL-labeled data achieves +1.58 F1 improvements over the pre-trained models that were trained with a state-of-the-art architecture[2] on the same SRL-labeled data. Similar experiments are conducted with *system-generated parse trees* instead of *gold parse trees*. Additionally, we also study whether inference with constraint is still beneficial even after learning with constraints via semi-supervised learning. By applying GBI after semi-supervised learning with syntactic-inconsistency loss on 1% of available SRL-labeled data, we demonstrate +3.67 F1 (extra +2.09 F1 by GBI) improvement over pre-trained model on the same labeled data. The material presented in this chapter is based on Mehta et al. (2018)[3].

[1] We use *SRL-labeled*, *SRL-unlabeled* data to distinguish instances with and without gold SRL labels regardless of other label information such as parse tree annotation.

[2] Since our submission of Mehta et al. (2018), which this chapter is based on, the state-of-the-art model for SRL has been updated. The previous state-of-the-art Peters et al. (2018b) with ELMo, which we apply our technique on, has been advanced by 1.6 F1 points Ouchi et al. (2018) as mentioned in Chapter 3 well.

[3] For Mehta et al. (2018), Lee and Mehta are the co-first authors with equal contributions.

4.1 Overview

Semantic role labeling (SRL), a.k.a shallow semantic parsing, identifies the arguments corresponding to each clause or proposition, i.e. its semantic roles, based on lexical and positional information. SRL labels non-overlapping text spans corresponding to typical semantic roles such as Agent, Patient, Instrument, Beneficiary, etc. This task finds its use in many downstream applications such as question-answering (Shen and Lapata, 2007), information extraction (Bastianelli et al., 2013), machine translation, etc.

Several SRL systems relying on large annotated corpora have been proposed (Peters et al., 2018c; He et al., 2017), and perform relatively well. A more challenging task is to design an SRL method for low resource scenarios (e.g. rare languages or domains) where we have limited annotated data but where we may leverage annotated data from related tasks. Therefore, in this chapter, we focus on building effective systems for low resource scenarios and illustrate our system's performance by simulating low resource scenarios for English.

SRL systems for English are usually built using the large annotated corpora of verb predicates and their arguments provided as part of the PropBank and OntoNotes v5.0 projects (Kingsbury and Palmer, 2002; Pradhan et al., 2013). These corpora are built by adding semantic role annotations to the constituents of previously-annotated syntactic parse trees in the Penn Treebank (Marcus et al., 1993). Traditionally, SRL relies heavily on using syntactic parse trees either from shallow syntactic parsers (chunkers) or full syntactic parsers and Punyakanok et al. (2008) shows significant improvements by using syntactic parse trees.

Recent breakthroughs motivated by end-to-end deep learning techniques (Zhou and Xu, 2015; He et al., 2017) achieve state-of-the-art performance without leveraging any syntactic signals, relying instead on ample role-label annotations. We hypothesize that by leveraging syntactic structure while training neural SRL models, we may achieve robust performance, especially for low resource scenarios. Specifically, we propose to leverage syntactic parse trees as hard constraints for the SRL task i.e., we explicitly enforce that the predicted argument spans of the SRL network must agree with the spans associated by the syntactic parse of the sentence via scoring function in the training objective. Moreover, we present a semi-supervised learning (SSL) based formulation, wherein we leverage syntactic parse trees for SRL-unlabeled data to build effective SRL for low resource scenarios.

We build upon the SRL system provided by Peters et al. (2018c) (with ELMo) that follows the architecture of He et al. (2017) which formulates SRL as a BIO tagging problem and use multi-layer highway bi-directional LSTMs. However, we differ in terms of our training objective. In addition to the log-likelihood objective, we also include *syntactic inconsistency loss* (defined in Section 4.2.3) which quantifies the hard constraint (spans associated with syntactic parse) violations in our training objective. In other words, while training our model, we enforce the outputs of our SRL system to agree with the spans associated with the syntactic parse of the sentence as much as possible. In summary, our contributions to low-resource SRL are:

1. Introducing hard syntactic constraint as a loss function (*syntactic-inconsistency loss*) during training time, as opposed to the previous studies which examined the effect of introducing similar constraints for neural SRL model in inference time (He et al., 2017).
2. Proposing a semi-supervised learning formulation for low-resource SRL by leveraging the

fact that the syntactic inconsistency loss does not require labels.

3. Performing experiments with varying amounts of SRL-unlabeled data, both with gold and system-generated parse tree, that show the effectiveness of semi-supervised learning for low-resource SRL.

4.2 Proposed Approach

We build upon an existing deep-learning formulation for SRL (He et al., 2017). First we revisit definitions introduced by He et al. (2017) and then discuss our approach on top of it.

4.2.1 Task definition

Given a sentence-predicate pair $(\mathbf{x}, v)$ with sentence $\mathbf{x} = x_1, x_2, \ldots, x_n$ and verb predicate v, SRL task is defined as predicting a sequence of tags $\mathbf{y} = y_1, y_2, \ldots, y_n$, where each x_i is word tokens and each y_i belongs to a set of BIO tags (Φ). So, for an argument span with semantic role ARG$_i$, the B-ARG$_i$ tag indicates that the corresponding token marks the beginning of the argument span, the I-ARG$_i$ tag indicates that the corresponding token is inside of the argument span, and the O tag indicates that token is outside of all argument spans. Let $n = |\mathbf{x}| = |\mathbf{y}|$ be the length of the sentence. Further, let *srl-spans*$(\mathbf{y})$ denote the set of all argument spans in the SRL tagging $\mathbf{y}$. Similarly, *parse-spans*$(\mathbf{x})$ denotes the set of all unlabeled parse constituents for the given sentence $\mathbf{x}$. Lastly, SRL-labeled data refers to sentence-predicate pairs with gold SRL labels and SRL-unlabeled data refers to set of sentences without such information.

4.2.2 Baseline model

He et al. (2017) proposed a deep bi-directional LSTM to learn a locally decomposed scoring function conditioned on the entire input sentence- $\sum_{i=1}^{n} \log p(y_i|\mathbf{x})$. To learn the parameters of a network θ, the conditional negative log-likelihood $\mathcal{L}(\theta)$ of a sample of training data $\mathcal{T} = \{\mathbf{x}^{(t)}, \mathbf{y}^{(t)}\}_{\forall t}$ is minimized, where $\mathcal{L}(\theta)$ is

$$\mathcal{L}(\theta) = -\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \sum_{i=1}^{|\mathbf{y}|} \log p(y_i|\mathbf{x};\theta). \tag{4.1}$$

Since Eq.(4.1) does not model any dependencies between the output tags, the predicted output tags tend to be structurally inconsistent. To alleviate this problem, He et al. (2017) search for the best $\hat{\mathbf{y}}$ by employing Astar-serach ($\mathtt{A}^\star$) over the space of all possibilities (Φ^n) using the scoring function $f(x, y)$, which incorporates log probability and structural penalty terms.

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}'\in\Phi^n}\, \mathbf{f}(\mathbf{x}, \mathbf{y}') \tag{4.2}$$

$$\text{where} \quad \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \log p(y_i|\mathbf{x}) - \sum_{c\in C} c(\mathbf{x}, \mathbf{y}_{1:i}) \tag{4.3}$$

where each constraint function c applies a non-negative penalty given the input $\mathbf{x}$ and a length-t prefix $\mathbf{y}_{1:t}$.

4.2.3 Structural Constraints

There are different types of structural constraints: BIO, SRL and syntactic constraints. *BIO constraints* define valid BIO transitions for sequence tagging. For example, B-ARG0 cannot be followed by I-ARG1. *SRL constraints* define rules on the role level and has three particular constraints, which are, (1) unique core roles constraint (U): each core rule cannot appear more than once in a sentence, (2) continuation roles constraint (C): a continuation role can only occur to the roles that already appeared in earlier spans, and (3) reference roles constraint (R): a span can only labeled as reference role if another span is labeled with the corresponding role (Punyakanok et al., 2008; Täckström et al., 2015). Lastly, syntactic constraints state that *srl-spans*$(\mathbf{y})$ have to be a subset of *parse-spans*$(\mathbf{x})$.

He et al. (2017) use BIO and syntactic constraints at decoding time by solving Eq. (4.2), where $f(x, y)$ incorporates those constraints, and report that SRL constraints do not show significant improvements over the ensemble model. In particular, by using syntactic constraints, He et al. (2017) achieve up to $+2$ F1 score on CoNLL-2005 dataset via $\mathtt{A}^*$ decoding.

Improvements of SRL system via use of syntactic constraints is consistent with other observations (Punyakanok et al., 2008). However, all previous work enforce syntactic consistency only during decoding step. We propose that enforcing syntactic consistency during training time would also be beneficial and show the efficacy experimentally on Section 4.3.3.

Enforcing Syntactic Consistency To quantify syntactic inconsistency, we define *disagreeing-spans* as

$$\text{disagreeing-spans}(\mathbf{x}, \mathbf{y}) \;=\; \{\text{span}_i \in \text{srl-spans}(\mathbf{y}) \mid \text{span}_i \notin \text{parse-spans}(\mathbf{x})\}. \tag{4.4}$$

Further, we define *disagreement rate*, $g(\mathbf{x}, \mathbf{y}) \in [0, 1]$, and *syntactic inconsistency score*, $s(\mathbf{x}, \mathbf{y}) \in [-1, 1]$, as follows:

$$g(\mathbf{x}, \mathbf{y}) = \frac{|\text{disagreeing-spans}(\mathbf{x}, \mathbf{y})|}{|\text{srl-spans}(\mathbf{y})|} \tag{4.5}$$

$$s(\mathbf{x}, \mathbf{y}) = 2 \times g(\mathbf{x}, \mathbf{y}) - 1. \tag{4.6}$$

Note that $g(\mathbf{x}, \mathbf{y})$ is defined on a sequence whereas $g(\cdot)$ in Eq. (3.6) for inference (GBI) was defined on a factorized spans in Chapter 3.

Syntactic Inconsistency Loss (SI-Loss) For a given $(\mathbf{x}, v)$, let us consider $\hat{\mathbf{y}}^{(t)}$ to be the best possible tag sequence (according to Eq.(4.2)) during epoch t of model training. Ideally, if our model is syntax-aware, we would have empty *disagreeing-spans* set and have $g(\mathbf{x}, \hat{\mathbf{y}}^{(t)}) = 0$ or $s(\mathbf{x}, \hat{\mathbf{y}}^{(t)}) = -1$. We define a loss component due to syntactic inconsistency (SI-Loss) as follows:

$$\mathcal{L}_{\text{SI}}(\theta) = s(\mathbf{x}, \hat{\mathbf{y}}^{(t)}) \sum_{i=1}^{|\hat{\mathbf{y}}^{(t)}|} \log p(\hat{y}_i^{(t)} | \mathbf{x}; \theta^{(t)}). \tag{4.7}$$

During training, we want to inject the syntactic constraint by minimizing the proposed syntactic inconsistency loss. In order to apply gradient-based learning on a non-differentiable scoring

function, we fix the score $s(\cdot)$ and multiply it to the gradient of probability term similar to that of REINFORCE(Williams, 1992). As one can observe, we use negative value of inconsistency score, $-s(\mathbf{x}, \mathbf{y})$, as a score function to promote (penalize) any SRL output $\mathbf{y}$ that is consistent (inconsistent) with the parse tree information.

4.2.4 Training with Joint Objective

Combining Eq.(4.1), a supervised loss, and Eq.(4.7), the SI-Loss, we propose a joint training objective. For a given sentence-predicate pair $(\mathbf{x}, v)$ and SRL tags $\mathbf{y}$, our joint loss (at mini-batch t) is defined as:

$$\mathcal{L}_{\text{Joint}}(\theta) = \alpha_1 \underbrace{\sum_{i=1}^{|\mathbf{y}|} -\log p(y_i|\mathbf{x}; \theta^{(t)})}_{Eq.\ (4.1)\ \text{supervised loss}} + \alpha_2 \underbrace{s(\mathbf{x}, \hat{\mathbf{y}}^{(t)}) \sum_{i=1}^{|\hat{\mathbf{y}}^{(t)}|} \log p(\hat{y}_i^{(t)}|\mathbf{x}; \theta^{(t)})}_{Eq.\ (4.7)\ \text{SI-loss}} \tag{4.8}$$

Here, α_1 and α_2 are weights (hyper-parameters)[4] for different loss components and are tuned using a development set. During training, we minimize joint loss - i.e., negative log-likelihood (or cross-entropy loss) and syntactic inconsistency (SI) loss simultaneously.

4.2.5 Semi-supervised learning formulation

In low resource scenarios, we have limited labeled data and larger amounts of unlabeled data. The obvious question is how to leverage large amounts of unlabeled data for training accurate models. In context of SRL, we propose to leverage SRL-unlabeled data in terms of parse trees.

Observing Eq.(4.7), one can notice that our formulation of SI-Loss is only dependent upon model's predicted tag sequence $\hat{\mathbf{y}}^{(t)}$ at a particular time point t during training and the given sentence and it does not depend upon gold SRL tags. We leverage this fact and formulate semi-supervised loss by computing SI-loss portion of Eq. (4.8) from SRL-unlabeled sentences.

In summary, with simplified notation $\log P(\mathbf{y}|\mathbf{x}; \theta) = \sum_{i=1}^{|\mathbf{y}|} \log p(\mathbf{y}_i|\mathbf{x}; \theta)$, we present our semi-supervised formulation of joint objective (SSL loss) as following:

$$\mathcal{L}_{\text{SSL}}(\theta) = \alpha_1 \underbrace{\sum_{(\mathbf{x}_j, \mathbf{y}_j) \in T} -\log P(\mathbf{y}_j|\mathbf{x}_j, \theta^{(t)})}_{\text{supervised}} + \alpha_2 \underbrace{\sum_{\mathbf{x}_k \in U} s(\mathbf{x}_k, \hat{\mathbf{y}}_k^{(t)}) \log P(\hat{\mathbf{y}}_k^{(t)}|\mathbf{x}_k, \theta^{(t)})}_{\text{unsupervised}} \tag{4.9}$$

where SRL-labeled data T only contributes to supervised loss and SRL-unlabeled data U contributes in terms of syntactic inconsistency objective. Note that on Eq. (4.8), two loss functions are evaluated at respective input samples $\mathbf{x}_j \in T, \mathbf{x}_k \in U$, whereas both loss functions on Eq. (4.9) are evaluated at the same instance $\mathbf{x}$.

[4]We use $\alpha_1 = \alpha_2$ for all the experiments after hyper-parameter tuning on the provided development set on CoNLL2012.

4.3 Experiments

4.3.1 Dataset

We evaluate our model's performance on the span-based SRL dataset from CoNLL-2012 shared task Pradhan et al. (2013). This dataset contains gold predicates as part of the input and also gold parse information corresponding to each sentence, which we use for defining hard constraints for the SRL task. We use the standard train/development/test split containing 278K/38.3K/25.6K sentences. There is approximately 10% disagreement between gold SRL-spans and gold parse-spans (we term these as noisy syntactic constraints). During training, we do not preprocess data to handle these noisy constraints, but for the analysis related to enforcing syntactic constraints during inference, we study both cases: with and without noisy constraints.

4.3.2 Model configurations

For the SOTA system proposed in Peters et al. (2018c), we use code from Allen AI[5] to implement our approach. We follow their initialization and training configurations. Let **B**$\underline{X}$**, J**$\underline{X}$ denote model trained with $\underline{X}\%$ of the SRL-labeled data with cross-entropy loss ($\mathcal{L}$) and joint loss ($\mathcal{L}_{\text{Joint}}$), respectively. Let, B$\underline{X}$-SI·$\underline{U}$x and B$\underline{X}$-SSL·$\underline{U}$x denote models trained with SI-loss ($\mathcal{L}_{\text{SI}}$) and semi-supervised loss ($\mathcal{L}_{\text{SSL}}$), respectively, on the pre-trained B$\underline{X}$ model where $\underline{X} \times \underline{U}$ amount of SRL-unlabeled data were used for further training. For example, if we trained on 2.8k instances (1% of training set) to pre-train a model $B1$, and used 5.6k unlabeled examples (2x times of labeled instance used) to train with SI-loss, then we will obtain B1-SI·2x after training. For all our results, unless expressed to use $\texttt{A}^\star$ or GBI, to satisfy BIO constraints, we run Viterbi decoding by default for inference.

For experiments with syntactic-inconsistency loss ($\mathcal{L}_{\text{SI}}$), we add an extra square loss, $\beta\|\theta - \theta_{\text{pre-train}}\|^2$ with β set to 0.005, to the Eq. (4.7) to keep the model θ close to the pre-trained model $\theta_{\text{pre-train}}$ to avoid catastrophic forgetting. We optimize with SGD with a step size of 0.01 and stop fine-tuning when the best development score is not updated more than 10 epochs.

For experiments with joint loss ($\mathcal{L}_{\text{Joint}}$) of Eq. (4.8) and semi-supervised loss ($\mathcal{L}_{\text{SSL}}$) of Eq. (4.9), we set the equal weights for supervised loss and syntactic-inconsistency loss, i.e. $\alpha_1 = \alpha_2 = 1.0$. we use SGD with a step size of 0.05 and stop fine-tuning when the best development score is not updated more than 10 epochs.

Finally, in order to examine whether the improvements from $\mathcal{L}_{\text{SI}}$, $\mathcal{L}_{\text{Joint}}$, and $\mathcal{L}_{\text{SSL}}$ are not some noisy byproducts of fine-tunning with a very small learning rate, we further applied cross entropy loss ($\mathcal{L}$) and optimized with SGD with a step size of 0.01 to the pre-trained model. We denote such models as B$\underline{X}$-micro as we are fine-tunning pre-trained B$\underline{X}$ with very small learning rate. Again, like all other learning, we stop fine-tuning when the best development score is not updated more than 10 epochs.

4.3.3 Results

We are interested in answering following questions.

[5]https://github.com/allenai/allennlp

Model/ Legend	Test F1	Average disagreement rate (%)
B100	84.40	14.69
B10	78.56	17.01
B1	67.28	21.17
J100	84.75 (+0.35)	14.48 (−1.43%)
J10	79.09 (+0.53)	16.25 (−4.47%)
J1	68.02 (+0.74)	20.49 (−3.21%)

Table 4.1: Comparison of baseline models (**B**) with the models trained with the joint objective (**J**). * Legend: **B**X**,** **J**X denotes model trained with $X\%$ of the SRL-labeled data with respective objective.

Base Model/ Legend	Test F1	Average disagreement rate (%)
B1	67.28	21.17
B1-SI·1x	67.67 (+0.39)	20.14 (−4.87%)
B1-SI·5x	**67.74 (+0.46)**	**19.93 (-5.86%)**
B1-SI·10x	67.71 (+0.43)	20.16 (−4.77%)
B10	78.56	17.01
B10-SI·1x	78.84 (+0.28)	16.17 (−4.94%)
B10-SI·5x	78.67 (+0.11)	16.47 (−3.17%)
B10-SI·10x	**78.76 (+0.20)**	**16.09 (-5.4%)**

Table 4.2: Training with **SI-loss** for varying sizes of SRL-unlabeled data on top of the pre-trained baseline models (B1, B10 on Table 4.1). * Legend: BX-SI·Ux denotes a model trained with SI-loss ($\mathcal{L}_{\text{SI}}$) on the pre-trained model BX where $X \times U$ amount of SRL-unlabeled data were used for further training.

- Q1. How often does the baseline model produce syntactically consistent outputs?
- Q2. Does our approach actually enforce syntactic constraints?
- Q3. Does our approach enforce syntactic constraints without compromising the quality of the system?
- Q4. How well does our SSL formulation perform, especially in low-resource scenarios?
- Q5. What is the difference in using the syntactic constraints in training time compared to using it at decoding time?

To answer (Q1-2), we report average disagreement rate on test data. To answer (Q3-5), we report overall F1-scores on test set. For experiments using SRL-unlabeled data, we report average results after running multiple experiments with different random samples of unlabeled data.

Does training with joint objective help? We trained 3 models (J1, J10, J100) with random 1% and 10% subsamples as well as 100% of the training set, with joint loss ($\alpha_1 = \alpha_2 = 0.5$ in Eq. (4.8)). For comparison, we trained 3 SOTA models (B1, B10, B100) with the same training

Base Model/ Legend	Test F1	Average disagreement rate (%)
B1	67.28	21.17
B1-micro	67.76 (+0.48)	20.75 (−1.98%)
B1-SSL·1x	68.45 (+1.17)	19.57 (−7.56%)
B1-SSL·5x	**68.86 (+1.58)**	**19.38 (-8.46%)**
B1-SSL·10x	68.61 (+1.33)	19.29 (−8.88%)
B10	78.56	17.01
B10-micro	78.86 (+0.3)	16.25 (−2.06%)
B10-SSL·1x	79.23 (+0.67)	16.03 (−5.76%)
B10-SSL·5x	79.25 (+0.69)	16.01 (−5.88%)
B10-SSL·10x	**79.34 (+0.78)**	**15.88 (-6.64%)**

Table 4.3: Training with semi-supervised objective (**SSL**) on top of the baseline models (**B1**, **B10** from Table 4.1), with the same SRL-labeled data used to train the baseline models and with varying sizes of SRL-unlabeled data. B1-micro, B10-micro represents running the same supervised loss that was used to pre-train B1, B10 but with smaller (micro) learning rate. The learning rates used in B1-micro, B10-micro were set to the same rate as other fine-tunning experiments. We have conducted these experiments to examine whether the gains are simply coming from running longer epochs of training. * Legend: BX-SSL·Ux denote model trained with semi-supervised loss ($\mathcal{L}_{\text{SSL}}$) on the pre-trained BX model where $X \times U$ amount of SRL-unlabeled data were used for further training.

sets. All models were trained for max 150 epochs and ended training earlier if F1 on validation set did not improve for 20 epochs.

Table 4.1 reports the results of this experiment. Answering (Q1), we can see that even state-of-the art model has average disagreement rate of 14.69% and that the F1 score and disagreement rate is inversely proportional. We see that models trained with joint objective (JX) improve over baseline models (BX), both in terms of F1 and average disagreement rate. These improvements provide evidence for answering (Q2-3) favorably. Further, gains are more in low resource scenarios because by training models jointly to satisfy syntactic constraints helps in better generalization when trained with limited SRL corpora.

Does SSL based training work for low-resource scenarios? To enforce syntactic constraints via SI-loss on SRL-unlabeled data, we further train the pre-trained model with two objectives in using unlabeled data: SI-loss ($\mathcal{L}_{\text{SI}}$) (in Table 4.2) and Semi-supervised loss ($\mathcal{L}_{\text{SSL}}$) (in Table 4.3).

Results related to SI-loss experiment are reported in Table 4.2. We see that with SI-loss improvements are more notable in terms of average disagreement rate as compared to F1. Results related to the semi-supervised loss are reported in Table 4.3. We report +1.58 F1 and +0.78 F1 improvement over B1 and B10, respectively, by training with semi-supervised loss using SRL-unlabeled data. Note that we cannot achieve these improvements by simply fine-tunning B$\underline{X}$ further with supervised loss, as seen with B$\underline{X}$-micro on Table 4.3. This provides evidence to answer (Q4) favorably. In general, the performance gains increase as the size of the SRL-unlabeled

Base Model/ Legend	Test F1	Average disagreement rate (%)
B1	67.28	21.17
B1-micro	67.76 (+0.48)	20.75 (−1.98%)
B1-SI·5x	67.74 (+0.46)	19.93 (−5.86%)
J1	68.61 (+1.33)	20.49 (−8.88%)
B1-SSL·5x	**68.86 (+1.58)**	**19.38 (-8.46%)**
B10	78.56	17.01
B10-micro	78.86 (+0.3)	16.25 (−2.06%)
B10-SI·10x	78.76 (+0.20)	16.09 (−5.4%)
J10	79.25 (+0.69)	16.01 (−5.88%)
B10-SSL·10x	**79.34 (+0.78)**	**15.88 (-6.64%)**

Table 4.4: Overall comparison of loss functions from experiments utilizing 1% (top) and 10% (bottom) of available SRL-labeled data. This table aggregates statistics of best-performing instance of each loss function (SI-loss, Joint, and semi-supervised loss (SSL)) from Table 4.1, 4.2, 4.3.

data increases for semi-supervised learning.

Lastly, in Table 4.4, we compare all presented loss functions together by aggregating statistics of best-performing instances of each loss functions from the Table 4.1, 4.3, Table 4.3. Both 1% and 10% cases show similar trend in that semi-supervised learning performs the best both in terms of F1 (higher the better) and average constraint violation rate (lower the better). While supervised loss with the small learning rate is similar in F1 with SI-loss, the average violation of constraint is lower for the model trained with SI-loss. In fact, the Table 4.4 shows the importance of learning syntactic constraint from unlabeled dataset. Both SI-loss and SSL which injects syntactic constraint using SRL-unlabeled data show lower syntactic constraint violation than other models.

Is it better to enforce syntactic consistency at decoding or at training time? To answer (Q5), we conducted three experiments: using syntactic constraints on (a) inference only, i.e. structured prediction, (b) training only, and (c) both training and inference steps. For the structured prediction, we consider A* decoding, as used in He et al. (2017), and GBI (Chapter 3. If neither A* decoding nor gradient-based inference is used, we use Viterbi algorithm to enforce BIO constraints. The performance is the best (bold on Table 4.5) when syntactic consistency is enforced both on training and inference steps, with +3.67, +2.1 F1 score improvement over B1 and B10 respectively, and we conclude that the effort of enforcing syntactic consistency at inference time is complementary to the same effort at training time.

While syntactic constraints help at both train and inference time, injecting constraints on train time is more robust than enforcing them on decoding time. The performance of GBI drops by $0.1 \sim 0.5$ in F1 score[6] when about 10% of parse data is noisy as can be seen in Table 4.5). On the other hand, the performance drop was around 0.1 in F1 score for SSL when the same amount

[6]The performance of A* drops too significantly given noisy constraints. We only consider GBI for the comparison of robustness.

Decoding	B10	B10-SSL·10x	B1	B1-SSL·5x
Viterbi	78.56	79.34	67.28	68.86
	Noisy syntactic constraints			
A*	72.95 (-5.61)	73.57 (-5.77)	63.77 (-3.51)	64.73 (-4.13)
Gradient-based	79.7 (+1.41)	80.21 (+0.87)	69.85 (+2.57)	**70.95** (**+2.1**)
	Noise-free syntactic constraints			
A*	78.87 (+0.31)	79.51 (+0.17)	67.97 (+0.69)	68.97 (+0.11)
Gradient-based	80.18 (+1.62)	**80.66** (**+1.32**)	69.97 (+2.69)	70.94 (+2.08)

Table 4.5: Comparison of different decoding techniques: Viterbi, A* He et al. (2017) and gradient based inference Lee et al. (2019) with noisy and noise-free syntactic constraints. Note that the (+/-) F1 are reported w.r.t Viterbi decoding on the same column.

Parse tree & Predicate type	B1-SSL5x		B10-SSL·10x		J100-SSL·3x	
	F1	Average Disagreement	F1	Average Disagreement	F1	Average Disagreement
Gold	68.86 (+1.58)	19.38 (-1.79)	79.34 (+0.78)	15.88 (-1.13)	- -	- -
System-predicted	68.57 (+1.29)	19.73 (-1.44)	79.01 (+0.45)	16.39 (-0.62)	84.87 (+0.47)	14.23 (-0.46)

Table 4.6: Comparing the best semi-supervised models in Table Table 4.3 that uses gold parse, from Treebank in CoNLL2012, with semi-supervised models that uses system-generated parse trees. The system-generated parse trees were created by running two off-the-shelf parsers, Berkeley Parser(Petrov et al., 2006) and ZPar(Zhu et al., 2013), on NYT dataset(Sandhaus, 2008). After selecting instances where both parses agree, the system-generated parse have approximately 80% agreement with gold SRL.

of noise was introduced.

4.3.4 Domain adaptation using output constraints

A machine Learning model will not perform as well on out-of-domain datasets as on the domain it was trained on, due to the difference on underlying distribution. We have observed such behaviors with neural models as well in earlier in section 3.3.1. To remedy this problem, the thesis asks a question 'Can we use the universal constraint independent of domain as a guidance to the domain adaptation'? We already have observed that using constraint information can improve performance on out-of-domain using structured prediction (GBI) in Chapter 3. We furthermore explore whether the proposed SSL approach in this chapter can remedy the difference of data distribution.

To examine whether the proposed SSL approach can help domain adaptation, the thesis

Genre / Task	Failure rate (%)		F1	
	before	after	before	after
	Syntactic Parsing			
Source domain (NW/WSJ)	13.90	-	84.40	-
Broadcast Conversation (BC)	19.32	11.16	66.93	74.23 (+7.30)
Broadcast News (BN)	11.67	10.56	78.77	79.04 (+0.27)
Pivot Corpus (PT)	9.84	6.63	85.99	87.06 (+1.07)
Telephone Conversation (TC)	10.01	9.49	65.99	66.13 (+0.14)
Weblogs (WB)	17.48	15.25	74.55	76.45 (+1.90)

Table 4.7: Evaluation of domain adaptation models further trained with structural loss on syntactic parsing application. F1 scores are reported on the whole test set per domain. The initial reference model was trained on WSJ section under NW on OntoNote v5.0. The structural loss training improves reference models on all cases and more than +1 F1 score except BN & TC. The table shows that learning with structural loss can be successfully applied to resolve performance degradation on out-of-domain data.

conducts experiments on the CoNLL2012 dataset. The experimental setting is similar to that of in Chapter 3.3.1. We use the NewsWire (NW) section of OntoNotes v5.0 English PropBank corpus as the source domain, and additional genres provided in the PropBank corpus (BC, BN, PT, TC, WB) as the target domains.

First, we examine how training with structural loss ($\mathcal{L}_{\text{SI}}$) Eq. (4.7) can help domain adaptation in case of syntactic parsing. We pre-trained the GNMT model for syntactic parsing with on NW section and then trained further with structural loss on the target domains. As Table 4.7 shows, the structural loss training on the target domain both reduces the failure rate and improves performance. In comparison to similar experiment using GBI in section 3.3.1, the F1 gains in this experiment (+) are relatively higher than GBI (+0.21 to +0.75 in F1), however, GBI reduces failure rate much more drastically (near 100% enforcement of constraint).

Second, we examine how semi-supervised training with the SSL loss ($\mathcal{L}_{\text{SSL}}$) can aid domain adaptation in the case of SRL. We pre-trained SRL network with ELMo embedding on the NW corpus and then apply the SSL approach, using the source domain as a supervised set and target domain as a unsupervised set with SSL loss formulation of Eq. (4.9). As Table 4.8 shows, SSL training with the joint objective usually reduces the failure rate and improves the performance, except the Telephone Conversation (TC) domain.

4.4 Related Work

Previous approaches for SRL (Pradhan et al., 2005; Koomen et al., 2005) constituted of cascaded systems with four subtasks: pruning, argument identification, argument labeling, and inference. Recent approaches (Zhou and Xu, 2015; He et al., 2017) proposed end-to-end systems for SRL using deep recurrent or bi-LSTM-based architecture with no syntactic inputs, and have achieved a competitive results on English SRL. Peters et al. (2018c) proposed ELMo, a deep contextualized word representation, and improved on He et al. (2017) by 3.2 F1-points. The baseline models (B1, B10, B100) in this chapter are based on Peters et al. (2018c),and our experiments obtain

Genre / Task	Failure rate (%)		F1	
	before	after	before	after
	SRL			
Source domain (NW)	18.10	-	77.45	-
Broadcast Conversation (BC)	26.86	19.70	68.05	71.17 (+3.12)
Broadcast News (BN)	18.51	15.19	49.92	74.96 (+1.25)
Pivot Corpus (PT)	10.01	9.01	83.50	85.86 (+2.36)
Telephone Conversation (TC)	19.09	20.15	72.70	71.61 (-1.09)
Weblogs (WB)	20.32	17.57	76.17	78.73 (+2.56)

Table 4.8: Evaluation of domain adaptation on SRL using the semi-supervised learning approach. F1 scores are reported on the whole test set per domain. The initial reference model was trained on the NW section on OntoNote v5.0. Except TC, a telephone conversation section, semi-supervised approach helps domain adaptation by giving more than +1 F1 score improvement. The table shows that semi-supervised learning can be successfully applied to resolve performance degradation on out-of-domain data.

similar performance. Since the submission of Mehta et al. (2018), which this chapter is based on, the performance of Peters et al. (2018c) for SRL has been surpassed by 0.9 (He et al., 2018) and 1.6 F1-points (Ouchi et al., 2018), respectively, by using a span-based model architecture. Additionally, BERT-based (Devlin et al., 2019) model was also proposed (Shi and Lin, 2019) recently resulting in a similar performance with state-of-the-art where BERT-small records on par and BERT-large records +0.3 F1 increment but still lower by 0.5 F1 score than the ensemble model of Ouchi et al. (2018).

Even with the end-to-end learning, inference still remains as a separate subtask and can be formalized as a constrained optimization problem. To solve this problem ILP (Punyakanok et al., 2008), *A* algorithm* (He et al., 2017) and gradient-based inference (GBI of Chapter 3) were employed. All of these works leveraged syntactic parses during inference and constraints were never used during training unless used as a cascaded system.

To the best of our knowledge, this work is the first attempt towards semi-supervised learning (SSL) for a span-based SRL model. Nonetheless, there has been some prior work in SSL for dependency-based SRL systems (Fürstenau and Lapata, 2009; Deschacht and Moens, 2009; Croce et al., 2010). Fürstenau and Lapata (2009) proposed to augment the dataset by finding similar unlabeled sentences and annotate accordingly. While interesting, this augmentation technique is hard to apply to span-based SRL, as it requires one to annotate the whole span. Deschacht and Moens (2009) and Croce et al. (2010) proposed to leverage the relations between words by learning a latent word distribution over the context, i.e. a language model. Our paper also incorporates this idea by using ELMo, as it is trained via a language model objective.

Along with this chapter, there were two concurrent works (Swayamdipta et al., 2018; Strubell et al., 2018) that tried to improve SRL by injecting prior knowledge of syntactic parsing. As discussed in section 2.3, both models take an approach of multi-task learning by changing the model structure and thus capturing the relation of the SRL and parsing in a soft manner. In contrast, this chapter focuses on injecting an explicit relation between the two tasks, by augmenting the training objective with loss that reflects a constraint, i.e. syntactic-inconsistency loss, in a model-

agnostic way without changing a model structure. In more detail, Swayamdipta et al. (2018) learns SRL and constituency parsing together in multi-task fashion by sharing common parameters. Strubell et al. (2018) uses dependency parse tree in order to guide attention mechanism in the self-attention model. All three works have shown significant improvements while the results are not directly comparable as each of them have different model architecture and evaluation set ups. By using syntactic information, Mehta et al. (2018) improved +0.47 F1 score on top of 84.4 F1 score on CoNLL2012 using baseline model of Peters et al. (2018c) and Swayamdipta et al. (2018) improved +0.8 F1 score on top of its semi-CRF's 83.0 F1 score baseline without ELMo. Lastly, Strubell et al. (2018) performs joint prediction of predicate and semantic-role labels and performs 83.12 F1 similar to its baseline (Tan et al., 2018) and better when dependency parse predictions are provided on inference time 83.38 F1. All three models demonstrated that injection of syntactic information to the SRL model is beneficial. However, among these three concurrent related work, this chapter uniquely focused on semi-supervised learning, in a model-agnostic way, in pursuit of improving low-resource settings.

Lastly, very recently, there was a follow up work (Nandwani et al., 2019) focused in learning with constraints. By expressing the constraint with soft logic (Brocheler et al., 2012) that is differentiable, Nandwani et al. (2019) proposes a primal-dual formulation. Following our experiment on this chapter, Nandwani et al. (2019) reports similar improvement in F1 score from 77.19 to 78.72 in F1 score on utilizing 10% of available training set, whereas this chapter reports F1 score improvement from 78.56 to 79.34 for the same setting.

4.4.1 Comparison to other constraint learning framework

As discussed in the related work section 2.2, the most notable frameworks to learn with constraint are Posterior Regularization (PR) (Ganchev et al., 2010) and Constrained-Driven Learning (CoDL) (Chang et al., 2007, 2012) where both frameworks solves constrained EM formulation.

The previous methods explores most credible outputs from unlabeled instances, using constraint information in the E-step of EM, to learn from such explored output. In contrast, this thesis is trying evaluate how good or bad the the model is by inspecting model's unconstrained output similar to Reinforcement Learning. By learning to penalize constraint-violating outputs the presented semi-supervised learning implants constraint function innately in the process.

In light of this difference, the proposed method brings one computation advantage: the ability to use very general constraints, global or semi-Markov constraints, without exponential complexity. As discussed in related work, running inference or sampling a sequence or from a posterior distribution with global constraint leads to a computation of exponential complexity. Thus, previous researches are typically too expensive to use with such general constraints, as one would have to run the expensive sampling steps multiple times in the learning process. Nonetheless, the proposed method does not suffer from this problem as the method evaluates on the outcome of an unconstrained inference step which has the linear complexity, or quadratic if we use Viterbi decoding. As an illustrative example, considering our SRL output with just length 7 and label space of 130, exponential complexity results in $\sim 10^{14}$ whereas linear and quadratic complexities are around 10^3 and 10^6, respectively.

4.5 Conclusion and Future Work

We presented a Syntactic-Inconsistency loss SI-loss to enforce SRL systems to produce outputs that agree with the provided parse tree. Furthermore, leveraging the fact that SI-loss does not require labeled data, we proposed a SSL formulation with joint objective constituting of SI-loss and supervised loss together. We show the efficacy of the proposed approach on low resource settings, +1.58, +0.78 F1 on 1%, 10% SRL-labeled data respectively, via further training on top of pre-trained SOTA model. We further show the structured prediction can be used as a complimentary tool and show performance gain of +3.67, +2.1 F1 over pre-trained model on 1%, 10% SRL-labeled data, respectively. Semi-supervised training from the scratch and examination of semi-supervised setting on large dataset remains as part of the future work.

Chapter 5

Semi-Supervised Learning with agreement constraint in multi-view models

In the course of this thesis, I have focused on a framework to enforce known output constraints. The major benefit of the constraint-injection approach is that it does not require a labeled data and thus can help to improve low-resource and domain-adaptation settings. However, it may be often difficult to identify a meaningful constraint for the given task without being proficient at it. Furthermore, the prior knowledge might require other types of labeled data or high-quality models to provide such constraint. For example, we need annotated parse trees or high-quality parser in order to apply the constraint we use for a SRL output and a parse tree. In this chapter, I focus on a more general constraint that can be applicable to any sequence-tagging problems. By solving sequence-tagging applications in a multi-view learning setup, I propose to apply agreement constraint between the views in order to promote coherence. Similar techniques to previous chapters are used in this chapter, and again similarly show significant improvements on low-resource settings as well as achieving state-of-the-art performance for the high-resource dependency parsing of Chinese corpus.

Multi-view learning makes use of diverse models arising from multiple sources of input or different feature subsets for the same task. For example, a given natural language processing task can combine evidence from models arising from character, morpheme, lexical, or phrasal views. The most common strategy to use multi-view data, especially popular in the neural network community, is to unify multiple representations into one unified vector through concatenation, averaging, or pooling, and then build a single-view model on top of the unified representation. As an alternative, we examine whether building one model per view and then unifying the different models can lead to improvements, especially in low-resource scenarios. More specifically, taking inspiration from Co-Training (Blum and Mitchell, 1998) methods, we propose a semi-supervised learning approach based on multi-view models through consensus promotion, and investigate whether this improves overall performance. To test the multi-view hypothesis, we use moderately low-resource scenarios for nine languages and test the performance of the multi-task model for part-of-speech tagging and dependency parsing. The proposed model shows significant improvements across the test cases, with average gains of -0.9 to +9.3 in labeled attachment score (LAS) for dependency parsing and gains of +2.2 to +7.8 universal POS-tagging score (UPOS) for tagging across 9 languages on extremely-low resource setting. We also investigate the effect of

unlabeled data on the proposed model by varying the amount of training data and by applying different domains of unlabeled data. The material presented in this chapter is based on Lim et al. (2020)[1].

5.1 Overview

Multi-view data consist of different manifestations of the same data, often in the form of different features, and such data are abundant in real-world applications (Xu et al., 2013). Color and texture information can be viewed as examples of multi-view data in image processing whereas character-, word- level representations, stem, prefix, and suffix are examples of multi-view data in Natural Language Processing (NLP).

The use of multi-view data has resulted in considerable success in various NLP problems. Combining different word representations at the character, token, or sub-word levels has proven to be helpful for dependency parsing (Botha et al., 2017; Andor et al., 2016), Part-of-Speech (POS) tagging (Plank et al., 2016), and other NLP tasks.

Given multiple views, a simple but popular approach is to unify multiple representations into a combined one through concatenation, averaging, or pooling. This approach is especially popular in neural networks as it is very straightforward to concatenate multiple representations without any modification of the model structure. All the aforementioned work also considered this approach. However, is it the best usage of multi-view data? A simple input concatenation can lead to overfitting problem as the model might ignore the specific statistical properties of each view (Zhao et al., 2017).

Recently, META-BILSTM (Bohnet et al., 2018) was proposed to extend the naive solution of concatenating input representations in the context of POS tagging, and it showed superior performance compared to simple view concatenation on input representations. META-BILSTM builds a single-view model of each view (lower layer) and concatenates the series of single-view-model outputs to form an input to the meta layer, as shown in Figure 5.2. All the components of META-BILSTM (per-view models and meta layer) are trained jointly, as expressed in Eq. (5.2).

In this chapter, we first examine whether META-BILSTM can be beneficial in the context of more complex tasks, namely multi-task learning in POS tagging and dependency parsing. The study then proposes CO-META, a semi-supervised approach, to improve each single-view model through the consensus promotion of the multiple single-view models on unlabeled data. The proposed CO-META is motivated by Co-Training (Blum and Mitchell, 1998), a classic approach to multi-view learning, which enables exploration of unlabeled data and is known to be helpful in low-resource settings. Overall, Co-Training and many of its variants improve the performance of multi-view models by maximizing agreement between the multi-view models on unlabeled data, and thus can improve performance in low-resource settings.

Thus, this study raises the question of whether classical Co-Training style approaches can further improve the META-BILSTM model in low-resource settings. Specifically, we explore two questions: (1) can models from different views learn from each other on unlabeled data? Moreover, (2) can this help the performance of low-resource models? We study whether improving each

[1]For Lim et al. (2020), Lee and Lim are the co-first authors with equal contributions.

multi-view model by promoting consensus in a Semi-Supervised Learning (SSL) fashion can lead to learning better meta models in the context of joint tagging and dependency parsing.

Once we apply META-BILSTM, we obtain several parsing models trained by each view. Then the main challenge that arises with regard to our SSL approach (CO-META) is deciding *what* and *how much* a single view should learn from other views. We suggest three different methods for determining *what* to learn from each other and named them as Sequence-based, Token-based voting, and the Ensemble-based approach. Then, to determine *how much* each view should learn from the determined example, we introduce an agreement score $g(\cdot)$, which serves as example weighting function, in section 5.3.3.

We employ META-BILSTM and our semi-supervised methods on top of the graph-based parser with a bi-affine classifier proposed by Dozat et al. (2017), and investigate the effectiveness of our approach on both low- and high-resource scenario experiment setups using the Universal Dependency 2.3 dataset (Zeman et al., 2018a). CO-META, the proposed model shows consistent improvement across the test cases, with an average of -0.9 to +9.3 Labeled Attachment Score (LAS) gains in low-resource (50 labeled instances) and +0.2 to +1.0 LAS gains in high-resource settings, respectively. The study also investigates whether the proposed method depends on unlabeled data by changing the amount and varying the domains of unlabeled data, and its effect on the proposed model. In summary, our contributions to joint parsing are as follows:

1. Proposal of a new formulation CO-META that leverages consensus promotion on top of a META-BILSTM model.
2. Analysis of the relation of each multi-view model performance to that of the meta model.
3. Exploring different semi-supervised scenarios, where the amount of unlabeled data and the domains of unlabeled data are varying.
4. Generalization of META-BILSTM and CO-META by expanding an additional-view model on top of the existing model using external word embedding.

5.2 Related Work

5.2.1 Dependency Parsing with Multi-Task Structure

Dependency parsing is an essential component of many NLP tools because of their ability to capture potentially complex relational information in a sentence. Formally, the goal of the dependency parsing is to produce tree structures for a sentence $x = (w_1, w_2 ...w_n)$ following a provided dependency grammar (De Marneffe et al., 2006; de Marneffe and Manning, 2008; Nivre et al., 2016). In general, a sentence's syntactic dependency tree consists of the dependency arc, from modifier, w_m, to the *Head*, w_h, and the dependency relation label on the arc, *Dep*, that defines the relation between w_m and w_h. Dependency parsing is widely used for NLP applications such as named entity recognition (Kazama and Torisawa, 2008), discourse understanding (Sagae, 2009), and information extraction (Culotta and Sorensen, 2004; Fares et al., 2018).

Recent breakthroughs in multi-task learning have made it possible to effectively perform different tasks with the same model. The multi-task approach enriches context-sensitive feature representations by learning different tasks using shared parameters (Hashimoto et al., 2016). In

NLP, this approach has been widely used to learn joint models performing tagging and parsing simultaneously, and all state-of-the-art (SOTA) models now use a multi-task structure. In general, given an input sentence x and a set of gold labels $y = (l_1, l_2...l_n)$, where each l_i consists of labels for tagging and parsing, the goal of the multi-task structure is to train a joint model that can provide simultaneously a POS tagger and a dependency parser.

There are many variants of multi-task learning for tagging and parsing. These variants consist in models sharing parameters between the tasks (Straka, 2018) and variants (Che et al., 2018; Lim et al., 2018). On top of this, recent systems trained with Language Model (LM) representations have shown even better results. One of these models, ELMo (Peters et al., 2018a), is trained with unsupervised textual representations using BiLSTM. Models with ELMo obtained the best performance in the 2018 CoNLL shared task (Che et al., 2018; Lim et al., 2018). Another more-recent and cutting-edge Language Model, BERT (Devlin et al., 2019), which is trained by bidirectional transformers with a masked language model strategy, shows outstanding results in parsing (Kondratyuk, 2019; Kulmizev et al., 2019). While many variants exist, all these models basically produce a single parser and tagger based on a single concatenated view. In contrast, Bohnet et al. (2018) proposed an approach to build several POS taggers trained by individual lexical representations and generated a multi-view model only for POS tagging.

5.2.2 Co-Training

The standard multi-view learning approaches try to learn a model by jointly optimizing all the multi-view models arising from different views as opposed to combining input level multi-view data. The most representative and one of the earliest multi-view learning methods is Co-Training (Blum and Mitchell, 1998). Co-Training and many of its variants (Nigam and Ghani, 2000; Muslea et al., 2002; Yu et al., 2011) try to maximize the mutual agreement of multi-view models on unlabeled data by promoting the *consensus* principle. The unified model is expected to perform better when each view provides some knowledge that the other views do not possess; that is, when different views hold *complementary* information.

5.3 Proposed Approach

This subsection details the model structures and loss functions that are used throughout this chapter. We first consider the baseline model, in section 5.3.1, introduced by Lim et al. (2018) and extend it to a multi-view model structure following Bohnet et al. (2018) in section 5.3.2. Then, in section 5.3.3, we present a new semi-supervised learning (SSL) approach. We name the baseline model as BASELINE, the extended multi-view model as META-BASE, and the SSL approach as CO-META.

5.3.1 BASELINE model

As it is known that using information from multiple views yield better performance, most SOTA multi-task parsers use both word-level and character-level views to get a lexical embedding $v_{1:n}^{(wc)}$ from a sequence of n words $w_{1:n}$. Most of these approaches simply concatenate the

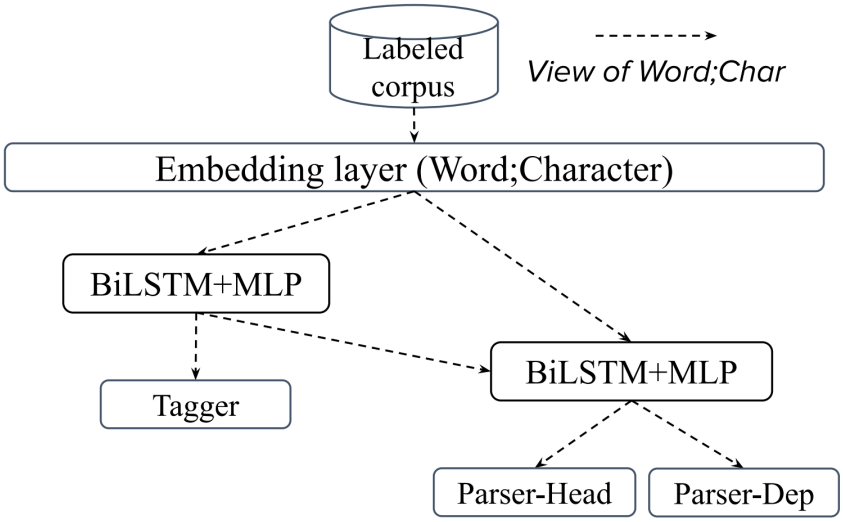


Figure 5.1: The structure of our baseline model: a multi-task POS tagger and dependency parser.

word embedding $v_i^{(w)}$ and the character-level embedding $v_i^{(c)}$ of w_i to form $v_i^{(wc)}$. For example, Figure 5.1 shows a multi-task parsing architecture for low-resource scenarios proposed by Lim et al. (2018) which obtains near state-of-the-art results on the CoNLL 2018 shared task (Zeman et al., 2018b)[2]. Specifically, the parser transforms the sequence of shared lexical representation $v_i^{(wc)}$ to a context-sensitive vector contextualized by BiLSTM with a hidden layer r_0 as:

$$h_i^{(pos)} = BiLSTM(r_0^{(pos)}, (v_1^{(wc)}, .., v_n^{(wc)}))_i$$
$$h_i^{(dep)} = BiLSTM(r_0^{(dep)}, (v_1^{(wc)}, .., v_n^{(wc)}))_i$$

The system uses vector $h_i^{(pos)}$ to predict *POS* with a Multi-layer Perceptron (MLP) classifier, and $h_i^{(dep)}$ to predict *Head* and *Dep* with a bi-affine classifier (Dozat and Manning, 2016). During training, it learns the parameters of the network θ that maximize the probability $P(y_j|x_j, \theta)$ from the training set T based on the conditional negative log-likelihood loss $\mathcal{L}_{\text{BASELINE}}$:

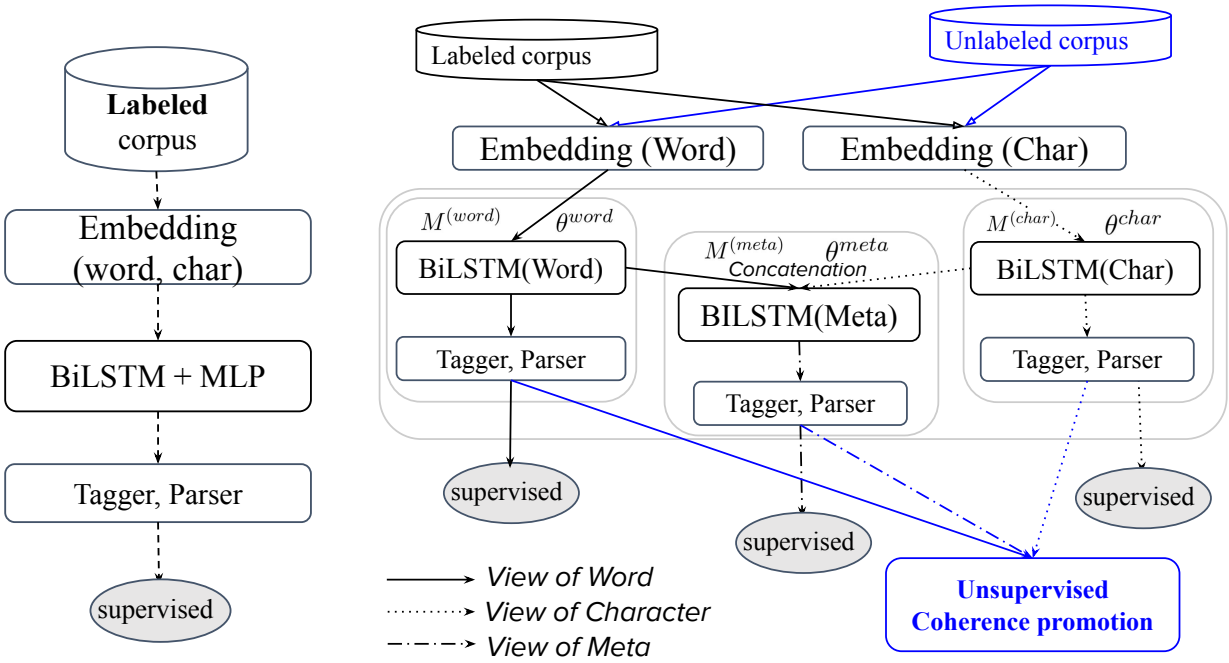
$$\mathcal{L}_{\text{BASELINE}}(\theta) = \sum_{(x_j, y_j) \in T} -\log P(y_j|x_j, \theta) \tag{5.1}$$

where $(x_j, y_j) \in T$ denotes an element from the training set T, y is a set of gold labels (l^{POS}, l^{Head}, l^{Dep}). The sequence of predicted labels $\hat{y}$ is obtained by simply taking argmax as follows:

$$\hat{y} = \arg\max_y P(y|x_j, \theta).$$

The model of Lim et al. is subsequently used as the BASELINE model.

[2]The parser achieved the 2nd and 4th ranks with regard to UAS and LAS, respectively, out of 27 teams in the CoNLL 2018 shared task.



(A) BASELINE (B) META-BASE (supervised loss only) (C) CO-META (with unsupervised loss)

Figure 5.2: These figures describe the overall structures of (A) BASELINE, (B) META-BASE, and (C) CO-META. The BASELINE model in (A) uses the embeddings from multiple view in a concatenated manner as an single input to a single model. In contrast, (B) META-BASE and (C) CO-META maintain separate BiLSTM layers for each view and then builds META-BILSTM on top of them. Both META-BASE and CO-META learn with supervised loss ($\mathcal{L}_{\text{META-BASE}}$), however, CO-META learns with an additional unsupervised loss g using an unlabeled corpus. While the figures describe a simple case where there are only two views (word, character), the model can easily expand with extra views as shown in our experiment with language-model (LM) embeddings (Table 5.6,5.7). In order to include language-model embeddings such as ELMo and BERT in our multi-view set up, we simply increase the number of views to be three (word, character, LM).

5.3.2 Supervised Learning on multi-view data (META-BASE)

Bohnet et al. (2018) increased the performance of tagger using META-BILSTM in the multi-view setting. In order to examine whether a similar approach would help the joint model of POS-tagging and dependency parsing, we propose the meta structure shown in Figure 5.2-(B). We use Lim et al. (2018)'s multi-task structure of tagging and parsing as our default structure for a single-view model (e.g. for $M^{(word)}$, $M^{(char)}$) and denote the overall system as META-BASE, as it serves as an additional baseline for the semi-supervised multi-view learning model (CO-META).

We define a model M^{vi} for each view $vi \in V$, where V is the set of all views. For example, Figure 5.2 contains different views for word, character, and meta, and overall view V is defined as $V = \{\text{word,char,meta}\}$. Each model M^{vi} consists of a BiLSTM^{vi} that contextualizes its view with a representation h_i^{vi} for word w, an MLP classifier to predict the POS tag, and a bi-affine

classifier to predict parsing outputs *Head* and *Dep*. The overall concatenation of the representation vectors h_i^{vi} from each view, for all $vi \in V$, becomes an input to the $M^{(meta)}$. For example, in Figure 5.2, M^{word} and M^{char} consume the word- and character-level embedding, respectively, and M^{meta} consumes the concatenation of two models' contextualized outputs as h^{wc} where h_i^{wc}=[h_i^{word};h_i^{char}]. Each M^{vi} is parameterized by the network parameter θ^{vi}, and the overall network parameter θ is defined as the union of the network parameters of all views, that is, $\theta = \cup_{vi \in V} \theta^{vi}$.

During supervised learning of a multi-view model using META-BILSTM, we train θ to maximize the probability $P(y_j|x_j, \theta) = \sum_{vi \in V} -\log P(y_j|x_j, \theta^{vi})$ which is simply the summation of the standard cross entropy loss across the views ($vi \in V$) for the input and labeled instance pair (x_j, y_j) in the training set T. Thus, our multi-view supervised model is trained by optimizing over the following supervised loss:

$$\mathcal{L}_{\text{META-BASE}}(\theta) = \sum_{(x_j, y_j) \in T} \sum_{vi \in V} -\log P(y_j|x_j, \theta^{vi}). \tag{5.2}$$

Observing Figure 5.2-(B), note that θ^{vi} is updated during optimization of both its own loss, $-\log P(y_j|x_j, \theta^{vi})$, and the loss for META-BILSTM, $-\log P(y_j|x_j, \theta^{meta})$.

5.3.3 CO-META

Coherence promotion. CO-META stands for the semi-supervised learning approach, which is inspired from Co-Training, on the meta structure. The main idea of Co-Training (Blum and Mitchell, 1998) is to augment training data with each model's confident prediction on unlabeled data so that each model can learn from other models' predictions. While not exactly following the Co-Train approach, we adopt the idea of one model teaching other models on unlabeled data.

We propose to promote coherence across multi-view models on the fly, by examining how similar outputs are across the views before each update. To do so we first extract the best possible result as $\hat{y}^*$ on a given instance x in unlabeled set U. We then impose an *agreement constraint* between $\hat{y}^*$ and each view(vi)'s output $\hat{y}^{vi}$ through the use of an *agreement score* $g(\hat{y}^*, \hat{y}^{vi})$. The resulting unsupervised loss $\mathcal{L}_{\text{Coherence}}$ is as follows:

$$\mathcal{L}_{\text{Coherence}}(\theta) = \sum_{vi \in V \setminus \{\text{meta}\}} \sum_{x \in U} -g(\hat{y}^*, \hat{y}^{vi}) \log P(\hat{y}^*|x, \theta^{vi}). \tag{5.3}$$

The *agreement score* $g(\hat{y}^*, \hat{y}^{vi})$ measures how much agreement the two arguments $\hat{y}^*, \hat{y}^{vi}$ have,

$$g(\hat{y}^{vi}, \hat{y}^{vj}) = \sum_{t=1}^{n} \frac{I(y_t^{vi}, y_t^{vj})}{n}, \tag{5.4}$$

where $I(\cdot)$ is a simple indicator function, n stands for the length of the sequence y, and y_t is the value of y at position t. Note that similar to Chapter 3,4, the constraint score function (*agreement score*) is a normalized count ranging 0 to 1, where 1 indicates perfect agreement. By optimizing to reward high-agreement cases in Eq. (5.3), we wish to promote the coherence across the views.

Another interpretation of $\mathcal{L}_{\text{Coherence}}$ in Eq. (5.3) is that we are assessing how much we can trust, when learning, the output $\hat{y}^*$. In the equation, the value of agreement score, $g(\hat{y}^*, \hat{y}^{vi})$, is acting as a confidence weight one should have in updating model θ^{vi} with instance $\hat{y}^*$. If the prediction $\hat{y}^{vi}$ has a similar structure to the extracted $\hat{y}^*$, then the vi-view model is aligned with the extracted output and thus can confidently learn from $\hat{y}^*$.

The idea of learning from a model's own prediction was explored by Nigam and Ghani (2000) in the context of self-training but without the agreement score. In our experiments, learning $\hat{y}^*$ without an agreement score was almost always worse than learning without it, often resulting in a negative effect. We present an ablation study on agreement score for the Chinese corpus in Figure 5.4.

Computing $\hat{y}^*$. We consider three methods to obtain $\hat{y}^*$: sequence-voting, token-voting, and ensemble-based computation.

- **Sequence-voting** selects the entire prediction of one model in one view vi^*, as $\hat{y}^* = \hat{y}^{vi^*}$, which has the highest likelihood for its prediction score, i.e. $vi^* = \text{argmax}_{vi \in V} P(\hat{y}^{vi}|x, \theta^{vi})$. In the sequence-voting approach, the view vi^* *only teaches other views and does not teach itself.*
- **Token-voting** selects the most popular label among the three models for each word w_m. When there is no agreement between the output of each model, we select the prediction of $M^{(meta)}$.
- **Ensemble-based** selects $\hat{y}^*$ using an ensemble method, that is, $\hat{y}^* = \textit{Softmax}(\sum_{vi \in V} P(\hat{y}^{vi}|x, \theta^{vi}))$.

5.3.4 Joint Semi-Supervised Learning

While the labeled data T is small in low-resource scenarios, we often have larger unlabeled data U. We thus need to leverage the supervised model Eq.(2) using unlabeled data. Since our $\mathcal{L}_{\text{Coherence}}$ only requires a prediction result $\hat{y}$, we can train both T and U as a joint loss ($\mathcal{L}_{\text{SSL}}$) as follows:

$$\begin{aligned} \mathcal{L}_{\text{SSL}}(\theta) &= \mathcal{L}_{\text{META-BASE}}(\theta) + \mathcal{L}_{\text{Coherence}}(\theta) \\ &= \sum_{(x_j, y_j) l \in T} \sum_{vi \in V} -\log P(y_j|x_j, \theta^{vi}) \\ &+ \sum_{vi \in V \setminus \{\text{meta}\}} \sum_{x_k \in U} -g(\hat{y}_k^*, \hat{y}_k^{vi}) \log P(\hat{y}_k^*|x_k, \theta^{vi}) \end{aligned} \tag{5.5}$$

where $T \subseteq U$ might apply to U, T when using T without labels. In what follows, let's call CO-META the training process with $\mathcal{L}_{\text{SSL}}$ on the meta-LSTM structure.

5.4 Experiments

5.4.1 Data Sets

We evaluate CO-META on the Universal Dependency 2.3 test set[3] for nine languages, following the criteria from de Lhoneux et al. (2017), with regard to typological variety, geographical distance, and the quality of the treebanks. Our testing languages are thus Ancient Greek, Chinese, Czech, English, Finnish, Greek, Hebrew, Kazakh, and Tamil. During training, we use pre-trained word embeddings[4] and unlabeled data[5] from the CoNLL 2018 shared task to initialize our word embedding $v^{(w)}$ and the SSL presented in the previous section. When we employ Language Models, we use pretrained models provided by Lim et al. (2018) for ELMo and Google[6] for BERT. We use the gold segmentation result for the training and test data.

5.4.2 Evaluation Metrics

There are two major evaluation metrics in dependency parsing. The Unlabeled Attachment Score (UAS) is used to evaluate the structure of a dependency graph. It measures to what extent the structure of the parsed tree is correct, without taking into account the labels on the different arcs of the tree. The Labeled Attachment Score (LAS) is the same as UAS, but takes into account dependency labels.

As for POS tagging, we report universal POS-tagging score (UPOS) score which measures the percentage of words that are assigned the correct POS label. We evaluate our tagger and parser based on the official evaluation metric provided by the CoNLL 2018 shared task[7].

5.4.3 Experimental Setup

To test the low-resource scenario following Guo et al. (2016), we sample the first 50 instances (Table 5.2) or 100 instances (Table 5.3) from the labeled data to form the training set. In addition, we conduct an ablation study to examine the effect of the proposed semi-supervised approach in various conditions: by increasing train set size with fixed unlabeled data for Chinese (Figure 5.3) and by varying the unlabeled dataset type and size when the training set is fixed for Greek (Table 5.5). The hyperparameter setup related to layers of each view and meta are presented in the Table 5.1. For simplicity and for the sake of comparison, we set the respective hyperparameters of each view's model and meta-layer to be identical to that of BASELINE model. We use a batch size of 2 (low-resource with 50, 100 instances of training set), 32 (high-resource training set with few thousands of training instances), and 10 (high-resource + BERT) respectively for different resource sizes. In order to reflect the joint loss function (5.5) we alternate supervised loss and unsupervised loss in weighted manner varying from a 8-to-1 ratio to a 25-to-1 ratio. We evaluate our models on the test sets, and report the average of the three best performing results, trained

[3] http://hdl.handle.net/11234/1-2895
[4] http://hdl.handle.net/11234/1-1989
[5] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989#
[6] https://github.com/google-research/bert
[7] https://universaldependencies.org/conll18/evaluation.html

Table 5.1: Hyperparameter setup for experiments

Component	**value**
$v^{(c)}$ (char) Dim.	100
$v^{(w)}$ (word) Dim.	100
$v^{(elmo)}$ Dim.	1024
$v^{(bert)}$ (multi) Dim.	768
$v^{(bert)}$ (base) Dim.	200
$h^{(word)}$ (word) output Dim.	400
$h^{(char)}$ (char) output Dim.	400
No. BiLSTM layers	2
MLP output (arc) Dim.	300
MLP output (dep) Dim.	300
MLP output (pos) Dim.	100
Dropout	0.3
Learning rate	0.002
Learning rate (BERT)	0.00001
β_1, β_2	0.9, 0.99
Epoch	1,000
Batch size (low-resource: 50, 100 labels.)	2
Batch size (high-resource: thousands of labels.)	32
Batch size (high-resource+Language model)	10
Gradient clipping	5.0

with different initial seeds, within 1,000 epochs. All the reported scores are based on the scores from the meta-layer output, unless otherwise stated.

5.5 Results

Our study has several goals: (1) to study the impact of multi-view based learning, META-BASE CO-META, on tagging and parsing in low-resource scenarios, (2) to check whether CO-META can increase the consensus between single-view models and the effect of this promoted consensus on the performance of each-view model and on the overall META-BILSTM system, (3) to study the effect of unlabeled data on CO-META, and finally (4) to investigate to what extent the efficacy of CO-META remains when the approach is applied to high-resource scenarios.

5.5.1 Results in Low-Resource Settings

Impact of Multi-View Learning. Table 5.2 shows the experimental results of $M^{(meta)}$, given 50 training instances, on the test data of each language. We see that the proposed Co-Training method CO-META shows average performance gains of -0.9 to +9.3 LAS points in parsing and +1.8 to 7.8 UPOS points in tagging compared to BASELINE.

Table 5.2: LAS and UPOS scores of $M^{(meta)}$ model output on the test set using **50** training sentences and unlabeled sentences based using CO-META, META-BASE, and our BASELINE model Lim et al. (2018). We report META-BASE, which does not use any unlabeled data, in order to decompose the performance gains into the gains due to META-BASE (supervised) and CO-META (semi-supervised).

		CO-META						META-BASE		BASELINE	
		TOKEN VOTING		SEQUENCE VOTING		ENSEMBLE					
corpus name	number of unlabeled	LAS	POS	LAS	POS	LAS	POS	LAS	POS	LAS	POS
cs_cac (Czech)	23478	47.4	79.4	47.4	79.7	**48.7**	**81.4**	45.9	79.0	39.4	74.6
fi_ftb (Finnish)	14981	21.7	43.2	22.0	44.7	21.8	43.5	21.9	**44.6**	**22.6**	39.2
en_ewt (English)	12543	45.1	75.7	46.3	**76.7**	**46.5**	76.3	45.4	75.2	42.8	71.1
grc_perseus (Ancient Greek)	11460	30.8	70.1	**31.7**	**70.9**	31.3	70.7	30.9	70.4	29.5	65.8
he_htb (Hebrew)	5240	47.9	76.9	47.8	77.2	**48.4**	**77.4**	47.6	76.7	45.1	75.2
zh_gsd (Chinese)	3997	36.1	70.7	35.1	70.8	**36.9**	**71.1**	35.1	70.6	34.8	68.7
el_bdt (Greek)	1162	60.0	84.3	**60.6**	83.2	60.5	**84.2**	57.8	82.6	51.7	80.0
ta_ttb (Tamil)	400	38.1	69.1	39.0	**69.7**	**40.0**	69.3	38.3	67.3	34.0	61.9
kk_ktb[8] (Kazakh)	12000	27.6	56.9	27.9	57.0	**28.7**	57.1	27.8	**57.7**	26.2	53.0
Average	-	39.4	69.6	39.8	70.0	**40.3**	**70.1**	39.0	69.3	36.2	65.5

Note that the proposed META-BASE approach also shows a LAS improvement of -0.7 to +6.5 and UPOS improvement of +1.6 to +5.4 over BASELINE as well. Breaking down the contribution of improvements, CO-META shows -0.3 to +2.8 LAS improvement and -1.3 to +2.4 UPOS improvement over META-BASE, and these improvements are comparable to the improvements of META-BASE over BASELINE. The average LAS score improvement of ENSEMBLE-based CO-META over META-BASE is +1.3 and the improvement from BASELINE to META-BASE is +2.8.

The experiment with 100 labeled instances (Figure 5.3) also shows that improvement of CO-META from META-BASE and META-BASE from BASELINE are comparable. The average improvement of LAS score ENSEMBLE-based CO-META over META-BASE is +1.4 and the improvement from BASELINE to META-BASE is +1.8 LAS score. We also observe that CO-META improves over BASELINE for Finnish, unlike the experiment with 50 labeled instances in Table 5.2.

Table 5.3: LAS and UPOS scores of $M^{(meta)}$ model on the test set using **100** training sentences. We see that CO-META improves over BASELINE for Finnish, unlike the results in Table 5.2 (50 sentences used)

		CO-META						META-BASE		BASELINE	
		TOKEN VOTING		SEQUENCE VOTING		ENSEMBLE					
corpus name	number of unlabeled	LAS	POS	LAS	POS	LAS	POS	LAS	POS	LAS	POS
cs_cac (Czech)	23478	55.0	83.6	54.9	82.9	**56.3**	**84.6**	54.1	84.0	50.8	81.6
fi_ftb (Finnish)	14981	28.5	50.7	29.0	50.9	**29.2**	**51.1**	29.0	50.5	27.6	49.7
en_ewt (English)	12543	56.7	81.3	**57.9**	**82.5**	**57.9**	82.0	56.5	82.3	55.1	80.6
grc_perseus (Ancient Greek)	11460	36.5	77.1	**38.1**	**78.0**	37.0	77.8	36.0	77.4	34.9	76.2
zh_gsd (Chinese)	3997	44.5	76.5	43.5	76.2	**45.3**	**76.9**	42.9	76.2	41.0	74.1
el_bdt (Greek)	1162	68.5	**88.7**	**69.1**	88.2	69.0	88.5	67.4	87.4	66.2	86.7
Average	-	48.3	76.3	48.8	76.5	**49.1**	**76.8**	47.7	76.3	45.9	74.8

Table 5.4: LAS on Greek(el_bdt) corpus for each model, with the average agreement score $g(\hat{y})$ comparing $M^{(word)}$ and $M^{(char)}$ over the entire test set using 100 training sentences.

Method	WORD	CHAR	META	AGREEMENT SCORE
SEQUENCE-VOTING	**61.8**	66.7	**69.1**	0.871
ENSEMBLE	61.4	**66.9**	69.0	**0.879**
WITHOUT	57.6	65.2	67.4	0.799

Comparison of CO-META Variants. When we compare the three proposed Co-Training approaches, one can see that the ENSEMBLE approach seems to work better than SEQUENCE VOTING, and TOKEN VOTING is always worst. Because the best-voted labels for each token are not guaranted to get an optimal structure over the parse tree at the sentence-level, the TOKEN-VOTING model has a relatively high chance of learning from an inconsistent graph, e.g. a dependency parse with multiple roots and cycles.

Interaction among the layers? A more detailed per-layer analysis of the LAS scores is available in Table 5.4 for the Greek corpus. Among the three views, CHAR always outperforms WORD, and all three views improve after using CO-META: there are improvements of 1.6-1.7 LAS points for META, 1.5-1.7 for CHAR and 3.8-4.2 for WORD.

We make three interesting observations. First, we note that the model with lower performance, namely, the WORD view in our example, always benefits the most from other better-performing views. Second, the evolution of low-performing views towards better results has a positive effect

Table 5.5: Scores of CO-META with the ENSEMBLE method on different domains of unlabeled data with 100 training sentences.

Labeled	**Unlabeled**	size	LAS	UAS	POS
el_bdt	el_bdt	1162	**69.0**	**75.6**	88.5
(Greek)	wikipedia	12000	68.7	75.1	**88.7**
	crawl	12000	68.3	74.8	88.4
zh_gsd	zh_gsd	3997	45.3	57.9	76.9
(Chinese)	wikipedia	12000	**46.3**	**59.1**	77.6
	crawl	12000	46.1	59.0	**77.8**

on the META view, and thus on the overall performance. While the score of CHAR increases by 1.5, META increases by 1.7. If the lower-performing-view model was not helping, then the improvement would be upper-bounded by the performance gain of the higher-performing model. Note that we do not update the META layer $\theta^{(meta)}$ when using CO-META, and so the gains result from the improvements of the single-view layers. Lastly, we can observe the AGREEMENT SCOREs between *word* and *char* views on the last column increase when we apply CO-META. As the higher AGREEMENT SCORE denotes that models predict a similar tree structure, we can confirm that CO-META indeed promotes consensus between the views.

Sensitivity to the Domain of the Unlabeled Set. In Table 5.5, we investigated a more realistic scenario for our semi-supervised approach for two languages, Chinese and Greek, by using out-of-domain data: Wikipedia and a crawled corpus. In the case of Chinese, the crawled out-of-domain corpus shows better results than the in-domain corpus for both ENTROPY-based and ENSEMBLE-based CO-META, by up to 1.1 UAS and 0.9 POS points. In contrast, for Greek, the in-domain corpus (el_bdt) shows a better result than the out-of-domain corpus even though the size of el_bdt is only about 13% of the others. We conjecture that as the Chinese has a large character set, the exposure to diverse characters helps learning regardless of the domain.

Effect of Training Size on Performance. Table 5.2 shows positive results for CO-META given fixed size training data. However, would CO-META be useful even with extremely low resource scenarios (<50 sentences)?, or in a more favorable scenario, when more resources are available for training (e.g. >1000 sentences)? To answer these questions, we conducted an experiment using the zh_gsd (Chinese) corpus with training sets of different sizes, but with a fixed set of 12k unlabeled data. The results are visible in Figure 5.3 (A,B).

Figure 5.3(A) shows our results for the lower resource scenario (with less than 50 sentences for training). CO-META outperforms META-BASE and BASELINE, except when only five sentences are used for training. We conjecture that this result is attributable to the fact that too little vocabulary is used to allow meaningful generalization. A similar behavior was observed for fi_ftb in Table 5.2: in this experiment, there is only 241 token available for fi_ftb, whereas other languages had on average of 1388 tokens. However, as observed in Figure 5.3, once we expand the labeled instances (>20 sentences), CO-META and META-BASE always outperform BASELINE, both in lower (3A) and higher resource (3B) settings. Also note that CO-META always outperforms META-BASE, including when one only has 5 labeled instances for training.

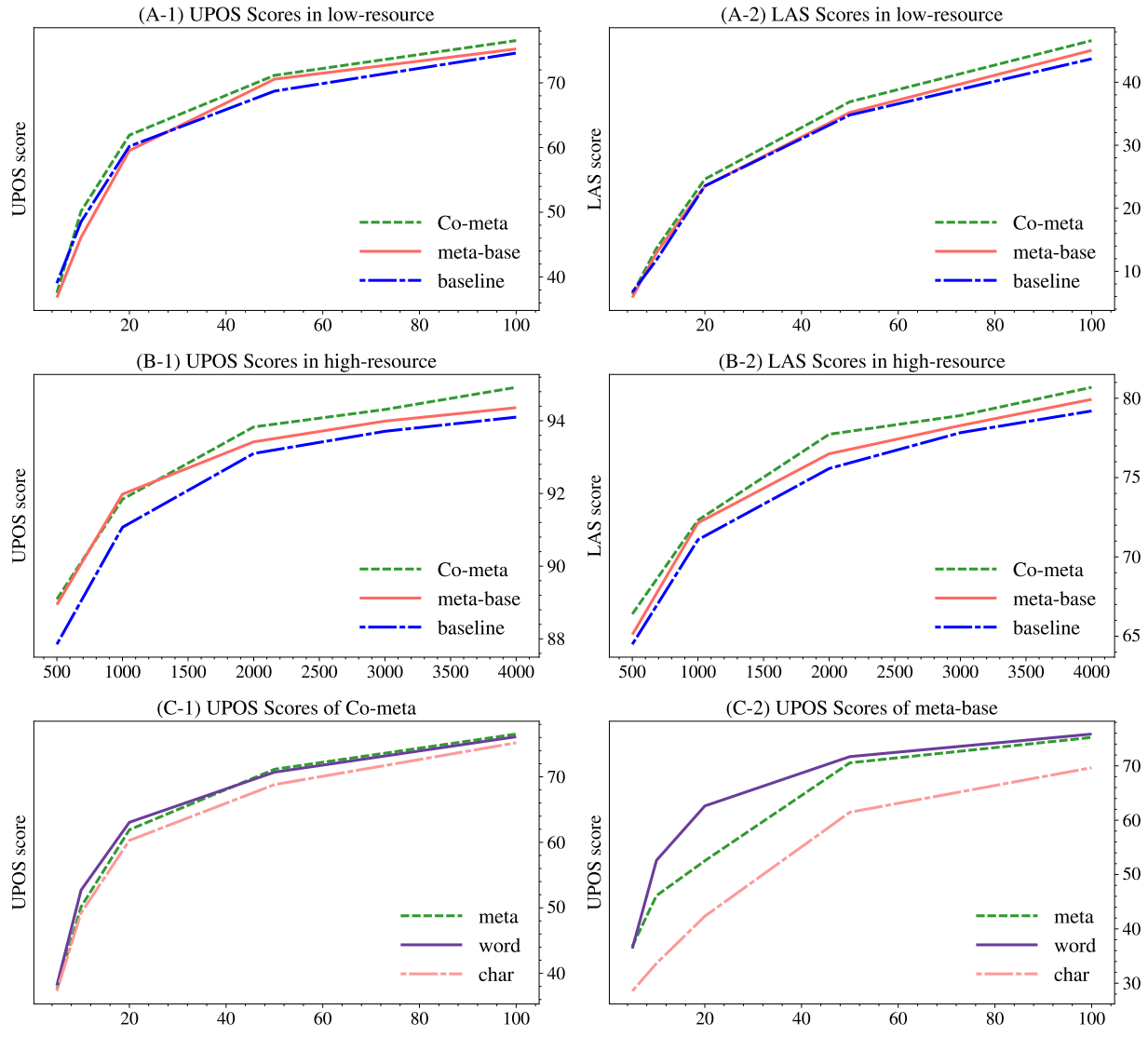


Figure 5.3: Evaluation results for Chinese (zh_gsd) on various sizes of the train set (5,10,20,50,100,500, 1k,2k,3k,4k) together with the fixed size of 12k unlabeled sentences. The test results show the effect of varying training set size, while the unlabeled set size is fixed, on the ENSEMBLE-based CO-META.

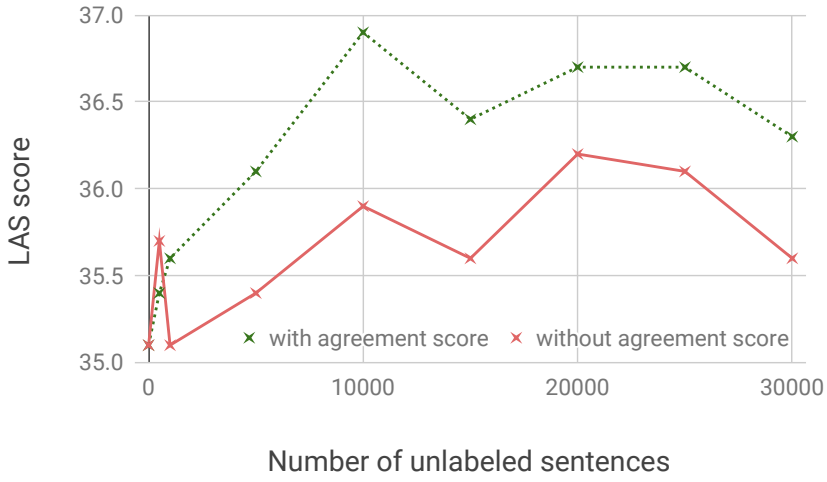


Figure 5.4: Evaluation results for Chinese (zh_gsd) based on different sizes of the unlabeled set and proposed models. We apply ENSEMBLE-based CO-META with the fixed size of 50 training sentences while varying the unlabeled set size (0, 500, 1000, 5000, . . . , 30000). Except the case of using 500 unlabeled set, in all seven cases, CO-META preformed better.

We can refine our analysis by examining the different layers of META-BASE and CO-META that appear on Figure 5.3 (A-1). META-BASE is detailed on Figure 5.3 (C-2) and CO-META on Figure 5.3 (C-1). In most cases, META view stays close to the highest performing view (the WORD layer for most cases). One interesting fact is that the WORD- as well as meta- layers of META models (both META-BASE and CO-META) outperform the BASELINE which is built on a combined view.

The biggest contrast between CO-META and META-BASE is the gap between the performances of the WORD and the CHAR layers. A closer look at META-BASE(C-2) seems to indicate that the performance of the META layer cannot differ too much from the lower-performing layer (CHAR in our case). When the gap between WORD and CHAR becomes too large (>5 points), then the performance gain of META layer is parallel to that of CHAR layer for training set size of 10–50 even when the WORD layer makes large performance gains. In contrast, the CO-META's META layer from 3(C-1) shows more stable performance, as the gap between CHAR and WORD is minimal as the two layers learn from each other.

To summarize from Table 5.2 and Figure 5.3, the proposed SSL approach is always beneficial for the META-BILSTM structure when comparing the LAS scores between CO-META and META-BASE. However, the META-BILSTM structure itself might not benefit when too few tokens exist in the train set. In general, we hypothesize that for META-BILSTM structure to be useful, the train set should consist of more than 300 tokens (more than 20 sentences).

Effect of agreement score. Lastly, we also ran ablation study (Figure 5.4) to examine the effect of the agreement score on Chinese corpus (zh_gsd). To compare to the current setup $\mathcal{L}_{\text{SSL}}$, where the agreement score is used as a weight, we tried running CO-META experiments

Table 5.6: LAS for the English (en_ewt) corpus for each model, with the external language models with the entire train set which has 12,543 labeled sentences.

Model	**LM**	LAS	UAS	POS
UDPIPE (Kondratyuk, 2019)	-	86.97	89.63	96.29
BASELINE (Lim et al., 2018)	-	86.82	89.63	**96.31**
METABASE	-	86.95	89.61	96.19
CO-META	-	**87.01**	**89.68**	96.17
BASELINE (Lim et al., 2018)	ELMo	88.14	91.07	96.83
METABASE	ELMo	88.28	91.19	96.90
CO-META	ELMo	88.25	91.19	96.84
UDIFY (Kondratyuk, 2019)	BERT-MULTI	88.50	90.96	96.21
UUPARSER (Kulmizev et al., 2019)	BERT-MULTI	87.80	-	-
BASELINE	BERT-MULTI	89.34	91.70	96.66
METABASE	BERT-MULTI	89.49	**92.01**	96.75
CO-META	BERT-MULTI	**89.52**	91.99	**96.80**

Table 5.7: LAS for the **Chinese (zh_gsd)** corpus for each model, with the BERT-Multilingual embedding using the entire training set. We observe much higher improvements than for English showed (see Table 5.6), probably because zh_gsd has a relatively small training set (3,997) and larger character sets than the training set (12,543) of English (en_ewt) corpus.

Model	**LM**	LAS	UAS	POS
UDPIPE	-	80.50	84.64	**94.88**
BASELINE	-	79.70	84.28	94.41
METABASE	-	80.32	84.58	94.72
CO-META	-	**80.71**	**84.99**	94.81
UDIFY	BERT-MULTI	83.75	87.93	95.35
UUPARSER	BERT-MULTI	83.7	-	-
METABASE	BERT-MULTI	83.90	88.07	**96.07**
CO-META	BERT-MULTI	**84.21**	**88.39**	**96.07**

without agreement scores by setting them all to 1. The train data was fixed with 50 instances and unlabeled set size varied With this configuration, we find that, the performance is always worse than CO-META, and thus we conclude that the proposed agreement score plays a major role in stabilizing the Co-Train approach.

5.5.2 Results in High-Resource Settings

Although the lack of annotated resources for many languages has given rise to low-resource approaches, several languages exist with plenty of resources. We thus need to examine whether our approach is also effective in more favorable setting, when large scale resources are available. A comprehensive overview is shown in Table 5.6, which compares different systems using no language model (first part of the table) and using ELMo (Peters et al., 2018a) or BERT (Devlin

et al., 2019) for English corpus (en_ewt). We conduct similar experiment for Chinese corpus (zh_gsd) as well and present in Table 5.7.

Table 5.6, 5.7 includes a comparison of our results using the approach presented in this chapter with four state-of-the-art systems. The first system is BASELINE (introduced in section 5.3.1), which obtained the best LAS measure for English in the 2018 CoNLL shared task. The second is UDPIPE (Straka, 2018) which was one of the best performing systems during the 2018 CoNLL shared task (best MLAS score, that combines tagging and parsing, and 2nd for the average LAS score). UDPIPE uses a multi-task learning approach with a loosely-joint LSTM layer between tagger and parser. The third system is UDIFY (Kondratyuk, 2019) (derived from UDPIPE), where the LSTM layer is replaced with BERT embedding, which is in turn fine-tuned during training. The fourth system is UUPARSER (Kulmizev et al., 2019) wherein concatenated word, character and BERT embeddings serve as an input, i.g., $h_i = [v^{(wc)}; v^{(bert)}]$.

Effect of CO-META On High-Resource Settings without LMs. By improving our baseline of our meta-LSTM and SSL approaches, we observe a slight improvement of up to 0.19 and 0.04 LAS points against the BASELINE and UDPIPE, respectively, for English corpus Table 5.6. In contrast, we find that both META-BASE and CO-META slightly underperform the BASELINE in tagging, is surprising. One possible reason might be that there is enough data to get accurate results using a supervised learning approach while SSL suffers from unexpected surface sequences. Another evidence of this is that SSL did not bring further improvement when using more than 10,000 training sentences. In contrast, Chinese corpus in Table 5.7, for which we had a relatively small train set (3,997), is positively affected by SSL, with a gain of up to 0.21 LAS points comparing to UDPIPE, and 1.01 points comparing to the BASELINE. We assume that the main reason for this is the character set. Languages with a bigger character set size and little training data gain more benefit from SSL.

Effect of CO-META On High-Resource Settings with LMs. While we train our model with a LM, we concatenate the last layer of the LM embedding with the input of the $BiLSTM^{(meta)}$ presented in the previous section. Finally, the input of our meta model consists of three different contextualized features as $[h_i^{(word)}; h_i^{(char)}; v_i^{(lm)}]$.

On average, adding a LM provides excellent results for both dependency parsing and POS tagging outperforming cases without LMs by large margins, up to 1.24 LAS point for ELMo and 2.51 LAS point for BERT for English in Table 5.6, and up to 3.51 LAS point for BERT for Chinese in Table 5.7. Furthermore, our parser with CO-META globally shows better results than the state-of-the-art parsers that use ELMo (Lim et al., 2018) and the BERT-Multilingual model (Kondratyuk, 2019). However, it should be noted that the UDIFY model used by Kondratyuk (2019) (which includes BERT-Multilingual as a LM) was first trained with 75 different languages using a Universal Dependency corpora and then tuned for English, and it is not clear how this training process affects the performance. Thus, we add the results of UUPARSER and BASELINE with BERT which fine-tune on only one language and still found that CO-META+BERT-MULTI shows better performance.

In this experiment, we generalized CO-META by adding an additional view: LM embedding. This version of CO-META can, surprisingly, improve by more than 1–1.7 (English) and around

1.5 (Chinese) LAS points compared to competing models even in a high-resource+LM setting. Note that we obtain the state-of-the-art performance for Chinese corpus with CO-META.

5.6 Conclusion

In this chapter, we have presented a multi-view learning strategy for joint POS tagging and parsing using Co-Training methods. Among three proposed (sequence-voting, token-voting, and ensemble-based) strategies for CO-META, the ensemble-based model yield the best result. This strategy is especially well suited for low-resource scenarios, when only a very small sample of annotated data is available, along with larger quantities of unlabeled data. Our experiment shows statistically significant gains (-0.9 to +9.3 points compared to the baseline), largely due to the proper integration of unlabeled data in the learning process. As future research, we wish to apply CO-META to other sequence-labeling tasks such as Named Entity Recognition and semantic role labeling.

Part III

Looking Forward

Chapter 6

Conclusions and Future work

In this chapter, we first summarize the key contributions of this thesis, and then present some potential directions that the presented methods can take.

6.1 Summary

In this thesis, we proposed methods that incorporate prior knowledge or constraints on the output space. The proposed method for injecting constraints in the output space is especially beneficial for the field of applied machine learning, since it is easy and intuitive to design constraints for each task a priori. Furthermore, in order for non-machine-learning experts to apply the presented methods, all the algorithms proposed in this thesis injects constraints by only with modifying the loss function without changing the model structure. To demonstrate the proposed constraint-injection methods, this thesis mainly focuses on natural language understanding tasks such as syntactic parsing and semantic role labeling where the constraints can be easily identified from its inherent syntactic structure. With this high-level motivation, the key contributions of this thesis are as follows.

First, we presented the Gradient-Based Inference (GBI) algorithm where the output constraint can be utilized in the inference procedure. Performing inference with a global constraint in a discrete space requires a combinatorial search that leads to exponential complexity. To overcome this combinatorial search problem in the output space, this thesis transforms it to a gradient-based search problem in the continuous model space motivated by the dual decomposition approach. GBI's approach is novel since it was the first approach to enforce output constraints using model updates on neural models, improving on after previous dual decomposition approaches for linear models (Rush et al., 2010; Chang et al., 2012). Through the experiments on semantic role labeling (SRL), constituency parsing and a toy transducer problem, GBI was shown to effectively enforce constraints while also improving the overall performance of neural models. The experiments were conducted in an extensive manner as GBI was examined on low-resource, high-resource, and on out-of-domain examples, and all results demonstrated positive effects of GBI. Particularly in SRL, the comparison with the A*-search algorithm, GBI exhibited faster, higher-performing, and more robust-to-noise results. The SRL experiments beat the previous state-of-the-art result(He et al., 2018) on the CoNLL2012 test set by injecting the syntactic constraint into an older, lower-

performing model (He et al., 2017). However, our SRL model has been superseded by the concurrent work (Ouchi et al., 2018) which uses different model structures.

One concern around applying GBI to neural models was on *catastrophic forgetting* of neural networks in that they can unlearn what they learned with a few gradient steps. Since GBI is applying gradient-based model updates, there was a risk of a neural network forgetting the main task functions in the process of enforcing subsidiary constraint. This thesis mitigates this problem by introducing an L2 regularizer, so that the model parameters do not drift too far away from the original learned model weights, which was demonstrated to be effective through extensive experiments.

Second, we extended the test time constraint injection of GBI to training time, from test-instance-based optimization on inference time to generalization to multiple instances at training time. In contrast to test-time inference, here there is no fixed learned model parameter that can serve as an anchor for L2 regularization. We proposed a joint loss function to stabilize the constraint-learning signal with the supervised signal. Furthermore, observing that constraint loss does not require labeled data, we presented a semi-supervised learning (SSL) framework with the aim of improving performance on low-resource scenarios. With experiments on SRL, the SSL framework showed that it can reduce constraint violations and improve the overall performance in a stable manner. The experiments were first performed using annotated parse tree information where the SSL improved the performance of 1%- and 10%- resource-SRL models by +1.58 and +0.78 F1 respectively. In order to show that this approach is also applicable in the real world setting, the same experiment was conducted using off-the-shelf parsers as well, which improved 100%-resource-SRL models by +0.47 F1. As GBI did, the SSL framework showed its strength on low-resource and out-of-domain settings. Additionally, the analysis of SSL with GBI suggests that the efforts on training time and on inference time (GBI) are complementary rather than mutually exclusive: the performance is best when efforts on train-time and inference-time methods are combined.

Lastly, we presented a coherence promotion method (CO-META) on a multi-view learning set up that can significantly help low-resource scenarios. By introducing *agreement constraint* between multiple views, we were able to apply a similar SSL framework as we did on SRL. While the previous chapters were geared toward injecting prior knowledge about a task, Chapter 5 proposed an agreement-constraint which can be applied to any sequence-labeling problem with multiple views. Besides presenting CO-META, this thesis first contributes by applying the recent META-BILSTM(Bohnet et al., 2018) approach to multi-view learning (META-BASE), previously applied to POS-tagging, to a more complex problem of joint modeling of POS and dependency parsing. Based on this, we introduced the CO-META framework which can further improve META-BASE using semi-supervised learning, and applies *agreement constraint* across the views using unlabeled data. Building on the competitive BASELINE model for dependency parsing, through experiments, we demonstrated that META-BASE and CO-META can significantly improve low-resource models across 9 different languages. With an ablation study varying the size of training data with fixed unlabeled data, we showed that CO-META can significantly help extremely-low and mid resource settings (train data ranging from 5-4k). We further saw a strong effect of CO-META on state-of-the-art models by plugging in external language models, and exhibited state-of-the-art result for Chinese dependency parsing. Whereas the first set of experiments focused on a trivial multi-view setup by only using word- and character- based embeddings for

CO-META, in applying the external language model, this thesis demonstrated that CO-META can also handle an expanded multi-view setup with word-, character-, and external-language-model (ELMo, BERT) - based embeddings. Along with studying the effect of train data, we examined the effect of unlabeled sets on CO-META by varying domain and amount of unlabeled data on Greek and Chinese corpora. Finally, we also inspected whether the improvements from CO-META are, in fact, a result of promoting coherence as we intended. In an ablation study, we showed that the performance difference between multiple views on META-BASE decrease by applying CO-META. This result offered the same conclusions as seen in other constraint-injection methods in this thesis, in that the overall performance was increased as well as enforcing the constraint.

6.2 Future work

Overall, this thesis presented ways to enforce output constraints when they are in the form of scalar scoring function. This leads to several future research questions. First, the relation of the constraint function to the main task should be further studied in terms of when the constraint injection will be most, and least, helpful. Second, a formal study on which unlabeled data to use and how much should be used in relation to the constraint function is required. While there are detailed experiments on the OntoNote5.0 (Weischedel et al., 2012) and CoNLL2018 dataset in this thesis, rigorous studies on conditions to when the presented semi-supervised would be successful are left as future work. Finally, studies on how to enforce multiple constraints simultaneously should also be conducted. One trivial approach would be forming multiple constraint functions as a single weighted scoring function. However, this question is related to the multi-objective research problem (Miettinen, 1999; Hwang and Masud, 2012) and is left for future research as well.

While there could be many potential directions, we introduce an immediate extension of this thesis in detail: multi-task learning with constraint loss as a regularizer. Then, we conclude with discussions on when the models presented could be useful and when it might be hard to apply.

6.2.1 multi-task systems with agreement scores

Many NLP tasks, for example, are related to each other and thus are solved in a multi-task learning framework. The usual multi-task learning simply combines a supervised signal from multiple tasks without taking care of specific relations between the tasks. However, extending this thesis, it would be noteworthy to examine whether we can utilize an agreement score across the multi-task output as a regularizer on the multi-task training process.

Generally, the style of loss function presented in this thesis can be expressed by rewriting the Eq. (4.9) to

$$\sum_{(x^{(i)},y_1^{(i)})\in D_1} XL(x^{(i)}, y_1^{(i)}; \theta_1) + \sum_{(x^{(j)},y_2^{(j)})\in D_2} f_{agree}(\hat{y}_1^{(j)}, y_2^{(j)} | x^{(j)}; \theta_1), \tag{6.1}$$

where the XL denotes cross entropy loss, $f_{agree}(y_1, y_2)$ denotes agreement score function between the output y_1, y_2, D_1 denotes a primary training set, D_2 is an additional dataset which could be

unlabeled, and $\hat{y}_k^{(j)}$ denotes output from model θ_k and input $x^{(j)}$. We express two related y_1 and y_2 with different subscripts where a subscript serves as a task id. It is worth noting that in order to measure $f_{agree}(y_1, y_2)$, the formats of y_1 and y_2 do not have to be identical, while it is necessary for them to be comparable in order to measure an agreement. For example, returning to the example of SRL application, we can think of $\hat{y}_1^{(j)}$ to be an SRL output and $y_2^{(j)}$ to be parse tree information for a sentence $x^{(j)}$, i.e. subscript 1 indicates SRL and 2 indicates syntactic parse tree. While SRL output $\hat{y}_1^{(j)}$ and parse tree information $y_2^{(j)}$ will not be in the same format, the spans of $y_1^{(j)}$ and $y_2^{(j)}$ can be used to measure an agreement score. In our previous experiment f_{agree} measures whether the spans of $\hat{y}_1^{(j)}$ are a subset of $y_2^{(j)}$.

Expanding the formulation to the multi-task setting we can simply extend the same equation onto two data distribution: D_1 from task 1 and D_2 from task 2.

$$\sum_{(x^{(i)}, y_1^{(i)}) \in D_1} \Big[XL(x^{(i)}, y_1^{(i)}; \theta_{com}, \theta_1) + f_{agree}(y_1^{(i)}, \hat{y}_2^{(i)}; \theta_{com}, \theta_2) \Big] + \sum_{(x^{(j)}, y_2^{(j)}) \in D_2} \Big[XL(x^{(j)}, y_2^{(j)}; \theta_{com}, \theta_2) + f_{agree}(\hat{y}_1^{(j)}, y_2^{(j)}; \theta_{com}, \theta_1) \Big] \tag{6.2}$$

We decompose the model parameter θ as common model part θ_{com} between task1 and task2 and differentiating part θ_i for each task i, i.e. $\theta = \theta_{com}, \theta_1, \theta_2$. Note that with the agreement score function, $f_{agree}(\hat{y}_1^{(j)}, y_2^{(j)}; \theta_{com}, \theta_1)$, we can now further train model of task1 (θ_1) even when we are training task2 using training set D_2 in a supervised way, and vice versa, using the same agreement function in a different direction $f_{agree}(y_1^{(i)}, \hat{y}_2^{(i)}; \theta_{com}, \theta_2)$ on D_1. While training with agreement function, which could be unstable, the supervised loss on another task can serve as a stable anchor, as we have seen in Chapter 4. With this formulation, there is one clear benefit. The model can learn how the outputs of multi-task should behave with each other even when the training instances are not annotated for both task1 and task2 jointly. Thus, this approach could serve as a method to learn disjoint training dataset on multiple NLP tasks given a reasonable agreement function.

With the same spirit, further extending the approach recursively, we can even try to leverage unlabeled dataset and perform semi-supervised learning:

$$\sum_{(x^{(i)}, y_1^{(i)}) \in D_1} \Big[XL(x^{(i)}, y_1^{(i)}; \theta_{com}, \theta_1) + f_{agree}(y_1^{(i)}, \hat{y}_2^{(i)}) \Big] + \sum_{(x^{(j)}, y_2^{(j)}) \in D_2} \Big[XL(x^{(j)}, y_2^{(j)}; \theta_{com}, \theta_2) + f_{agree}(\hat{y}_1^{(j)}, y_2^{(j)}) \Big] + \sum_{x^{(k)} \in D_{unlabeld}} f_{agree}(\hat{y}_1^{(k)}, \hat{y}_2^{(k)}). \tag{6.3}$$

6.2.2 Future applications

This thesis focused on expressing the output constraint as a simple scalar constraint function, and the method presented in this thesis shows one way of utilizing this proposed constraint function. Specifically, this thesis shows that simple normalized count of error in a sequence could be used to

effectively enforce structural constraints. As the proposed constraint function is generic and easy to formulate, we believe the presented methods can easily extend to other applications swiftly.

A direct application could be on applying CO-META to sequence-labeling tasks and helping low-resource scenarios as we did for dependency paring. CO-META can be easily extended to other labeling tasks such as named entity recognition (NER) and SRL as there was no explicit knowledge used in introducing CO-META, except that there should be coherence across views. While applying CO-META to other sequence-labeling tasks will be a straightforward extension, as CO-META requires multiple copies of models, the memory footprint can be a problem given limited resources. Thus, CO-META will be easier to apply on tasks where smaller models are preferred.

Second, as discussed in the previous subsection, the work presented in this thesis can comfortably expand to multi-task problems. Returning to the example of SRL and syntactic parsing, we believe a similar approach of injecting syntactic information can be applied to other span-based models such as co-reference resolution and shallow discourse relation problems. Another example is neural machine translation (NMT) where various multi-task approaches try to learn syntactic parser and tagger jointly with NMT. While there are existing efforts to enforce syntactic tree structure through top-down decoding (Gū et al., 2018) or by manipulating attentions (Eriguchi et al., 2016), it would be interesting to apply agreement scores to the unconstrained output as this thesis did. Through use of scoring function, one could evaluate whether translation preserves syntactic structure or whether aligned tokens have a matching POS-tag type and use this signal to further learn. Although omitting detailed examples for brevity, question answering could benefit from the presented work with logic constraint from multiple task.

As we have discussed various possible extensions, we should also consider when the presented methods are difficult to utilize. The presented methods inject information through evaluation of constraint function, and thus, it would be laborious to apply these methods on a task where it takes hours or days to evaluate constraint. For example, some topological consistency measures might take days, and even years, to compute (Otter et al., 2017). While this measure might be useful in generating objects with some topological constraints, the presented methods in this thesis would not be suitable, as these require multiple evaluation of such expensive metric.

To summarize, this thesis concentrated on injecting output constraints to the black-box neural models in a model-agnostic way. While it is tempting to change the model structure per constraint injection, this model-agnostic framework provides a capability to easily extend this framework to different tasks and different model architectures as presented in this thesis. With this benefit, while this thesis mainly focused on structural knowledge on the output space, the presented techniques are amenable to extend toward other models and constraints. This may contribute toward regularizing neural models to behave more coherently to real-world constraints.

Bibliography

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042, 2016. URL http://arxiv.org/abs/1603.06042. 5.1

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR, arXiv preprint arXiv:1409.0473*, 2014. 3.2.7

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 65–69, 2013. 4.1

David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, 2016. 3.4

David M. Blei, Andrew Bagnell, and Andrew K. McCallum. Learning with scope, with application to information extraction and classification. In *Uncertainty in Artificial Intelligence (UAI)*, 2002. 3

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998. 1.2, 5, 5.1, 5.3.3

Bernd Bohnet, Ryan T. McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. *CoRR*, abs/1805.08237, 2018. URL http://arxiv.org/abs/1805.08237. II, 5.1, 5.2.1, 5.3, 5.3.2, 6.1

Jan A. Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan T. McDonald, and Slav Petrov. Natural language processing with small feed-forward networks. *CoRR*, abs/1708.00214, 2017. URL http://arxiv.org/abs/1708.00214. 5.1

Matthias Brocheler, Lilyana Mihalkova, and Lise Getoor. Probabilistic similarity logic. *arXiv preprint arXiv:1203.3469*, 2012. 4.4

Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *ACL*, 2007. 4.4.1

Ming-Wei Chang, Lev-Arie Ratinov, Nicholas Rizzolo, and Dan Roth. Learning and inference with constraints. 2008. 2.1

Ming-Wei Chang, Lev Ratinov, and Dan Roth. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, Sep 2012. ISSN 1573-0565. doi: 10.1007/

s10994-012-5296-5. URL https://doi.org/10.1007/s10994-012-5296-5. 2.1, 2.1, 2.2, 2.2, 4.4.1, 6.1

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium, October 2018. ACL. URL http://www.aclweb.org/anthology/K18-2005. 5.2.1

Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006. 3.1.2

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, October 2014. 3

Michael Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003. 2.3

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011. 2.3

Danilo Croce, Cristina Giannone, Paolo Annesi, and Roberto Basili. Towards open-domain semantic role labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 237–246, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1858681.1858706. 4.4

Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42Nd Annual Meeting on ACL*, ACL '04, Stroudsburg, PA, USA, 2004. ACL. doi: 10.3115/1218955.1219009. URL https://doi.org/10.3115/1218955.1219009. 5.2.1

Dipanjan Das, André FT Martins, and Noah A Smith. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 209–217. Association for Computational Linguistics, 2012. 2.1, 2.1

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement. In *In Proceedings of the 15th Treebanks and Linguistic Theories Workshop*, pages 99–110, 2017. 5.4.1

Marie-Catherine de Marneffe and Christopher D. Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA, 2008. ACL. ISBN 978-1-905593-50-7. URL http://dl.acm.org/citation.cfm?id=1608858.1608859. 5.2.1

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating typed dependency parses from phrase structure parses. In *Lrec*, volume 6, pages 449–454, 2006. 5.2.1

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. 2.2

Koen Deschacht and Marie-Francine Moens. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 21–29, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-59-6. URL http://dl.acm.org/citation.cfm?id=1699510.1699514. 4.4

Daniel Deutsch, Shyam Upadhyay, and Dan Roth. A general-purpose algorithm for constrained sequential inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 482–492, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-1045. URL https://www.aclweb.org/anthology/K19-1045. 3.4

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805. 2.3

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423. 4.4, 5.2.1, 5.5.2

Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734, 2016. URL http://arxiv.org/abs/1611.01734. 5.3.1

Timothy Dozat, Peng Qi, and Christopher D. Manning. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada, August 2017. ACL. URL http://www.aclweb.org/anthology/K/K17/K17-3002.pdf. 5.1

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *NAACL-HLT*, pages 199–209, 2016. 3.4

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Tree-to-sequence attentional neural machine translation. *arXiv preprint arXiv:1603.06075*, 2016. 6.2.2

Murhaf Fares, Stephan Oepen, Lilja Øvrelid, Jari Björne, and Richard Johansson. The 2018 shared task on extrinsic parser evaluation: On the downstream utility of English universal dependency parsers. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 22–33, Brussels, Belgium, October 2018. ACL. URL http://www.aclweb.org/anthology/K18-2002. 5.2.1

Hagen Fürstenau and Mirella Lapata. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 220–228, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1609067.1609091. 4.4

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *J. Mach. Learn. Res.*, 11:2001–2049, August 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1859918. 2.2, 2.2, 4.4.1

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017. 3.2.6, 3

Gerald Gazdar, Ewan Klein, Geoffrey K Pullum, and Ivan A Sag. *Generalized phrase structure grammar*. Harvard University Press, 1985. 2.3

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Neural Information Processing Systems (NIPS)*, 2015. 3.2.3

Jetic Gū, Hassan S Shavarani, and Anoop Sarkar. Top-down tree structured decoding with syntactic connections for neural machine translation and parsing. *arXiv preprint arXiv:1809.01854*, 2018. 6.2.2

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. A representation learning framework for multi-source transfer parsing. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 5.4.3

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. *CoRR*, abs/1611.01587, 2016. URL http://arxiv.org/abs/1611.01587. 5.2.1

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 473–483, 2017. (document), 3.2.6, 4, 3.9, 3.10, 3.4, 4, 4.1, 1, 4.2, 4.2.2, 4.2.2, 4.2.3, 3.2.8, 4.5, 4.4, 6.1

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2058. URL https://www.aclweb.org/anthology/P18-2058. 4, 4.4, 6.1

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric P. Xing. Harnessing deep neural networks with logical rules. In *Association for Computational Linguistics (ACL)*, 2016. 3.4

C-L Hwang and Abu Syed Md Masud. *Multiple objective decision making—methods and applications: a state-of-the-art survey*, volume 164. Springer Science & Business Media, 2012.

6.2

Jun'ichi Kazama and Kentaro Torisawa. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *proceedings of ACL-08*, pages 407–415, 2008. 5.2.1

Paul Kingsbury and Martha Palmer. From treebank to propbank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. European Language Resources Association (ELRA), 2002. URL http://www.aclweb.org/anthology/L02-1283. 4.1

Daniel Kondratyuk. 75 languages, 1 model: Parsing universal dependencies universally. *CoRR*, abs/1904.02099, 2019. URL http://arxiv.org/abs/1904.02099. 5.2.1, 5.6, 5.5.2, 5.5.2

Terry Koo, Alexander M Rush, Michael Collins, Tommi Jaakkola, and David Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298. Association for Computational Linguistics, 2010. 3.1, 3.1.1, 3.4

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 181–184. Association for Computational Linguistics, 2005. 4.4

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. Deep contextualized word embeddings in transition-based and graph-based dependency parsing–a tale of two parsers revisited. *arXiv preprint arXiv:1908.07397*, 2019. 5.2.1, 5.6, 5.5.2

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *Machine Learning*, pages 1378–1387, 2016. 3

Jay Yoon Lee, Michael Wick, Jean-Baptiste Tristan, and Jaime Carbonell. Enforcing output constraints via sgd: A step towards neural lagrangian relaxation. *AKBC workshop*, 2017. 1.1, 3.2.8

Jay Yoon Lee, Sanket Vaibhav Mehta, Michael Wick, Jean-Baptiste Tristan, and Jaime Carbonell. Gradient-based inference for networks with output constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4147–4154, 2019. (document), 1.2, 1, 3, 4.5

KyungTae Lim, Cheoneum Park, Changki Lee, and Thierry Poibeau. SEx BiST: A multi-source trainable parser with deep contextualized lexical representations. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 143–152, Brussels, Belgium, October 2018. ACL. URL http://www.aclweb.org/anthology/K18-2014. (document), 5.2.1, 5.3, 5.3.1, 5.3.1, 5.3.2, 5.4.1, 5.2, 5.6, 5.5.2

KyungTae Lim, Jay Yoon Lee, Jaime Carbonell, and Thierry Poibeau. Semi-supervised learning on meta structure: Multi-task tagging and parsing in low-resource scenarios. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 1.2, 2, 5, 1

Gordon Lyon. Syntax-directed least-errors anallysis for context-free languages: A practical approach. *Programming Languages*, 17(1), January 1974. 3.1.1

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993. 4.1

Sanket Vaibhav Mehta, Jay Yoon Lee, and Jaime G. Carbonell. Towards semi-supervised learning for deep semantic role labeling. *CoRR*, abs/1808.09543, 2018. URL http://arxiv.org/abs/1808.09543. 1.1, 1.2, 1, 4, 2, 3, 4.4

K. Miettinen. *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management Science. Springer US, 1999. ISBN 9780792382782. URL https://books.google.com/books?id=ha_zLdNtXSMC. 6.2

Marcus Mitchell, B Santorini, MA Marcinkiewicz, and A Taylor. Treebank-3 ldc99t42 web download. *Philidelphia: Linguistic Data Consortium*, 3:2, 1999. 3.2.7

Ion Muslea, Steven Minton, and Craig A Knoblock. Active+ semi-supervised learning= robust multi-view learning. In *ICML*, volume 2, pages 435–442, 2002. 5.2.2

Yatin Nandwani, Abhishek Pathak, Parag Singla, et al. A primal dual formulation for deep learning with constraints. In *Advances in Neural Information Processing Systems*, pages 12157–12168, 2019. 4.4

Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998. 2.2, 2.2

Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93, 2000. 5.2.2, 5.3.3

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia, May 2016. European Language Resources Association. URL https://www.aclweb.org/anthology/L16-1262. 5.2.1

Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1):17, 2017. 6.2.2

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. A span selection model for semantic role labeling. In *EMNLP*, pages 1630–1642. Association for Computational Linguistics, 2018. 1, 4, 2, 4.4, 6.1

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018a. URL http://arxiv.org/abs/1802.05365. 5.2.1, 5.5.2

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018b. 3, 1, 3.2.6, 3, 2

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018c. 4, 4.1, 4.3.2, 4.4

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics, 2006. (document), 4.6

Barbara Plank, Anders Søgaard, and Yoav Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *CoRR*, abs/1604.05529, 2016. URL `http://arxiv.org/abs/1604.05529`. 5.1

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 217–220. Association for Computational Linguistics, 2005. 4.4

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, 2013. 1.2, 3.2.6, 4.1, 4.3.1

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, 2008. 3.2.1, 3.2.1, 3.4, 4.1, 4.2.3, 4.4

Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Computational Natural Language Learning (CoNNL)*, 2009. 3.2.2

Sebastian Riedel and Andrew McCallum. Fast and robust joint models for biomedical event extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–12. Association for Computational Linguistics, 2011. 2.1, 2.1

Alexander M. Rush and Michael Collins. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362, 2012. 3.1, 3.1.1, 3.4

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics, 2010. 2.1, 2.1, 3.1, 3.1.1, 3.4, 6.1

Kenji Sagae. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, pages 81–84, Stroudsburg, USA, 2009. ACL. URL `http://dl.acm.org/citation.cfm?id=1697236.1697253`. 5.2.1

Rajhans Samdani, Ming-Wei Chang, and Dan Roth. Unified expectation maximization. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 688–698. Association for

Computational Linguistics, 2012. 2.2, 2.2

Evan Sandhaus. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752, 2008. (document), 4.6

Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007. 4.1

Peng Shi and Jimmy Lin. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*, 2019. 4.4

David Sontag, Amir Globerson, and Tommi Jaakkola. *Introduction to Dual Decomposition for Inference*. MIT Press, optimization in machine learning edition, January 2010. URL `https://www.microsoft.com/en-us/research/publication/introduction-to-dual-decomposition-for-inference/`. 2.1

Milan Straka. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium, October 2018. ACL. URL `http://www.aclweb.org/anthology/K18-2020`. 5.2.1, 5.5.2

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*, 2018. 2.3, 4.4

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems (NIPS)*, 2014. 3

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. Syntactic scaffolds for semantic structures. *CoRR*, abs/1808.10485, 2018. 2.3, 2, 4.4

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3: 29–41, 2015. 4.2.3

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. Deep semantic role labeling with self-attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 4, 4.4

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2.3

Oriol Vinyals, Luksz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *NIPS*, 2015. 3, 3.2.2, 3.2.2, 3.4

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Jess Kaufman, Michelle Franchini, Mohammed El Bachouti, Nianwen Xue, Martha Palmer, Jena D. Hwang, Claire Bonial, Jinho Choi, Aous Mansouri, Maha Foster, Abdel aaati Hawwary, Mitchell Marcus, Ann Taylor, Crag Greenberg, Eduard Hovy, Robert Belvin, and Ann Houston. Ontonotes release 5.0. In *LDC Catalog*, 2012. 6.2

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforce-

ment learning. *Machine Learning*, 8:229–256, 1992. 2.2, 3.1.2, 4.2.3

Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. In *Empirical Methods in Natural Language Processing*, pages 1296–1306, 2016. 3.4

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR, arXiv preprint arXiv:1609.08144*, 2016. 3.2.7

Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *CoRR*, abs/1304.5634, 2013. URL `http://arxiv.org/abs/1304.5634`. 5.1

Shipeng Yu, Balaji Krishnapuram, Rómer Rosales, and R Bharat Rao. Bayesian co-training. *Journal of Machine Learning Research*, 12(Sep):2649–2680, 2011. 5.2.2

Dan Zeman et al. Universal Dependencies 2.2 – CoNLL 2018 shared task development and test data, 2018a. URL `http://hdl.handle.net/11234/1-2184`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, `http://hdl.handle.net/11234/1-2184`. 5.1

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium, October 2018b. ACL. URL `http://www.aclweb.org/anthology/K18-2001`. 5.3.1

Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017. 5.1

Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1127–1137, 2015. 4.1, 4.4

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, 2013. (document), 4.6