Injecting constraints into neural NLP models

Jay Yoon Lee

November 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jaime Carbonell, Chair
William Cohen
Graham Neubig
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.

Abstract

The goal of this thesis proposal is injecting knowledge/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 simple rules necessary for well-formed structures unless there are immense amount of training data. The thesis proposes that not all the aspects of the model have to be learned from the data itself and injecting simple knowledge/constraints into the neural models can help low-resource tasks as well as improving state-of-the-art models.

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

Focusing on deterministic constraints on the output values, this thesis proposal first applies output constraints on inference time via proposed 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.

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

Lastly, the thesis proposes to extend the completed work to generalized span-based models and to domain adaptation where target domain is unlabeled. Moreover, the thesis proposal promises to explore additional methodology that might bring bigger gains through constraint injection compared to currently proposed approaches.

In the next sections, I provide brief description on the applications I worked with the constraints I injected per problem.

Contents

List of Tables

Chapter 1

Introduction

With the recent advances of 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) problems. However, even the best-performing models fail to satisfy simple constraint necessary for well-formed structures unless there is an immense amount of training data.

To this end, this thesis proposal raises a question 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 a black box function for modeling relation between an input $\mathbf{x}$ and output $\mathbf{y}$. The common approach is maximizing the conditional likelihood of $\mathbf{y}$ given $\mathbf{x}$ on training data. While this black box modeling approach can be beneficial in learning complex functions that we do not know how to model, the thesis proposes 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 proposal studies whether the injection of ancillary knowledge or constraints to the neural models lead to not only better satisfaction of constraints but also to better performance especially on, but not limited to, low-resource settings.

The goal of this proposed thesis is to provide systematic way of injecting knowledge/constraints into neural models and to show its efficacy, primarily for NLP tasks. The proposal focuses on NLP problems as the constraint satisfaction is an important aspect of the application. NLP often requires reasoning between multiple tasks and the multi-task setting can be expressed as a primary task and ancillary tasks or simple constraints to satisfy. Furthermore, in automated NLP system, model's output is often used as an input to the downstream tasks such as question answering and recommendation system and simple constraint violation can lead to unrecoverable errors for the downstream tasks.

To solve the constrained problem, cascaded system was often employed in the pre-neural era. For example, in semantic role labeling (SRL), the model has to identify the relationship between a verb predicate and 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. In the pre-neural era, a typical SRL system would first output a set of possible spans using a syntactic parser and then sequentially find most likely labels on provided spans. Nonetheless, thanks to representation power of neural network, end-to-end learning showed the best efficacy: system outputting the spans and labels simultaneously in case of SRL.

Most current state-of-the-art models in NLP are learned in end-to-end fashion. However, as irony 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 (set A) that they have to be within the the parsing constituents (set B), i.e. $A \subset B$. 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 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. Even so, can we actually learn more from the fact that the neural representation is not even satisfying simple constraint? The thesis proposal further asks whether the simple measure of constraint violation can be an indicator of how bad a model is and whether this signal can be leveraged to learn a better model.

To address the question, the thesis proposal focuses on the structural knowledge of the output space as a constraint and use 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, in the proposed work [20, 24]. The first benefit in focusing on the knowledge of output space is that it is intuitive as we can directly enforce logical/linguistic constraints. Another advantage of structural knowledge on output space is that it often does not require labeled dataset to evaluate. Leveraging this advantage, I enforce the structural constraints to improve inference procedure using the proposed gradient-based inference (GBI) method, to improve low-resource models via semi-supervised learning and also to improve on domain adaptation where the target domain is unlabeled.

1.1 Organization of the proposal

The organization of the thesis proposal is provided as follows. First, chapter 2 illustrates applications and constraints used to address the proposed constraint injection methods. Part I (chapter 3, 4) introduces the completed work. Chapter 3 presents an inference algorithm, Gradient-Based Inference (GBI), that can utilize the provided constraint function. In chapter 4, going beyond structured inference, the thesis proposal shows that the constraint injection can be used on training time as well. This chapter further proposes semi-supervised learning framework and shows significant improvement on low-resource settings. Part II (chapter 5, 6) provides potential new directions for completion of the thesis. Part II proposes to *explore* new applications (chapter 5) and new methods (chapter 6) with provided time schedule at the beginning.

Chapter 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.

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 "it is really like this, just look at the bus sign" the goal is to tag the two-argument verb "is" as the predicate, "it" as its first argument and the prepositional phrase "like this" as its second 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 representing both the regimentation of tokens into contiguous segments and the semantic role of the those segments.

$\langle$" So it 's a very mixed bag . "$\rangle \longrightarrow$ `sssr!ssssrr!srrr!rr!ssrrrrrr!`

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 2.1: A shift-reduce example for which the method successfully enforces constraints. 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.

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. 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. [1] To continue with our example, the original SRL sequence-labeling might incorrectly label "really like this" as the second argument rather than "like this." Since according to the parse tree "really" is part of the verb phrase, thus while the tree contains the spans "is really like this" and "like this" it does not contain the span "really like this." The hope is that enforcing the BIO labeling to agree with the actual parse spans would benefit SRL. Indeed, we notice that this is indeed the case, and that our hypothetical example is actually a real data-case from our experiments, which we describe later.

2.2 Syntactic parsing

As a second illustrative example, consider a 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 seq2seq model [38]. Let us suppose we linearize the tree using a sequence of shift (`s`) and reduce (`r,r!`) commands that control an implicit shift reduce parser. Intuitively, 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 and the interpretation of the symbol `r` is that we start (or continue) reducing (popping) the top elements of the stack, 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 type-free version of the parse tree `(S (NP the ball) (VP is (NP red)))` for the input sentence "the ball is red". 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).

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 $m_{\mathbf{x}}$, 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 loss $g(\mathbf{y}, \mathcal{L}^x)$ for this task simply counts the errors of each of the three types.

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

[1] Note that even when using ground-truth parse-trees for side-information that the ground-truth SRL spans might not agree with the ground-truth parse-spans.

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 [38] 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 [32] 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*.

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 simple artificial languages 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 [12]. 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 problem-specific constraints that may arise from the application of T on the input sentence. For example, if the source language $\mathcal{L}_S$ lacks the symbol $\mathtt{a}$ and the transducer transforms certain patterns in the source language to aaa, then it must be the case that the number of $\mathtt{a}$'s in the target sequence is a multiple of three. If the network is unable to learn such constraints from the training data, then our method is applicable and we can evaluate its performance for such simple cases.

Part I

Completed work

Chapter 3

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

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: sequence transduction, syntactic parsing, and semantic role labeling. In each case, the algorithm not only satisfies constraints, but improves accuracy, even when the underlying network is state-of-the-art.

3.1 Problem definition

Suppose we have trained a sequence-to-sequence (seq2seq) network [6, 19, 36] to perform a structured prediction task such as syntactic constituency parsing [38]. 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 down-stream 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 (SOTA) 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. Though, 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, by iteratively adjusting 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 [5], but does so in a way to satisfy the constraints.

The paper 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 outline, this chapter presents the following points for the thesis proposal:

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 a SOTA network by 1.03 F1 [26]. This improvement happens to surpass a $\mathtt{A}^*$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.

3.2 Constraint-aware inference in neural networks

Our goal is to design 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 [17, 33, 34], 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 revolves around Lagrangian relaxation, we emphasize that the purpose is merely to provide intuition and motivate design choices.

3.2.1 Problem definition and motivation

Typically, a neural network parameterized by weights W is a function from an input $\mathbf{x}$ to an output $\mathbf{y}$. The network has an associated compatibility function $\Psi(\mathbf{y}; \mathbf{x}, W) \rightarrow \mathbb{R}_+$ that measures how likely an output $\mathbf{y}$ is given an input $\mathbf{x}$ under weights W. 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 interested in the following optimization problem:

$$\begin{aligned} \max_{\mathbf{y}} \quad & \Psi(\mathbf{x}, \mathbf{y}, W) \\ \text{s.t.} \quad & \mathbf{y} \in \mathcal{L}^{\mathbf{x}} \end{aligned} \tag{3.1}$$

Simple greedy inference are no longer sufficient since the outputs might violate the global constraints (i.e., $\mathbf{y} \notin \mathcal{L}^{\mathbf{x}}$). Instead, suppose we had a function $g(\mathbf{y}, \mathcal{L}) \rightarrow \mathbb{R}_0^+$ that measures a loss between output $\mathbf{y}$ and a grammar $\mathcal{L}$ such that $g(\mathbf{y}, \mathcal{L}) = 0$ if and only if there are no grammatical errors 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 [22]. We could then express the constraints as an equality constraint and minimize the Lagrangian:

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

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

Instead, we might circumvent these issues if we optimize over a 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, observe that the network's weights 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. 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.

3.2.2 Algorithm

Instead of solving the aforementioned impractical optimization problem, we propose to optimize a "dual" set of model parameters W_λ over the constraint function while regularizing W_λ to stay close to the originally learned weights W. The objective function is as follows:

$$\begin{aligned} \min_{W_\lambda} \quad & \Psi(\mathbf{x}, \hat{\mathbf{y}}, W_\lambda) g(\hat{\mathbf{y}}, \mathcal{L}) + \alpha \|W - W_\lambda\|_2 \\ \text{where} \quad & \hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} \Psi(\mathbf{x}, \mathbf{y}, W_\lambda) \end{aligned} \tag{3.3}$$

Although this objective deviates from the original optimization problem, it is reasonable because by definition of the constraint loss $g(\cdot)$, the global minima must correspond to outputs that satisfy all constraints. Further, we expect to find high-probability optima if we initialize $W_\lambda = W$. 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. Crucially, the optimization problem yields computationally efficient subroutines that we exploit in the optimization algorithm.

To optimize the objective, the algorithm alternates maximization to find $\hat{\mathbf{y}}$ and minimization w.r.t. W_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 loss g. Then, given a fixed $\hat{\mathbf{y}}$, we minimize the objective with respect to the W_λ by performing stochastic gradient descent (SGD). Since $\hat{\mathbf{y}}$ is fixed, the constraint loss term becomes a constant in the gradient; thus, making it easier to employ external black-box constraint losses (such as those based on compilers) that may not be differentiable. As a remark, note the similarity to REINFORCE [39]: 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. Furthermore, GBI also works on sequence-tagging problem, SRL (Section 3.3.1), where next output does not depend on the current output, which is far from REINFORCE setting.

Algorithm 1 Constrained inference for neural nets

```
Inputs: test instance $\mathbf{x}$, input specific CFL $\mathcal{L}^{\mathbf{x}}$, pretrained weights $W$
$W_\lambda \leftarrow W$ #reset instance-specific weights
**while** not converged **do**
    $\mathbf{y} \leftarrow f(\mathbf{x}; W_\lambda)$ #perform inference using weights $W_\lambda$
    $\nabla \leftarrow g(\mathbf{y}, \mathcal{L}^{\mathbf{x}}) \frac{\partial}{\partial W_\lambda} \Psi(\mathbf{x}, \mathbf{y}, W_\lambda) + \alpha \frac{W - W_\lambda}{\|W - W_\lambda\|_2}$ #compute constraint loss
    $W_\lambda \leftarrow W_\lambda - \eta \nabla$ #update instance-specific weights with SGD or a variant thereof
**end while**
```

3.3 Experiments

In this section we study our algorithm on three different tasks: SRL, syntactic constituency parsing and a synthetic sequence transduction task. All three tasks require hard constraints, but they play

Network	Failure rate(%)	Inference	Conv rate(%)	Failure set						Test set	
				Average (%) Disagreement		F1		Exact Match (%)		F1	
				before	after	before	after	before	after	before	after
SRL-100	9.82	GBI	42.25	44.85	24.92	48.00	59.70 (+11.7)	0.0	19.90	84.40	85.63 (+1.23)
		`A*`	40.40		33.91		48.83 (+0.83)		13.79		84.51 (+0.11)
SRL-70	10.54	GBI	46.22	45.54	23.02	47.81	59.37 (+11.56)	0.0	19.57	83.55	84.83 (+1.28)
		`A*`	44.42		32.32		50.49 (+2.68)		16.12		83.90 (+0.35)
SRL-40	11.06	GBI	47.89	45.71	22.42	46.53	58.83 (+12.3)	0.0	19.45	82.57	84.03 (+1.46)
		`A*`	44.74		32.17		46.53 (+2.88)		15.15		82.98 (+0.41)
SRL-10	14.15	GBI	44.28	47.14	24.88	44.19	54.78 (+10.59)	0.0	15.28	78.56	80.18 (+1.62)
		`A*`	43.66		32.80		45.93 (+1.74)		12.28		78.87 (+0.31)
SRL-1	21.90	GBI	52.85	50.38	21.45	37.90	49.00 (+11.10)	0.0	12.83	67.28	69.97 (+2.69)
		`A*`	48.96		30.28		41.59 (+3.69)		11.25		67.97 (+0.69)

Table 3.1: Comparison of the GBI vs. `A*`inference procedure for SRL. We report avg. disagreement rate, F1-scores and exact match for *failure set* (columns 5-10) and F1-score for whole test set (last 2 columns). Also we report performances on 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
	BS-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 3.2: Parsing Networks with various performances (BS-9 means beam size 9). `Net1,2` are trained on GNMT seq2seq model whereas `Net3-5` are trained on lower-resource and simpler seq2seq model to test GBI on wide range of model performances.

Net	Failure (n/2415)	Conv rate	F1 (Failure set)		F1 (whole test)	
			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 3.3: Evaluation of the GBI on syntactic parsing using GNMT seq2seq. Note that GBI without beam search performs higher than BS-9 on Table 3.2.

a different role in each. In the sequence transduction tasks 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 . We provide an example-based case studies per application in Appendix .1.

We are interested in answering the following questions (Q1) how well does the neural network learn the constraints from data (Q2) for cases in which the network is unable to learn the constraints perfect, can GBI actually enforce the constraints (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, and (Q5) does GBI also work on 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 various proposed questions, we define some terminologies 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.

3.3.1 Semantic Role Labeling

We employ the SOTA SRL network with ELMo embeddings, which is essentially a multi-layer highway bi-LSTM that produces BIO output predictions for each input token [26]. For data we use OntoNotes v5.0, which has ground-truth for both SRL and syntactic parsing [29]. We evaluate GBI on the test-set (25.6k examples), out of which consistent parse information is available for 81.25% examples (we only include side-information in terms of constraints for this subset).

We repeat our same experimental procedure as before, but for this task and also train multiple networks, SRL-X, with varying amounts (X%) of training dataset. We train our network at test time for max 10 iterations and use SGD with learning rate of 0.01. From Table 3.1, 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 SOTA network (SRL-100) that does not incorporate the constraints (they report 84.6 F1, we obtain a similar 84.4 F1 with their network, and achieve 85.63 after enforcing constraints with our inference). Further, 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 avg. 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 3.1, we can see that by applying GBI on SRL-100, the avg. disagreement rate on 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 are similar to those we observe in the parsing task, and provide additional evidence for answering Q1-3 favorably.

To enforce constraints during inference, He et al. proposed to employ constrained-$\mathtt{A}^*$decoding. For the sake of fair comparison with GBI, we consider $\mathtt{A}^*$decoding, as used in [13] and report results for SRL-X networks. We see from Table 3.1, that GBI procedure consistently out-performs $\mathtt{A}^*$decoding on all evaluation metrics, thus, proving superiority of our approach. Additionally, in Appendix .2, we introduce additional case study of noisy constraint where GBI shows its superiority compared $\mathtt{A}^*$again.

3.3.2 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 Section 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) [10]. 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[1] 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).

In order to study our algorithm on a wide range of accuracy regimes (section 3.3.4), 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 (`Net1,2`) using a highly competitive seq2seq architecture for machine translation, GNMT [41] with F1 scores, 86.78 and 87.33, respectively. And, to study wider range of accuracies, we train a simpler architecture with different hyper parameters and obtain nets (Net3-5). For all models, we employ Glorot initialization, and basic attention [1]. See Table 3.2 for a summary of the networks, hyper-parameters, and their performance.

We report the behavior of the constraint-satisfaction method on Table 3.3 for Net1-2, and on Table 3.4 for Net3-5. Across all the experimental conditions (Table 3.3, 3.4), the conversion

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

Net	Infer method	Failure (n/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 3.4: Evaluation of the GBI on simpler, low-resource seq2seq networks. In here, we also evaluate whether GBI can be used in combination with different inference techniques: greedy and beam search of various width.

rates are high, often above 80 and sometimes above 90 supporting Q2. Note that beam search can also increase constraint satisfaction with conversion rate going as high as 51.74% (164/317) in case of Net3 with beam size 9. However, as the quality of the model increases, the conversion rate became minuscule; in case of Net1,2 the conversion rate was less than 14% with beam 9; Net1 converted 26 out of 187 and Net2 converted 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 this 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 failure-set from 73.54 to 79.68 F1, resulting in an overall improvement from 86.54 to 87.57 F1 (entire test-set).

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%.

3.3.3 Simple Transduction Experiment

In our final experiment we focus on a simple sequence transduction task in which we find that despite learning the training data perfectly, the network fails to learn the constraint in a way that generalizes to the test set.

For our task, we choose a simple transducer, similar to those studied in recent work [12]. The source language $\mathcal{L}_S$ is `(az|bz)*` and the target language $\mathcal{L}_T$ is `(aaa|zb)*`. The transducer is

	Failure rate (%)	Conversion rate (%)	F1 on failure set	
			before	after
Genre \Task	***Syntactic Parsing***			
Broadcast Conversation (BC)	19.3	98.8	56.4	59.0 (+2.6)
Broadcast News (BN)	11.7	98.1	63.2	68.8 (+5.6)
Pivot Corpus (PT)	9.8	97.8	71.4	75.8 (+4.4)
Telephone Conversation (TC)	10.1	86.2	56.9	57.6. (+0.7)
Weblogs (WB)	17.6	95.3	62.0	63.2 (+1.2)
Genre \Task	***SRL***			
Broadcast Conversation (BC)	26.86	53.88	39.72	52.4 (+12.68)
Broadcast News (BN)	18.51	55.19	39.28	50.58 (+11.3)
Pivot Corpus (PT)	10.01	62.34	47.19	63.69 (+16.5)
Telephone Conversation (TC)	19.09	54.62	47.7	58.04 (+10.34)
Weblogs (WB)	20.32	44.13	47.6	57.39 (+9.39)

Table 3.5: Evaluation of GBI method on out-of-domain data on the applications of syntactic parser and SRL. F1 scores are reported on the *failure set*. SRL model was trained on NW and the syntactic parser was trained on WSJ section , which is subsection of NW, on OntoNote v5.0. Except PT, which is new and old Testament, all failure rate on out-domain data is higher than that of in-domain (11.9% for parsing and 18.1% for SRL) as suspected. The table shows that GBI can be successfully applied to resolve performance degradation on out-of-domain data.

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, $T(\texttt{bzazbz}) \mapsto \texttt{zbaaazb}$. The training set comprises 1934 sequences of length 2–20 and the test set contain sentences of lengths 21-24. We employ shorter sentences for training to require 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). 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.

3.3.4 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 study GBI on high quality reference 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 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. Similarly, for *syntactic parsing*, we train additional low-quality models `Net`3-5 with a simpler uni-directional encoders/decoders, and on different training data portion of 25%, 75%, and 100%. (Table 3.2). We evaluate GBI on each of them in Table 3.1, 3.4 and find further evidence in support of favorable answers to Q2 (satisfying constraints) and Q3 (improving F1 accuracy) by favorably answering Q4. Moreover, while not reported fully due to page limits, we examined both tasks over 20 experiments with different baseline networks in combination to different inference strategies, and we found GBI favorable in all but one case (but by just 0.04 compared to employing post-processing).

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 3.4, 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 on beam search inference.

3.3.5 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 actual out-of-domain data where model quality can suffer. For SRL, we train SOTA network with ELMo embedding on NewsWire (NW) section of OntoNotes v5.0 English PropBank corpus and then test on the additional genres provided in the PropBank 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 3.5, the failure rate for NW trained SRL network in general is higher for out-of-genre data with highest being 26.86% for BC(vs. 18.10% NW). Further, by enforcing constraints, we see significant gains on failure set in terms of F1 score across all genres (ranging from 9.39-16.5 F1), thus, providing additional evidences for answering Q5.

Similar to SRL, we train a GMNT seq2seq model on the Wall Street Journal of NW section in OntoNotes v5.0 Treebank which shares the same genre classification with PropBank. [2] The F1 on the within-genre data (test set of WSJ) is 85.03, but the F1 on these genres is much lower, ranging from the mid-forties on BC (46.2–78.5 depending on the subcategory) to the low-eighties on BN (68.3–81.3. depending on the subcategory). Indeed, we find that overall the F1 is lower

[2]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.

and in some cases, like WB, the failure rate is much higher (17.6% for WB *vs.* 11.9% for WSJ). Following the same experimental protocol as on the PTB data, we report the results in Table 3.5 (aggregating over all subcategories in each genre). 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. Again, we find support for Q2, Q3 and Q5.

3.4 Related work

Recent work has considered applying neural networks to structured prediction; for example, structured prediction energy networks (SPENs) [3]. 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 [15]. 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 $\texttt{A}^*$decoding [13] or integer linear programming [30]. For parsing, previous work in enforcing hard constraints has focused on post-processing [38] or building them into the decoder via sampling [9] or search constraints [40].

Finally, as previously mentioned, our method highly resembles dual decomposition and more generally Lagrangian relaxation for structured prediction [17, 33, 34]. 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. 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 SOTA SRL, 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.

Chapter 4

Learning with Output Constraints

Neural models have shown several state-of-the-art performances on Semantic Role Labeling (SRL). However, the neural models require an immense amount of semantic-role corpora and are thus not well suited for low-resource languages or domains. The paper proposes a semi-supervised semantic role labeling method that outperforms the state-of-the-art in limited SRL training corpora. The method is based on explicitly enforcing syntactic constraints by augmenting the training objective with a syntactic-inconsistency loss component and uses SRL-unlabeled instances to train a joint-objective LSTM. On CoNLL-2012 English section, the proposed semi-supervised training with 1%, 10% SRL-labeled data and varying amounts of SRL-unlabeled data achieves +1.58, +0.78 F1, respectively, over the pre-trained models that were trained on SOTA architecture with ELMo on the same SRL-labeled data. Additionally, by using the syntactic-inconsistency loss on inference time, the proposed model achieves +3.67, +2.1 F1 over pre-trained model on 1%, 10% SRL-labeled data, respectively.

4.1 Problem definition

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 [35], information extraction [2], machine translation, etc.

Several SRL systems relying on large annotated corpora have been proposed [13, 27], 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 built using large annotated corpora of verb predicates and their arguments provided as part of the PropBank and OntoNotes v5.0 projects [16, 29]. These corpora are built by adding semantic role annotations to the constituents of previously-annotated syntactic parse trees in the Penn Treebank [23]. Traditionally, SRL relies heavily on using syntactic parse trees either from shallow syntactic parsers (chunkers) or full syntactic parsers and Punyakanok et al. shows significant improvements by using syntactic parse trees.

Recent breakthroughs motivated by end-to-end deep learning techniques [13, 42] 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 implied 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 state-of-the-art SRL system by [13, 27], where we formulate 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 implied by syntactic parse) violations in our training objective. In other words, while training our model, we enforce the outputs of our system to agree with the spans implied by the syntactic parse of the sentence as much as possible. In summary, our contributions to low-resource SRL are:

1. A novel formulation which leverages syntactic parse trees for SRL by introducing them as hard constraints while training the model.
2. Experiments with varying amounts of SRL-unlabeled data that point towards semi-supervised learning for low-resource SRL by leveraging the fact that syntactic inconsistency loss does not require labels.

4.2 Proposed Approach

We build upon an existing deep-learning approach to SRL [13]. First we revisit definitions introduced by [13] and then discuss about our formulation.

4.2.1 Task definition

Given a sentence-predicate pair $(\mathbf{x}, v)$, SRL task is defined as predicting a sequence of tags $\mathbf{y}$, where each y_i belongs to a set of BIO tags (Ω). So, for an argument span with semantic role ARG$_i$, B-ARG$_i$ tag indicates that the corresponding token marks the beginning of the argument span and I-ARG$_i$ tag indicates that the corresponding token is inside of the argument span and 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/unlabeled data refers to sentence-predicate pairs with/without gold SRL tags.

4.2.2 State-of-the-Art (SOTA) Model

He et al. 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, the conditional negative log-likelihood $\mathcal{L}(\mathbf{w})$ of a sample of training data $\mathcal{T} = \{\mathbf{x}^{(t)}, \mathbf{y}^{(t)}\}_{\forall t}$ is minimized, where $\mathcal{L}(\mathbf{w})$ is

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

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, [13] searches for the best $\hat{\mathbf{y}}$ over the space of all possibilities (Ω^n) using the scoring function $f(x, y)$, which incorporates log probability and structural penalty terms. The details of scoring function is on Appendix Eq.(7).

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

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: unique core roles (U), continuation roles (C) and reference roles (R) [30]. Lastly, syntactic constraints state that *srl-spans*$(\mathbf{y})$ have to be subset of *parse-spans*$(\mathbf{x})$.

[13] 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, [13] achieves up to $+2$ F1 score on CoNLL-2005 dataset via A* decoding.

Improvements of SRL system via use of syntactic constraints is consistent with other observations [30]. However, all previous works 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*$(\mathbf{x}, \mathbf{y}) = \{span_i \in \text{srl-spans}(\mathbf{y}) \mid span_i \notin \text{parse-spans}(\mathbf{x})\}$. 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:

$$g(\mathbf{x}, \mathbf{y}) = \frac{|\text{disagreeing-spans}(\mathbf{x}, \mathbf{y})|}{|\text{srl-spans}(\mathbf{y})|} \tag{4.3}$$

$$s(\mathbf{x}, \mathbf{y}) = 2 \times g(\mathbf{x}, \mathbf{y}) - 1 \tag{4.4}$$

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 $g(\mathbf{x}, \hat{\mathbf{y}}^{(t)}) = 0$ or $s(\mathbf{x}, \hat{\mathbf{y}}^{(t)}) = 1$. We define a loss component due to syntactic inconsistency (*SI-Loss*) as follows:

$$\text{SI-Loss} = s(\mathbf{x}, \hat{\mathbf{y}}^{(t)}) \sum_{i=1}^{|\hat{\mathbf{y}}^{(t)}|} \log p(\hat{y}_i^{(t)} | \mathbf{x}; \mathbf{w}^{(t)}) \tag{4.5}$$

During training, we want to minimize SI-Loss.

4.2.4 Training with Joint Objective

Based on Eq.(4.1), a supervised loss, and Eq.(4.5), 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 training objective (at epoch t) is defined as:

$$\text{Joint loss} = -\alpha_1 \sum_{i=1}^{|\mathbf{y}|} \log p(y_i | \mathbf{x}; \mathbf{w}) + \alpha_2 \, s(\mathbf{x}, \hat{\mathbf{y}}^{(t)}) \sum_{i=1}^{|\hat{\mathbf{y}}^{(t)}|} \log p(\hat{y}_i^{(t)} | \mathbf{x}; \mathbf{w}^{(t)}) \tag{4.6}$$

Here, α_1 and α_2 are weights (hyper-parameters) 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 loss.

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.(4.5), 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 in our SSL formulation to compute SI-loss from SRL-unlabeled sentences.

Let *sup-s* be a batch of SRL-labeled sentences and *usup-s* be a batch SRL-unlabeled sentences only with parse information. In SSL setup, we propose to train our model with joint objective where *sup-s* only contributes to supervised loss Eq.(4.1) and *unsup-s* contributes in terms of syntactic inconsistency objective Eq.(4.5) and combine them according to Eq.(4.6) to train them with joint loss.

4.3 Experiments

4.3.1 Dataset

We evaluate our model's performance on span-based SRL dataset from CoNLL-2012 shared task [29]. This dataset contains gold predicates as part of the input sentence and also gold parse information corresponding to each sentence which we use for defining hard constraints for

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 joint objective (**J**).

Base Model/ Legend	Test F1	Average disagreement rate (%)
B10	78.56	17.01
B10-SI1x	78.84 (+0.28)	16.17 (−4.94%)
B10-SI5x	78.67 (+0.11)	16.47 (−3.17%)
B10-SI10x	78.76 (+0.20)	16.09 (−5.4%)
B1	67.28	21.17
B1-SI1x	67.67 (+0.39)	20.14 (−4.87%)
B1-SI5x	67.74 (+0.46)	19.93 (−5.86%)
B1-SI10x	67.71 (+0.43)	20.16 (−4.77%)

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).

SRL task. We use standard train/development/test split containing 278K/38.3K/25.6K sentences. Further, there is approx. 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. [1]

4.3.2 Model configurations

For the SOTA system proposed in [27], we use code from Allen AI[2] 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 and joint training objective, respectively. B$\underline{X}$-SI$\underline{U}$x and B$\underline{X}$-J$\underline{U}$x denote model trained with SI-loss and Joint loss, 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. To satisfy BIO constraints, we run Viterbi decoding by default for inference.

[1]We preprocessed data for inference by simply deleting syntactic parse trees for the sentences where we have disagreement and perform standard Viterbi decoding for those sentences and note that this preprocessing scheme was never used during training.

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

Base Model/ Legend	Test F1	Average disagreement rate (%)
B10	78.56	17.01
B10-further	78.86 (+0.3)	16.25 (-2.06%)
B10-J1x	79.23 (+0.67)	16.03 (-5.76%)
B10-J5x	79.25 (+0.69)	16.01 (-5.88%)
B10-J10x	**79.34 (+0.78)**	**15.88 (-6.64%)**
B1	67.28	21.17
B1-further	67.76 (+0.48)	20.75 (-1.98%)
B1-J1x	68.45 (+1.17)	19.57 (-7.56%)
B1-J5x	**68.86 (+1.58)**	**19.38 (-8.46%)**
B1-J10x	68.61 (+1.33)	19.29 (-8.88%)

Table 4.3: Training with joint objective (**J**) 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.

4.3.3 Results

We are interested in answering following questions. (Q1) how well 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, and lastly (Q5) what is the difference in using the syntactic constraints in training time compared to using it at decoding time. To answer (Q1-2) favorably we report average disagreement rate computed over test split. To answer (Q3-5), we report overall F1-scores on CoNLL-2012 test set (using standard evaluation script). For experiments using SRL-unlabeled data, we report average results after running multiple experiments with different random samples of it.

Does training with joint objective help? We trained 3 models with random 1%, 10% and whole 100% of the training set with joint objective ($\alpha_1 = \alpha_2 = 0.5$). For comparison, we trained 3 SOTA models with the same training sets. All models were trained for max 150 epochs and with a patience of 20 epochs. Table 4.1 reports the results of this experiment. 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 (Q1-3) 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 pre-trained model with two objectives in SSL set up: (a) SI-loss (Table 4.2) and (b) joint objective (Table 4.3)

For experiment (a), we use square loss, $\beta\|W - W_{\text{pre-train}}\|^2$ regularizer to keep the model W close to the pre-trained model $W_{\text{pre-train}}$ to avoid catastrophic forgetting (β set to 0.005). We

Decoding	B10	B10-J10x	B1	B1-J5x
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 4.4: Comparison of different decoding techniques: Viterbi, A* [13] and gradient based inference [20] with noisy and noise-free syntactic constraints. Note that the (+/-) F1 are reported w.r.t Viterbi decoding on the same column.

optimize with SGD with learning rate of 0.01, $\alpha_2 = 1.0$, patience of 10 epochs. We see that with SI-loss improvements are significant in terms of average disagreement rate as compared to F1.

For experiment (b), we train B1 and B10 with joint objective in SSL set-up (as discussed in Section 4.2.5). We use SGD with learning rate of 0.05, $\alpha_1 = \alpha_2 = 1.0$ and patience of 10 epochs. We report +1.58 F1 and +0.78 F1 improvement over B1 and B10, trained with 5% and 100% SRL-unlabeled data, respectively. Note that we cannot achieve these improvements with simply fine-tunning B$\underline{X}$ with supervised loss, as seen with B$\underline{X}$-further on Table 4.3. This provides evidence to answer (Q4) favorably. In general, the performance gains increase as the size of the SRL-unlabeled data increases.

Is it better to enforce syntactic consistency on decoding or on 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 [13] and gradient-based inference [20], which optimizes loss function similar to SI-loss on Eq.(4.5) per example basis. 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 4.4) when syntactic consistency is enforced both on training and inference steps, +3.67, +2.1 F1 score improvement over B1 and B10 respectively, and we conclude that the effort of enforcing syntactic consistency on inference time is complementary to the same effort on training time. However, note that the overall performance increases as the benefit from enforcing syntactic consistency with SSL is far greater compared to marginal decrement on structured prediction.
While syntactic constraints help both train and inference, injecting constraints on train time is far more robust compared to enforcing them on decoding time. The performance of the structured prediction drops rapidly when the noise in the parse information is introduced (x column of Table 4.4). On the other hand, SSL was trained on CoNLL2012 data where about 10% of the gold SRL-spans do not match with gold parse-spans and even when we increase noise level to 20% the

performance drop was only around 0.1 F1 score.

4.4 Related Work

The traditional approaches for SRL [18, 28] constituted of cascaded system with four subtasks: pruning, argument identification, argument labeling, and inference. Recent approaches [13, 42] proposed end-to-end system for SRL using deep recurrent or bi-LSTM-based architecture with no syntactic inputs and have achieved SOTA results on English SRL. Lastly, [27] proposed ELMo, a deep contextualized word representations, and improved the SOTA English SRL by 3.2 F1-points.

Even on the end-to-end learning, inference still remains as a separate subtask and would be formalized as a constrained optimization problem. To solve this problem ILP [30], *A* algorithm* [13] and gradient-based inference [20] were employed. Further, all of these works leveraged syntactic parse during inference and was never used during training unless used as a cascaded system.

To the best of our knowledge, this work is the first attempt towards SSL span-based SRL model. Nonetheless, there were few efforts in SSL in dependency-based SRL systems [7, 8, 11]. [11] proposed to augment the dataset by finding similar unlabeled sentences to already labeled set and annotate accordingly. While interesting, the similar augmentation technique is harder to apply to span-based SRL as one requires to annotate the whole span. [7, 8] proposed to leverage the relation between words by learning latent word distribution over the context, i.e. language model. Our paper also incorporates this idea by using ELMo as it is trained via language model objective.

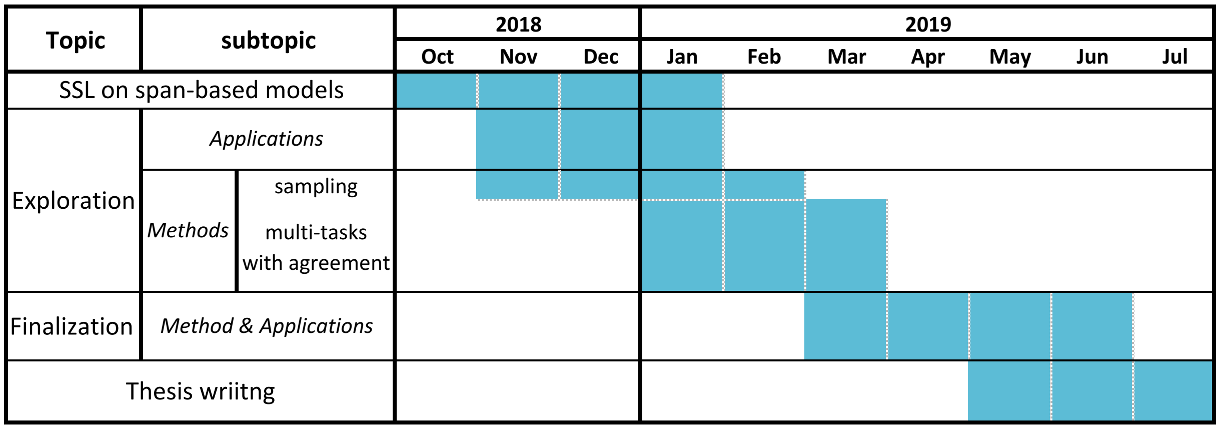
4.5 Conclusion and Future Work

We presented a SI-loss to enforce SRL systems to produce syntactically consistent outputs. Further, 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.

Part II

Proposed additional work

The following chapters illustrates the set of proposed work which are in exploration for the completion of the thesis. While the details of the proposed applications (chapter 5) and proposed methods (chapter 6) are described in the following chapters, the schedule for proposed additional work is as follows:



Topic	subtopic		2018			2019						
			Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
SSL on span-based models			🟦	🟦	🟦	🟦						
Exploration	*Applications*			🟦	🟦	🟦						
	Methods	sampling		🟦	🟦	🟦	🟦					
		multi-tasks with agreement				🟦	🟦	🟦				
Finalization	*Method & Applications*							🟦	🟦	🟦	🟦	
Thesis wriitng										🟦	🟦	🟦

Figure 4.1: Time table for the proposed additional work.

Chapter 5

Additional applications

5.1 Domain adaptation using output constraints

Machine Learning models do not perform as well on out-of-domain dataset compared to the domain it was trained on due to the difference on underlying distribution. We have observed such behaviors on neural models as well on earlier chapter. As 3.5 shows, most of out-of-domain dataset performs worse on major metric and have higher failure rate. To remedy this problem, the proposed work 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 the proposed work, we explore whether the proposed SSL approach 4 can remedy the difference of data distribution and we expect a synergy between the structured prediction and SSL approach as we have seen on 4.4, i.e. we expect the combined benefit to be larger than individual benefits.

5.1.1 Preliminary Experiments

For the proposed work, I have conducted preliminary experiment on the CoNLL2012 dataset. The experiment setting is similar to 3.5. We define source domain as NewsWire (NW) section of OntoNotes v5.0 English PropBank corpus, and additional genres provided in the PropBank corpus (BC, BN, PT, TC, WB) as the target domain.

First, we examine how training with structural loss 4.5 can help domain adaptation in case of syntactic parsing. We pre-trained GNMT model for syntactic parsing with on NW section and then trained further with structural loss 4.5 on the target domains. As table 5.1 shows, the structural loss training on target domain both reduces the failure rate and improves the performance.

Secone, we examine how SSL training with joint objective eq. 4.6 can aid domain adaptation in case of SRL. We pre-trained SRL network with ELMo embedding on NW corpus and then perform SSL approach using the source domain as a supervised set and target domain as a unsupervised set on our joint loss formulation eq. 4.6. As table 5.2 shows, the SSL training with joint objective both reduces the failure rate and improves the performance except the Telephone Conversation.

The remaining works to be done on this proposed additional work is a comparison with

	Failure rate (%)		F1	
	before	after	before	after
Genre \Task	***Syntactic Parsing***			
Broadcast Conversation (BC)	19.3	11.2	66.93	74.23 (+7.3)
Broadcast News (BN)	11.7	10.6	78.77	79.04 (+0.27)
Pivot Corpus (PT)	9.8	6.6	85.97	87.06 (+1.09)
Telephone Conversation (TC)	10.0	9.5	66.06	66.13 (+0.07)
Weblogs (WB)	17.4	15.3	74.55	76.45 (+1.9)

Table 5.1: 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.

	Failure rate (%)		F1	
	before	after	before	after
Genre \Task	***SRL***			
Broadcast Conversation (BC)	26.86	19.70	68.05	71.17 (+3.12)
Broadcast News (BN)	18.51	55.19	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 5.2: Evaluation of domain adaptation on SRL using semi-supervised learning approach. F1 scores are reported on the whole test set per domain. The initial reference model trained on 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.

possible alternatives such as self-training and thorough analysis. While not included in this proposal as the comparison was not rigorously done, initial comparison with self-training on syntactic parsing indicates that while self-training shows mediocre improvement across genres, the constraint injection shows significant improvements on specific genres (BC,WB) and par with self-training on other genres. As the improvement is not uniform across different genres, interesting question left is analyzing when the constraint injection works best as a tool of domain adaptation and when we should not use it.

5.2 Extending Semi-Supervised approach to general span-based models.

There are multiple span-based tasks: SRL, co-reference resolution, and Shallow Discourse Relation (SDR) problems. Traditional approaches have resorted to cascaded systems where the syntactic parsers produce candidate spans, however, with the advent of the deep models, SOTA models are trained without particular information of syntactic parse tree. For one case of span-based model (SRL), we have showed that injecting the knowledge of syntactic parse tree can still enhance the span level accuracy as well as the overall accuracy in chapter 4. As we are not assuming any task-specific features in injecting the information of syntactic parse tree in the application of SRL, we believe this approach can be naturally extended to other span-based models such as co-reference resolution, FrameNet SRL, and SDR.

The challenge remaining in this proposed work is that there has been newer architectures introduced in SOTA model in SRL [14] and co-reference resolution [21]which shares the same core span-based neural representation. These work improve previous approach [13], which our completed work also used as a reference model, by considering all possible spans as compared to previous BIO sequence representation. As the new model have higher capacity compared to the reference model this thesis previously studied, it is a non-trivial task to incorporate the syntactic parse tree information into the new span-based models.

While the new SOTA model considers much richer span information, the model still only train on the cross entropy function of gold labels. With this fact and based on our initial analysis on the SOTA model's span-level accuracy, we believe the new span-based model can also benefit from the proposed parse tree injection. We will examine whether the proposed approach can bring improvement on both low-resource and full=resource models as completed work did on SRL. Although challenging, successful incorporation of the structural constraint to a different span-based neural architecture will further generalize the proposed parse tree injection method on chapter 4.

Chapter 6

Exploratory: New methods

The completed work performed knowledge injection to the model through the constraint violation signals acquired form the unlabeled dataset, previously unseen on the initial training period. While the completed work on semi-supervised learning showed efficacy on low-resource, full-resource and preliminary results on domain adaptation, the model did not learn about constraints from the primary training set itself. To be more precise, the completed work depended on a pre-trained reference model. Given a pre-trained model as a starting point, I further trained the model to satisfy particular knowledge that I inject with help of unlabeled dataset. As I pointed out earlier, one limitation with this approach is that it cannot learn much of constraint from the training set as the pre-trained model overfits to the training set and the training set will lack signals that will help constraint injection.

In the proposed work, I am interested in exploring the best way to inject constraint from the scratch without any pre-trained model from the primary training set itself. In this effort, as well as direct application of joint objective similar to eq.(4.6), I will consider the two approaches: 6.1 Multi-task learning with agreement score and 6.2 Sampling negative examples.

6.1 Multi-task system with agreement score

Many NLP tasks require reasoning between multiple sub tasks. Leveraging this fact, it would be interesting to examine whether we can utilize the agreement score of the multi-task output as a regularizer on the training process. While completed work used the explicit constraint and an agreement score that is easy to access and cheap to evaluate, if we consider NLP tasks in general, an explicit constraint might not exist, might be hard to compute, or might require labeled data to evaluate. However, even before resorting to the linguistic knowledge, for the NLP problems that can be formulated as a multi task problem, we already know that multiple tasks have to agree upon. Utilizing this simple knowledge, I propose to use the agreement score from shared task as a guiding signal for training procedure.

Generally, what I have done in my completed work can be expressed as equation 6.1 at a high level rewriting the equation 4.6,

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

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 primary training set, D_2 is additional dataset which could be unlabeled set, and lastly $\hat{y}_k^{(j)}$ denotes output from model θ_k and input $x^{(j)}$. It is worth noting for agreement score $f_{agree}(y_1, y_2)$, I express y_1, y_2 with different subscript as the output format of the two variables do not have to be identical while it is necessary for them to be comparable. For example, returning to the example of SRL application, we can think of $\hat{y}_1^{(j)}$ to be an SRL output format and $y_2^{(j)}$ to be parse tree information given a sentence $x^{(j)}$. While SRL output c and parse tree information $y_2^{(j)}$ will not be in the same format, the spans of $y_2^{(j)}$ and $y_2^{(j)}$ are comparable to measure agreement score. In our previous experiment f_{agree} measures whether spans of $\hat{y}_1^{(j)}$ is 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 as equation 6.2.

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

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. $\theta = \theta_{com}, \theta_1, \theta_2$. Also extending the approach recursively, we can try to leverage unlabeled dataset and perform semi-supervised learning as equation 6.3. While the last agreement component taken from two system output might not be unstable signal by itself, I believe the supervised signals and agreement scores from D_1, D_2 will anchor the learning signals.

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

6.1.1 Related Work

Recently, there was a work in SRL which tries to learn the simplified syntactic parser in multi-task fashion and shows improvement in the SRL performance [37]. This recent work only incorporated supervised loss, i.e. $\sum_{(x^{(i)}, y_1^{(i)}) \in D_1} XL(x^{(i)}, y_1^{(i)}; \theta_{com}, \theta_1) + \sum_{(x^{(j)}, y_2^{(j)}) \in D_2} XL(x^{(j)}, y_2^{(j)}; \theta_{com}, \theta_2)$ and do not incorporate the agreement score, which is major novelty in the proposed work. Thus it would be interesting to compare the proposed model to [37] and analyze the direct effect of agreement score.

6.2 Sampling negative examples

In this proposed work, I want to investigate why even the SOTA networks fail in satisfying simple constraints as we observed earlier on chapters 3,4. I hypothesize that the failure is due to the lack of introduction of *negative samples*, which refer to constraint-violating samples, in the training process. For the classification problem, it is well established that training set should have balanced positive and negative examples in order to learn a good classifier from it. Therefore, I wish to introduce negative samples in terms of constraint to the framework of NLP learning.

The completed work is already introducing a negative examples in the learning through evaluation of model-generated output. I believe this exploration step is a crucial factor in the success of completed work as much as the augmentation of objective function with of constraint loss. In fact, this is closely related to the recent success of Reinforcement Learning (RL) [4, 31] approaches for sequences in NLP problems. While the supervised approach has been successful, they often do not get exposed to a wrong examples or in other words the supervised method does not expose a model to an ill state in Markov chain. This is often called exposure bias [31] and RL approaches remedied this problem partially. I believe behind success of RL approach, exposure to the negative samples and ill states on Markov chain remedied the exposure bias besides the notion of reward.

With the aforementioned motivation, the thesis proposes to systematically introduce the negative samples in the training set. To systematically learn from negative samples, the proposed work considers the following two aspect: *which* negative samples are useful and *how* to learn from negative samples.

To address the first aspect, the thesis proposes to sample "slightly" wrong examples motivated from the ideas in active learning. In uncertainty sampling principle of active learning, it is well known that the instances near the classification boundary are more helpful in learning the classifier compared to the instances far from the boundary. Nonetheless, in the proposed setting, the constraints are provided and we do not need separate classifier for constraint violation. We can simply draw negative samples according to the distribution of constraint violation score $g(\mathbf{y}, \mathcal{L})$, where $g(\mathbf{y}, \mathcal{L}) \rightarrow \mathbb{R}_+$ on section 3.2.1 quantifies constraint violation score proportional to degree of violation. Lastly, note that the proposed approach is completely different from approaches in RL nor from my complete work. On both cases, the sampling process solely depend on the model state whereas the proposed approach draws negative samples from the constraint function similar to RAML [25].

To address the second aspect, the proposal reminds the reader the fact that "constraint" is often not a primal objective for the task. It is left as a question whether simply augmenting the training set with negative samples would improve the model as RAML did. The RAML draws the samples proportional to the reward of the primal task whereas the proposed work draws samples from subsidiary reward, i.e. constraint violation, and it is uncertain how this will affect the learning process. The point in consideration is that even if good negative samples are given, we have to find a proper way to learn primal task leveraging the negative samples from constraint violation. The proposed work promises to explore two approaches as provided in the following subsection 6.2.2. The following subsections describe details of proposed method for learning with negative samples.

6.2.1 Drawing with negative samples

It is not easy to sample a sequence by some arbitrary score as its sample space is often exponentially large. Consider the most simple example of generating sentence with length n and vocabulary size V. Even in this simple case, V^n score evaluations are required to compute the normalizer for the sentence distribution. Most NLP applications have to consider much larger support space. For example, in generation problems such as machine translation and summarization, its sample spaces have to consider arbitrary length outputs rather than a fixed length n. The variable-length output space leads to infinitely large space and even with the pruning, by neglecting certain length, the sample space is much larger than the simple case of V^n.

The form of the score function also determines the level of difficulty on sampling. If we were to give the same score on all the samples, again considering the most simple case, the sampling procedure becomes much easier as we can simply draw each token uniformly. It is especially hard to smaple when the score function is computed in intractable ways such as grammar scoring function with hundreds of hand-crafted rules. Therefore, I propose to use Metropolis-Hastings (MH) sampling in order to sample sequences with arbitrary score. With the distribution of interest q that we want to draw from, we can draw a sample from q using MH sampling in the following way. We can simply draw a new sample y' from proposal distribution p, and then accept with $A(y', y)$ given the previous sample y as following:

$$\begin{aligned} A(y', y) \quad &= \min\left(1, \frac{f(y'|y*)}{f(y_{\text{prev}}|y*)}\frac{p(y_{\text{prev}})}{p(y')}\right) \\ \text{where} \quad & y' \sim p(y|y_{\text{prev}}),\ f(y|y*;\tau) \propto q(y|y*;\tau). \end{aligned}$$

Note that in order to use MH sampling, we need to sample from proposal distribution p and need to access the proposal probability of the drawn sample. I propose to explore following two proposal distributions:

- *Using pre-trained model*: The SOTA pre-trained model as a proposed sampler will produce output y not always perfect but also not too far away from the true output y^* given an input. The better pre-trained model is the sampler will be more efficient as the proposal distribution will closer to the true distribution.
- *Using edit-distance*: If we use edit-distance as a score for proposed sampler, we can create this sampler for any sequence and a way to efficiently sample from edit-distance was shown in [25].

6.2.2 Learning strategy with negative samples

Given we have good negative samples respect to constraint violation, there still need to be a proper ways to learn primal task from it. The following are two directions I propose to explore:

- *Max-margin training*: The maximization of the margin between true output and negative sample as equation 6.4 is one way to learn from negative samples.

$$\begin{aligned} \max_\theta \quad & \sum_{(x,y^*)\in D} \log P\left(y^* \mid x;\theta\right) - \log P\left(y_{\text{neg}} \mid x;\theta\right) \\ \text{where} \quad & y_{neg} \sim q\left(y*, \tau \mid g(y, \mathcal{L})\right) \end{aligned} \tag{6.4}$$

where y^*, y_{neg} denotes true output and negative sample, respectively, and the definitions of rules $\mathcal{L}$ and constraint violation evaluation function $g(\mathbf{y}, \mathcal{L}) \to \mathbb{R}_+$ carries from the section 3.2.1. The learning process will draw outputs similar to the true output with constraint violation, the model will naturally learn to avoid negative samples.

- *Using reward of a primal task*: As RL approaches promotes or penalizes the explored instances with a reward, we propose to assign the reward of the primal task to the negative samples as in equation 6.5.

$$\begin{aligned} \max\nolimits_\theta \quad & \sum_{(x,y^*)\in D} r_{\text{primal}}(y_{\text{neg}}, y^*) \log P(y_{\text{neg}}|x;\theta) \\ \text{where} \quad & y_{\text{neg}} \sim q\left(y^*, \tau \mid g(y, \mathcal{L})\right) \end{aligned} \tag{6.5}$$

where $r_{\text{primal}}(y_{\text{neg}}, y^*)$ is a reward from the primal task. We can also extend this to the joint loss of cross cross entropy loss XL similar to joint loss in 4.6,

$$\begin{aligned} \max\nolimits_\theta \quad & \sum_{(x,y^*)\in D} XL(x, y^*;\theta) + r_{\text{primal}}(y_{\text{neg}}, y^*) \log P(y_{\text{neg}}|x;\theta) \\ \text{where} \quad & y_{\text{neg}} \sim q\left(y^*, \tau \mid g(y, \mathcal{L})\right). \end{aligned} \tag{6.6}$$

By using the reward of the primary score, we can effectively explore the negative samples and also learn the primary task from the negative samples. How this approach will work in comparison to our previous joint learning will be an interesting analysis.

Appendix

.1 Example-based case study for GBI.

⟨“ it is really like this , just look at the bus signs . ”⟩
⟶ `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	0.012	50.0%
6	B-ARG1 B-V B-ARG2 I-ARG2 I-ARG2 O O O O O O O O	0.049	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 1: A semantic role labeling example for which the method successfully enforces syntactic constraints. The initial output has inconsistent span 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"..

⟨“ 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%

Table 2: A shift-reduce example for which the method successfully enforces constraints. 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.

`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: A sequence transduction example for which enforcing the constraints improves accuracy. Red indicates errors.

.2 Analyzing the behavior of different inference procedures in presence of noisy constraints

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 4: Comparison of different inference procedures: Viterbi, A*[13] and GBI with noisy and noise-free constraints. Note that the (+/-) F1 are reported w.r.t Viterbi decoding on the same column.

.3 Structured prediction formulation

He et al. proposed to incorporate such structural dependencies at decoding time by augmenting the loglikelihood function with penalization terms for constraint violations

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \log p(y_i|\mathbf{x}) - \sum_{c \in C} c(\mathbf{x}, \mathbf{y}_{1:i}) \tag{7}$$

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

Bibliography

[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. 3.3.2

[2] 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

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

[4] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1171–1179, Cambridge, MA, USA, 2015. MIT Press. URL `http://dl.acm.org/citation.cfm?id=2969239.2969370`. 6.2

[5] 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. 3.1

[6] 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. 3.1

[7] 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

[8] 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

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

[10] Marcus Mitchell et al. Treebank-3 ldc99t42 web download. In *Philidelphia: Linguistic Data Consortium*, 1999. 3.3.2

[11] 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

[12] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Neural Information Processing Systems (NIPS)*, 2015. 2.3, 3.3.3

[13] 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), 3.3.1, 3.4, 4.1, 4.2, 4.2.2, 4.2.2, 4.2.3, 4.4, 4.3.3, 4.4, 5.2, 4, .3

[14] 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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 364–369, 2018. URL `https://aclanthology.info/papers/P18-2058/p18-2058`. 5.2

[15] 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

[16] 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

[17] 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. 3.2, 3.4

[18] 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

[19] 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. 3.1

[20] Jay Yoon Lee, Michael Wick, Sanket Vaibhav Mehta, Jean-Baptiste Tristan, and Jaime

Carbonell. Gradient-based inference for networks with output constraints. *arXiv preprint arXiv:1707.08608v2*, 2017. (document), 1, 4.4, 4.3.3, 4.4

[21] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics, 2017. URL `http://aclweb.org/anthology/D17-1018`. 5.2

[22] Gordon Lyon. Syntax-directed least-errors anallysis for context-free languages: A practical approach. *Programming Languages*, 17(1), January 1974. 3.2.1

[23] 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

[24] 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

[25] Mohammad Norouzi, Samy Bengio, zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. Reward augmented maximum likelihood for neural structured prediction. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1723–1731. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6547-reward-augmented-maximum-likelihood-for-neural-structured-predicti` `pdf`. 6.2, 6.2.1

[26] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018. 3.1, 3.3.1

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

[28] 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

[29] 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. 3.3.1, 4.1, 4.3.1

[30] 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. 3.4, 4.1, 4.2.3, 4.4

[31] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732, 2015. 6.2

[32] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Computational Natural Language Learning (CoNNL)*, 2009. 2.2

[33] 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. 3.2, 3.4

[34] 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. 3.2, 3.4

[35] 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

[36] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems (NIPS)*, 2014. 3.1

[37] Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. Syntactic scaffolds for semantic structures. *CoRR*, abs/1808.10485, 2018. 6.1.1

[38] Oriol Vinyals, Luksz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *NIPS*, 2015. 2.2, 3.1, 3.4

[39] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. 3.2.2

[40] 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

[41] 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. 3.3.2

[42] 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