Annotating and Modeling Shallow Semantics Directly from Text

Luheng He

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Luke Zettlemoyer, Chair

Yejin Choi

Noah Smith

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Annotating and Modeling Shallow Semantics Directly from Text

Luheng He

Chair of the Supervisory Committee:
Associate Professor Luke Zettlemoyer
Computer Science and Engineering

One key challenge to understanding human language is to find out the word to word semantic relations, such as "who does what to whom", "when", and "where". Semantic role labeling (SRL) is the widely studied challenge of recovering such predicate-argument structure. SRL is designed to be consistent across syntactic alternations , which can potentially benefit downstream applications such as information extraction, machine translation, and summarization. However, the performance of SRL system is limited by the amount of training data and its dependence on the intermediate syntactic representation. In this thesis, our goal is to develop annotation frameworks and learning models for recovering semantic structures directly from text, in an end-to-end manner.

We first introduce question-answer driven semantic role labeling (QA-SRL), an annotation framework that allows us to gather SRL information from non-expert annotators. Different from the traditional SRL formalisms (e.g. PropBank), this new task does not depend on predefined syntactic structure or frame ontology. It is simple and intuitive enough that we can train any native speaker to provide annotation, as long as they can understand the meanings of sentences.

We also develop two general-purpose, syntax-independent neural models that lead to significant performance gains, including a over 40% error reduction over long-standing pre-neural performance levels on PropBank. Our first model, DeepSRL, uses highway BiLSTMs to make local BIO-tagging decisions for each token. While significantly out-performing previous systems, DeepSRL cannot jointly process multiple predicates or incorporate span-level features.

To address these limitations, we further introduce a span-based neural model called the Labeled Span Graph Networks (LSGNs). Inspired by a recent state-of-the-art coreference resolution model, LSGNs build contextualized representations for all spans in the input text, and use lightweight classifiers to make independent edge labeling decisions. With LSGNs, we are able to model all predicate words and argument spans jointly, the first end-to-end result we know of. LSGNs also lead to a unified view for many NLP structures involving span-labeling or span-span relations. In addition to SRL and coreference resolution, LSGNs also achieve state-of-the-art performance when applied to named entity recognition (NER) without any feature engineering. This opens up exciting future directions to build a single, unified model for end-to-end, document-level semantic analysis.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

My five years at the University of Washington is the happiest, most productive time in my life so far. I am extremely lucky to be a part of this amazing community where I was surrounded by many smart and supportive people.

First, I would like to express my sincerest gratitude to my advisor Luke Zettlemoyer. He has always been tremendously encouraging and patient. He gives me both freedom and guidance, and encourages me to take risks and make my own decisions.

My journey in NLP started with my former advisor Ben Taskar at the University of Pennsylvania. Ben has taught me the fundamentals of machine learning and NLP research. His wisdom and mentorship have never stopped influencing me.

I would also like to thank the rest of my thesis committee, Yejin Choi, Noah Smith, and Gina Levow, for their insightful comments and advice on research and career. I want to thank Ido Dagan for the discussions and encouragement on my QA-SRL work. I would also like to thank Dipanjan Das and Tom Dean for their invaluable guidance during my internships at Google.

I am lucky to have collaborated with some of the best researchers in the world: Nicholas FitzGerald, Jennifer Gillenwater, Kenton Lee, Omer Levy, Mike Lewis, Victoria Lin, Yi Luan, Julian Michael, Sameer Singh, and Gabriel Stanovsky.

I want to thank everyone from UW-NLP for the interesting daily discussions, exchange of ideas, and sleepless deadlines that we have pulled together. In particular, I would like to thank my awesome officemates: Eunsol Choi, Ariel Holtzman, Phoebe Mulcaire, Hao Peng, and Swabha Swayamdipta, and our GPUs that kept us warm and cozy during the Seattle winters.

I am grateful to my CSE graduate advisors, Lindsay Michimoto and Elise DeGoede Dorough. Their help and support really make me feel that I belong to the community.

Last but not least, I want to thank Qiao Zhang for being my boyfriend, my best friend, my climbing and workout buddy, for his love and support that has kept me sane and strong. I would like to thank my mother, Aiming Hang, and my father, Youhua He, for showing me the wonder of scientific discovery and the joy of programming when I was a little girl.

DEDICATION

to my parents.

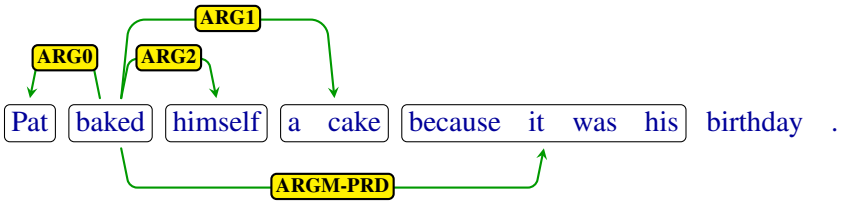
Chapter 1
INTRODUCTION

One key challenge to understanding human language is to find out the word to word semantic relations, such as "who does what to whom", "when", and "where". Semantic role labeling (SRL) is the widely studied challenge of recovering such predicate-argument structure, typically for verbs, as shown in Figure 1.1a. The formalism is designed to capture explicit predicate-argument structures with meaningful labels, and maintain consistency across syntactic alternations such as *John broke the window* and *The window broke* [Palmer et al., 2005]. SRL has shown promises in many real-world NLP applications, such as information extraction [Christensen et al., 2010], question answering [Shen and Lapata, 2007], and machine translation [Liu and Gildea, 2010].

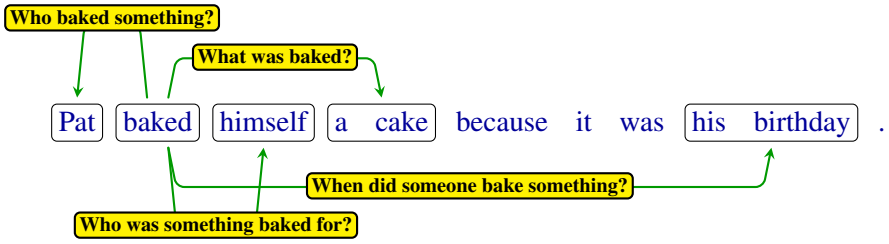
The goal of my thesis work is to develop an accurate and robust SRL system that can work for many different domains and applications. In this thesis, I will describe our solution for building a system that is able to predict SRL structures directly from natural language, without relying on any intermediate preprocessing (e.g. syntactic parsing). This solution is two-folds: 1) designing cheaper, more scalable ways for collecting SRL data, so that we can re-train our SRL model for any new domain; and 2) building accurate, end-to-end models that are able to predict the SRL structures directly from raw text.

1.1 Previous Approaches for SRL Annotation

Existing resources for SRL annotations include the Proposition Bank (or PropBank) [Palmer et al., 2005], the FrameNet [Ruppenhofer et al., 2006], and the OntoNotes [Weischedel et al., 2011] corpus. These corpora are high-quality, as they are created by expert annotators following very detailed annotation guidelines. These corpora have been widely used in training state-of-the-art SRL systems. As an example, Figure 1.1a shows the SRL structure under the PropBank formalism,



(a) SRL dependencies under the PropBank formalism [Palmer et al., 2005].



(b) Question-answer driven semantic annotation.

Figure 1.1: A comparison between PropBank (top) and our QA-SRL annotation scheme (bottom; details in Chapter 4). The sentence contains one verbal predicates *baked*. Each dependency shows a predicate-argument relation between a predicate word and an argument span, with the semantic relation defined by the corresponding label or question.

where “bakes” is a verbal predicate, and the phrases “Pat”, “himself”, “a cake” are its semantic arguments, labeled with corresponding SRL roles.

1.1.1 Limitations

The requirement for expert annotation potentially limits the application and deployment of SRL in real applications, especially when the input texts come from new domains such conversations, web forums, or scientific literature. For example, if we would like to collect SRL data in the biomedical domain, recruiting annotators who have both linguistic training and domain knowledge would be time-consuming and expensive.

Another limitation of traditional SRL annotations is their dependence on the constituency syntax [Marcus et al., 1994] and predefined ontology (or frame files). For example, PropBank is annotated on top of the Penn Treebank syntax, and contains thousands of frame files. This means that, when creating new SRL corpora, we would have to preprocess the texts with syntactic parsers, which are far from perfect. For annotators, determining which frame set to use is a necessary step, which also requires non-trivial amount of training. For example, to annotate the SRL structure for an occurrence of the verb "play", the annotator needs to first decide on which of the following frame to use: "play.01: play a game", "play.02: play a role", or "play.10: play a trick", etc. [1]

1.2 Previous Approaches for SRL Modeling

Thanks to the availability of annotated corpora such as PropBank [Palmer et al., 2005] and OntoNotes [Pradhan et al., 2013], statistical modeling for SRL has seen great success. Most of the traditional models take a pipelined approach that consists of several stages: 1) argument extraction and pruning, 2) argument labeling, and 3) post-processing. These systems also rely heavily on syntactic processing. Since most PropBank arguments are syntactic constituents [Bonial et al., 2010], these systems typically start with parsing the sentence and extracting candidate arguments using hand-engineered rules. Syntactic features, such as part-of-speech tags and dependency labels, are widely used for the argument labeling stage [Punyakanok et al., 2008, Täckström et al., 2015]. As a final post-processing step, they often use integer linear programming (ILP) to enforce global structural constraints [Punyakanok et al., 2008, Toutanova et al., 2008].

1.2.1 Limitations

These syntax-dependent SRL systems typically suffer from *cascading errors* [Finkel et al., 2006], which accumulate from the intermediate processing stages. The cascading errors will be further amplified when the systems are applied to out-of-domain data, because the embedded syntactic parsers are usually trained on the same domain (i.e. newswire) as the SRL systems, and are likely

[1] Source: http://verbs.colorado.edu/propbank/framesets-english-aliases/play.html

to make more mistakes on out-of-domain samples.

Most of these pipeline-based SRL systems are also dependent on gold standard predicates, as opposed to predicted ones. CoNLL 2005 [Carreras and Màrquez, 2005] and CoNLL 2012 [Pradhan et al., 2013] are two mostly widely used shared tasks for span-based SRL systems. However, both tasks assume gold predicates as part of the input. We argue that it is important to evaluate SRL systems on raw sentences, and have the systems predict all predicates as well as their arguments. This raises a more challenging structured prediction problem, as there can be at most n possible predicates, and n^2 possible argument spans.

1.3 Overview of Our Approach

In this thesis, I will address the above mentioned limitations — dependency on syntax, expert annotators, and gold predicates — by describing our new methods for annotating and modeling SRL structures directly from natural language texts.

1.3.1 Scalable SRL Annotation

We would like to find a way to collect very large amounts of training data at a lower cost. Intuitively, we would like to argue that:

Anyone who understands the meaning of a sentence should be able to provide useful annotation.

Hiring non-expert annotators for semantic annotation means that it is no longer feasible to follow the traditional SRL formalisms, which usually depend some underlying *syntactic structures* and have a large, predefined *ontology* [Palmer et al., 2005, Ruppenhofer et al., 2006]. Based on this key insight, we introduced Question-Answer driven Semantic Role Labeling (QASRL) [He et al., 2015], a novel annotation framework where annotators write wh-questions[2] about a given verb and provide answers to these questions using words from the sentence. As illustrated in Figure 1.1b, the question-answer pairs represent underlying semantic relations, and can be used to train an SRL system.

[2]Questions starting with words such as *what, who, where, when, why* and *how*.

Different from the traditional SRL annotations, this new task does not depend on predefined syntactic structure or frame ontology. It is simple and intuitive enough that we can train any native speaker to provide annotation, as long as they can understand the meanings of sentences. This is because we are *using natural language to annotate natural language*.

Our experiments demonstrated high quality data annotation with very little annotator training and established baseline performance levels for the task. We hired non-expert, part-time annotators on Upwork (previously oDesk) to label over 3,000 sentences (nearly 8,000 verbs) across two domains (newswire and Wikipedia) at a cost of approximately $0.50 per verb. We show that the data is high quality, rivaling PropBank in many aspects including coverage, and easily gathered in non-newswire domains. Our hope is that this approach will generalize not only across different domains in English. For example, we would be able to hire domain experts to annotate new data in the biomedical domain, without requiring them to have any linguistic knowledge.

1.3.2 End-to-end Modeling for SRL

Traditional learning models for SRL often rely on hand-engineered features induced from intermediate syntactic representations, such as part-of-speech tags and syntactic dependencies [Punyakanok et al., 2008, Toutanova et al., 2008]. Recent breakthroughs involving end-to-end deep models for SRL without syntactic input [Zhou and Xu, 2015, Marcheggiani et al., 2017] seem to overturn the long-held belief that syntactic parsing is a prerequisite for this task. Inspired by this, we show that these results can be pushed further using deep highway bidirectional LSTMs with constrained decoding, again significantly moving the state of the art (another 2 points on CoNLL 2005).

Our DeepSRL system [He et al., 2017] combines a number of best practices in the recent deep learning literature. Following Zhou and Xu [2015], we treat SRL as a BIO[3] tagging problem and use deep bidirectional LSTMs. However, we differ by (1) simplifying the input and output layers, (2) introducing highway connections [Srivastava et al., 2015, Zhang et al., 2016], (3) using recur-

[3]Begin, inside, and outside.

rent dropout [Gal and Ghahramani, 2016], (4) decoding with BIO-constraints, and (5) ensembling with a product of experts. Our model gives a 10% relative error reduction over previous state of the art on the test sets of CoNLL 2005 and 2012. We also report performance with predicted predicates to encourage future exploration of end-to-end SRL systems.

We also present detailed error analyses to better understand the performance gains, including (1) design choices on architecture, initialization, and regularization that have a surprisingly large impact on model performance; (2) different types of prediction errors showing, e.g., that deep models excel at predicting long-distance dependencies but still struggle with known challenges such as PP-attachment errors and adjunct-argument distinctions; (3) the role of syntax, showing that there is significant room for improvement given oracle syntax but errors from existing automatic parsers prevent effective use in SRL.

1.3.3 Labeled Span Graph Networks

As a final step, we propose a unified neural architecture called Labeled Span Graph Networks (LSGNs). LSGNs build fixed-length representations for all spans in an input text and use lightweight classifiers to make decisions about how to best link, rank, or label them. The representations are built with deep BiLSTMs and include a span internal attention mechanism, following recent work [Lee et al., 2017]. The classifiers are trained to make independent predictions for each text span. LSGNs support three types of decisions: *span linking* — predicting labeled edges between pairs of spans; *span ranking*— predicting the best parent span for a given child span; and *span labeling*— predicting labels for individual spans. Individual LSGNs can include one or more of these classifier types, computed over shared span embeddings.

To demonstrate their generality, we study four different LSGNs. We show that a state-of-the-art coreference resolution model [Lee et al., 2017] is an LSGN with *span ranking* potentials. We also introduce new LSGNs for named entity recognition NER) and SRL that provide strong performance gains over existing state-of-the-art BIO tagging models. For SRL, the LSGN formulation is particularly novel. The ability to predict overlapping spans enables joint modeling of all predicates and arguments, unlike current SRL models that must be run independently for each verb in

a sentence. Finally, we also introduce an LSGN that predicts all three tasks jointly in a single forward pass, on top of shared span representations. This approach saves significant computation with only modest loss in accuracy, and empirically demonstrates the generality of the LSGN modeling paradigm.

Compared to previous systems, including DeepSRL [2017], LSGN has the following advantages: 1) Instead of relying on gold (or predicted) predicates as part to the input, LSGN is able to jointly predict all predicates and their arguments in one forward pass. 2) LSGNs can be used as a general span-based architecture for a range of NLP tasks, and can be used for predicting SRL, coreference clusters and named entities with shared span representations. Both advantages make the model easier to use in real applications.

1.4 Thesis Outline

In Chapter 2, we will provide a brief overview of the background and related work on SRL formalisms and several types of SRL models. In Chapter 4, we will introduce our work on gathering syntax-agnostic, non-expert annotation for semantic role labeling, and applying such annotation to improve a CCG parser. In Chapter 5 and 6, we will present the technical details of our end-to-end SRL models: DeepSRL and LSGN. Finally, we conclude and discuss on future directions in Chapter 7.

Chapter 2

BACKGROUND

In this chapter, we will provide a brief overview of the background information for this thesis. Section 2.1 reviews the semantic role labeling (SRL) formalism and discusses some of the details in the PropBank annotation process. Section 2.2 presents a high-level overview of some of the most widely used structured prediction approaches to SRL modeling, including the syntax-dependent pipelined approaches as well as the more recent neural models.

2.1 Semantic Role Labeling

Semantic role labeling models information such as "who did what to whom, when and where" in sentences. Formally, SRL can be defined as a set of labeled predicate-argument dependencies, where the labels are semantic relations, or "roles", defined specific to the SRL formalism. SRL is designed to be consistent across syntactic alternation. Consider the following example from Palmer et al. [2005]:

1. John *broke* the window.

2. The window *broke*.

In the above example, both sentences contain *broke* as a verbal predicate. In both sentences, *the window* are arguments of the predicate *broke*, and semantically correspond to *the thing being broken*, while syntactically, the first occurrence is the subject of *broke*, and the second one is the object.

The job of an automatic SRL system is to identify such predicate-argument relations. These relations are typically labeled with the corresponding SRL roles, which distinguishes between different types of relationships. The set of possible SRL role labels, and the how they are interpreted,

depends on the SRL formalism and the set of pre-defined frame files. In the first example, the SRL relations under the PropBank formalism are broke $\xrightarrow{\text{ARG0}}$ John and broke $\xrightarrow{\text{ARG1}}$ the window, according to the PropBank frame definition[1] shown in Table 2.1.

Roleset id: break.01, break, cause to not be whole

Role	**Description**
Arg0-PAG	breaker
Arg1-PPT	thing broken
Arg2-MNR	instrument
Arg3-PRD	pieces
Arg4-DIR	arg1 broken away from waht?

Table 2.1: Frame definition for the roleset *break.01* from the PropBank frame inventory.

Apart from PropBank, there exist other semantic formalisms such as the FrameNet [Ruppenhofer et al., 2006], NomBank [Meyers et al., 2004], Abstract Meaning Representation (AMR) [Banarescu et al., 2013], etc. Since most of my thesis work focuses on the PropBank formalism, the rest of this chapter will be used to go through more detailed explanations of PropBank and its annotation process. Section 3.1 will provide a brief overview of the other semantic formalisms.

2.1.1 PropBank Annotation for SRL

Most of the modeling work of this thesis focuses on the PropBank [Palmer et al., 2005] formalism. PropBank captures 1) sentence-level semantics, where the predicate-argument relations do not extend across more than one sentence; 2) span-based arguments, where the arguments are continuous spans in the text (instead of head words); and 3) arguments are defined in each frame file/role set, in a verb-specific way.

[1] Unified Verb Index: https://verbs.colorado.edu/verb-index

Frames and Role Sets In PropBank, the frame files contains verb-specific descriptions for the core arguments that can be associated with a particular verb. Currently, PropBank has defined 10,513 frames, including 6,029 verbs, 2,572 nouns, and 1,912 adjectives.[2] Different senses of each verb (or noun/adjective) can trigger very different frames, and they are defined in different *role sets*. For example, the frame for "play" contains 6 role sets, covering senses such as "play a game", "play a role", or "perform music".

Relation to Syntax The original PropBank dataset is annotated as an additional layer on top of the Penn Treebank [Marcus et al., 1994]. According to the PropBank Annotation Guideline [Babko-Malaya, 2005, Bonial et al., 2010], most of the arguments are syntactic constituents. Multi-constituent spans are represented with the special continuation (C-X) tag.

Core Arguments vs. Modifiers PropBank differentiates between the core arguments and the modifier arguments (sometimes called adjuncts). The core arguments are numbered from Arg0 to Arg5, with the exact semantic meaning defined in the verb-specific frame files. Core arguments with the same label can map to different semantic roles for different verbs. For example, the Arg2 of role set *break.01* means "the instrument used for breaking", while for another role set *buy.01*, Arg2 means "the seller".

Different from the core arguments. modifier arguments are shared across all verb frames. In PropBank, modifier role labels have the prefix "AM-" (meaning "arg-modifier). They are used to capture relations that are applicable to all kinds of verbs, such as reciprocals (REC), time (AM-TMP), location (AM-LOC), manner (AM-MNR), and purpose (AM-PRC). The 2005 version of PropBank [Babko-Malaya, 2005] has 14 modifier labels, while the 2010 version [Bonial et al., 2010] has 18 modifier labels.

C-Args and R-Args Continuation arguments (with the prefix "C-" as in "C-A0") and reference arguments (with the prefix "R-") are two special types of roles to handle the multi-constituent

[2] According to: https://verbs.colorado.edu/propbank/framesets-english/

arguments in PropBank. During evaluation, continuation arguments (C-X) are combined with their base arguments (preceding argument with label X) and treated as a single argument, while the reference roles (R-X) are treated as separate labels.

2.1.2 Annotation Process for PropBank

According to the PropBank annotation guidelines [Babko-Malaya, 2005, Bonial et al., 2010], annotators would go through the following steps: 1) Select the appropriate roleset for the target predicate. 2) Assign argument labels according to the frame file definition. 3) If the target verb is not covered by any existing frame file, a new one needs to be created. As we will further discuss in Chapter 4, it could be difficult for untrained annotators to go through this kind of annotation process in a reliable way.

2.1.3 Span-based versus Dependency-based SRL

While the original PropBank-style SRL is span-based, in the sense that each argument can contain more than one word, dependency-based SRL is also a widely used formalism [Surdeanu et al., 2008, Hajič et al., 2009]. The dependency-based SRL formalism is converted from span-based SRL using head rules, and evaluated with labeled F1 over the word-word dependencies. Different from the CoNLL 2005 SRL task, the CoNLL 2008 task 1) includes nominal predicates from the NomBank [Meyers et al., 2004]; 2) requires the systems to correctly predict all the predicates and predicate sense. The CoNLL 2009 task is multi-lingual and provides a binary indication for the argument-bearing predicates as part of the input. This thesis will be mainly focusing on annotations and models for span-based SRL.

2.2 Structured Prediction for SRL

Structured prediction is the core problem of many NLP tasks and applications. Usually, we would like to learn a model f that maps an input X to an output Y. Different from other machine learning applications such as classification or regression, the output of a structured prediction problem $Y \in$

$\mathcal{Y}$ is often a combinatorial structure (such as a sequence of labels, a tree, or an arbitrary graph), while the space of all possible outputs $\mathcal{Y}$ can be exponential in its size.

Typically, the model is parameterized by a set of learnable weights θ, which are learned by optimizing some loss function $\mathcal{L}$ with respect to the gold structure Y^*:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f_\theta(X, Y^*), Y^*)$$

At inference time, we wish to find the best output structure $\hat{Y}$ that maximizes the model score under the learned parameters θ. The inference problem can be formulated as follows:

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}} f_\theta(X, Y) \tag{2.1}$$

Because the space of all possible output structures $\mathcal{Y}$ is usually too large to exhaustively enumerate, structured prediction models usually use dynamic programming [Eisner and Satta, 1999, Lafferty et al., 2001] or inexact search [Collins and Roark, 2004] when searching for the prediction $\hat{Y}$.

2.2.1 Structured Prediction Models for SRL

Formally, the SRL prediction problem is defined as follows: given input words $\mathbf{w} = w_1, \ldots, w_n$, and a target predicate $v \in \{1, \ldots, n\}$, the model should predict the set of labeled argument spans $Y = \{a_1, \ldots, a_k\}$. Each output argument span a_i can be represented as a tuple $(s_{\text{start}}, s_{\text{end}}, l)$, where $1 \leq s_{\text{start}} \leq s_{\text{end}} \leq n$ are the start and the end indices of the span. $l \in \mathcal{L}$ is the role label assigned to argument span.

Predicate Identification According to the CoNLL 2005 shared task, the target predicates v are given as part of the input, which means that these models only need to predict arguments for the "gold" predicates. However, in real applications, the predicates need to be predicted. Although most of the predicates in the original PropBank are verbs, the OntoNotes dataset does contain some nominal predicates. With both verbal and nominal predicates, predicate identification can be a non-trivial task, but is often overlooked in the span-based SRL modeling literature.

Span-based SRL Models SRL could be directly modeled as a span-labeling problem: Given each predicate, the model identifies spans in the sentence corresponding to its arguments, and assigns role labels to these argument spans. For example, one simple way to model the structure is to factorize the model into local classifiers for labeling each answer span:

$$p(Y \mid X) \propto \prod_{(s_{\text{start}}, s_{\text{end}}) \in \mathcal{S}} p(l \mid \mathbf{w}, v, s_{\text{start}}, s_{\text{end}}; \theta) \tag{2.2}$$

Here $\mathcal{S}$ denotes the set of candidate arguments, which can contain as many as n^2 spans in the input text. But naively modeling all possible argument spans can be computational intensive and can cause sample sparsity. Most models go through a preprocessing step to prune the space of candidate argument using a syntactic parse tree and a set of hand-engineered rules.

The decomposed local labeling probabilities $p(l \mid \mathbf{w}, v, s_{\text{start}}, s_{\text{end}}; \theta)$ could be parameterized either as a log-linear feature-rich model [Toutanova et al., 2008, Täckström et al., 2015] or with a neural network [FitzGerald et al., 2015]. Syntactic features, such as head words, dependency parents, and part-of-speech tags, have been extensively used in these SRL models.

Global Constraints Most of the span-based SRL systems enforce task-specific global constraints when decoding the output structure. These constraints [Punyakanok et al., 2008, Toutanova et al., 2008] include:

- **Non-overlapping spans:** Arguments of the same predicate cannot overlap with each other.
- **Unique core argument:** Each predicate cannot have duplicate core arguments of the same type (Arg0-Arg5).
- **Continuation role:** If the prediction contains continuation role (C-X), it should also contain an argument of the base type (X) before the C-X argument.
- **Reference role:** If the prediction contains reference role (R-X), it should also contain an argument of the base type (X) (can be either before of after the R-X argument).

These constraints are often enforced with integer linear programming (ILP) [Punyakanok et al., 2008, Toutanova et al., 2008]. Täckström et al. [2015] used dynamic programming to enforce a subset of them.

Tagging-based SRL Models SRL can also be considered as a BIO tagging problem [Màrquez et al., 2008, Collobert et al., 2011, Zhou and Xu, 2015]. Given an input sentence $\mathbf{x}$ and a target predicate p in the sentence, the output SRL structure can be considered as a set of *non-overlapping* argument spans, labeled with the corresponding SRL roles. This can be solved with the BIO-tagging approach and a sequential tagging model. Specifically, each word in the sentence $\mathbf{x}_i$ is labeled with a BIO tag, as shown in Figure 2.1.

B-Arg0	I-Arg0	O	O	O	B-V	B-Arg1	I-Arg1	I-Arg1	I-Arg1
Many	tourists	visit	Disney	to	***meet***	their	favorite	cartoon	characters

Figure 2.1: SRL as a BIO tagging problem. The input is the sequence of words and a target predicate "meet", and the output is the sequence of BIO tags, corresponding to the set of labeled arguments spans "Arg0:Many tourists" and "Arg1:their favorite cartoon characters".

Different from the span-based model, tagging-based models are often factorized with respect to tag n-grams. For example, a second-order globally normalized tagging model would look like:

$$p(Y \mid X) \propto \prod_{i=1}^{|\mathbf{w}|} \phi(t_i, t_{i-1}, \mathbf{w}, v) \tag{2.3}$$

In general, the tagging model can be implemented with feature-rich conditional random fields (CRFs) [Lafferty et al., 2001] or neural sequential taggers [Collobert et al., 2011, Zhou and Xu, 2015]. Tagging-based SRL models naturally follow the non-overlapping constraint, but could still produce predictions with duplicate core arguments or inconsistent continuation roles.

2.3 Summary

This chapter presented a brief overview of the PropBank SRL formalism and some structured prediction methods for span-based SRL. In Chapter 3, we will present more detailed discussions on the literature of semantic formalisms and SRL models. As we will see in Chapter 3, our question-answer driven SRL (QA-SRL) annotation framework is closely related to the PropBank formalism, but is more free-formed and more intuitive for non-annotators to understand. In Chapter 5 and 6, we will introduce two neural SRL models, based on BIO-tagging and span-labeling, respectively. Both models are trained in an end-to-end manner, without relying on any syntactic processing or heuristic argument pruning.

Chapter 3

RELATED WORK

This chapter reviews several lines of related work. Section 3.1 reviews several annotated corpora for semantic structures, either with expert or non-expert annotations. Section 3.2 covers a range of span-based SRL models, from the traditional ones with syntactic features, to the newer end-to-end neural SRL models. Finally, we also investigate different classes of span-based NLP models in Section 3.3.

3.1 Linguistic Annotation for Semantic Structures

The success of syntactic annotation projects such as the Penn Treebank [Marcus et al., 1993] has led to numerous efforts to create semantic annotations for large corpora. The major distinguishing features of our QA-SRL approach (Chapter 4) are that it is not tied to any linguistic theory and that it can be annotated by non-experts with minimal training. In the rest of this section, we will survey related work on several expert-annotated SRL corpora and some other semantic representations.

3.1.1 Expert-annotated SRL Corprora

Existing SRL task formulations are closely related to our work on QA-SRL, as the annotations aim at recovering labeled predicate-argument structures in texts. FrameNet [Baker et al., 1998] contains a detailed lexicon of verb senses and thematic roles. However, this complexity increases the difficulty of annotation. While the FrameNet project is decades old, the largest fully annotated corpus contains about 3,000 sentences [Das et al., 2014]. With QA-SRL, we were able to annotate over 3,000 sentences within weeks in He et al. [2015], and about 75,000 sentences in 9 days using crowdsourcing later in FitzGerald et al. [2018]. PropBank [Kingsbury and Palmer, 2002], NomBank [Meyers et al., 2004] and OntoNotes [Hovy et al., 2006] circumvent the need for a large

lexicon of roles, by defining the core semantic roles in a predicate-specific manner. However, this means that frames need to be created for every verb, and it requires experts to distinguish between different senses and different roles. Our QA-SRL scheme takes a different approach than FrameNet or PropBank, by using templated yet natural-sounding questions (e.g. "What was avoided?") to express different semantic roles (e.g. *Arg1* or *Undesirable_situation*[1]).

3.1.2 Other Meaning Representations

The QA-SRL annotation scheme is also related to recent, more general semantic annotation efforts. Abstract Meaning Representation [Banarescu et al., 2013] can be viewed as an extension of PropBank with additional semantic information. Sentences take 8-13 minutes to annotate—which is slower than ours, but the annotations are more detailed. Universal Cognitive Conceptual Annotation (UCCA) [Abend and Rappoport, 2013] is an attempt to create a linguistically universal annotation scheme by using general labels such as *argument* or *scene*. The UCCA foundational layer does not distinguish semantic roles, so *Frogs eat herons* and *Herons eat frogs* will receive identical annotation — thereby discarding information which is potentially useful for translation or question answering. They report similar agreement with PropBank to our approach (roughly 90%), but annotator training time was an order-of-magnitude higher (30-40 hours). The Groningen Meaning Bank [Basile et al., 2012] project annotates text by manually correcting the output of existing semantic parsers. They show that some annotation can be crowdsourced using "games with a purpose" — however, this does not include its predicate-argument structure, which requires expert knowledge of their syntactic and semantic formalisms. Reisinger et al. [2015] study crowdsourcing semantic role labels based on Dowty's proto-roles, given gold predicate and argument mentions. This work directly complements our focus on labeling predicate-argument structure. More recently, Michael et al. [2018] used Amazon Mechanical Turk[2] to crowdsource free-formed question-answer pairs that captures relations over all possible pairs of words in a sentence.

QA-SRL circumvents the need for large frame inventory by using natural language question-

[1] Based on the PropBank and FrameNet frame for "avoiding" respectively.

[2] https://www.mturk.com/

answer pairs (see details in Chapter 4) to express predicate-argument structures. This idea of expressing the meaning of natural language in terms of natural language is related to natural logic [MacCartney and Manning, 2007], in which they use natural language for logical inference. Similarly, we model predicate-argument structure of a sentence with a set of question-answer pairs. While existing work on natural logic has relied on small entailment datasets for training, our method allows practical large-scale annotation of training data. It also worth noting that Abend and Rappoport [2017] provides a broad overview of some of the semantic representations.

3.1.3 Improving Parsers with Non-expert Annotation

Crowdsourcing is a widely used approach to build datasets with lower costs and at larger scale, often with a tradeoff for annotation quality [Snow et al., 2008]. For structured tasks, crowdsourcing can be particularly challenging, as it is difficult to formulate the problem as simple microtasks for untrained annotators. Crowdsourcing has been used for improving structured prediction models such as named entity recognition [Werling et al., 2015] and prepositional phrase (PP) attachment [Jha et al., 2010]. These works usually focus on a particular substructure (e.g. PP attachment decisions), rather than having crowd workers annotating the entire linguistic structures.

As an extension to the QA-SRL scheme, our work on human-in-the-loop (HITL) parsing (Section 4.2) is most related to that of Duan et al. [2016], which automatically generates paraphrases from n-best parses and gained significant improvement by re-training from crowd-sourced judgments on two out-of-domain datasets. Choe and McClosky [2015] improve a parser by creating paraphrases of sentences, and then parsing the sentence and its paraphrase jointly. Instead of using paraphrases, we build on the question-answer driven approach of QA-SRL, using automatically generated multiple-choice questions to provide supervision for CCG syntactic attachment.

3.2 Models for Semantic Role Labeling

In this section, we will survey a range of learning models for predicting the SRL structure. While more recent syntax-free neural SRL models — including our proposed models DeepSRL (Chapter

5) and LSGN (Chapter 6) — consistently out-perform traditional syntax-dependent systems, syntax is still shown by recent works to be helpful as side information. We also include a brief overview of unsupervised SRL systems.

3.2.1 SRL Systems with Syntactic Information

Traditional approaches to SRL have used syntactic parsers to identify constituents and model long-range dependencies, and enforced global consistency using integer linear programming [Punyakanok et al., 2008] or dynamic programs [Täckström et al., 2015]. Syntactic features and syntactic information are also widely used on other related semantic tasks such as FrameNet parsing [Das et al., 2014] and dependency-based SRL [Johansson and Nugues, 2008]. Neural methods have been employed on top of syntactic features [FitzGerald et al., 2015, Roth and Lapata, 2016], where sparse, one-hot syntactic features such as dependency paths are embedded into dense vectors.

Another way to incorporate syntax while avoiding cascading errors is to build joint models for syntax and semantics. Swayamdipta et al. [2016] proposed a joint model for dependency-based syntax and semantics based on the stack LSTM model, while Lewis et al. [2015] jointly models CCG (Combinatory Categorical Grammar) parsing and dependency-based SRL. More recently, Swayamdipta et al. [2017] uses constituency prediction as an auxiliary objective to improve FrameNet prediction, while Strubell et al. [2018] uses syntactic dependencies to guide self-attentions in PropBank SRL. Our experiments on DeepSRL (Chapter 5) and LSGN (Chapter 6) show that relatively simple, end-to-end trained neural methods have a remarkable ability to learn long-range dependencies, syntactic constituency structure, and global constraints without coding task-specific mechanisms for doing so.

3.2.2 Systems with less or no Syntax

An alternative line of work has attempted to reduce the dependency on syntactic input for SRL models. Collobert et al. [2011] first introduced an end-to-end neural-based approach with sequence-level training and used convolutional neural networks to model the context window. However, their

best system fell short of traditional feature-based systems. Neural methods have also been used as classifiers in transition-based SRL systems [Henderson et al., 2013, Swayamdipta et al., 2016]. Most recently, several successful LSTM-based[3] architectures [Hochreiter and Schmidhuber, 1997] have achieved state-of-the-art results in English span-based SRL [Zhou and Xu, 2015], Chinese SRL [Wang et al., 2015], and dependency-based SRL [Marcheggiani et al., 2017] with little to no syntactic input. Our DeepSRL model is mostly related to Zhou and Xu [2015], treating SRL as a BIO tagging problem, but out-performs it by combining a number of deep learning practices. More recently, Tan et al. [2018] further pushes the state of the art with a self-attention based architecture [Vaswani et al., 2017].

3.2.3 Unsupervised SRL

There has also been work on trying to solve SRL in an unsupervised way [Lang and Lapata, 2014, Luan et al., 2016, Titov and Klementiev, 2012]. Lang and Lapata [2011] used linguistic-motivated heuristics to identify arguments and treated subsequent argument labeling task as a clustering problem. Titov and Khoddam [2015] assumed gold arguments, and tackled argument labeling as a discrete autoencoder problem, where the model is trained to recover argument span from latent labels. Previous work on unsupervised SRL usually rely heavily on predicted syntax, making them unsuitable for the real low-resource setting.

3.3 Neural Span-based Models

DeepSRL and LSGN are both end-to-end neural models for SRL, but take different approaches to model the argument spans. There have been two major types of neural models for identifying and labeling spans: BIO-taggers [Collobert et al., 2011, Chiu and Nichols, 2016] and semi-Markov segmenting models [Kong et al., 2016]. The latter has the advantage of incorporating span-level features, but both models are constrained to predict a set of *non-overlapping* spans. In Chapter 6, we will introduce the LSGN model which uses expressive span-level features while making no

[3]Long short-term memory

restrictions on the set of output spans (e.g. nested and overlapping spans are allowed).

This flexibility is particularly well matched for end-to-end span-based SRL. Here, although each verb is guaranteed to have non-overlapping arguments, the arguments for different verbs can overlap or be nested within each other. Perhaps because of this challenge, existing deep neural models assume gold predicates and predict arguments one verb at a time [He et al., 2017, Zhou and Xu, 2015]. Yang and Mitchell [2017] use span-level features in a relational model, but still do BIO-tagging for pruning.

The LSGN model draws heavily from the recent work on end-to-end coreference resolution [Lee et al., 2016a], which enumerates all possible mention spans up to a certain length, and builds fixed-length span representations. The idea of enumerating (potentially overlapping) spans and building span representations is also applied to other NLP tasks such as question-answering [Lee et al., 2016a] and constituency parsing [Stern et al., 2017].

Chapter 4
ANNOTATING SRL WITHOUT EXPERTS

To enable crowd-scale SRL data collection, we need to design an annotation framework that is simple and intuitive enough for non-expert annotators to understand. Our goal is to make sure that anyone who understands the meaning of a sentence should be able to provide annotation with very little training. This motivated us to move away from traditional SRL task definitions and to design a novel annotation scheme called Question-Answer Driven Semantic Role Labeling (QA-SRL). We later showed that this kind of non-expert annotation can be used directly to improve a near state-of-the-art syntactic parser. This chapter is based on work originally described in He et al. [2015, 2016].

4.1 Question-Answer Driven Semantic Role Labeling (QA-SRL)

We introduce a new question-answer driven SRL task formulation (QA-SRL), which uses question-answer pairs to label verbal predicate-argument structure. For example, for the sentence in Figure 4.1, we can ask a short question containing a verb, e.g. *"Who finished something?"*, and whose answer is a phrase from the original sentence, in this case *"UCD."* The answer tells us that *"UCD"* is an argument of "finished," while the question provides an indirect label on the role that *"UCD"* plays. Enumerating all such pairs, as we will see later, provides a relatively complete representation of the original verb's arguments and modifiers.

The QA-SRL task formulation has a number of advantages. It can be easily explained to non-expert annotators with a short tutorial and a few examples. Moreover, the formulation does not depend on any pre-defined inventory of semantic roles or frames, or build on any existing grammar formalisms. Nonetheless, as we will show, it still represents the argument and modifier attachment decisions that have motivated previous SRL definitions, and which are of crucial importance for

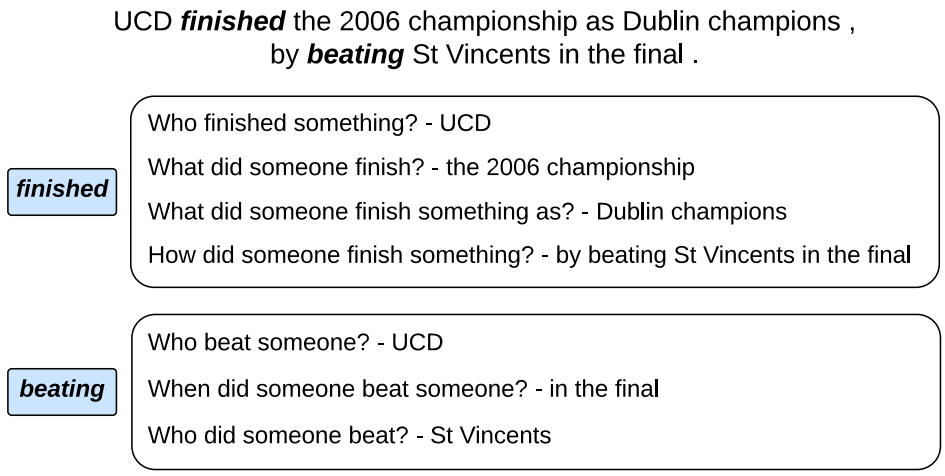


Figure 4.1: QA-SRL annotations for a Wikipedia sentence. The two target predicates "finished" and "beating" are provided to the annotators via a high-recall part-of-speech tagger, and the question-answer pairs are composed by our non-expert annotators using our templates.

semantic understanding in a range of NLP tasks, such as machine translation [Liu and Gildea, 2010] and coreference resolution [Ponzetto and Strube, 2006]. The annotations also, perhaps surprisingly, capture other implicit arguments that cannot be read directly off of the syntax, as was required for previous SRL approaches. For example, in *"It was his mother's birthday, so he was going to play her favorite tune"*, annotators created the QA pair *"When would someone play something? His mother's birthday"* which describes an implicit temporal relation. Finally, QA-SRL data can be easily examined, proofread, and improved by anyone who speaks the language and understands the sentence; we use natural language to label the structure of natural language.

We present a scalable approach for QA-SRL annotation and baseline models for predicting QA pairs. Given a sentence and target word (the verb), we ask annotators to provide as many question-answer pairs as possible, where the question comes from a templated space of wh-questions[1] and the answer is a phrase from the original sentence. This approach guides annotators to quickly

[1]Questions starting with a wh-word, such as *who*, *what*, *when*, *how*, etc.

Field	Description	Example of Values	No. Values
WH*	Question words (wh-words)	who, what, when, where, why, how, how much	7
AUX	Auxiliary verbs	is, have, could, is n't	36
SBJ	Place-holding words for the subject position	someone, something	2
TRG*	Some form of the target word	built, building, been built	≈ 12
OBJ1	Place-holding words for the object position	someone, something	2
PP	Frequent prepositions (by, to, for, with, about) and prepositions (unigrams or bigrams) that occur in the sentence	to, for, from, by	≈ 10
OBJ2	Similar to OBJ1, but with more options	someone, something, do, do something, doing something	9

Table 4.1: Fields in our question annotation template, with descriptions, example values, and the total number of possible values for each. **WH*** and **TRG*** are required; all other fields can be left empty.

construct high quality questions within a very large space of possibilities.

Experiments demonstrate high quality data annotation with very little annotator training and establish baseline performance levels for the task. We hired non-expert, part-time annotators on Upwork (previously oDesk) to label over 3,000 sentences (nearly 8,000 verbs) across two domains (newswire and Wikipedia) at a cost of approximately $0.50 per verb. We show that the data is high quality, rivaling PropBank in many aspects including coverage, and easily gathered in non-newswire domains.[2] The baseline performance levels for question generation and answering

[2]Our hope is that this approach will generalize not only across different domains in English, as we show in this paper, but also to other languages. We will leave those explorations to future work.

WH*	**AUX**	**SBJ**	**TRG***	**OBJ1**	**PP**	**OBJ2**	
Who			built	something			?
What	had	someone	said				?
What	was	someone	expected		to	do	?
Where	might	something	rise		from		?

Table 4.2: Four example questions written with our question annotation template.

reinforce the quality of the data and highlight the potential for future work on this task.

Because QA-SRL is designed to be an open-ended, flexible annotation scheme with minimal syntactic assumptions, it can be easily extended to many domains. For example, we can collect data for biomedical text by hiring people who have biomedical background but no linguistic training. Furthermore, QA-SRL also has the potential to be applied to other languages, because the concept of questions and answers is not English-specific.

4.1.1 Annotation Task Design

We annotate verbs with pairs of questions and answers that provide information about predicate-argument structure. Given a sentence s and a verbal predicate v in the sentence, annotators must produce a set of wh-questions that contain v and whose answers are phrases in s.

To speed annotation and simplify downstream processing, we define a small grammar over possible questions. The questions are constrained with a template with seven fields, $q \in$ **WH** $\times$ **AUX** $\times$ **SBJ** $\times$ **TRG** $\times$ **OBJ1** $\times$ **PP** $\times$ **OBJ2**, each associated with a list of possible options. Descriptions for each field are shown in Table 4.1. The grammar is sufficiently general to capture a wide-range of questions about predicate-argument structure—some examples are given in Table 4.2.

The precise form of the question template is a function of the verb v and sentence s, for two of the fields. For the **TRG** field, we generate a list of inflections forms of v using the Wiktionary

Sentence	CoNLL-2009		QA-SRL	
(1) Stock-fund managers, meantime, went into October with less cash on hand than they **held** earlier this year.	A0	they	Who had held something?	Stock-fund managers / they
	AM-TMP	year	When had someone held something?	earlier this year
			What had someone held?	less cash on hand
			Where had someone held something?	on hand
(2) Mr. Spielvogel **added** pointedly: " The pressure on commissions did n't begin with Al Achenbaum. "	A0	Spielvogel	Who added something?	Mr. Spielvogel
	A1	did	What was added?	" The pressure on commissions did n't begin with Al Achenbaum . "
	AM-MNR	pointedly	How was something added?	pointedly
(3) The . . . company said net **rose** to $108.8 million, or $1.35 a share, from $90.5 million , or $1.12 a share, a year ago.	A1	net	What rose?	net
	A3	$/ago	What did something rise from?	$ 90.5 million , or $ 1.12 a share
	A4	to	What did something rise to?	$ 108.8 million , or $ 1.35 a share
			When did something rise?	a year ago
(4) Mr. Agnew was vice president of the U.S. from 1969 until he **resigned** in 1973 .	A0	he	Who resigned from something?	Mr. Agnew
	AM-TMP	in	When did someone resign from something?	1973
			What did someone resign from?	vice president of the U.S.
(5) Mr. Gorbachev badly **needs** a diversion from the serious economic problems and ethnic unrest he faces at home .	A0	Gorbachev	Who needs something?	Gorbachev/he
	A1	diversion	What does someone need?	a diversion from . . . he faces at home
	AM-ADV	badly	How does someone need something?	badly
			What does someone need sth. from?	the serious economic . . . at home
(6) Even a federal measure in June allowing houses to **add** research fees to their commissions did n't stop it .	A0	houses	What added something?	houses
	A1	fees	What was added?	research fees
	A2	to		
			When was something added?	June
(7) This year , Mr. Wathen says the firm will be able to **service** debt and still turn a modest profit .	A0	firm	Who will service something?	the firm
	A1	debt	What will be serviced?	debt
			When will something be serviced?	this year
(8). . . , the soft-spoken clarinetist announced that . . . and that it was his mother 's birthday , so he was going to **play** her favorite tune from the record .	A0	he	Who would play something?	the soft-spoken clarinetist / he
	A1	tune	What would be played?	her favorite tune from the record
			When would someone play something?	his mother 's birthday

Table 4.3: Comparison between CoNLL-2009 relations and QA-SRL annotations. While closely related to PropBank predicate-argument relations, QA pairs also contain information about within-sentence co-reference (Ex 4, 5, 8), implicit or inferred relations (Ex 3, 6, 7, 8) and roles that are not defined in PropBank (Ex 1, 4). Annotation mistakes are rare, but for example include missing pronouns (Ex 4) and prepositional attachment errors (Ex 5).

dictionary. For the **PP** field, the candidates are all the prepositions that occurred in the sentence s, and some frequently-used prepositions - *by*, *to*, *for*, *with*, and *about*. We also include preposition bigrams (e.g., *out for*) from s.

Answers are constrained to be a subset of the words in the sentence but do not necessarily have to be contiguous spans. We also allow questions to have multiple answers, which is useful for annotating graph structured dependencies such as those in examples 3 and 6 in Table 4.3.

4.1.2 Data Collection

We annotated over 3000 sentences (nearly 8,000 verbs) in total across two domains: newswire (PropBank) and Wikipedia. Table 4.4 shows the full data statistics. In the newswire domain, we sampled sentences from the English training data of CoNLL-2009 shared task [Hajič et al., 2009], excluding questions and sentences with fewer than 10 words. For the Wikipedia domain, we randomly sampled sentences from the English Wikipedia, excluding questions and sentences with fewer than 10 or more than 60 words.

In each sentence, we need to first identify the candidates for verbal predicates. In principle, a separate stage of annotation could identify verbs—but for simplicity, we instead used POS-tags. We used gold POS-tags for newswire, and predicted POS-tags (using Stanford tagger [Toutanova et al., 2003]) in Wikipedia. Annotators can choose to skip a candidate verb if they are unable to write questions for it. Annotators skipped 136 verbs (3%) in Wikipedia data and 50 verbs (1.5%) in PropBank data.

For annotation, we hired 10 part-time, non-expert annotators from Upwork (previously oDesk) and paid $10 per hour for their work. The average cost was $0.58 per verb ($1.57 per sentence) for newswire text and $0.45 per verb ($1.01 per sentence) on the Wikipedia domain. The annotators are given a short tutorial and a small set of sample annotations (about 10 sentences). Annotators were hired if they showed good understanding of English and our task. The entire screening process usually took less than 2 hours.

Writing QA pairs for each sentence takes 6 minutes on average for Wikipedia and 9 minutes on newswire, depending on the length and complexity of the sentence and the domain of the text.

Dataset	Sentences	Verbs	QAs
newswire-train	744	2020	4904
newswire-dev	249	664	1606
newswire-test	248	652	1599
Wikipedia-train	1174	2647	6414
Wikipedia-dev	392	895	2183
Wikipedia-test	393	898	2201

Table 4.4: Annotated data statistics.

	All Roles	Core	Adjuncts
Precision	81.4	85.9	59.9
Recall	86.3	89.8	63.6

Table 4.5: Agreement with gold PropBank (CoNLL-2009) for all roles, core roles, and adjuncts. Precision is the percentage of QA pairs covering exactly one PropBank relation. Recall is the percentage of PropBank relations covered by exactly one QA pair.

4.1.3 Data Quality

Agreement with Gold PropBank Data (CoNLL-2009) PropBank is the most widely used annotation of predicate-argument structure. While our annotation captures different information from PropBank, it is closely related. To investigate the similarity between the annotation schemes, we measured the overlap between the newswire domain (1241 sentences) of our QA-SRL dataset and the PropBank dataset.

For each PropBank predicate that we have annotated with our scheme, we compute the agreement between the PropBank arguments and the QA-SRL answers. We ignore modality, reference, discourse and negation roles, as they are outside the scope of our current annotation. An annotated answer is judged to match the PropBank argument if either (1) the gold argument head is within the annotated answer span, or (2) the gold argument head is a preposition and at least one of its children is within the answer span.

We measure the macro-averaged precision and recall of our annotation against PropBank, with the proportion of our QA-pairs that are match a PropBank relation, and the proportion of PropBank relations covered by our annotation. The results are shown in Table 4.5, and demonstrate high

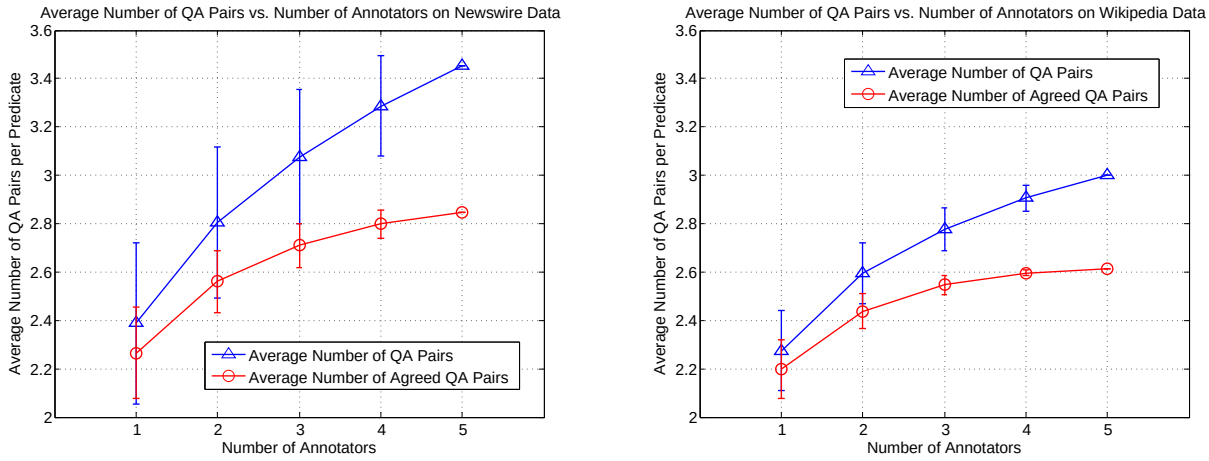


Figure 4.2: Inter-annotator agreement measured on 100 newswire sentences and 108 Wikipedia sentences, comparing the total number of annotators to the number of unique QA pairs produced and the number of agreed pairs. A pair is considered agreed if two or more annotators produced it.

overall agreement with PropBank. Agreement for core arguments [3] is especially strong, showing much of the expert linguist annotation in PropBank can be recovered with our simple scheme. Agreement for adjuncts is lower, because the annotated QAs often contain inferred roles, especially for *why*, *when* and *where* questions (See examples 4, 7 and 8 in Table 4.3). These inferred roles are typically correct, but outside of the scope of PropBank annotations; they point to exciting opportunities for future work with QA-SRL data. On the other hand, the adverbial arguments in PropBank are sometimes neglected by annotators, thus becoming a major source of recall loss.

Table 4.6 shows the overlap between our annotated question words and PropBank argument labels. There are many unsurprising correlations—*who* questions are strongly associated with PropBank agents (A0), and *where* and *when* questions correspond to PropBank temporal and locative roles, respectively. Some types of questions are divided much more evenly among PropBank roles,

[3] In PropBank, A0-A5 are the core arguments. In QA-SRL, the core arguments include QA pairs with a question that starts with *Who* or *What*.

	WHO	WHAT	WHEN	WHERE	WHY	HOW	HOW MUCH
A0	1575	414	3	5	17	28	2
A1	285	2481	4	25	20	23	95
A2	85	364	2	49	17	51	74
A3	11	62	7	8	4	16	31
A4	2	30	5	11	2	4	30
A5	0	0	0	1	0	2	0
ADV	5	44	9	2	25	27	6
CAU	0	3	1	0	23	1	0
DIR	0	6	1	13	0	4	0
EXT	0	4	0	0	0	5	5
LOC	1	35	10	89	0	13	11
MNR	5	47	2	8	4	108	14
PNC	2	21	0	1	39	7	2
PRD	1	1	0	0	0	1	0
TMP	2	51	341	2	11	20	10

Table 4.6: Co-occurrence of wh-words in QA-SRL annoations and role labels in PropBank.

such as *How much*. These cases show how our questions can produce a more easily interpretable annotation than PropBank labels, which are predicate-specific and can be difficult to understand without reference to the frame files.

Together, these results suggest that non-experts can annotate much of the information contained in PropBank, and produce a more easily interpretable annotation.

Inter-Annotator Agreement To judge the reliability of the data, we measured agreement on a portion of the data (100 sentences in the newswire domain and 108 sentences in the Wikipedia domain) annotated by five annotators.

Measuring agreement is complicated by the fact that the same question can be asked in multiple ways—for example *"Who resigned?"* and *"Who resigned from something?"*—and annotators may choose different, although usually highly overlapping, answer spans. We consider two QA pairs to be equivalent if (1) they have the same wh-word and (2) they have overlapping answer spans. In

this analysis, *Who* and *What* are considered to be the same wh-word.

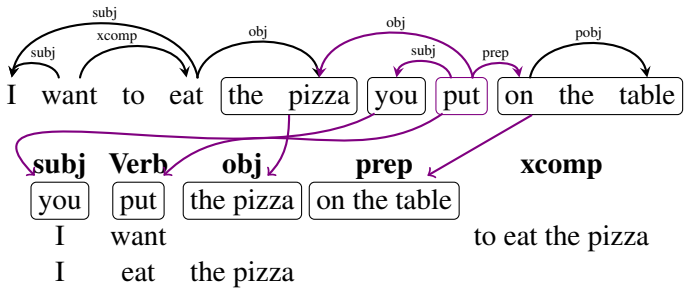
Figure 4.2 shows how the number of different QA pairs (both overall and agreed) increases with number of annotations. A QA pair is considered to be agreed upon if it is proposed by at least two of the five annotators. After five annotators, the number of agreed QA pairs starts to asymptote. A single annotator finds roughly 80% of the agreed QA pairs that are found by five annotators, suggesting that high recall can be achieved with a single stage of annotation. To further improve precision, future work should explore a second stage of annotation where annotators check each other's work, for example, by answering each other's questions.

4.2 Human-in-the-Loop Parsing with Q/A Annotation

QA-SRL is a cheap, scalable way to gather non-expert annotation, but we would also like to show its application to some other tasks, such as syntactic parsing. As a proof-of-concept, we introduced a human-in-the-loop (HITL) framework to improve CCG (Combinatory Categorial Grammar) [Hockenmaier and Steedman, 2007] parsing using QA-SRL style question-answer pairs. The HITL parser can automatically query non-expert annotators to disambiguate difficult attachment decisions. The queries are multiple choice questions, with the question prompt and candidate answer phrases automatically generated from a n-best list of parses. For example, given the sentence "*Pat ate the cake on the table that I baked last night.*", the parser can automatically generate the following query "*What did someone bake? 1) the table 2) the cake*" to disambiguate the dependencies $\langle$baked $\rightarrow$ cake$\rangle$ and $\langle$baked $\rightarrow$ table$\rangle$. We showed that by collecting human judgments and using them as soft decoding constraints, we can improve the accuracy over a near state-of-the-art CCG parser [Lewis et al., 2016].

4.2.1 From CCG Parses to Q/A Pairs

Our annotation task consists of multiple-choice *what*-questions that admit multiple answers. To generate them, we produce question–answer (QA) pairs from each parse in the 100-best scored output of a CCG parser and aggregate the results together.



Dependency	Question	Answer
want → I	What wants to eat something?	I
eat → I	What would eat something?	I
eat → pizza	What would something eat?	the pizza
put → you	What put something?	you
put → pizza	What did something put?	the pizza
on → table	What did something put something on?	the table

Figure 4.3: From a labeled dependency graph, we extract phrases corresponding to every argument of every verb using simple heuristics. We then create questions about dependencies, adding a *would* modal to untensed verbs and placing arguments to the left or right of the verb based on its CCG category. We only generate QA pairs for *subj*, *obj*, and *pobj* dependencies. To identify multiple answer options, we create QA pairs from all parses in the 100-best list and pool equivalent questions with different answers.

We designed the approach to generate queries with high **question confidence**—questions should be simple and grammatical, so annotators are more likely to answer them correctly—and high **answer uncertainty**—the parser should be uncertain about the answers, so there is potential for improvement.

Our questions only apply to core arguments of verbs where the argument phrase is an NP, which account for many of the parser's mistakes. Prepositional phrase attachment mistakes are also a large source of errors—we tried several approaches to generate questions for these, but the greater ambiguity and inconsistency among both annotators and the gold parses made it difficult to extract meaningful signal from the crowd.

Generating Question–Answer Pairs Figure 4.3 shows how we generate QA pairs. Each QA pair corresponds to a dependency such that if the answer is correct, it indicates that the dependency is in the correct parse. We determine a verb's set of arguments by the CCG supertag assigned to it in the parse (see Steedman [2000] for an introduction to CCG). For example, in Figure 4.3 the word *put* takes the category $((S\backslash NP)/PP)/NP$ (not shown), indicating that it has a subject, a prepositional phrase argument, and an object. CCG parsing assigns dependencies to each argument position, even when the arguments are re-ordered (as with *put* $\rightarrow$ *pizza*) or span long distances (as with *eat* $\rightarrow$ *I*).

To reduce the chance of parse errors causing nonsensical questions (for example, *What did the pizza put something on?*), we replace all noun phrases with *something* and delete unnecessary prepositional phrases. [4]

Grouping QA Pairs into Queries After generating QA pairs for every parse in the 100-best output of the parser, we pool the QA pairs by the head of the dependency used to generate them, its CCG category, and their question strings. We also compute marginalized scores for each question and answer phrase by summing over the scores of all the parses that generated them. Each pool becomes a query, and for each unique dependency used to generate QA pairs in that pool, we add

[4]The exception to this is with copular predicates, where we include the span of the argument in the question.

Dataset	Sentences	Covered	Queries	Q/S
CCG-Dev	1913	1155	1904	1.7
CCG-Test	2407	1460	2511	1.7
Bioinfer	500	360	680	1.9

Table 4.7: Sentence coverage, number of queries annotated, and average number of queries per sentence (Q/S).

a candidate answer to the query by choosing the answer phrase that has the highest marginalized score for that dependency. For example, if some parses generated the answer phrase *pizza* for the dependency *eat* $\rightarrow$ *pizza*, but most of the high-scoring parses generated the answer phrase *the pizza*, then only *the pizza* appears as an answer.

From the resulting queries, we filter out questions and answers whose marginalized scores are below a certain threshold and queries that only have one answer choice. This way we only ask confident questions with uncertain answer lists.

4.2.2 Reducing Cost and Improving Scalability

In our original QA-SRL paper, we hired freelancers[5] without linguistic background to annotate sentences from the newswire (Penn Treebank) and the Wikipedia domain. They are paid \$10 per hour, resulting in an average cost of \$1–\$1.5 per sentence. To further reduce cost and increase annotation speed, we automatically generate questions and a set of candidate answers, transforming the annotation task into a multiple-choice task, which is more crowdsourable. In HITL parsing, we used a simple template-based question generator, reduced annotation cost to \$0.8 per sentence, and was able to collect answers for 200 questions per hour (each question is answered by 5 different annotators) on a crowdsourcing platform[6].

[5]Freelancers are hired from `www.upwork.com`

[6]`www.crowdflower.com`

k-Agreed	CCG-Dev	CCG-Test	Bioinfer
5	48.0%	40.2%	47.7%
$\geq$ **4**	76.6%	68.0%	75.0%
$\geq$ **3**	94.9%	91.5%	94.0%

Table 4.8: The percentage of queries with at least k annotators agreeing on the exact same set of answers.

Table 4.7 shows how many sentences we asked questions for and the total number of queries annotated. We collected annotations for the development and test set for CCGbank [Hockenmaier and Steedman, 2007] as in-domain data and the test set of the Bioinfer corpus [Pyysalo et al., 2007] as out-of-domain. The CCGbank development set was used for building question generation heuristics and setting hyperparameters for re-parsing.

5 annotators answered each query; on CCGbank we required 85% accuracy on test questions and on Bioinfer we set the threshold at 80% because of the difficulty of the sentences. Table 4.8 shows inter-annotator agreement. Annotators unanimously chose the same set of answers for over 40% of the queries; an absolute majority is achieved for over 90% of the queries.

4.2.3 Reparsing with Q/A Pairs

To improve the output of the parser, we re-parse each sentence with an augmented scoring function that penalizes parses for disagreeing with annotators' choices. If q is a question, a is an answer to q, d is the dependency that produced the QA pair $\langle q, a \rangle$, and $v(a)$ annotators chose a, we add re-parsing constraints as follows:

- If $v(\textit{None of the above}) \geq T^{+}$, penalize parses that agree with q's supertag on the verb by w^{t} amount.

- If $v(a) \leq T^{-}$, penalize parses containing d by w^{-}.

- If $v(a) \geq T^+$, penalize parses that do not contain d by w^+.

where T^+, T^-, w^t, w^-, and w^+ are hyperparameters. We incorporate these penalties into the parsing model during decoding. By using soft constraints, we mitigate the risk of incorrect annotations worsening a high-confidence parse.

We use Lewis et al. [2016]'s CCG parser (which is the state of the art by then) for our baseline. We chose the following set of hyperparameters based on performance on development data (CCG-Dev): $w^+ = 2.0$, $w^- = 1.5$, $w^t = 1.0$, $T^+ = 3$, $T^- = 0$. In the Bioinfer dataset, we found during development that the pronoun/subspan heuristics were not as useful, so we did not use them in re-parsing.

Data	**All**		**Changed**		
	Lewis16	**HITL**	**Lewis16**	**HITL**	**Pct.**
CCG-Dev	87.9	88.4	83.9	87.1	12%
CCG-Test	88.1	88.3	84.2	85.9	10%
Bioinfer	82.2	82.8	-	-	-

Table 4.9: CCG parsing accuracy with human in the loop (HITL) versus the state-of-the-art baseline (Lewis16) in terms of labeled F1 score. For both in-domain and out-domain, we have a modest gain over the entire corpus. We also show improvements of CCG parsing accuracy on changed sentences (Changed) for in-domain data. We achieved significant improvement over the 10%–12% (Pct.) sentences that were changed by re-parsing.

Results Table 4.9 shows our end-to-end parsing results. The larger improvement on out-of-domain sentences shows the potential for using our method for domain adaptation. There is a much smaller improvement on test data than development data, which may be related to the lower annotator agreement reported in Table 4.8.

There was much larger improvement (1.7 F1) on the subset of sentences that are changed after

re-parsing, as shown in Table 4.9. This suggests that our method could be effective for semi-supervised learning or re-training parsers.

4.3 Summary

In this chapter, we introduced QA-SRL, an annotation scheme that allows non-experts to annotate predicate-argument structures with question-answer pairs. Based on the idea of "using natural language to annotated natural langueage", we avoid defining detailed annotation guideline or large frame inventories, making the data collection process cheaper, more scalable, and more intuitive to people without linguistic training. Comparisons against the PropBank dataset showed that the QA-SRL data we have collected are high-quality. In our follow-up work, human-in-the-loop parsing, we further demonstrated that this kind of question-answer-driven annotation can be used to improve the performance of a state-of-the-art CCG parser.

With QA-SRL, we have taken the first steps towards the goal of building robust SRL systems for all domains. In the future, we would like to build neural models that are able to learn from and to predict QA-SRL data. For example, we can use this model to generate natural language questions and gather validation judgments from crowd workers, expanding the QA-SRL datasets with even lower cost. In Chapter 5 and 6, we will describe two accurate neural SRL models for span-based SRL. Although these models are designed to predict the PropBank SRL, they can be potentially adapted for learning QA-SRL structures [FitzGerald et al., 2018].

Chapter 5

LEARNING SRL WITHOUT SYNTAX

In the previous chapter, we described an annotation scheme that enables us to collect large amounts of training data, which would be beneficial for training deep neural models. However, the annotations in this data are in natural language and only focus on semantics. They do not include, for example, the detailed syntactic annotations that are typically required to train semantic role labeling systems.

In this chapter, we present a new deep learning method that can learn from semantic supervision alone, using labels from PropBank that are closely related to our annotation scheme, as discussed in the last chapter. We introduce DeepSRL, a BIO-based model that significantly improves the previous state of the art of PropBank SRL. We use a stacked highway BiLSTM architecture with constrained decoding, while observing a number of recent best practices for initialization and regularization. Our 8-layer ensemble model achieves 83.2 F1 on the CoNLL 2005 test set and 83.4 F1 on CoNLL 2012, roughly a 10% relative error reduction over the previous state of the art. We will also present extensive empirical analysis of these gains, showing that (1) deep models excel at recovering long-distance dependencies but can still make surprisingly obvious errors, and (2) that there is still room for syntactic parsers to improve these results. This chapter is based on work originally described in He et al. [2017].

5.1 Introduction

Recently breakthroughs on SRL involving end-to-end deep models for SRL without syntactic input [Zhou and Xu, 2015, Marcheggiani et al., 2017] seem to overturn the long-held belief that syntactic parsing is a prerequisite for this task [Punyakanok et al., 2008]. In this chapter, we show that this result can be pushed further using deep highway bidirectional LSTMs with constrained decoding,

again significantly moving the state of the art (another 2 points on CoNLL 2005). We also present a careful empirical analysis to determine what works well and what might be done to progress even further.

Our DeepSRL model combines a number of best practices in the recent deep learning literature. Following Zhou and Xu [2015], we treat SRL as a BIO tagging problem and use deep bidirectional LSTMs. However, we differ by (1) simplifying the input and output layers, (2) introducing highway connections [Srivastava et al., 2015, Zhang et al., 2016], (3) using recurrent dropout [Gal and Ghahramani, 2016], (4) decoding with BIO-constraints, and (5) ensembling with a product of experts. Our model gives a 10% relative error reduction over previous state of the art on the test sets of CoNLL 2005 and 2012. We also report performance with predicted predicates to encourage future exploration of end-to-end SRL systems.

We present detailed error analyses to better understand the performance gains, including (1) design choices on architecture, initialization, and regularization that have a surprisingly large impact on model performance; (2) different types of prediction errors showing, e.g., that deep models excel at predicting long-distance dependencies but still struggle with known challenges such as PP-attachment errors and adjunct-argument distinctions; (3) the role of syntax, showing that there is significant room for improvement given oracle syntax but errors from existing automatic parsers prevent effective use in SRL.

5.2 The DeepSRL Model

Two major factors contribute to the success of our deep SRL model: (1) applying recent advances in training deep recurrent neural networks such as highway connections [Srivastava et al., 2015] and RNN-dropouts [Gal and Ghahramani, 2016],[1] and (2) using an A* decoding algorithm [Lewis and Steedman, 2014, Lee et al., 2016b] to enforce structural consistency at prediction time without adding more complexity to the training process.

Formally, our task is to predict a sequence $\boldsymbol{y}$ given a sentence-predicate pair $(\boldsymbol{w}, v)$ as input.

[1]We thank Mingxuan Wang for suggesting highway connections with simplified inputs and outputs. Part of our model is extended from his unpublished implementation.

Each $y_i \in \boldsymbol{y}$ belongs to a discrete set of BIO tags $\boldsymbol{T}$. Words outside argument spans have the tag O, and words at the beginning and inside of argument spans with role r have the tags B_r and I_r respectively. Let $n = |\boldsymbol{w}| = |\boldsymbol{y}|$ be the length of the sequence.

Predicting an SRL structure under our model involves finding the highest-scoring tag sequence over the space of all possibilities $\mathcal{Y}$:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{w}, \boldsymbol{y}, v) \tag{5.1}$$

We use a deep bidirectional LSTM (BiLSTM) to learn a locally decomposed scoring function conditioned on the input: $\sum_{t=1}^{n} \log p(y_t \mid \boldsymbol{w}, v)$.

To incorporate additional information (e.g., structural consistency, syntactic input), we augment the scoring function with penalization terms:

$$f(\boldsymbol{w}, \boldsymbol{y}) = \sum_{t=1}^{n} \log p(y_t \mid \boldsymbol{w}, v) - \sum_{c \in \mathcal{C}} c(\boldsymbol{w}, y_{1:t}, v) \tag{5.2}$$

Each constraint function c applies a *non-negative* penalty given the input $\boldsymbol{w}$ and a length-t prefix $y_{1:t}$. These constraints can be hard or soft depending on whether the penalties are finite.

Deep BiLSTM Model

Our model computes the distribution over tags using stacked BiLSTMs, which we define as follows:

$$\boldsymbol{i}_{l,t} = \sigma(\mathbf{W}_{\mathrm{i}}^{l}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}_{\mathrm{i}}^{l}) \tag{5.3}$$

$$\boldsymbol{o}_{l,t} = \sigma(\mathbf{W}_{\mathrm{o}}^{l}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}_{\mathrm{o}}^{l}) \tag{5.4}$$

$$\boldsymbol{f}_{l,t} = \sigma(\mathbf{W}_{\mathrm{f}}^{l}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}_{\mathrm{f}}^{l} + 1) \tag{5.5}$$

$$\tilde{\boldsymbol{c}}_{l,t} = \tanh(\mathbf{W}_{\mathrm{c}}^{l}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}_{\mathrm{c}}^{l}) \tag{5.6}$$

$$\boldsymbol{c}_{l,t} = \boldsymbol{i}_{l,t} \circ \tilde{\boldsymbol{c}}_{l,t} + \boldsymbol{f}_{l,t} \circ \boldsymbol{c}_{t+\delta_l} \tag{5.7}$$

$$\boldsymbol{h}_{l,t} = \boldsymbol{o}_{l,t} \circ \tanh(\boldsymbol{c}_{l,t}) \tag{5.8}$$

where $x_{l,t}$ is the input to the LSTM at layer l and timestep t. δ_l is either 1 or -1, indicating the directionality of the LSTM at layer l.

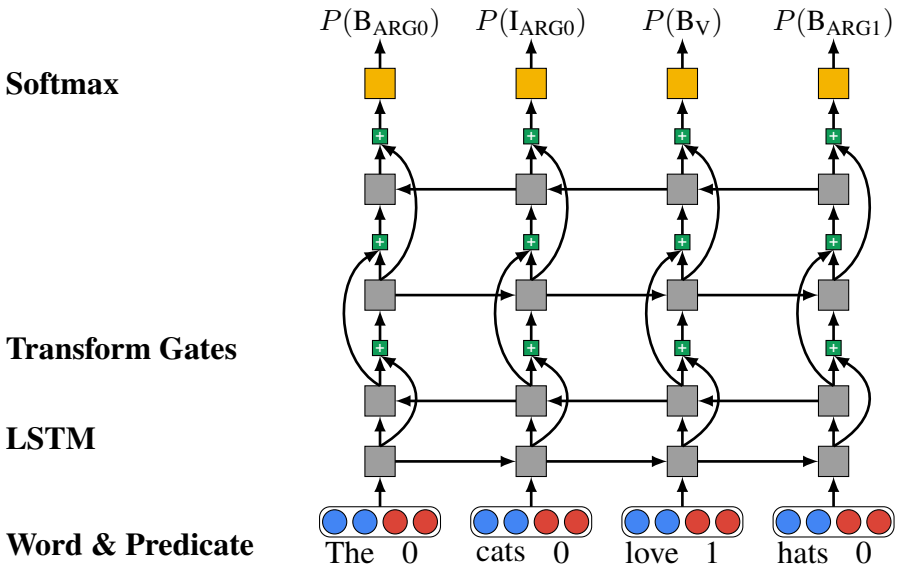


Figure 5.1: Highway LSTM with four layers. The curved connections represent highway connections, and the plus symbols represent transform gates that control inter-layer information flow.

To stack the LSTMs in an interleaving pattern, as proposed by Zhou and Xu [2015], the layer-specific inputs $x_{l,t}$ and directionality δ_l are arranged in the following manner:

$$\boldsymbol{x}_{l,t} = \begin{cases} [\mathbf{W}_{\text{emb}}(w_t), \mathbf{W}_{\text{mask}}(t = v)] & l = 1 \\ \boldsymbol{h}_{l-1,t} & l > 1 \end{cases} \tag{5.9}$$

$$\delta_l = \begin{cases} 1 & \text{if } l \text{ is even} \\ -1 & \text{otherwise} \end{cases} \tag{5.10}$$

The input vector $\boldsymbol{x}_{1,t}$ is the concatenation of token w_t's word embedding and an embedding of the binary feature $(t = v)$ indicating whether w_t word is the given predicate.

Finally, the locally normalized distribution over output tags is computed via a softmax layer:

$$p(y_t \mid \boldsymbol{x}, v) \propto \exp(\mathbf{W}^y_{\text{tag}} \boldsymbol{h}_{L,t} + \boldsymbol{b}_{\text{tag}}) \tag{5.11}$$

Highway Connections To alleviate the vanishing gradient problem when training deep BiLSTMs, we use gated highway connections [Zhang et al., 2016, Srivastava et al., 2015]. We include *transform gates* $\boldsymbol{r}_t$ to control the weight of linear and non-linear transformations between layers (See Figure 5.1). The output $\boldsymbol{h}_{l,t}$ is changed to:

$$\boldsymbol{r}_{l,t} = \sigma(\mathbf{W}_{\mathrm{r}}^{l}[\boldsymbol{h}_{l,t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\mathrm{r}}^{l}) \tag{5.12}$$

$$\boldsymbol{h}'_{l,t} = \boldsymbol{o}_{l,t} \circ \tanh(\boldsymbol{c}_{l,t}) \tag{5.13}$$

$$\boldsymbol{h}_{l,t} = \boldsymbol{r}_{l,t} \circ \boldsymbol{h}'_{l,t} + (1 - \boldsymbol{r}_{l,t}) \circ \mathbf{W}_{\mathrm{h}}^{l}\boldsymbol{x}_{l,t} \tag{5.14}$$

Recurrent Dropout To reduce over-fitting, we use dropout as described in Gal and Ghahramani [2016]. A shared dropout mask $\boldsymbol{z}_l$ is applied to the hidden state:

$$\widetilde{\boldsymbol{h}}_{l,t} = \boldsymbol{r}_{l,t} \circ \boldsymbol{h}'_{l,t} + (1 - \boldsymbol{r}_{l,t}) \circ \mathbf{W}_{\mathrm{h}}^{l}\boldsymbol{x}_{l,t} \tag{5.15}$$

$$\boldsymbol{h}_{l,t} = \boldsymbol{z}_l \circ \widetilde{\boldsymbol{h}}_{l,t} \tag{5.16}$$

$\boldsymbol{z}_l$ is shared across timesteps at layer l to avoid amplifying the dropout noise along the sequence.

5.3 Constrained Decoding for SRL

Constrained A Decoding*

The approach described so far does not model any dependencies between the output tags. To incorporate constraints on the output structure at decoding time, we use A* search over tag prefixes for decoding. Starting with an empty sequence, the tag sequence is built from left to right. The score for a partial sequence with length t is defined as:

$$f(\boldsymbol{w}, y_{1:t}, v) = \sum_{i=1}^{t} \log p(y_i \mid \boldsymbol{w}, v) - \sum_{c \in \mathcal{C}} c(\boldsymbol{w}, y_{1:i}, v) \tag{5.17}$$

An admissible A* heuristic can be computed efficiently by summing over the best possible tags for all timesteps after t:

$$g(\boldsymbol{w}, y_{1:t}, v) = \sum_{i=t+1}^{n} \max_{y_i \in \boldsymbol{T}} \log p(y_i \mid \boldsymbol{w}, v) \tag{5.18}$$

Exploration of the prefixes is determined by an agenda $\mathcal{A}$ which is sorted by $f(\boldsymbol{w}, y_{1:t}) + g(\boldsymbol{w}, y_{1:t})$. In the worst case, A* explores exponentially many prefixes, but because the distribution $p(y_t \mid \boldsymbol{w})$ learned by the BiLSTM models is very peaked, the algorithm is efficient in practice. We list some example constraints as follows:

BIO Constraints These constraints reject any sequence that does not produce valid BIO transitions, such as B_{ARG0} followed by I_{ARG1}.

SRL Constraints Punyakanok et al. [2008], Täckström et al. [2015] described a list of SRL-specific global constraints:

- Unique core roles (U): Each core role (ARG0-ARG5, ARGA) should appear at most once for each predicate.

- Continuation roles (C): A continuation role C-X can exist only when its base role X is realized before it.

- Reference roles (R): A reference role R-X can exist only when its base role X is realized (not necessarily before R-X).

We only enforce U and C constraints, since the R constraints are more commonly violated in gold data and enforcing them results in worse performance (see discussion in Section 5.6).

Syntactic Constraints We can enforce consistency with a given parse tree by rejecting or penalizing arguments that are not constituents. In Section 5.6, we will discuss the motivation behind using syntactic constraints and experimental results using both predicted and gold syntax.

5.4 PropBank and OntoNotes Results

We measure the performance of our SRL system on two PropBank-style, span-based SRL datasets: CoNLL-2005 [Carreras and Màrquez, 2005] and CoNLL-2012 [Pradhan et al., 2013][2]. Both datasets provide gold predicates (their index in the sentence) as part of the input. Therefore, each provided predicate corresponds to one training/test tag sequence. We follow the train-development-test split for both datasets and use the official evaluation script from the CoNLL 2005 shared task for evaluation on both datasets.

Model Setup

Our network consists of 8 BiLSTM layers (4 forward LSTMs and 4 reversed LSTMs) with 300-dimensional hidden units, and a softmax layer for predicting the output distribution.

Initialization All the weight matrices in BiLSTMs are initialized with random orthonormal matrices as described in Saxe et al. [2013].

All tokens are lower-cased and initialized with 100-dimensional GloVe embeddings pre-trained on 6B tokens [Pennington et al., 2014] and updated during training. Tokens that are not covered by GloVe are replaced with a randomly initialized UNK embedding.

Training We use Adadelta [Zeiler, 2012] with $\epsilon = 1e^{-6}$ and $\rho = 0.95$ and mini-batches of size 80. We set RNN-dropout probability to 0.1 and clip gradients with norm larger than 1. All the models are trained for 500 epochs with early stopping based on development results. [3]

The network for predicate detection (Section 5.5) contains 2 BiLSTM layers with 100-dimensional hidden units, and is trained for 30 epochs. For end-to-end evaluation, all arguments predicted for the false positive predicates are counted as precision loss, and all arguments for the false negative predicates are considered as recall loss.

---

[2]We used the version of OntoNotes downloaded at: `http://cemantix.org/data/ontonotes.html`.

[3]Training the full model on CoNLL 2005 takes about 5 days on a single Titan X Pascal GPU.

Ensembling We use a product of experts [Hinton, 2002] to combine predictions of 5 models, each trained on 80% of the training corpus and validated on the remaining 20%. For the CoNLL 2012 corpus, we split the training data from each sub-genre into 5 folds, such that the training data will have similar genre distributions.

Constrained Decoding We experimented with different types of constraints on the CoNLL 2005 and 2012 development sets. Only the BIO hard constraints significantly improve over the ensemble model. Therefore, in our final results, we only use BIO hard constraints during decoding. [4]

	Development				WSJ Test				Brown Test				Combined
Method	P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.	F1
Ours (PoE)	**83.1**	**82.4**	**82.7**	**64.1**	**85.0**	**84.3**	**84.6**	**66.5**	**74.9**	**72.4**	**73.6**	**46.5**	**83.2**
Ours	81.6	81.6	81.6	62.3	83.1	83.0	83.1	64.3	72.9	71.4	72.1	44.8	81.6
Zhou	79.7	79.4	79.6	-	82.9	82.8	82.8	-	70.7	68.2	69.4	-	81.1
FitzGerald (Struct.,PoE)	81.2	76.7	78.9	55.1	82.5	78.2	80.3	57.3	74.5	70.0	72.2	41.3	-
Täckström (Struct.)	81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	74.3	68.6	71.3	39.8	-
Toutanova (Ensemble)	-	-	78.6	58.7	81.9	78.8	80.3	60.1	-	-	68.8	40.8	-
Punyakanok (Ensemble)	80.1	74.8	77.4	50.7	82.3	76.8	79.4	53.8	73.4	62.9	67.8	32.3	77.9

Table 5.1: Experimental results on CoNLL 2005, in terms of precision (P), recall (R), F1 and percentage of completely correct predicates (Comp.). We report results of our best single and ensemble (PoE) model. The comparison models are Zhou and Xu [2015], FitzGerald et al. [2015], Täckström et al. [2015], Toutanova et al. [2008] and Punyakanok et al. [2008].

[4] A* search in this setting finds the optimal sequence for all sentences and is therefore equivalent to Viterbi decoding.

	Development				Test			
Method	P	R	F1	Comp.	P	R	F1	Comp.
Ours (PoE)	**83.5**	**83.2**	**83.4**	**67.5**	**83.5**	**83.3**	**83.4**	**68.5**
Ours	81.8	81.4	81.5	64.6	81.7	81.6	81.7	66.0
Zhou	-	-	81.1	-	-	-	81.3	-
FitzGerald (Struct.,PoE)	81.0	78.5	79.7	60.9	81.2	79.0	80.1	62.6
Täckström (Struct.)	80.5	77.8	79.1	60.1	80.6	78.2	79.4	61.8
Pradhan (revised)	-	-	-	-	78.5	76.6	77.5	55.8

Table 5.2: Experimental results on CoNLL 2012 in the same metrics as above. We compare our best single and ensemble (PoE) models against Zhou and Xu [2015], FitzGerald et al. [2015], Täckström et al. [2015] and Pradhan et al. [2013].

Results

In Table 5.1 and 5.2, we compare our best single and ensemble model with previous work. Our ensemble (PoE) has an absolute improvement of 2.1 F1 on both CoNLL 2005 and CoNLL 2012 over the previous state of the art. Our single model also achieves more than a 0.4 improvement on both datasets. In comparison with the best reported results, our percentage of completely correct predicates improves by 5.9 points. While the continuing trend of improving SRL without syntax seems to suggest that neural end-to-end systems no longer needs parsers, our analysis in Section 5.6 will show that accurate syntactic information can improve these deep models.

5.5 SRL without Gold Predicates

Predicate Detection

While the CoNLL 2005 shared task assumes gold predicates as input [Carreras and Màrquez, 2005], this information is not available in many downstream applications. We propose a simple

	Predicate Detection			End-to-end SRL (Single)			End-to-end SRL (PoE)			
Dataset	P	R	F1	P	R	F1	P	R	F1	Δ F1
CoNLL 2005 Dev.	97.4	97.4	97.4	80.3	80.4	80.3	81.8	81.2	81.5	-1.2
WSJ Test	94.5	98.5	96.4	80.2	82.3	81.2	82.0	83.4	82.7	-1.9
Brown Test	89.3	95.7	92.4	67.6	69.6	68.5	69.7	70.5	70.1	-3.5
CoNLL 2012 Dev.	88.7	90.6	89.7	74.9	76.2	75.5	76.5	77.8	77.2	-6.2
CoNLL 2012 Test	93.7	87.9	90.7	78.6	75.1	76.8	80.2	76.6	78.4	-5.0

Table 5.3: Predicate detection performance and end-to-end SRL results using predicted predicates. Δ F1 shows the absolute performance drop compared to our best ensemble model with gold predicates.

model for end-to-end SRL, where the system first predicts a set of predicate words $\boldsymbol{v}$ from the input sentence $\boldsymbol{w}$. Then each predicate in $\boldsymbol{v}$ is used as an input to argument prediction. We independently predict whether each word in the sentence is a predicate, using a binary softmax over the outputs of a bidirectional LSTM trained to maximize the likelihood of the gold labels.

Table 5.3 shows the predicate detection F1 as well as end-to-end SRL results with predicted predicates.[5] On CoNLL 2005, the predicate detector achieved over 96 F1, and the final SRL results only drop 1.2-3.5 F1 compared to using the gold predicates. However, on CoNLL 2012, the predicate detector has only about 90 F1, and the final SRL results decrease by up to 6.2 F1. This is at least in part due to the fact that CoNLL 2012 contains some nominal and copula predicates [Weischedel et al., 2013], making predicate identification a more challenging problem.

Ablations Figure 5.2 shows learning curves of our model ablations on the CoNLL 2005 development set. We ablate our full model by removing highway connections, RNN-dropout, and or-

[5]The frame identification numbers reported in Pradhan et al. [2013] are not comparable, due to errors in the original release of the data, as mentioned in Täckström et al. [2015].

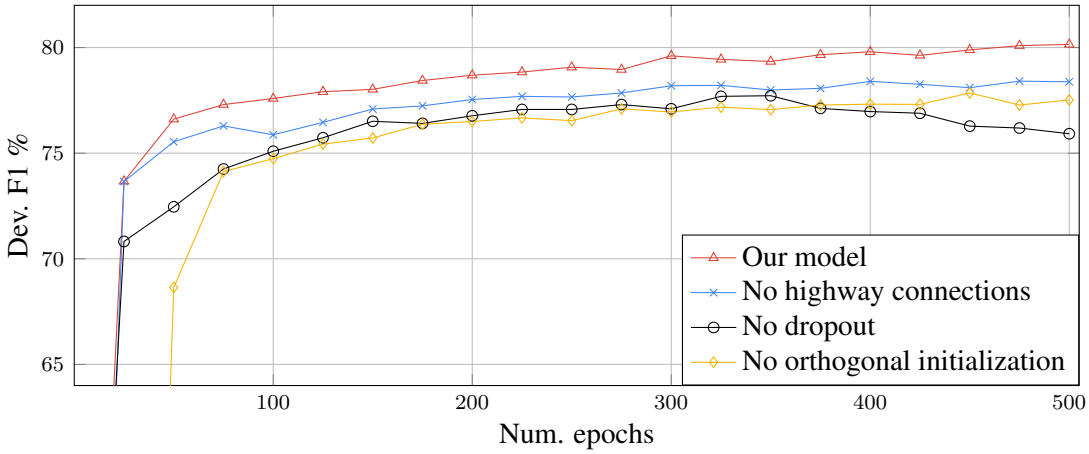


Figure 5.2: Smoothed learning curve of various ablations. The combination of highway layers, orthonormal parameter initialization and recurrent dropout is crucial to achieving strong performance. The numbers shown here are without constrained decoding.

thonormal initialization independently. Without dropout, the model overfits at around 300 epochs at 78 F1. Orthonormal parameter initialization is surprisingly important—without this, the model achieves only 65 F1 within the first 50 epochs. All 8 layer ablations suffer a loss of more than 1.7 in absolute F1 compared to the full model.

5.6 Error Analysis: What Works and What's Next

To better understand our deep SRL model and its relation to previous work, we address the following questions with a suite of empirical analyses:

- What is the model good at and what kinds of mistakes does it make?
- How well do LSTMs model global structural consistency, despite conditionally independent tagging decisions?
- Is our model implicitly learning syntax, and could explicitly modeling syntax still help?

Operation	Description	%
Fix Labels	Correct the span label if its boundary matches gold.	29.3
Move Arg.	Move a unique core argument to its correct position.	4.5
Merge Spans	Combine two predicted spans into a gold span if they are separated by at most one word.	10.6
Split Spans	Split a predicted span into two gold spans that are separated by at most one word.	14.7
Fix Boundary	Correct the boundary of a span if its label matches an overlapping gold span.	18.0
Drop Arg.	Drop a predicted argument that does not overlap with any gold span.	7.4
Add Arg.	Add a gold argument that does not overlap with any predicted span.	11.0

Table 5.4: Oracle transformations paired with the relative error reduction after each operation. All the operations are permitted only if they do not cause any overlapping arguments.

All the analysis in this section is done on the CoNLL 2005 development set with gold predicates, unless otherwise stated. We are also able to compare to previous systems whose model predictions are available online [Punyakanok et al., 2005, Pradhan et al., 2005].[6]

Error Types Breakdown

Inspired by Kummerfeld et al. [2012], we define a set of oracle transformations that fix various prediction errors *sequentially* and observe the relative improvement after each operation (see Table 5.4). Figure 5.3 shows how our work compares to the previous systems in terms of different types of mistakes. While our model makes a similar number of labeling errors to traditional syntax-based systems, it has far fewer missing arguments (perhaps due to parser errors making some arguments difficult to recover for syntax-based systems).

[6]Model predictions of CoNLL 2005 systems: `http://www.cs.upc.edu/~srlconll/st05/st05.html`

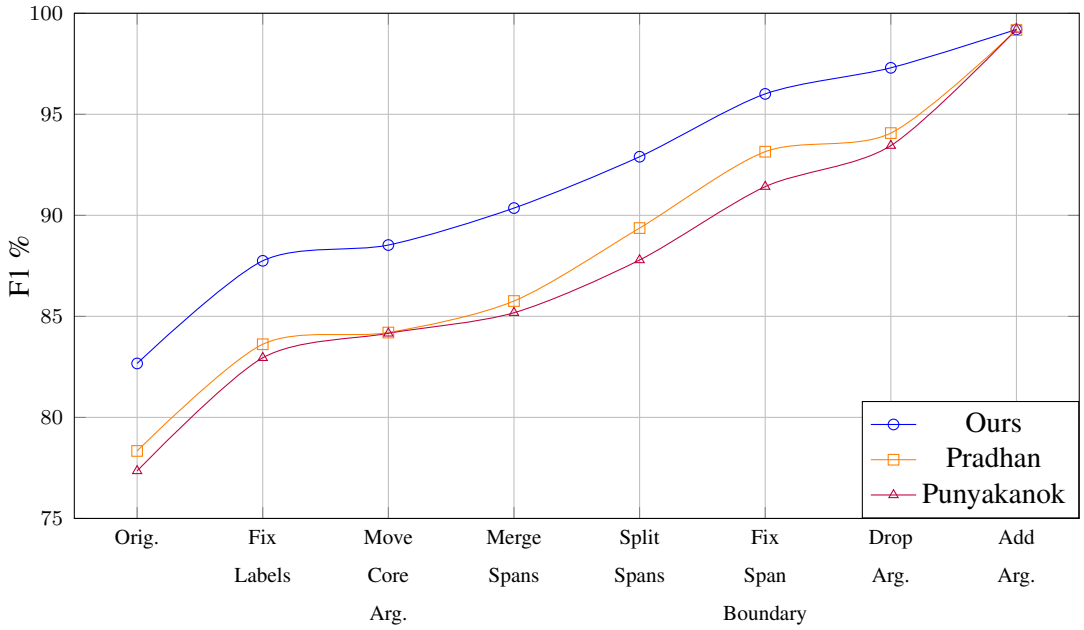


Figure 5.3: Performance after doing each type of oracle transformation in sequence, compared to two strong non-neural baselines. The gap is closed after the *Add Arg.* transformation, showing how our approach is gaining from predicting more arguments than traditional systems.

Label Confusion As shown in Table 5.4, our system most commonly makes labeling errors, where the predicted span is an argument but the role was incorrectly labeled. Table 5.5 shows a confusion matrix for the most frequent labels. The model often confuses ARG2 with AM-DIR, AM-LOC and AM-MNR. These confusions can arise due to the use of ARG2 in many verb frames to represent semantic relations such as direction or location. For example, ARG2 in the frame *move.01* is defined as *Arg2-GOL: destination*. [7] This type of argument-adjunct distinction is known to be difficult [Kingsbury et al., 2002], and it is not surprising that our neural model has many such failure cases.

[7]Source: Unified verb index: `http://verbs.colorado.edu`.

pred. \ gold	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	55	11	13	4	0	0	0	0	0
A1	78	-	46	0	0	22	11	10	25	14
A2	11	23	-	48	15	56	33	41	25	0
A3	3	2	2	-	4	0	0	0	25	14
ADV	0	0	0	4	-	0	15	29	25	36
DIR	0	0	5	4	0	-	11	2	0	0
LOC	5	9	12	0	4	0	-	10	0	14
MNR	3	0	12	26	33	0	0	-	0	21
PNC	0	3	5	4	0	11	4	2	-	0
TMP	0	8	5	0	41	11	26	6	0	-

Table 5.5: Confusion matrix for labeling errors, showing the percentage of predicted labels for each gold label. We only count predicted arguments that match gold span boundaries.

Attachment Mistakes A second common source of error is reflected by the *Merge Spans* transformation (10.6%) and the *Split Spans* transformation (14.7%). These errors are closely tied to prepositional phrase (PP) attachment errors, which are also known to be some of the biggest challenges for linguistic analysis [Kummerfeld et al., 2012]. Figure 5.4 shows the distribution of syntactic span labels involved in an attachment mistake, where 62% of the syntactic spans are prepositional phrases. For example, in *Sumitomo financed the acquisition from Sears*, our model mistakenly labels the prepositional phrase *from Sears* as the ARG2 of *financed*, whereas it should instead attach to *acquisition*.

Long-range Dependencies

To analyze the model's ability to capture long-range dependencies, we compute the F1 of our model on arguments with various distances to the predicate. Figure 5.5 shows that performance tends to degrade, for all models, for arguments further from the predicate. Interestingly, the gap between shallow and deep models becomes much larger for the long-distance predicate-argument structures. The absolute gap between our 2 layer and 8 layer models is 3-4 F1 for arguments that are within 2 words to the predicate, and 5-6 F1 for arguments that are farther away from the predicate.

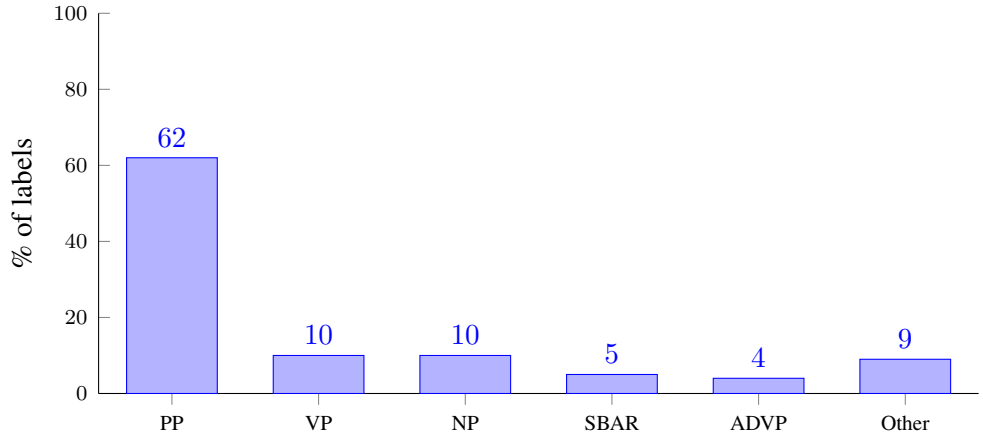


Figure 5.4: For cases where our model either splits a gold span into two ($Z \rightarrow XY$) or merges two gold constituents ($XY \rightarrow Z$), we show the distribution of syntactic labels for the Y span. Results show the major cause of these errors is inaccurate prepositional phrase attachment.

Surprisingly, the neural model performance deteriorates less severely on long-range dependencies than traditional syntax-based models.

Structural Consistency

We can quantify two types of structural consistencies: the BIO constraints and the SRL-specific constraints. Via our ablation study, we show that deeper BiLSTMs are better at enforcing these structural consistencies, although not perfectly.

BIO Violations The BIO format requires argument spans to begin with a B tag. Any I tag directly following an O tag or a tag with different label is considered a violation. Table 5.6 shows the number of BIO violations per token for BiLSTMs with different depths. The number of BIO violations decreases when we use a deeper model. The gap is biggest between 2-layer and 4-layer models, and diminishes after that.

Although the deeper models generate impressively accurate token-level predictions, they still

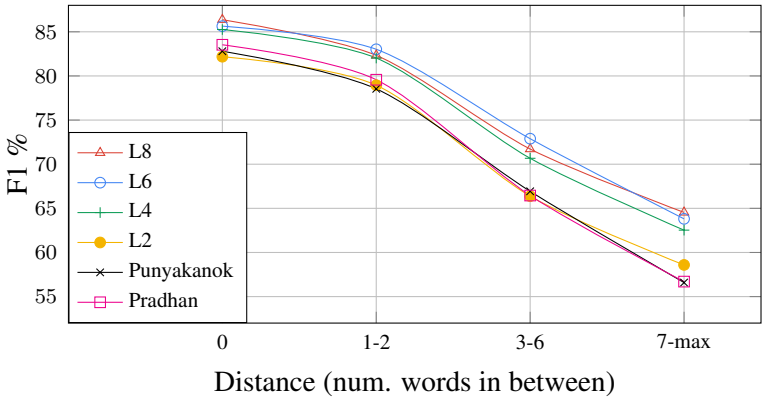


Figure 5.5: F1 by surface distance between predicates and arguments. Performance degrades least rapidly on long-range arguments for the deeper neural models.

	Accuracy		Violations	Avg. Entropy	
Model (no BIO)	F1	Token	BIO	All	BIO
L8+PoE	81.5	91.5	0.07	0.02	0.72
L8	80.5	90.9	0.07	0.02	0.73
L6	80.1	90.3	0.06	0.02	0.72
L4	79.1	90.2	0.08	0.02	0.70
L2	74.6	88.4	0.18	0.03	0.66

Table 5.6: Comparison of BiLSTM models without BIO decoding. We compare F1 and token-level accuracy (Token), averaged BIO violations per token (BIO), overall model entropy (All) model entropy at tokens involved in BIO violations (BIO). Increasing the depth of the model beyond 4 does not produce more structurally consistent output, emphasizing the need for constrained decoding.

make enough BIO errors to significantly hurt performance—when these constraints are simple enough to be enforced by trivial rules. We compare the average entropy between tokens involved in BIO violations with the averaged entropy of all tokens. For the 8-layer model, the average entropy

on these tokens is 30 times higher than the averaged entropy on all tokens. This suggests that BIO inconsistencies occur when there is some ambiguity. For example, if the model is unsure whether two consecutive words should belong to an ARG0 or ARG1, it might generate inconsistent BIO sequences such as B_{ARG0}, I_{ARG1} when decoding at the token level. Using BIO-constrained decoding can resolve this ambiguity and result in a structurally consistent solution.

SRL Structure Violations The model predictions can also violate the SRL-specific constraints commonly used in prior work [Punyakanok et al., 2008, Täckström et al., 2015]. As shown in Table 5.7, the model occasionally violates these SRL constraints. With our constrained decoding algorithm, it is straightforward to enforce the unique core roles (U) and continuation roles (C) constraints during decoding. The constrained decoding results are shown with the model named *L8+PoE+SRL* in Table 5.7.

Although the violations are eliminated, the performance does not significantly improve. This is mainly due to two factors: (1) the model often already satisfies these constraints on its own, so the number of violations to be fixed are relatively small, and (2) the gold SRL structure sometimes violates the constraints and enforcing hard constraints can hurt performance.

Can Syntax Still Help SRL?

The Propbank-style SRL formalism is closely tied to syntax [Bonial et al., 2010, Weischedel et al., 2013]. In Table 5.7, we show that 98.7% of the gold SRL arguments match an unlabeled constituent in the gold syntax tree. Similar to some recent work [Zhou and Xu, 2015], our model achieves strong performance without directly modeling syntax. A natural question follows: are neural SRL models implicitly learning syntax? Table 5.7 shows the trend of deeper models making predictions that are more consistent with the gold syntax in terms of span boundaries. With our best model (*L8+PoE*), 94.3% of the predicted arguments spans are part of the gold parse tree. This consistency is on par with previous CoNLL 2005 systems that directly model constituency and use predicted parse trees as features (Punyakanok, 95.3% and Pradhan, 93.0%).

			SRL-Violations		
Model or Oracle	F1	Syn %	U	C	R
Gold	100.0	98.7	24	0	61
L8+PoE	82.7	94.3	37	3	68
L8	81.6	94.0	48	4	73
L6	81.4	93.7	39	3	85
L4	80.5	93.2	51	3	84
L2	77.2	91.3	96	5	72
L8+PoE+SRL	82.8	94.2	5	1	68
L8+PoE+AutoSyn	83.2	96.1	113	3	68
L8+PoE+GoldSyn	85.0	97.6	102	3	68
Punyakanok	77.4	95.3	0	0	0
Pradhan	78.3	93.0	84	3	58

Table 5.7: Comparison of models with different depths and decoding constraints (in addition to BIO) as well as two previous systems. We compare F1, unlabeled agreement with gold constituency (Syn%) and each type of SRL-constraint violations (**U**nique core roles, **C**ontinuation roles and **R**eference roles). Our best model produces a similar number of constraint violations to the gold annotation, explaining why deterministically enforcing these constraints is not helpful.

	CoNLL-05		CoNLL-2012 Dev.						
	Dev.	Test	BC	BN	NW	MZ	PT	TC	WB
L8+PoE	82.7	84.6	81.4	82.8	82.8	80.4	93.6	84.8	81.0
+AutoSyn	83.2	84.8	81.5	82.8	83.2	80.6	93.7	84.9	81.1

Table 5.8: F1 on CoNLL 2005, and the development set of CoNLL 2012, broken down by genres. Syntax-constrained decoding (+AutoSyn) shows bigger improvement on in-domain data (CoNLL 05 and CoNLL 2012 NW).

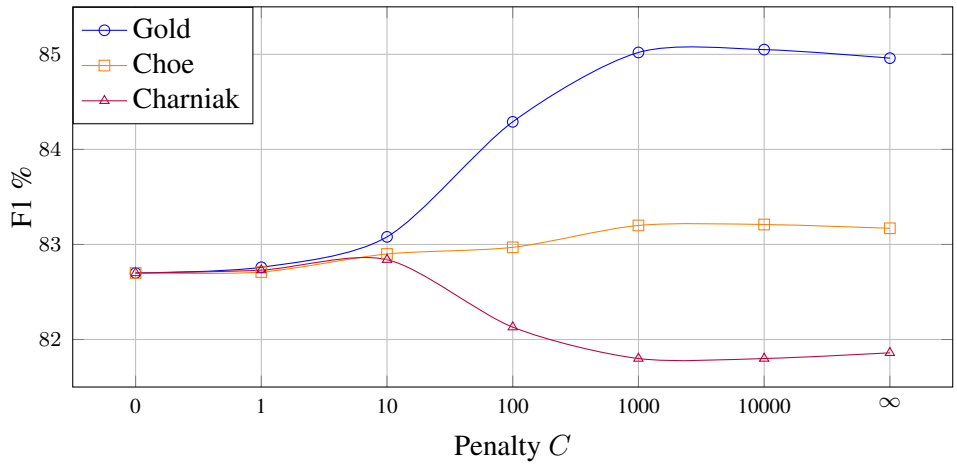


Figure 5.6: Performance of syntax-constrained decoding as the non-constituent penalty increases for syntax from two parsers (from Choe and Charniak [2016] and Charniak [2000]) and gold syntax. The best existing parser gives a small improvement, but the improvement from gold syntax shows that there is still potential for syntax to help SRL.

Constrained Decoding with Syntax The above analysis raises a further question: would improving consistency with syntax provide improvements for SRL? Our constrained decoding algorithm described in Section 5.3 enables us to inject syntax as a decoding constraint without having to re-train the model. Specifically, if the decoded sequence contains k arguments that do not match any unlabeled syntactic constituent, it will receive a penalty of kC, where C is a single parameter dictating how much the model should trust the provided syntax. In Figure 5.6, we compare the SRL accuracy with syntactic constraints specified by gold parse or automatic parses. When using gold syntax, the predictions improve up to 2 F1 as the penalty increases. A state-of-the-art parser [Choe and Charniak, 2016] provides smaller gains, while using the Charniak parser [Charniak, 2000] hurts performance if the model places too much trust in it. These results suggest that high-quality syntax can still make a large impact on SRL.

A known challenge for syntactic parsers is robustness on out-of-domain data. Therefore we provide experimental results in Table 5.8 for both CoNLL 2005 and CoNLL 2012, which consists of 8 different genres. The penalties are tuned on the two development sets separately ($C = 10000$ on CoNLL 2005 and $C = 20$ on CoNLL 2012). On the CoNLL 2005 development set, the predicted syntax gives a 0.5 F1 improvement over our best model, while on in-domain test and out-of-domain development sets, the improvement is much smaller. As expected, on CoNLL 2012, syntax improves most on the newswire (NW) domain. These improvements suggest that while decoding with hard constraints is beneficial, joint training or multi-task learning could be even more effective by leveraging full, labeled syntactic structures.

5.7 Summary

In this chapter, we presented a BIO-based deep SRL moodel with a 10% relative error reduction over the previous state of the art. Our ensemble of 8 layer BiLSTMs incorporated some of the recent best practices such as orthonormal initialization, RNN-dropout, and highway connections, and we have shown that they are crucial for getting good results with deep models. Extensive error analysis sheds light on the strengths and limitations of our deep SRL model, with detailed comparison against shallower models and two strong non-neural systems. While our deep model is better at

recovering long-distance predicate-argument relations, we still observe structural inconsistencies, which can be alleviated by constrained decoding.

Different from most prior work that assumes gold predicates as part of the input, we also experimented with the setup of "end-to-end" SRL, where all predicates and arguments need to be predicted. We observed significant performance drop when using predicted predicates instead of gold standard ones. In the next chapter, we will introduce another SRL model based on classifying span-span relations, which has the advantage of being able to 1) jointly predict all predicates and arguments in an end-to-end manner, and 2) be generalized to a range of NLP tasks.

Chapter 6

LABELED SPAN GRAPH NETWORKS

In the previous chapter, we introduced DeepSRL, a BIO-based neural model with relatively simple architecture. DeepSRL is able to out-perform more complicated pipelined systems without any syntactic processing. However, it still assumes gold or predicted predicates as part of the input, and predicts arguments in a predicate-by-predicate manner. In this chapter, we will take a different approach, tackling the SRL problem by directly enumerating predicate words and argument spans, and classifying the predicate-argument relations. Specifically, we introduce the Labeled Span Graph Networks (LSGNs), a versatile framework for modeling spans and span-span relations. As we will show our LSGN model will be able to jointly predict all predicates and their arguments in the sentence, while at the same time significantly improve the state-of-the-art when combined with contextualized word representations [Peters et al., 2018]. Experiments on coreference resolution and named entity recognition further showcase the generality of this framework. Part of the work described in this chapter is presented in He et al. [2018].

6.1 Introduction

Many NLP problems involve predicting relationships between spans of text. For example, coreference resolution (COREF) systems must predict if pairs of spans corefer (e.g. to determine if *their* refers to *many tourists* in Figure 6.1) while semantic role labeling (SRL) models relationships between predicate and argument spans (e.g. to know the purpose of the *visit* event is *to meet their favorite cartoon characters*, again in Figure 6.1). Recent work has introduced very high performing, but custom designed neural models for many such tasks, including SRL [Yang and Mitchell, 2017, He et al., 2017], COREF [Lee et al., 2017] and named entity recognition (NER) [Li et al., 2017]. In this paper, we introduce Labeled Span Graphs Networks (LSGNs), a unified architecture

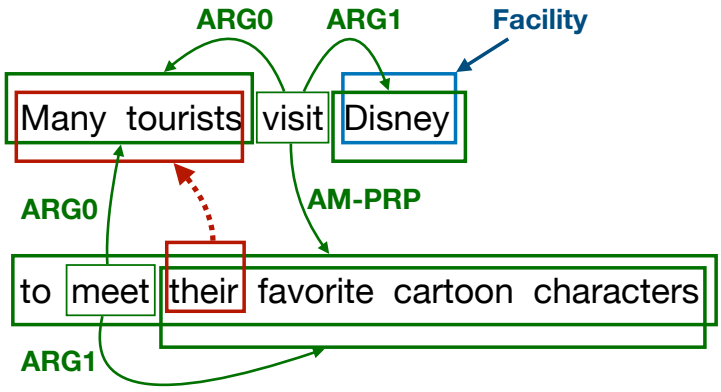


Figure 6.1: An example sentence with the span-edge structures from three different NLP tasks. Solid lines represent the predicate-argument structure in SRL. Dashed lines represent the coreferent link between mentions. The named entity label is represented with a single pointer from the entity type. The spans very often overlap or nest with each other.

for span-level modeling of text that performs well on all of these tasks.

LSGNs build fixed-length representations for all spans in an input text and use lightweight classifiers to make decisions about how to best link, rank, or label them. The representations are built with deep BiLSTMs and include a span internal attention mechanism, following recent work [Lee et al., 2017]. The classifiers are trained to make independent predictions for each text span. LSGNs support three types of decisions as seen in Figure 6.1: *span linking* — predicting labeled edges between pairs of spans; *span ranking*— predicting the best parent span for a given child span; and *span labeling*— predicting labels for individual spans. Individual LSGNs can include one or more of these classifier types, computed over shared span embeddings.

To demonstrate their generality, we study four different LSGNs. We show that a state-of-the-art COREF model [Lee et al., 2017] is an LSGN with *span ranking* potentials. We also introduce new LSGNs for NER and SRL, that provide strong performance gains over existing state-of-the-art BIO tagging models. For SRL, the LSGN formulation is particularly novel. The ability to predict overlapping spans enables joint modeling of all predicates and arguments, unlike current SRL models

Case	Parts	Targets	Applications
Linking	$\mathcal{S} \times \mathcal{S}$	$\mathcal{L}$	**SRL**, COREF (mention classification), relation extraction
Ranking	$\mathcal{S} \times \mathcal{L}$	$\mathcal{S}$	**COREF (mention ranking)**, dependency parsing
Labeling	$\mathcal{S}$	$\mathcal{L}$	**NER**, chunking, sentiment analysis

Table 6.1: Applicable tasks with different LSGN cases. The tasks highlighted in bold are the ones we report performance on.

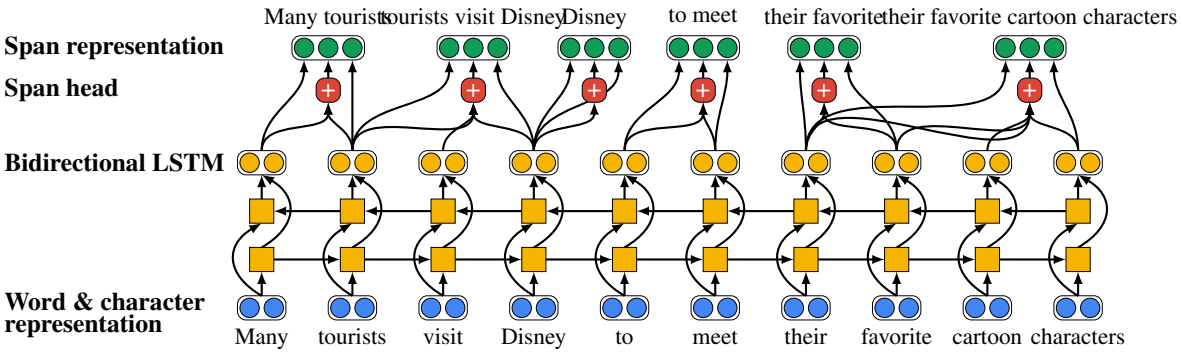


Figure 6.2: Building the general span representations $\mathbf{g}(s)$ from BiLSTM outputs. For clarity, we only show one BiLSTM layer and a small subset of the spans are shown here.

that must be run independently for each verb in a sentence. Finally, we also introduce an LSGN that predicts all three tasks jointly in a single forward pass, on top of shared span representations. This approach saves significant computation with only modest loss in accuracy, and empirically demonstrates the generality of the LSGN modeling paradigm.

6.2 Task Definition

We define the labeled span graph (LSG) prediction problem as follows: Given a sequence of words $X = w_1, \ldots, w_n$, we wish to predict a graph Y such that each node corresponds to a span $s \in \mathcal{S}$,

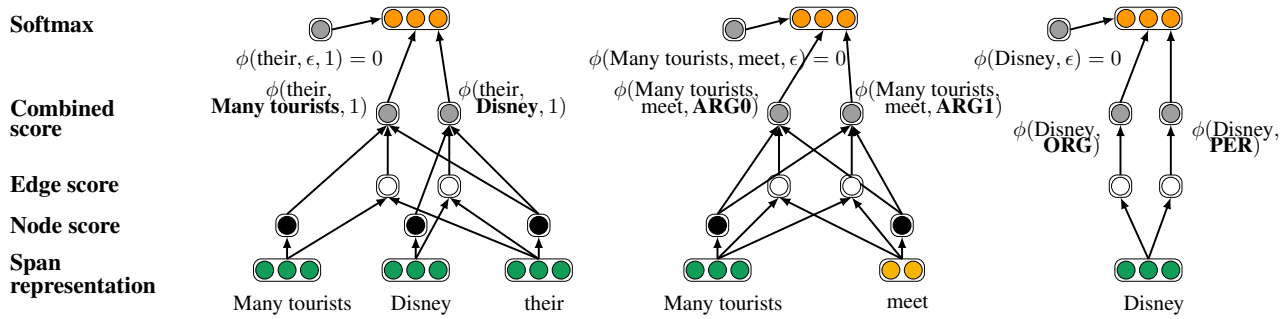


Figure 6.3: The task-specific modules of our model show a local softmax in each case. From left to right: COREF, SRL and NER. Black nodes represent unary factors, and white node represent relational factors. In SRL, all the predicates are spans of length 1, so their representations are the BiLSTM output of the token.

with $\mathcal{S} = \{(w_i, \ldots, w_j) \mid 1 \leq i \leq j \leq n\}$, and each edge is a relation between two spans, tagged with a discrete set of labels $\mathcal{L}$. We do not impose any constraints on the spans in Y. For example, they can be overlapping or nested.

LSGs are built by enumerating a set of parts (e.g. a pair of text spans) and choosing a target for each (e.g. providing a label). Varying the part and target definitions allows for very general modeling of different tasks as LSGNs, as summarized in Table 6.1. There are three general cases:

Span linking involves giving a target label l to a part $r = (s_{\text{c}}, s_{\text{p}})$ that pairs a child text span s_{c} with a parent s_{p}. A null label is predicted when there should not be an edge from s_{c} to s_{p} in the output graph. Many SRL and relation extraction models can be seen as instances of linking LSGNs.

Span ranking involves picking a target span s_{p} to link to a part $r = (s_{\text{c}}, l)$ containing a child span s_{c} and label l. In practice, the label l is often constrained to a degenerate singleton set $\mathcal{L}$. A null span may also be predicted, indicating that there is no edge of type l from s_{c}. Mention ranking coreference models and unlabeled dependency parsers can be seen as types of ranking LSGNs with degenerate edge labels.

Span labeling involves giving a target label l to a part $r = s$ defined by a single span s. This can be seen as a special case of span linking, in which s_{p} is a dummy span. NER and chunking can be modeled with span labeling LSGNs.

A single LSGN can include random variables to model multiple decision types, as we will see in the next section, allowing for rich sharing of the span embeddings for very different problems.

6.3 The LSGN Model

The full model $P(Y \mid X)$, includes a random variable y_r for every part $r \in R$ defined in Section 6.2, which ranges over its associated targets. For example, span linking random variables predict an edge label for all pairs of text spans.

The complete distribution is defined as:

$$P(Y \mid X) = \prod_{r \in R} P(y_r \mid X) \tag{6.1}$$

$$P(y_r \mid X) = \frac{\exp(\phi(r, y_r))}{\sum_{t' \in T \cup \{\epsilon\}} \exp(\phi(r, t'))} \tag{6.2}$$

Here we use ϵ to represent both the null label and the null span introduced in Section 6.2.

$\phi(r, t)$ is a scoring function for a possible (part, target) combination. For *span linking* and *span ranking*, the factors ϕ are defined on each pair of spans $(s_{\text{c}}, s_{\text{p}})$ and label l, decomposed into two unary scores on the child and parent spans, as well as a label-specific score for the span-span relation:

$$\begin{aligned} \phi(r, t) =& \phi(s_{\text{c}}, s_{\text{p}}, l) \\ =& \Phi_{\text{c}}(s_{\text{c}}) + \Phi_{\text{p}}(s_{\text{p}}) + \Phi^{(l)}_{\text{rel}}(s_{\text{c}}, s_{\text{p}}) \end{aligned} \tag{6.3}$$

For *span labeling*, the factor scores are simply defined over span-label pairs: $\phi(s, l) = \Phi^{(l)}_{\text{rel}}(s)$. Score for the null target is set to a constant: $\phi(r, \epsilon) = 0$, similar to logistic regression.

Learning For each input sample X, we want to minimize the negative log likelihood of the gold structure Y^*:

$$\mathcal{J}(X) = -\log P(Y^* \mid X) \tag{6.4}$$

$$= -\sum_{r \in R} \log P(y_r^* \mid X) \tag{6.5}$$

It is straightforward to extend the Equation (6.5) to a marginalized loss function:

$$\mathcal{J}_{\text{mg}}(X) = -\sum_{r \in R} \log \sum_{t \in Y^*(r)} P(y_r = t \mid X) \tag{6.6}$$

where $Y^*(r)$ represents a set of gold target values. For example, we can have more than one correct labels for *span linking*, or have several correct parent spans for *span ranking*.

Inference An LSGN can model $O(N^2)$ possible spans, where N is the maximum sentence length in our input.[1] Therefore, in both the *span linking* and *span ranking* cases, the model needs to consider $O(N^4|\mathcal{L}|)$ parts, which is computationally impractical. To overcome this issue, we generalize the pruning strategy from Lee et al. [2017].

We define two beams B_{c} and B_{p} for storing the candidate child and parent spans, respectively. The spans in the child beam B_{c} are ranked by their unary score $\Phi_{\text{c}}(\cdot)$, and those in B_{p} are ranked by $\Phi_{\text{p}}(\cdot)$. The sizes of the beams are limited by $\lambda_{\text{c}}N$ and $\lambda_{\text{p}}N$. Spans that fall out of the beam do not participate in computing the edge factor $\Phi_{\text{rel}}^{(l)}(\cdot,\cdot)$, reducing the overall number of relational factors evaluated by the model to $O(N^2|\mathcal{L}|)$, in addition to computing $O(N^2)$ unary factors.

We also limit the maximum width of spans to a fixed number $W \ll N$, which further reduces the number of computed unary factors to $O(N)$.

Multitask learning (MTL) As we will explain more detail in Section 6.3.1, our neural architecture allows straightforward sharing of span representations $\mathbf{g}$ among different tasks. To train a single model for multiple tasks, we optimize a weighted sum of the D task-specific losses:

[1] We assume spans do not cross-sentence boundary.

$\mathcal{J}_{\text{mtl}}(X) = \sum_{d=1}^{D} \lambda_d \mathcal{J}_d(X)$, where the λ_d are hyperparameters. The factor scores ϕ and the beams $B_{\text{c}}, B_{\text{p}}$ are task specific and not shared.

6.3.1 Neural Architecture

As shown in Figure 6.2, our neural model builds span representations using the outputs of bi-directional LSTMs to capture long-range contexts, and uses feed-forward networks to compute the factor scores described in Section 6.3. This is the same architecture as a recent end-to-end COREF model [Lee et al., 2017], extended to include to the *span linking* and *labeling* cases.

6.3.2 General Architecture

Word-level contexts The bottom layer consists of pretrained word embeddings and character-based representations for each word w_i:

$$\mathbf{x}_i = [\textsc{WordEmb}(w_i); \textsc{CharCNN}(w_i)] \tag{6.7}$$

We use m stacked bidirectional LSTMs with highway-connections [Zhang et al., 2016] to contextualize the word-level information:

$$\bar{\mathbf{x}}_i^m = \textsc{Highway}(\bar{\mathbf{x}}_i^{m-1}), \quad m = 1, \ldots, M \tag{6.8}$$

where the inputs of the BiLSTM $\bar{\mathbf{x}}_i^0 = \mathbf{x}_i$

Span representation As the central building block in our model, the span representations need to be able to capture information such as boundaries, content, and surrounding context. Our span representations contain the following parts: left and right end points from the BiLSTM outputs $(\bar{\mathbf{x}}^m_{\textsc{Start}(s)}, \bar{\mathbf{x}}^m_{\textsc{End}(s)})$, a soft head word $\mathbf{x}_{\text{h}}(s,t)$, and embedding span width features $\mathbf{f}_{\text{s}}(s)$.

$$\mathbf{g}(s) = [\bar{\mathbf{x}}^m_{\textsc{Start}(s)}; \bar{\mathbf{x}}^m_{\textsc{End}(s)}; \mathbf{x}_{\text{h}}(s); \mathbf{f}_{\text{s}}(s)] \tag{6.9}$$

For each span s, a *soft head representation* $\mathbf{x}_{\text{h}}(s)$ is computed as an attention mechanism over word inputs $\mathbf{x}$ in the span, where the attention weights $\mathbf{a}(s)$ are computed via a linear layer over

the BiLSTM outputs $\bar{\mathbf{x}}^m$.

$$\mathbf{x}_{\text{h}}(s) = \mathbf{x}_{\textsc{Start}(s):\textsc{End}(s)}\mathbf{a}(s)^\intercal \tag{6.10}$$

$$\mathbf{a}(s) = \textsc{SoftMax}(\mathbf{w}_\alpha^\intercal \bar{\mathbf{x}}^m_{\textsc{Start}(s):\textsc{End}(s)}) \tag{6.11}$$

$\mathbf{x}_{\textsc{Start}(s):\textsc{End}(s)}$ is a shorthand for stacking a list of vectors $\mathbf{x}_t$, where $\textsc{Start}(s) \leq t \leq \textsc{End}(s)$

Scoring The scoring functions $\Phi(\cdot)$ are implemented with feed-forward networks, with span representations $\mathbf{g}(s)$ as inputs:

$$\Phi_{\text{c}}(s) = \mathbf{w}_{\text{c}}^\intercal \text{MLP}_{\text{c}}(\mathbf{g}(s)) \tag{6.12}$$

$$\Phi_{\text{p}}(s) = \mathbf{w}_{\text{p}}^\intercal \text{MLP}_{\text{p}}(\mathbf{g}(s)) \tag{6.13}$$

$$\Phi^{(l)}_{\text{rel}}(s_{\text{c}}, s_{\text{p}}) = \mathbf{w}_{\text{r}}^{(l)\intercal} \text{MLP}_{\text{r}}(\mathbf{u}(s_{\text{c}}, s_{\text{p}})) \tag{6.14}$$

$$\mathbf{u}(s_{\text{c}}, s_{\text{p}}) = [\mathbf{g}(s_{\text{c}}); \mathbf{g}(s_{\text{p}}); \mathbf{f}_{\text{r}}(s_{\text{c}}, s_{\text{p}})] \tag{6.15}$$

$\mathbf{f}_{\text{r}}(s_{\text{c}}, s_{\text{c}})$ is a relational feature function on the a pair of spans. The exact definition of $\mathbf{f}_{\text{r}}$ is optional and task specific.

6.3.3 Task Specific Architectures

SRL (Span linking) We restrict the space of all possible predicates (*parent* spans) to a length of 1, and their representations are reduced to $\bar{\mathbf{x}}^m_{\textsc{Start}(s)}$.

In standard span-based SRL tasks where gold predicates are given as part of the input, we can easily adapt our model by setting all the predicate scores $\Phi_{\text{p}}(s)$ to 0, and restrict the parent beam B_{p} to the gold predicates only.

COREF (Span ranking) We use the same model architectures described in Lee et al. [2017], using shared unary scoring function and beams for mentions (children) and antecedents (parents), by setting $\Phi_{\text{c}} = \Phi_{\text{p}}$, $B_{\text{c}} = B_{\text{p}}$, and $\lambda_{\text{c}} = \lambda_{\text{p}}$. For each child span s_{c}, the space of all possible parents are restricted to all spans before s_{c} in the text, and normalized within the span s.

The relational features $\mathbf{f}_{\text{r}}$ contain element-wise multiplication of the span representations, binned distance, and metadata features, as described in Lee et al. [2017]. Given a mention, all antecedents in the same cluster are considered a valid *parent*, so the model optimizes for the marginalized log-likehood $\mathcal{J}_{\text{mg}}$.

NER (Span labeling) The factor scores are simply computed by an MLP over the span representation $\Phi_{\text{rel}}^{(l)}(s) = \mathbf{w}_{\text{r}}^{(l)\intercal}\text{MLP}_{\text{r}}(\mathbf{g}(s))$.

6.4 Experiments

We perform experiments on the OntoNotes 5.0 [Pradhan et al., 2013] and CoNLL 2005 [Carreras and Màrquez, 2005] benchmarks. OntoNotes is annotated with multiple layers of linguistic structures, including SRL, COREF, and NER. CoNLL 2005 is a PropBank SRL dataset, with the Penn Treebank split.

SRL tasks We conduct SRL experiments with two setups. In the *end-to-end* setup, a system takes a tokenized sentence as input, and predicts all predicates and their arguments as output, and is evaluated on the micro-averaged F1 on correctly predicting (predicate, argument span, label) tuples. For comparison with most previous systems, we also report results with gold predicates.

OntoNotes splits Table 6.2 shows the data statistics on various splits of OntoNotes. Following previous work, we use the CoNLL 2012 train/dev/test for coreference resolution and named-entity recognition. We found that some sentences in the OntoNotes 5.0 train/dev split have missing predicates, which is unsuitable for training end-to-end SRL systems. Therefore, our end-to-end SRL and multitask systems are trained on the smaller but cleaner CoNLL 2012 splits. For experiments with gold predicates, we use the full OntoNotes 5.0 train/dev split and the CoNLL 2012 test set.

	CoNLL 2012			OntoNotes5	
	Train	Dev	Test	Train	Dev
Documents	280,2	343	348	11,401	1,491
Sentences	75,187	9,603	9,479	115,812	15,680
Named entities	81,828	11,066	11,257	128,738	20,354
Predicates	190,334	24,065	26,715	253,070	35,297
Coref. Clusters	35,142	4,545	4,532	35,142	4,545

Table 6.2: Data statistics (in number of thousands) for the CoNLL 2012 split and the train/development split of OntoNotes5. Compared to CoNLL 2012, the OntoNotes5 train/development splits contain more NER and SRL annotations, but no additional coreference cluster.

	CoNLL 05 In-domain (WSJ)				Out-of-domain (Brown)			CoNLL 2012 (OntoNotes)			
End-to-End	Dev	P	R	F1	P	R	F1	Dev	P	R	F1
LSGN+ELMo	**85.3**	**84.8**	**87.2**	**86.0**	**73.9**	**78.4**	**76.1**	**83.0**	**81.9**	**84.0**	**82.9**
DeepSRL+PoE	81.5	82.0	83.4	82.7	69.7	70.5	70.1	77.2	80.2	76.6	78.4
LSGN	**81.6**	**81.2**	**83.9**	**82.5**	**69.7**	**71.9**	**70.8**	**79.4**	**79.4**	**80.1**	**79.8**
DeepSRL	80.3	80.2	82.3	81.2	67.6	69.6	68.5	75.5	78.6	75.1	76.8

Table 6.3: End-to-end SRL results for CoNLL 2005 and CoNLL 2012, compared to previous systems, including our DeepSRL model (Chapter 5). CoNLL 05 contains two test sets: WSJ (in-domain) and Brown (out-of-domain). The Dev column shows F1 on the development set.

	WSJ	Brown	OntoNotes
LSGN+ELMo	**87.4**	**80.4**	**85.5**
Peters et al. [2018]+ELMo	-	-	84.6
Tan et al. [2018]+PoE	86.1	74.8	83.9
DeepSRL+PoE	84.6	73.6	83.4
FitzGerald et al. [2015]+PoE	80.3	72.2	80.1
LSGN	83.9	73.7	82.1
Tan et al. [2018]	**84.8**	**74.1**	**82.7**
DeepSRL	83.1	72.1	81.7
Yang and Mitchell [2017]	81.9	72.0	-
Zhou and Xu [2015]	82.8	69.4	81.1

Table 6.4: Experiment results with gold predicates, compared to previous systems including DeepSRL (Chapter 5).

6.5 SRL Results

As shown in Table 6.3,[2] our LSGN model outperforms the previous best pipeline system DeepSRL (Chapter 5) by an F1 difference of anywhere between 1.3 and 6.0 in every setting. The improvement is larger on the Brown test set, which is out-of-domain, and the CoNLL 2012 test set, which contains nominal predicates. On all datasets, our model is able to predict over 40% of the sentences completely correctly.

To compare with additional previous systems, we also conduct experiments with gold predicates by constraining our predicate beam to be gold predicates only. As shown in Table 6.4, our model significantly out-performs DeepSRL, but falls short of Tan et al. [2018], a very recent self-attention-based [Vaswani et al., 2017] BIO-tagging model that was developed concurrently with our work. Compared to DeepSRL, the larger GloVe embeddings and the character-based represen-

[2]For the end-to-end setting on CoNLL 2012, we used a subset of the train/dev data from the previous work due to noise in the dataset, there the dev. number is not directly comparable.

	End-to-End SRL				COREF					NER			
	Dev	Test			Dev	Test				Dev	Test		
Method	F1	P	R	F1	F1	MUC	B^3	$CEAF_{\phi_4}$	F1	F1	P	R	F1
LSGN +ELMo	**83.0**	**81.9**	84.0	**82.9**	72.5	**79.7**	**69.9**	**66.8**	**72.1**	**88.2**	**89.6**	89.2	**89.4**
LSGN-MTL+ELMo	82.7	80.8	**84.5**	82.6	**72.6**	79.2	69.0	65.8	71.4	87.7	89.2	**89.6**	**89.4**
DeepSRL+PoE	77.2	80.2	76.6	78.4									
Lee et al. [2017] +PoE					69.0	77.2	66.6	62.6	68.8				
LSGN	**79.4**	**79.4**	**80.1**	**79.8**	**69.9**	**77.7**	**67.5**	**64.1**	**69.7**	**87.2**	**89.1**	**88.9**	**89.0**
LSGN-MTL	79.0	78.5	79.9	79.2	**69.9**	77.2	66.8	63.1	69.1	85.8	88.4	87.6	88.0
DeepSRL	75.5	78.6	75.1	76.8									
Lee et al. [2017]					67.7	75.8	65.0	60.8	67.2				
Li et al. [2017]										85.1	88.0	86.5	87.2
Strubell et al. [2017]										-	-	-	87.0
Chiu and Nichols [2016]										-	86.0	86.5	86.2

Table 6.5: Experimental results of end-to-end SRL, COREF and NER on the CoNLL 2012 development and test set. The DeepSRL system is described in Chapter 5. Our COREF results outperform Lee et al. [2017] due to larger model capacity and better hyperparameters.

tations are especially effective on the out-of-domain Brown test set. By adding the contextualized ELMo representations, we are able to out-perform all previous systems, including Peters et al. [2018], which applies ELMo to the SRL model introduced in Chapter 5.

6.6 COREF, NER, and MTL Results

Table 6.5 shows experiment results on end-to-end SRL, COREF and NER, all trained, developed and tested on the same CoNLL 2012 split. Our LSGN network is able to achieve state-of-the-art performance on all three tasks. Note that our COREF instance is the same as Lee et al. [2017], the performance gain on COREF comes only from better hyperparameter tuning (modeling wider

spans and using deeper BiLSTM contextualization layers).

The MTL row shows performance on each task when we train a single LSGN with shared span representations $\mathbf{g}$ and equal loss weights (0.33). All tasks suffer a modest decrease in performance compared to individually trained models. However, in practice, being able to produce SRL, COREF and NER with a single forward pass saves significant computation. It is also possible to explore multi-task learning techniques [Søgaard and Goldberg, 2016, Ruder et al., 2017] to further improve performance on all tasks.

6.7 Analysis

We conduct analysis mainly on the SRL task, because it is the first application of LSGN, and has more interesting long-range dependencies. LSGN's architecture differs significantly from previous BIO systems in terms of both input and decision space. To better understand our model's strengths and weaknesses, we perform three analyses following Lee et al. [2017] and DeepSRL (Chapter 5), studying (1) the effectiveness of beam pruning, (2) the ability to capture long-range dependencies, and (3) the ability to predict globally consistent SRL structures. The analyses are performed on the development sets without using ELMo embeddings. For comparability with previous works, analyses (2) and (3) are performed on the CoNLL 05 development set with gold predicates.

Span pruning Table 6.6 shows the statistics of various types of spans in the CoNLL 2012 training set. While arguments and mentions can contain up to two hundred words, most of them are under 30 words. We also list the average number of spans per token for each type. Interestingly, when compared to the actual λ we use for beam pruning, we have to include about twice as many spans as the actual span-to-token proportion.

Figure 6.4 shows the predicate and argument spans kept in the beam, sorted with their unary scores. Our model efficiently prunes unlikely argument spans and predicates, significantly reduces the number of edges it needs to consider. Figure 6.5 shows the recall of predicate words on the CoNLL 2012 development set (without using ELMo). By retaining $\lambda_{\mathrm{p}} = 0.4$ predicates per word, we are able to keep over 99.7% argument-bearing predicates. Compared to having a part-of-speech

Span type	Max. width	99% width	Span/token	λ
Arguments	201	24	0.46	0.8
Predicates	1	1	0.15	0.4
Mentions	94	17	0.12	0.4
Named Entities	28	6	0.06	-

Table 6.6: Statistics for different span types on the CoNLL 2012 training set, include the maximum span width (max. width) and the 99 percentile width (99%), the averaged number of spans per token (span/token) and the actual pruning factor we used (λ).

B_c	Φ_c	B_p	Φ_p
by ambulance	2.5	says	0.1
her mother ... ambulance	2.2	transported	0.0
her mother	2.2	ambulance	-8.3
Priscilla	1.9	been	-11.3
should	1.8		
transported by ambulance	-0.3		
Priscilla says .... ambulance	-2.2		
ambulance	-3.2		

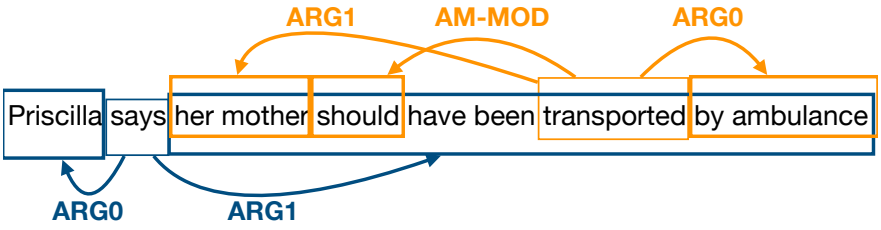


Figure 6.4: Top: The candidate arguments and predicates in the beams B_c and B_p after pruning, along with their unary scores. Bottom: Predicted SRL relations with two identified predicates and their arguments.

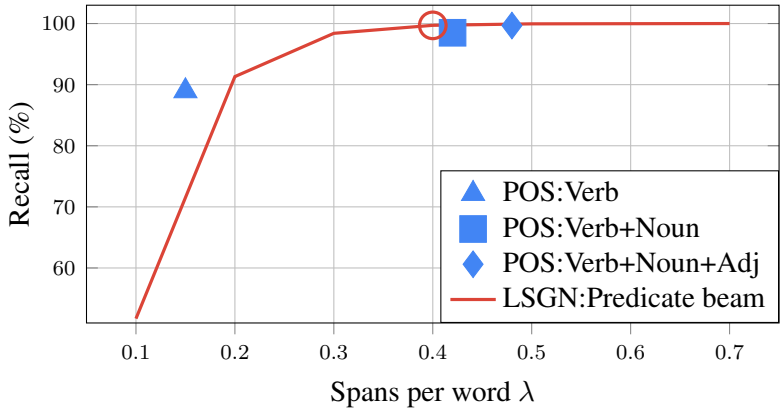


Figure 6.5: Recall of gold argument-bearing predicates on the CoNLL 2012 development data as we increase the number of predicates kept per word. POS:X shows the gold predicate recall from using certain pos-tags identified by the NLTK part-of-speech tagger [Bird, 2006].

tagger (POS:X in Figure 6.5), our joint beam pruning allowing the model to have a soft trade-off between efficiency and recall.[3]

Figure 5.5 shows the performance breakdown by binned distance between arguments to the given predicates. Our model is better at accurately predicting arguments that are farther away from the predicates, even compared to the ensembled DeepSRL model (Chapter 5) that has a higher overall F1. This is very likely due to architectural differences; a BIO tagger needs to memorize and propagate predicate information *through* many BiLSTM layers, while we directly have predicates and arguments interact via an MLP classifer *after* the BiLSTM encoding.

Global consistency As mentioned in Chapter 5, our BIO-based SRL system has good agreement with gold syntactic span boundaries (94.3%) but falls short of previous syntax-based systems [Punyakanok et al., 2008]. By directly modeling span information, our model achieves comparable syntactic agreement (95.0%) to Punyakanok et al. [2008] without explicitly modeling syntax.

[3]The predicate ID accuracy of our model is not comparable with that reported in Chapter 5, since our model does not predict non-argument-bearing predicates.

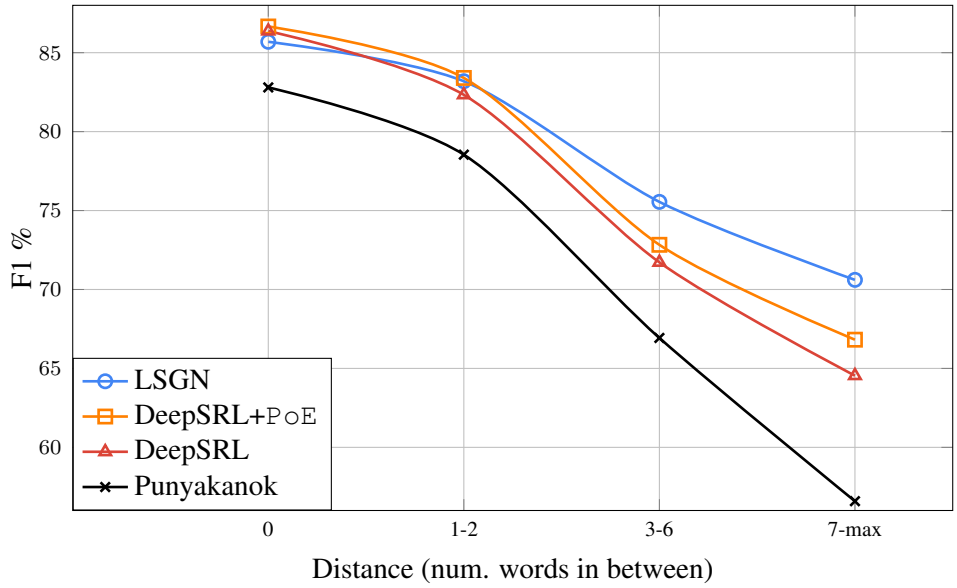


Figure 6.6: F1 by surface distance between predicates and arguments. The LSGN model performs better on arguments farther away from the predicates, even compared to previous PoE results that has higher overall F1.

On the other hand, our model suffers from other global consistency issues. For example, on the CoNLL 2005 test set, our model has lower complete-predicate accuracy (62.6%) than the BIO systems (DeepSRL and Tan et al. [2018]) (64.3%-66.4%). Table 5.7 shows its violations of global structural constraints[4] compared to previous systems.

Our model made more constraint violations compared to previous systems. For example, our model predicts duplicate core arguments[5] (shown in the U column in Table 5.7) more often than previous work. This is due to the fact that our model uses independent classifiers for labeling each predicate-argument pair, making it difficult for them to even implicitly track the decisions made for several arguments with the same predicate.

The LSGN+`decode` row in Table 6.7 shows SRL performance after enforcing the U-constraint

[4]Punyakanok et al. [2008] described a list of global constraints for SRL systems, e.g., there can be at most one core argument of each type for each predicate.

[5]Arguments with labels ARG0,ARG1,...,ARG5 and AA.

			SRL-Violations		
Model/Oracle	SRL F1	Syn %	U	C	R
Gold	100.0	98.7	24	0	61
LSGN+`decode`	82.4	95.1	0	8	104
LSGN	82.3	95.0	69	7	105
DeepSRL+`Poe`	82.7	94.3	37	3	68
DeepSRL	81.6	94.0	48	4	73
Punyakanok	77.4	95.3	0	0	0

Table 6.7: Comparison on the CoNLL 05 development set against previous systems in terms of unlabeled agreement with gold constituency (Syn%) and each type of SRL-constraints violations (**U**nique core roles, **C**ontinuation roles and **R**eference roles).

using dynamic programming [Täckström et al., 2015] at decoding time. Constrained decoding at test time is effective at eliminating all the core-role inconsistencies (shown in the U-column), but did not bring significant gain on the end result (shown in SRL F1).

6.8 Summary

In this chapter, we proposed LSGN, a general framework for span-level modeling. We show that in addition to a recent state-of-the-art end-to-end COREF model [2017], SRL and NER are also instances of LSGN. For SRL, LSGN enables us to jointly predict all predicates and argument spans in a sentence in an end-to-end manner, significantly improving the state of the art. We also show it is possible to learn an LSGN model that jointly predicts all three tasks with shared span representations, to tradeoff computational efficiency with modest accuracy loss. In the future, we could incorporate higher-order inference methods [Lee et al., 2018] to relax this assumption. It would also be interesting to combine our span-based architecture with the self-attention layers [Tan et al., 2018, Strubell et al., 2018] for more effective contextualization.

Chapter 7

CONCLUSION

In this thesis, we introduced new methods for annotating and modeling the semantic role labels from text, without relying on intermediate syntactic representations. Our goal is to build accurate and robust SRL systems that can benefit from the scalable annotation scheme and the end-to-end models introduced in this thesis.

In Chapter 4, we introduced QA-SRL, an annotation framework using question-answer pairs to represent predicate-argument structures. QA-SRL has a number of advantages — it is low cost, easily interpretable, and can be performed with very little training or linguistic expertise. These advantages come from the relatively open nature of the QA-SRL task, which does not depend on any linguistic theory of meaning or make use of any frame or role ontologies. We are simply using natural language to annotate natural language. As an extension to QA-SRL, we also built a human-in-the-loop framework for correcting attachment mistakes in CCG parsing, by having untrained annotators answering multiple-choice questions. These crowd-sourced annotations improve performance, particularly on out-of-domain data, demonstrating for the first time that untrained annotators can improve state-of-the-art parsers.

In Chapter 5, we presented a syntax-independent, end-to-end neural model for PropBank SRL that achieved a 10% relative error reduction over the previous state of the art. Our ensemble of 8 layer BiLSTMs incorporated some of the recent best practices such as orthonormal initialization, RNN-dropout, and highway connections, and we have shown that they are crucial for getting good results. However, we observed significant performance drop when using predicted predicates instead of gold standard ones. This motivates us to build another SRL model that could potentially address the limitation.

Finally, in Chapter 6, we proposed LSGN, a general framework for span-level modeling, which

addresses the limitations of our DeepSRL model by jointly modeling all predicates and their arguments in one output structure. We show that in addition to a recent state-of-the-art end-to-end coreference resolution model [2017], SRL and NER are also instance of LSGN. We also show it is possible to learn an LSGN model that jointly predicts all three tasks with shared span representations, to tradeoff computational efficiency with modest accuracy loss. This opens up exciting opportunities for end-to-end, document-level semantic understanding.

7.1 Future Work

We have made the first steps by collecting non-expert annotations with the QA-SRL framework and by building neural models for PropBank SRL. There are many possible future directions, such as modeling and applying QA-SRL data, and modeling document-level semantic representations by combining SRL with other NLP tasks.

Modeling and Using QA-SRL Annotations While QA-SRL (Chapter 4) described two simple baselines for predicting QA-SRL questions and answers, predicting the entire QA-SRL structure (including predicates, questions, and answers) could be challenging. Apart from the usual challenges in SRL modeling as we have discussed in Chapter 5 and 6, a QA-SRL model needs to generate token sequences for the questions instead of discrete role labels. Recently, FitzGerald et al. [2018] collected larger amounts of QA-SRL data and explored several neural architectures for QA-SRL, setting strong baselines for future work. For example, it would be interesting to experiment multi-task learning on QA-SRL and PropBank SRL using our LSGN model described in Chapter 6.

Apart from the modeling challenges, it is also an open question how we could use QA-SRL structures in downstream tasks. In QA-SRL, the predicate-argument relations are described with templated questions instead of discrete role labels, which are difficult to encode as features in downstream models. In the future, we could explore approaches that embed QA-SRL questions as fixed-length representations to be used in other tasks.

Document-level understanding with SRL Another interesting future direction is to extend sentence-level SRL tasks to document-level understanding, by combining SRL, coreference resolution, and other NLP tasks. For example, by predicting the SRL structure in a sentence "They filed the report.", we would know that "they" is the agent for "filing", but not who "they" are referring to unless we could connect the pronoun to its antecedents via coreference resolution. The LSGN model we have introduced in Chapter 6 could be used for multi-task learning with SRL, coreference resolution, and NER. In the future, we could extend such a model for producing more interesting document-level representations, and applying them to downstream tasks such as question-answering and information extraction.

BIBLIOGRAPHY

Omri Abend and Ari Rappoport. Universal conceptual cognitive annotation (ucca). In *ACL*, 2013.

Omri Abend and Ari Rappoport. The state of the art in semantic representation. In *ACL*, 2017.

Olga Babko-Malaya. Propbank annotation guidelines. *URL: http://verbs. colorado. edu*, 2005.

Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *COLING/ACL*, 1998.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *LAC@ACL*, 2013.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. Developing a large semantically annotated corpus. In *LREC*, 2012.

Steven Bird. Nltk: the natural language toolkit. In *COLING/ACL*, 2006.

Claire Bonial, Olga Babko-Malaya, Jinho D Choi, Jena Hwang, and Martha Palmer. Propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*, 2010.

Xavier Carreras and Lluís Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL*, 2005.

Eugene Charniak. A maximum-entropy-inspired parser. In *NAACL*, 2000.

Jason P. C. Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370, 2016.

Do Kook Choe and Eugene Charniak. Parsing as language modeling. In *EMNLP*, 2016.

Do Kook Choe and David McClosky. Parsing paraphrases with joint inference. In *ACL*, 2015.

Janara Christensen, Stephen Soderland, Oren Etzioni, et al. Semantic role labeling for open information extraction. In *FAMLBR@NAACL*, 2010.

Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *ACL*, 2004.

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

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56, 2014.

Manjuan Duan, Ethan Hill, and Michael White. Generating disambiguating paraphrases for structurally ambiguous sentences. In *LAW@ACL*, 2016.

Jason Eisner and Giorgio Satta. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *ACL*, 1999.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *EMNLP*, 2006.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. Semantic role labeling with neural network factors. In *EMNLP*, 2015.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. Large-scale qa-srl parsing. In *ACL*, 2018.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *CoNLL*, 2009.

Luheng He, Mike Lewis, and Luke Zettlemoyer. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *EMNLP*, 2015.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. Human-in-the-loop parsing. In *EMNLP*, 2016.

Luheng He, Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. Deep semantic role labeling: What works and what's next. In *ACL*, 2017.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL*, 2018.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998, 2013.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Julia Hockenmaier and Mark Steedman. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33:355–396, 2007.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90% solution. In *NAACL*, 2006.

Mukund Jha, Jacob Andreas, Kapil Thadani, Sara Rosenthal, and Kathleen McKeown. Corpus creation for new genres: A crowdsourced approach to pp attachment. In *MTurk@NAACL*, 2010.

Richard Johansson and Pierre Nugues. Dependency-based semantic role labeling of propbank. In *EMNLP*, 2008.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Paul Kingsbury and Martha Palmer. From treebank to propbank. In *LREC*, 2002.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. Adding semantic annotation to the penn treebank. In *HLT*, 2002.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. Segmental recurrent neural networks. In *ICLR*, 2016.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *EMNLP*, 2012.

John D. Lafferty, Andrew D McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

Joel Lang and Mirella Lapata. Unsupervised semantic role induction via split-merge clustering. In *ACL*, 2011.

Joel Lang and Mirella Lapata. Similarity-driven semantic role induction via graph partitioning. *Computational Linguistics*, 40(3):633–669, 2014.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*, 2016a.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Global neural ccg parsing with optimality guarantees. In *EMNLP*, 2016b.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. End-to-end neural coreference resolution. In *EMNLP*, 2017.

Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. *NAACL*, 2018.

Mike Lewis and Mark Steedman. A* ccg parsing with a supertag-factored model. In *EMNLP*, 2014.

Mike Lewis, Luheng He, and Luke Zettlemoyer. Joint a* ccg parsing and semantic role labeling. In *EMNLP*, 2015.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. Lstm ccg parsing. In *NAACL*, 2016.

Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou, and Wei-Yun Ma. Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks. In *EMNLP*, 2017.

Ding Liu and Daniel Gildea. Semantic role features for machine translation. In *COLING*, 2010.

Yi Luan, Yangfeng Ji, Hannaneh Hajishirzi, and Boyang Li. Multiplicative representations for unsupervised semantic role induction. In *ACL*, 2016.

Bill MacCartney and Christopher D Manning. Natural logic for textual inference. In *ACL-PASCAL@ACL*, 2007.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *CoNLL*, 2017.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *HLT*, 1994.

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.

Lluís Màrquez, Xavier Carreras, Kenneth C Litkowski, and Suzanne Stevenson. Semantic role labeling: an introduction to the special issue. *Computational linguistics*, 2008.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. The nombank project: An interim report. In *NAACL 2004 workshop: Frontiers in corpus annotation*, 2004.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. Crowdsourcing question-answer meaning representations. In *NAACL*, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

Simone Paolo Ponzetto and Michael Strube. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *NAACL*, 2006.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. Semantic role chunking combining complementary syntactic views. In *CoNLL*, 2005.

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 *CoNLL*, 2013.

Vasin Punyakanok, Peter Koomen, Dan Roth, and Wen-tau Yih. Generalized inference with multiple semantic role labeling systems. In *CoNLL*, 2005.

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.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 2007.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. Semantic proto-roles. *TACL*, 3:475–488, 2015.

Michael Roth and Mirella Lapata. Neural semantic role labeling with dependency path embeddings. In *ACL*, 2016.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*, 2017.

Josef Ruppenhofer, Michael Ellsworth, Miriam RL Petruck, Christopher R Johnson, and Jan Scheffczyk. Framenet ii: Extended theory and practice, 2006.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21, 2007.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP*, 2008.

Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*, 2016.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *NIPS*, 2015.

Mark Steedman. *The syntactic process*. 2000.

Mitchell Stern, Jacob Andreas, and Dan Klein. A minimal span-based neural constituency parser. In *ACL*, 2017.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. Fast and accurate entity recognition with iterated dilated convolutions. In *EMNLP*, 2017.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-Informed Self-Attention for Semantic Role Labeling. *ArXiv e-prints*, 2018.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL*, 2008.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. Greedy, joint syntactic-semantic parsing with stack lstms. In *CoNLL*, 2016.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*, 2017.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. Efficient inference and structured learning for semantic role labeling. *TACL*, 3:29–41, 2015.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. Deep semantic role labeling with self-attention. In *AAAI*, 2018.

Ivan Titov and Ehsan Khoddam. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *NAACL*, 2015.

Ivan Titov and Alexandre Klementiev. A bayesian approach to unsupervised semantic role induction. In *EACL*, 2012.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, 2003.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. Chinese semantic role labeling with bidirectional recurrent neural networks. In *EMNLP*, 2015.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 2011.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 2013.

Keenon Werling, Arun Tejasvi Chaganty, Percy S Liang, and Christopher D Manning. On-the-job learning with bayesian decision theory. In *NIPS*, 2015.

Bishan Yang and Tom M. Mitchell. A joint sequential and relational model for frame-semantic parsing. In *EMNLP*, 2017.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. Highway long short-term memory rnns for distant speech recognition. In *ICASSP*, 2016.

Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL*, 2015.

Appendix A

HYPERPARAMETERS FOR LSGN

This part of the appendix describes the hyper-parameters we used for the LSGN (introduced in Chapter 6). Aiming for a unified architecture, we share most of the hyperparameters and features in our model across the different tasks in LSGN. Our code and pretrained models are be publicly available at the author's website.

Representation sizes The word embeddings are fixed 300-dimensional GloVe embeddings [Pennington et al., 2014] (context window size of 2 for head word embeddings, and window size of 10 for LSTM inputs), normalized to be unit vectors. Out-of-vocabulary words are represented by a vector of zeros. In the character CNN, characters are represented as learned 8-dimensional embeddings. The convolutions have window sizes of 3, 4, and 5 characters, each consisting of 50 filters. We use the same span width feature and COREF mention-pair features as described in Lee et al. [2017].

Network sizes We use 3 stacked bidirectional LSTMs with highway connections and 200 dimensional hidden states. Each MLP consists of two hidden layers with 150 dimensions and rectified linear units [Nair and Hinton, 2010].

Inference For all tasks, we model spans up to length 30 because SRL has longer spans (See Table 6.6). For SRL, we use $\lambda_c = 0.8$ for pruning arguments, $\lambda_p = 0.4$ for pruning predicates. At decoding time, we use dynamic programming (a simplified version of Täckström et al. [2015]) to predict a set of non-overlapping arguments for each predicate [1]. For COREF we use $\lambda_p = \lambda_c = 0.4$.

[1] This is mainly a constraint enforced by the official CoNLL evaluation script.

The maximum number of antecedents (parents) is set to 250, and the crossing-spans are greedily pruned from the beam.

Training We use Adam [Kingma and Ba, 2014] with initial learning rate 0.001 and decay rate of 0.1% every 100 steps. The LSTM weights are initialized with random orthonormal matrices [Saxe et al., 2013]. We apply 0.5 dropout to the word embeddings and character CNN outputs and 0.2 dropout to all hidden layers and feature embeddings. In the LSTMs, we use variational dropout masks that are shared across timesteps [Gal and Ghahramani, 2016], with 0.4 dropout rate.

Batching At training time, we randomly shuffle all the documents and then batch at sentence level. Each batch contains at most 40 sentences and 700 words. For the multitask experiments, we limit each batch to contain at most 35 sentences and 600 words due to a 12G GPU memory limit. All models are trained for at most 320,000 steps with early stopping on the development set, which takes less than 48 hours on a single Titan X GPU.

ELMo To further improve performance, we also add the 3-layer contextualized word representations ELMo [Peters et al., 2018] to the input of the BiLSTMs (in the +ELMo rows). Since ELMo can be applied to most previous neural systems, the improvement from using ELMo is orthogonal to our contribution. In Table 6.3, 6.4, and 6.5, we organize all the results into two categories: the comparable single model systems, and the models augmented with ELMo or ensembling.