Syntactic Inductive Biases for Natural Language Processing

Swabha Swayamdipta

CMU-LTI-19-004

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Noah A. Smith (Co-Chair)
Chris Dyer (Co-Chair)
Jaime Carbonell
Luke Zettlemoyer (University of Washington)

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies*

Syntactic Inductive Biases for Natural Language Processing

Swabha Swayamdipta

CMU-LTI-19-004
May 2019

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Noah A. Smith, Co-Chair, University of Washington
Chris Dyer, Co-Chair, Google DeepMind
Jaime Carbonell, CMU
Luke Zettlemoyer, University of Washington

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies.*

To the women who came before me.

Abstract

With the rise in availability of data for language learning, the role of linguistic structure is under scrutiny. The underlying syntactic structure of language allows for composition of simple elements into more complex ones in innumerable ways; generalization to new examples hinges on this structure. We define a syntactic inductive bias as a signal that steers the learning algorithm towards a syntactically robust solution, over others. This thesis explores the need for incorporation of such biases into already powerful neural models of language.

We describe three general approaches for incorporating syntactic inductive biases into task-specific models, under different levels of supervision. The first method calls for joint learning of entire syntactic dependency trees with semantic dependency graphs through direct supervision, to facilitate better semantic dependency parsing. Second, we introduce the paradigm of scaffolded learning, which enables us to leverage inductive biases from syntactic sources to predict a related semantic structure, using only as much supervision as is necessary. The third approach yields general-purpose contextualized representations conditioned on large amounts of data along with their shallow syntactic structures, obtained automatically. The linguistic representations learned as a result of syntactic inductive biases are shown to be effective across a range of downstream tasks, but their usefulness is especially pronounced for semantic tasks.

Acknowledgments

The important thing is not to stop questioning; never lose a holy curiosity.

— Albert Einstein.

My journey in natural language processing has been incredibly fast-paced, given the field has evolved tremendously over the last few years. I have been fortunate to have many wonderful people with whom I have shared this journey.

First and foremost, I thank my advisor, Noah Smith, for his relentless support despite his ever-increasing responsibilities, his encouragement to always ask questions and revise opinions, and his ability of making the best of any situation. I have also been fortunate to have Chris Dyer as my advisor, whom I thank for being a steady source of technical insight, and inspiring me to be creative. I thank both my advisors for teaching by example, the value of logical narrative in research.

Prior to CMU, I was a Masters student at Columbia, where I was first introduced to NLP research via a research assistantship with Owen Rambow. I cannot thank Owen enough for providing me with this opportunity, and taking a chance on me. I am ever grateful to Michael Collins for teaching me the fundamentals of NLP and machine learning, and for sparking and nurturing my enthusiasm for this research.

I thank Luke Zettlemoyer, for serving on my committee and always being open to intelligent advice and comments on earlier drafts of this theses. A special thanks to Yejin Choi and Kathy McKeown for being absolute pathbreakers and showing me what sometimes seemed impossible.

Research is a long arduous process, and impossible to do alone. I have been lucky to work with several wonderful collaborators during my PhD—Sam Thomson, Kenton Lee, Brendan Roof, Phoebe Mulcaire, Hao Peng, Miguel Ballesteros, Omer Levy, Roy Schwartz, Nelson Liu, Sam Bowman, Lingpeng Kong, Nathan Schneider, Archana Bhatia, David Bamman, Jesse Dodge, Austin Matthews, Brendan O'Connor, Jeff Flanagan, Alon Lavie, Greg Hanneman, Waleed Ammar, Yulia Tsvetkov, and Eva Schlinger. I am also lucky to have many friends who I have spent several hours with, talk about life in and out of grad school, making sense of the madness—Maarten Sap, Dallas Card, Lucy Lin, Elizabeth Clark, Sofia Serrano, Nicholas Fitzgerald, Yonatan Bisk, Eunsol Choi, Yangfeng Ji, Ana Marasović, Mike Lewis, Dan Garrette, Mark Yatskar, Priyam Parashar, Rohan Ramanath, Suruchi Shah, Rahul Goutam, Pallavi Baljekar, Manaal Faruqui, Pradeep Dasigi, Sujay Jauhar, Christine Betts, Erika

Wolfe, Luheng He, and Amandalynne Paullada.

I thank Ankur Parikh, for hosting me at Google AI during a most enjoyable internship, which fundamentally changed my approach to research, and also Tom Kwaitkowski, Dipanjan Das, Emily Pitler, and Minjoon Seo for many fruitful discussions. I thank Matt Peters and several folks from AI2, where I interned in Fall 2018 and wrote the last chapter of this thesis. I am excited to start a new chapter of my career there, as a post-doctoral young investigator.

I am forever grateful to my dad, Sudhanshu, for nurturing my curiosity, and for patiently reading to me all of his books, and never tiring from my incessant questions. My mom, Mrunmayee is the bravest woman I know, I am indebted to her for all my strength, and I am still learning from her how it's never too late to try. I thank my sister, Swati for always inspiring me to be creative and exploratory, and my brother-in-law Lokanath for his indefatigable encouragement. My little niece, Namami, has filled my life with so much joy, I thank her for sharing her stories, princesses and play dough with a very far-away Mausi.

A special thanks to Suchin Gururangan, who brings meaning to everything in life, makes me believe in limitless possibilities, and yet grounds me with a sense of warmth and security, so essential in the uncertain world of scientific research. I also thank him for being a fantastic collaborator and for always being up for brainstorming sessions. I thank two important members of my family in Seattle—Kili, for being the best dog ever, and Dextie the cat for being 100% lovable.

This thesis is dedicated to all the women who paved the way for me and others like me; your numerous sacrifices have made magic happen.

— *April 2017, Seattle.*

Contents

Chapter 1

Introduction

The artificial intelligence revolution has put natural language processing in the limelight. NLP aims at facilitating seamless communication between humans and machines. The central challenge therefore is taking into consideration the full expanse of natural language, in its broad manifestation across domains (newswire, email, tweets), genres (satire, fiction); and style (word choice, fluency). The key to understanding this diverse expanse of natural language, whether by humans or machines, lies in the ability to navigate complex mappings between words and meanings.

Deep learning (LeCun et al., 2015) which mimics the behavior of neurons in the brain, has re-emerged as one viable solution to this challenge. The expressive power of this learning paradigm has impacted several fields such as computer vision (Krizhevsky et al., 2012), control (Mnih et al., 2015), robotics (Levine et al., 2016) and several subareas within machine learning itself (Goodfellow et al., 2016). In particular, deep learning has revolutionized NLP, ushering in large and unprecedented improvements across machine translation (Sutskever et al., 2014), generation (Graves, 2013), dialog generation (Wen et al., 2015), and structured prediction (Collobert et al., 2011; Luong et al., 2015), among many other applications.

Certain key advancements have been the primary drivers of modern NLP systems. Words and concepts which were previously represented as a set of sparse discrete features can now be represented as dense, real-valued vectors (Mikolov et al., 2013; Pennington et al., 2014) under deep learning. Continuous representations can capture fine grained similarities between objects, compared to discrete representations which are sparser and can only attain characterize coarser relationships. The representations themselves are parametric, and can be updated via optimizing task-specific objective functions. In aggregate, continuous representations have boosted all of NLP.

However, deep neural models depend on large amounts of computational power, and tend to be data hungry. On the other hand, only limited amount of labeled data is available[1] for NLP tasks, particularly those that are key to advancement of language technologies. These include, but are not limited to, tasks such as semantic parsing which delineate the complete semantics of text and provide salient information for a variety of applications such as information extraction and question answering. What's more, the datasets for such tasks are almost exclusively in English and a handful of other languages. Without access to sufficient data, neural approaches tend to suffer by overfitting to the available samples, and being unable to generalize to new, unseen data. In stark contrast, humans easily comprehend anything from news to Gen Z[2] slang to Faulkner, given just a *few examples* (Saffran, 2003). The uniquely human ability to generalize involves learning language in terms of a set of **latent structures governing language** that are largely invariant across diverse usages (Chomsky, 1957).

How can we make our machines generalize the way we do? One solution is to incorporate this latent linguistic structure into our learning algorithms, as prior knowledge. Before the advent of deep learning, machine learning approaches for language used linguistic structure to guide the design of discrete features for words and concepts.[3] In an attempt to efficiently consume enormous amounts of data, however, deep models often make simplistic assumptions about language (Vinyals et al., 2015), only capturing surface-level phenomena. While the trend is to move towards more powerful deep learning models which can consume massive amounts of unstructured data (Radford et al., 2018; Devlin et al., 2018), many have argued[4] that true learning of natural language can only be possible by *additionally* taking into account **linguistic structure** (Manning, 2015; Dyer, 2016), which is necessary for generalization. Access to these structures would ensure our machines acquire the ability to generalize, emulating humans.

In this thesis, we try to address this very question—how to leverage existing sources of rich, expert-annotated knowledge about the structure of language, along with the power of deep learning methods to improve linguistic representation learning? The focus in this thesis is on one particular flavor of linguistic structure—syntax, under the scope of single sentence structure.[5] We will primarily discuss supervised learning approaches, with the exception of Chapter 5, where we discuss both supervised and unsupervised learning. In most of our approaches, we consider two sources of supervision—one being syntax. However, this can be generalized to multiple sources given syntax

[1]NLP has seen the proliferation of large datasets collected via crowd-source annotations (Bowman et al., 2015; Williams et al., 2017; Rajpurkar et al., 2016). However, recent work has shown that these datasets tend to contain a majority of trivially solvable examples (Gururangan et al., 2018), reducing the impact of the scale.

[2]`https://www.businessinsider.com/generation-z`

[3]Even the early deep learning methods for language used discrete features converted into vector representations (Täckström et al., 2015; Roth and Woodsend, 2014), but as more powerful models have emerged, the trend is to move towards end-to-end models without reliance on structure (Zhou and Xu, 2015).

[4]Mirella Lapata. Translating from Multiple Modalities to Text and Back. `https://bit.ly/2VAItTZ`

[5]We address document-level relationships for coreference resolution in Chapter 4, however, our models do not give differential treatment to documents and sentences. There exists sophisticated structure outside of sentences, such as those evidenced in discourse analysis (Hearst, 1997); this thesis does not address structure at that level.

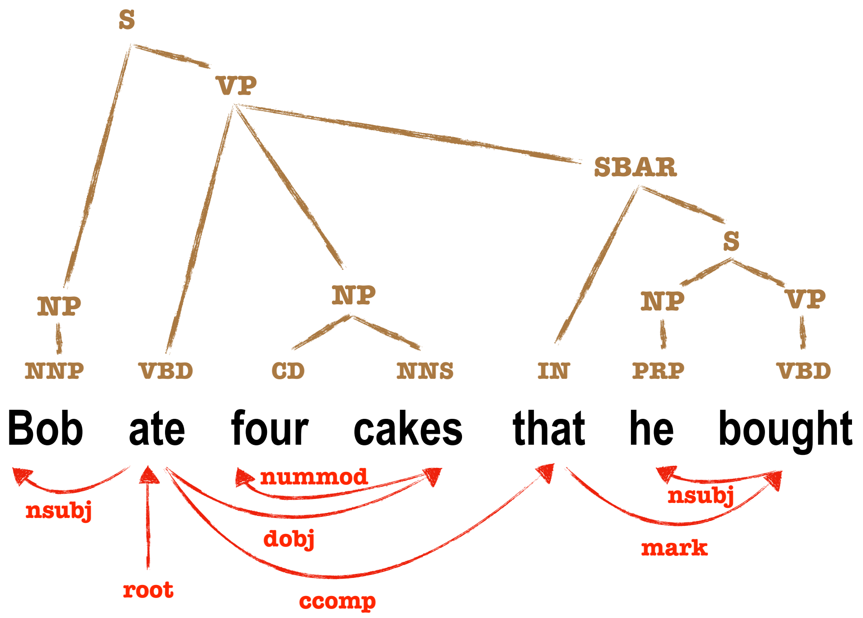


Figure 1.1: Syntactic formalisms. The phrase-syntactic tree of the sentence is shown above it in brown, and a dependency tree below the sentence in red. Under the first representation, terminal / leaf nodes in the tree denote the words in the sentence, and pre-terminals indicate part-of-speech tags. Non-leaf nodes in the phrase-structured tree correspond to syntactic constituents such as noun phrases (`NP`), and verb phrases (`VP`). A special node `S` indicates the root of the tree. In the dependency tree, all the nodes are words in the sentence. Edges are directed from the parent to the child node and are labeled with relationships between them.

remains one of them.

1.1 Linguistic Structure: Syntax

Syntax provides a skeletal framework for the construction of the meaning of a sentence. The structure of a sentence is characterized by the relationships between the words of the sentence. Syntax provides the mechanism to group together words into phrases, and those phrases into larger ones, recursively. Eventually this process yields a tree, with words in the sentence as it leaves. Syntactic trees can either be phrase-structured or dependency-based. Phrase-structured trees contain internal nodes for elemental phrases or constituents; the Penn TreeBank (Marcus et al., 1993) is the definitive resource for

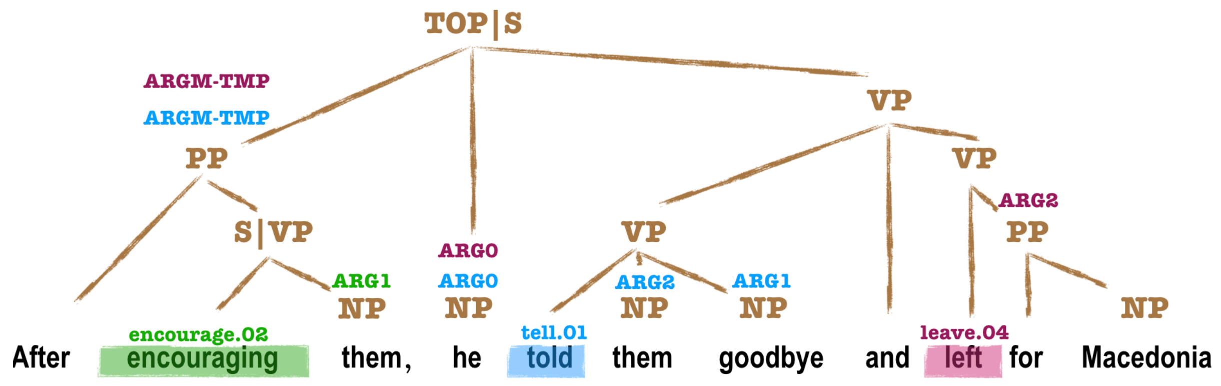


Figure 1.2: Syntax and semantics are closely related. The phrase-syntactic tree is shown in brown above the sentence. Semantic role labeling (SRL) structures from PropBank (Palmer et al., 2005) are shown alongside, in green, blue and magenta. Under SRL, words in the sentence that indicate stand-alone events are selected as predicates. These are shown as highlighted leaf nodes—"encouraging", "told" and "left". Each predicate is disambiguated to its relevant sense shown above it. Arguments to the predicates are are annotated on top of syntactic nodes, with the role labels color-coded by the predicate. SRL substructures (predicates, arguments) thus fully overlap with phrase-syntactic nodes.

phrase-syntactic annotations. Syntactic dependency trees forgo the internal node annotations—all the nodes (including non-leaf nodes) in a dependency tree are words in the sentence. Such trees have been annotated in the Universal Dependency TreeBank (Silveira et al., 2014).[6] Fig. 1.1 shows both syntactic formalisms underlying an example sentence.

Syntax is governed by an enumerable set of rules that allow words and phrases to be composed systematically. Chomsky (1957) postulated that given that a language has a large but finite vocabulary, this results in an infinite set of possible constructions. Put simply, any new idea can be represented in language, and we have the capacity to understand it based on the the same set of rules on a given vocabulary. Syntax provides a key mechanism to facilitate *generalization* to new, unseen examples.

We argue for the need of syntax-aware models in NLP to aid generalization. Non-syntactic models, in contrast, rely purely on statistical correlation in language. Not only does this further complicate an already challenging task, but also makes syntax-agnostic models vulnerable to errors via overreliance on co-occurrence patterns, ultimately rendering them incapable of generalizing well. We aim to learn representations of language which are explicitly aware of syntax, in order to leverage its compositionality, or the ability to combine smaller elements into larger ones on the fly.[7]

[6]There exist several other formalisms for syntax not explored in this thesis, such as lexical functional grammars (Bresnan et al., 2015).

[7]One could argue that if children can learn language without explicit lessons in syntax, why can't deep

1.1.1 Syntactic Inductive Biases

The goal of learning syntax-aware models is to select a solution which is more syntactically plausible compared to others. A preference for one solution over its contenders is referred to as an **inductive bias** (Mitchell, 1980) in machine learning. Inductive biases act as soft constraint, and could be expressed as priors in a Bayesian setting, or as regularizers in an optimization problem. Throughout this thesis, we will propose methods which express **preferences towards syntactically robust solutions** via appropriate syntactic inductive biases (Chapter 2). These biases will be incorporated in addition to the primary learning objective. Our hope is that such a design promotes generalizability towards unseen data.

In particular, the inductive biases that we employ come from joint learning, multi-task learning, and stage-wise or pipelined learning. Under joint learning (Chapter 3), a single objective is optimized towards predicting multiple structures for the same input. Joint learning induces biases that prefer both structures. Multi-task learning (Caruana, 1997) simplifies the learning problem by optimizing multiple objectives (Chapter 4). It involves sharing some parameters between the tasks, as well as retaining task-specific parameters. As a result, the inductive bias offered by each task objective is weakened, but this comes at the benefit of disentangling the inference. Finally, we use stage-wise pretraining in Chapter 5, by optimizing two objectives in a pipelined fashion. This induces a bias by encouraging the later model to rely on the predictions of the first training phase explicitly. Under each of these settings, one of the objectives involves syntactic prediction, hence inducing a syntactic bias. The other objective is task-specific, and will be addressed next.

1.1.2 Semantics

The semantic structure of the sentence is typically given by a graph. Unlike syntactic trees, semantic graphs contain several complexities, such as multiple parents, edge reentrancies and disconnected components. Moreover, semantic annotations vary greatly across formalisms such as PropBank (Palmer et al., 2005), AMRBank (Banarescu et al., 2012), FrameNet (Baker et al., 1998), UCCA (Abend and Rappoport, 2013), semantic proto-roles (Reisinger et al., 2015) and broad-coverage semantic dependencies (Oepen et al., 2014). Fig. 1.4 shows a sentence and its annotations under three different semantic formalisms, as well as two syntactic formalisms. Efforts to collect annotations and maintain resources, are labor-intensive, as well as fragmented across formalisms. The total amount of available annotations for a supervised learning problem is thus sparser, making semantic structured prediction an arduous learning challenge to date.

learning models? Cognitive linguistics for language acquisition has shown repeatedly that language-awareness is a uniquely human trait (Pinker, 1993). Given that the artificial intelligence we have built is far less sophisticated than the human brain, this is an extraordinary requirement for current models.

Given that most applications ultimately care about the meaning of sentences in order to understand, process and extract information from them, semantic analysis is core to natural language processing. Throughout this thesis, we will be visiting several semantic structured prediction tasks. In Chapter 3 we address a dependency-based semantic role labeling task. In Chapter 4 we address three different span-based semantic tasks: frame-semantic role labeling, PropBank semantic role labeling and coreference resolution. Finally, Chapter 5 centers around contextualized representations for word tokens, which are general enough to have wide applicability, and hence need to be meaning-preserving.

As algorithms for the semantic analysis of natural language sentences have developed, the role of *syntax* has been repeatedly revisited. Syntax offers an incomplete but potentially useful view of semantic structure. Fig. 1.2 shows the syntactic tree and the semantic predicate-argument graph for a sentence; notice the large overlap between the two structures. Linguistic theories have argued for a very tight integration of syntactic and semantic processing (Steedman, 2000; Copestake and Flickinger, 2000), and many systems have used syntactic dependency or phrase-based parsers as preprocessing for semantic analysis (Gildea and Palmer, 2002; Punyakanok et al., 2008; Das et al., 2014). Because annotated training datasets for semantics will always be limited, we expect that syntax will continue to offer useful inductive bias, encouraging semantic models toward better generalization.

1.1.3 Contextualized Representation Learning

As syntax provides a blueprint for ascertaining what a valid sentence is, it bears an intimate relationship with language models, which answer the same question from a statistical perspective. Phrase-structure syntax, just like language models, also determines how sentences can be generated. As a result, there has been a long history of syntactically structured language models which estimate the joint probability of sentences and their underlying phrase-structure trees (Jelinek and Lafferty, 1991; Chelba and Jelinek, 2000; Roark, 2001; Emami and Jelinek, 2005; Dyer et al., 2016).

Language models have risen in importance as pretraining objectives. They result in contextualized representations of word tokens (CWR) which are given by internal states of language models trained at a large scale (Peters et al., 2018a; Radford et al., 2018; Devlin et al., 2018). These representations have been immensely successful due to the representational power offered via large-scale unsupervised training. However, most language models based on deep learning tend to treat language as a sequence of tokens. Given that language is organized hierarchically as a tree, such sequential biases are inappropriate for learning syntactic generalizations.

Inspired by syntactic language models, we want to investigate the role of syntax in producing CWR. Towards this, we seek to incorporate syntactic inductive biases in addition to the sequential biases in language models. However, this introduces a challenge—the success of CWR can be attributed to scale, and syntax can be expensive to produce accurately. Hence, using syntactic language models for producing CWR would be computationally prohibitive. In Chapter 5, we address this problem and suggest a trade-off between large-scale language modeling and incorporation of rich syntactic structure.

Syntactic Supervision Throughout this thesis, we will take into account syntactic supervision for different tasks, as outlined above. The syntax to take into consideration should structurally correlate with the target structure of interest. In general, incorporation of any kind of linguistic structure is beneficial when it can be done **efficiently**, possible when structures can be broken down into simple units which are learnable when training deep models on large datasets. Full syntactic processing is computationally expensive, however, and might not even be necessary. Our models are designed to preserve efficiency of computation—hence, we select a syntactic representation which is both closely related to the task, and hence useful, as well as cheap to acquire. We address the central question: is there a way for semantic analyzers to benefit from syntax with minimal computational cost of syntactic processing? In Chapter 3 we will discuss a task guided by syntactic dependency trees, and in Chapter 4 phrase-structured trees. In the latter, we only take into account *parts* of the syntactic tree which correspond highly to the substructures we are interested in. In Chapter 5 we use a very simple approximation of a phrase-syntactic tree given by shallow syntactic structure. The selection of these sources of supervision is also motivated by the ease of procuring relevant data.

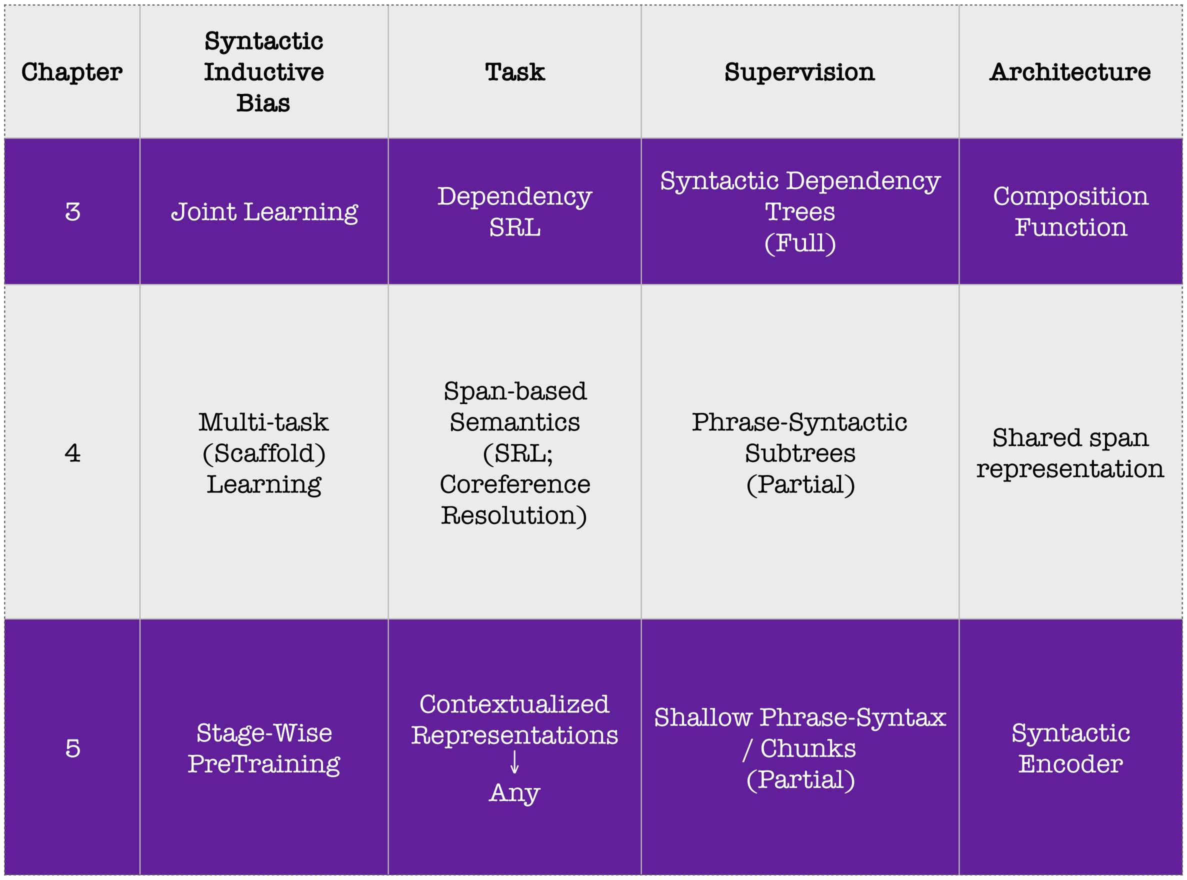
Thesis Goal The goal of this thesis is to study the role of **syntactic inductive biases** in deep neural models of language. Towards this end, we will consider two broad applications—semantic structure prediction and contextualized representation learning. For each task, we will ask two broad questions:

- What is a good syntactic inductive bias for this task? This will involve designing learning objectives which take into account syntax along with the task of interest.
- How can we leverage deep learning architectures to enable learning multiple structures such that there is an adequate amount of information sharing?

1.2 Road Map

This thesis is organized as follows.

First, we will visit some background on various syntactic inductive biases in Chapter 2. Next, we will discuss the three major contributions of this thesis, each organized



Chapter	Syntactic Inductive Bias	Task	Supervision	Architecture
3	Joint Learning	Dependency SRL	Syntactic Dependency Trees (Full)	Composition Function
4	Multi-task (Scaffold) Learning	Span-based Semantics (SRL; Coreference Resolution)	Phrase-Syntactic Subtrees (Partial)	Shared span representation
5	Stage-Wise PreTraining	Contextualized Representations ↓ Any	Shallow Phrase-Syntax / Chunks (Partial)	Syntactic Encoder

Figure 1.3: Road map for this thesis. Each row summarizes a chapter—syntactic inductive biases used for the tasks addressed, the supervision signal that allows incorporation of the bias, and the architectural element that facilitates this integration. The source of supervision can be full or partial, indicating how much of the full syntactic analysis of a sentence is consumed. In Chapter 5, the pretraining task results in contextualized representations, which could then be plugged into *any* downstream task.

as a chapter. See Fig. 1.3 for a diagrammatic plan for this thesis.

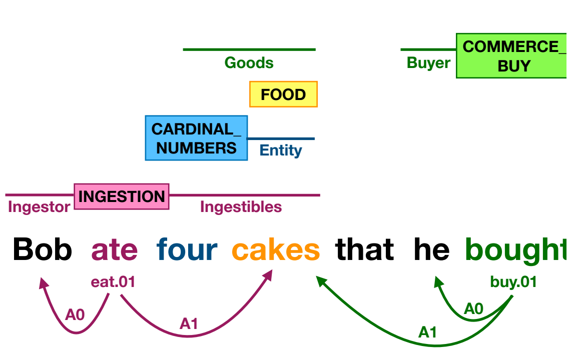
In Chapter 3, we will address the task of semantic dependency parsing, by taking into account syntactic dependency trees. Our approach involves *jointly* learning the entire syntactic dependency tree for a given sentence along with its semantic graph. Joint models have frequently been proposed as a way to avoid cascading errors in NLP pipelines. Unlike prior approaches, our approach involves learning intermediate *continuous* representations of syntactic and semantic structure jointly, where the syntactic inductive biases are captured in continuous composed representations of syntax and semantics. Our model surpassed the performance of all prior joint models, achieving competitive performance overall with the state-of-the-art in seven different languages, and two shared tasks. This work was published as Swayamdipta et al. (2016).

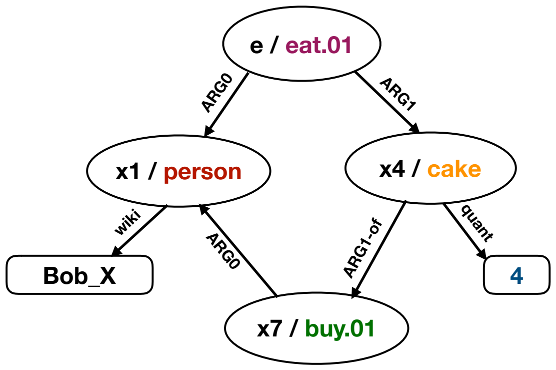
In Chapter 4, we introduce a multi-task learning paradigm where one of the tasks is prediction of a shallow approximation of syntax, for several span-based semantic structured prediction tasks. Here, we consider three different semantic tasks, each involving classification of spans of text as semantic arguments (see an example in Figure 2.2). The key observation we made was that semantic argument spans almost always correspond to valid syntactic phrases. The syntactic inductive biases are incorporated in learning sentential span representations, which are also helpful for predicting semantic arguments. We use a multi-task setting to learn the entire frame-semantic graph from FrameNet annotations, and *only relevant parts* (constituents) of the syntactic tree from The Penn TreeBank (Marcus et al., 1993), as a **syntactic scaffold**. Our experiments demonstrate that the syntactic scaffold offers a significant boost to state-of-the-art baselines for two flavors of semantic role labeling and a coreference resolution task. This work is presented in Swayamdipta et al. (2017) and Swayamdipta et al. (2018b).

In Chapter 5, we explore the question of whether exposing language models to shallow syntactic biases is required for improved contextualized representations. We construct a probe for these models by building a richer, syntax-aware language model. Here we consider a very shallow approximation of syntax, given by base phrase chunks, which can be predicted efficiently and with high accuracy on large corpora which we need to train our language model. We use a hierarchical syntactic encoder, internal states of which capture the syntactic inductive biases given by the shallow, predicted syntax. We observe that while such a syntactic encoder does not hurt performance, it also does not significantly improve performance across tasks, leading to the conclusion that large language models already capture the information to be provided by shallow, predicted syntax.

Overall, we demonstrate that syntactic biases are most helpful for tasks where the desired output is closely aligned with syntactic structure. Throughout this thesis, we will emphasize on carefully selecting the kind of supervision we want to exploit which will be the most beneficial for higher task accuracy, without sacrificing model efficiency. The methods presented throughout this thesis are general learning paradigms for incorporating structural biases, and are not limited solely to syntactic structure, nor are the downstream applications limited to the ones discussed in this thesis. For instance, analogous semantic biases could be useful for capturing deeper phenomena as observed in discourse and pragmatics.

Because of its ability to provide generalization to new, unseen examples, linguistic structure has been and will remain a cornerstone in natural language processing. Even as models of language become more sophisticated and powerful, we hypothesize this capability to be substantive in the years to come; linguistic structural biases will remain relevant. However, the particulars of structural bias design and incorporation are still open to exploration; we anticipate progress on this front. Perhaps the advancements will move towards capturing structural biases implicitly. The work in this thesis is built around the close relationship of linguistic structure to generalization—this central premise will continue to hold even as we make huge strides of progress in NLP.





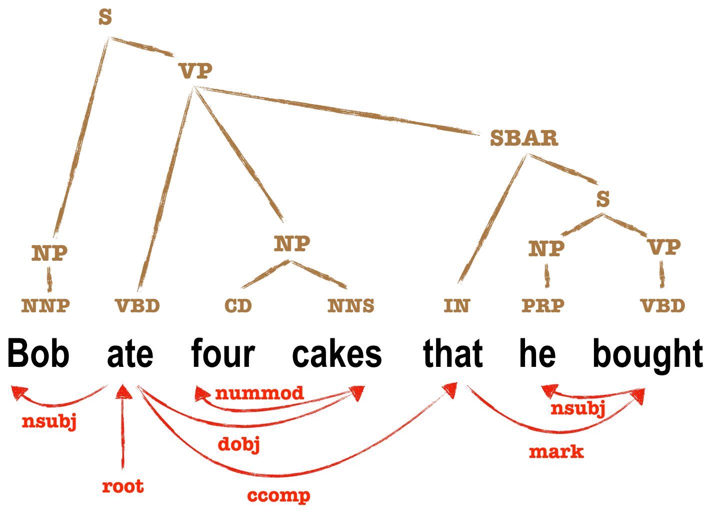


Figure 1.4: Different semantic annotations for a sentence. Top Left: FrameNet (Baker et al., 1998) (above) and PropBank (Palmer et al., 2005) (below) semantic graphs and for an example sentence. PropBank dependency (Surdeanu et al., 2008) nodes are individual tokens of the sentence, Target words and phrases are highlighted in the sentence, and their lexical units are shown italicized below. Frames are shown in colored blocks, and frame element segments are shown horizontally alongside the frame. Top Right: Abstract meaning representation graph (Banarescu et al., 2012) for the same sentence, annotations from Gruzitis and Barzdins (2016). Bottom: Syntactic constituency tree in grey above the sentence and (universal) dependency tree in red below.

Chapter 2

Background: Structural Inductive Biases

Representation learning with linguistic structure necessitates special attention to selection of algorithms which can make the most of diverse structural resources. Algorithms for such tasks could include traditional pipelining of various structures (Chapter 5), joint algorithms for multiple structures (Chapter 3), multitask learning algorithms (Chapter 4), and those which treat structures as latent variables (Swayamdipta et al., 2018a). We motivated the use of structural inductive biases in Chapter 1 (§1.1). In this chapter, we will review some common approaches to incorporate such biases into language learning.

2.1 Stage-Wise Learning or Pipelines

In a typical pipeline, two tasks T_1 and T_2 are separately trained, with the output of T_2 used to define the inputs to T_1 (Wolpert, 1992). Using syntax as T_2 in a pipeline is perhaps the most common approach for semantic structure prediction (Toutanova et al., 2008; Yang and Mitchell, 2017; Wiseman et al., 2016). Many systems have used syntactic dependency or phrase-based parsers as preprocessing for semantic analysis (Gildea and Palmer, 2002; Xue and Palmer, 2004; Punyakanok et al., 2008; Das et al., 2014). The most common approach of incorporating syntactic features is to have access to the complete syntactic parse tree of the text under consideration, beforehand. This involves syntactic parsing of sentences, the output of which is fed in a pipeline to the semantic parser. Pipelines generally tend to achieve state-of-the-art performance on structured prediction tasks (Toutanova et al., 2008; Björkelund et al., 2010; FitzGerald et al., 2015; Yang and Mitchell, 2017). One reason pipelines often dominate is that they make available the complete syntactic parse tree at all stages of T_1 processing (or inference).

However, pipelines introduce the problem of cascading errors (T_2's mistakes affect

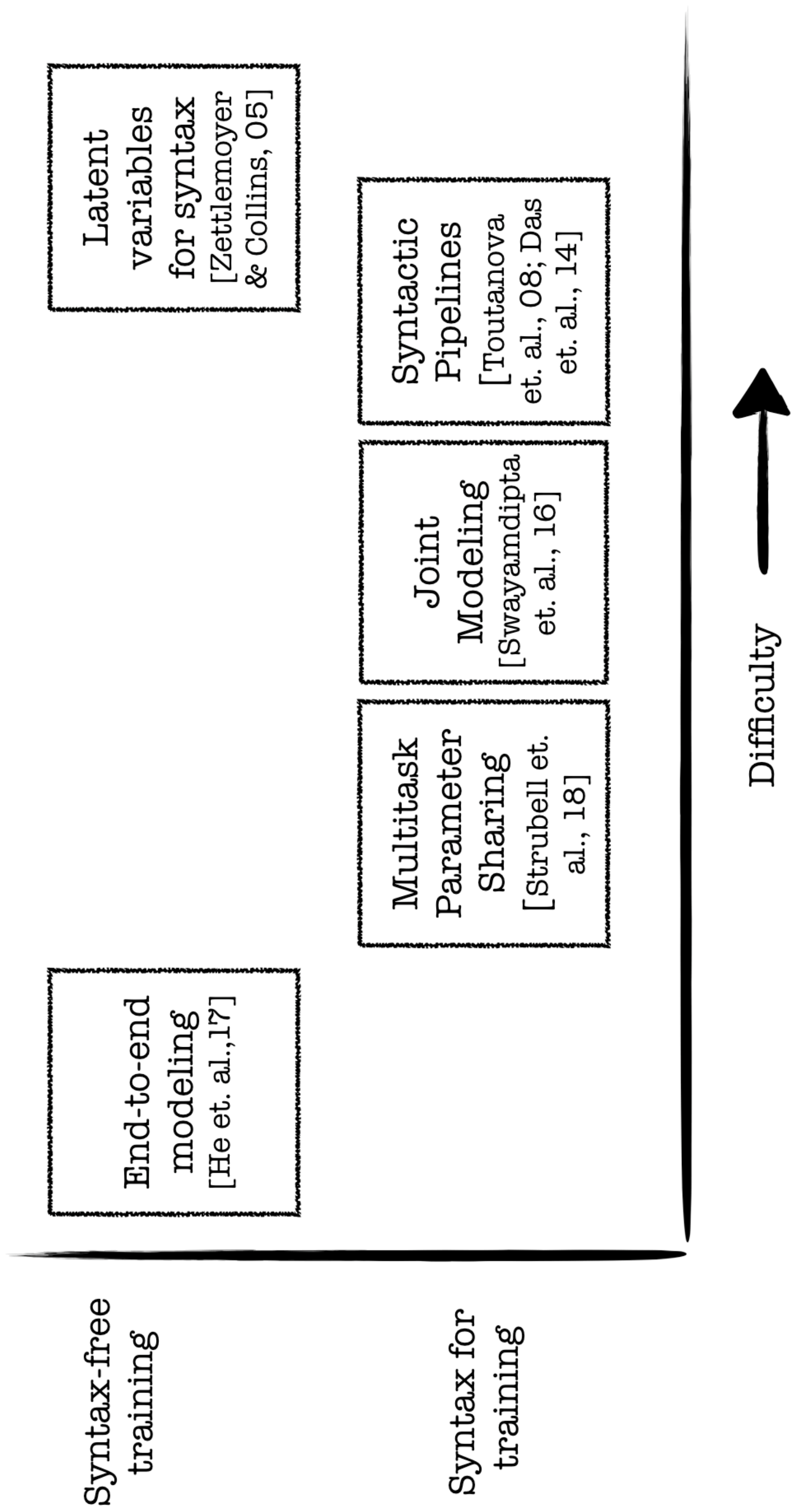

Syntax-free training
Syntax for training
End-to-end modeling [He et. al.,17]
Multitask Parameter Sharing [Strubell et. al., 18]
Joint Modeling [Swayamdipta et. al., 16]
Syntactic Pipelines [Toutanova et. al., 08; Das et. al., 14]
Latent variables for syntax [Zettlemoyer & Collins, 05]
Difficulty

Figure 2.1: Different kinds of inductive biases taking into account different degrees of syntax.

the performance, and perhaps the training, of T_1; He et al., 2013). Syntactic parsers are trained almost exclusively on the news domain Marcus et al. (1993), and generalizing outside of it has proven difficult (McClosky, 2010). Errors in syntactic parsing thus gravely hurt the quality of semantic graphs (Punyakanok et al., 2008), and this effect is especially marked in non-news text. Dependence on syntax also means that features must be re-engineered every time we consider a new semantic graph formalism. Further, extraction of these intricately designed features increases run time cost (He et al., 2013). To date, remedies to cascading errors are so computationally expensive as to be impractical (e.g., Finkel et al., 2006).

Pretraining with language models is another example of a stage-wise learning approach (Peters et al., 2018a; Radford et al., 2018; Devlin et al., 2018). In this setting, T_1 is a language model; representations learned as a side-effect are applied to downstream tasks T_2. These representations have been immensely successful due to the representational power offered via large-scale unsupervised training. In Chapter 5 we will see a stage-wise pretraining approach for syntax-aware language modeling.

2.2 Joint Learning

The motivation behind joint learning of syntactic and semantic representations is that any one task is helpful in predicting the other (Lluís and Màrquez, 2008; Lluís et al., 2013; Henderson et al., 2013; Swayamdipta et al., 2016). This typically requires joint prediction of the outputs of T_1 and T_2, which tends to be computationally expensive at both training and test time.

Joint models have frequently been proposed as a way to avoid cascading errors in NLP pipelines; varying degrees of success have been attained for a range of joint syntactic-semantic analysis tasks (Sutton and McCallum, 2005; Henderson et al., 2008; Toutanova et al., 2008; Johansson, 2009; Lluís et al., 2013, *inter alia)*. Both structures can be combined to form a joint graph with distinct edges for syntactic and semantic relations. Joint models can potentially model the uncertainty of a syntactic decision—a feat pipelines cannot achieve. In Chapter 3, we will address the task of semantic dependency parsing, by taking into account syntactic dependency trees.

While joint models are excellent representation learners, they require complex inference algorithms which need to account for individual structural constraints. A simplification is offered by multitask learning, where each task gets to retain some of its own parameters, while sharing the rest with others.

2.3 Multi-Task Learning

As a first step towards building linguistic structure-aware representations, we need to select informative structures, annotations for which do not always agree. However, taken together, different linguistic annotations provide interdependent views on the same underlying object: the meaning of a sentence. This makes it possible to synthesize multiple annotations to learn better representations.

Multitask learning (Caruana, 1997) is a collection of techniques in which two or more tasks are learned from data with at least some parameters shared. A range of examples have recently been explored in NLP, showing in particular how neural architectures achieve better performance when learned for multiple tasks together (Collobert et al., 2011; Luong et al., 2015; FitzGerald et al., 2015; Chen et al., 2017; Peng et al., 2017; Hashimoto et al., 2017).

Neural architectures have often yielded performance gains when trained for multiple tasks together (Collobert et al., 2011; Luong et al., 2015; Chen et al., 2017; Hashimoto et al., 2017). In particular, performance of semantic role labeling tasks improves when done jointly with other semantic tasks (FitzGerald et al., 2015; Peng et al., 2017, 2018a). Hershcovich et al. (2018) proposed a multitask learning setting for universal syntactic dependencies and UCCA semantics (Abend and Rappoport, 2013).

This framework is particularly useful when each task deserves slightly different treatment, particularly in the structural assumptions - such is the case for learning shallow syntax with full semantic structures (Chapter 4).

Multiple Semantic Sources Not only is syntax complementary to semantics, various semantic formalisms can also be complementary to each other. Recent approaches have experimented with learning from multiple semantic sources. Peng et al. (2018a) used supervision from two different, structurally divergent semantic formalisms—semantic dependency graphs and span-based semantic graphs. They observed gains in performance in both formalisms when considering cross-task features. Similarly, in Mulcaire et al. (2018), the same semantic dependency formalism was considered across seven different languages, resulting in improvements in languages which are resource poor, when incorporating supervision from English semantic dependencies.

Fig. 1.4 illustrates different broad-coverage semantic formalisms—frame-semantic graphs from FrameNet (Baker et al., 1998), semantic role labeling (SRL) structures from PropBank (Palmer et al., 2005), and abstract meaning representations (AMR) from AMRBank (Banarescu et al., 2012) for a single sentence, along with its syntactic trees in dependency and phrase-structure forms.

2.3.1 Varying Degrees of Supervision

While leveraging supervision for different flavors of linguistic structure is beneficial, it can be expensive to learn the intricacies of each kind of structure. This is particularly true when one of the sources is syntax, which is constrained to be tree-structured. Moreover, in Chapter 3, we will see that learning syntax is helpful for semantics, but we did not observe improvements in the reverse direction. Hence, the amount of syntax learnt could be reduced such that it is still beneficial to semantics, and yet does not expend too much modeling effort.

Guided by this intuition, in Chapter 4, we will downgrade syntactic learning to an auxiliary task which learned *shallow* syntactic structures, only to aid the primary task of semantic structure learning. We will use a multi-task setting to learn the entire semantic structure, and *only some relevant parts* of the syntactic tree.

2.4 Latent Variable Learning

Another solution is to treat the output of T_2 as a (perhaps structured) latent variable. This approach obviates the need of supervision for T_2 and requires marginalization (or some approximation to it) in order to reason about the outputs of T_1. Syntax as a latent variable for semantics was explored by Zettlemoyer and Collins (2005) and Naradowsky et al. (2012).

Peng et al. (2018b) treated semantic dependency labels as latent when trying to predict span-based arguments. Another application is question answering with distant supervision (Swayamdipta et al., 2018a). Our model had access to the question, the answer string, and lengthy documents which contained the answer string during training, but no knowledge of which exact passage in the document answered the question. Since the answer string could appear in different contexts, where not every occurrence could answer the question, we used a latent variable to model this uncertainty. The lengthy document itself was represented as a simple bag-of-embeddings. This approach was able to surpass the performance of more powerful recurrent neural models which relied on truncating the document for managing space and time complexity.

2.5 End-to-end Learning

End-to-end learning refers to training a possibly complex task using a single objective such that gradient-based learning can be applied to all the modules together. End-to-end approaches have regained popularity with the advent of deep learning, for applications

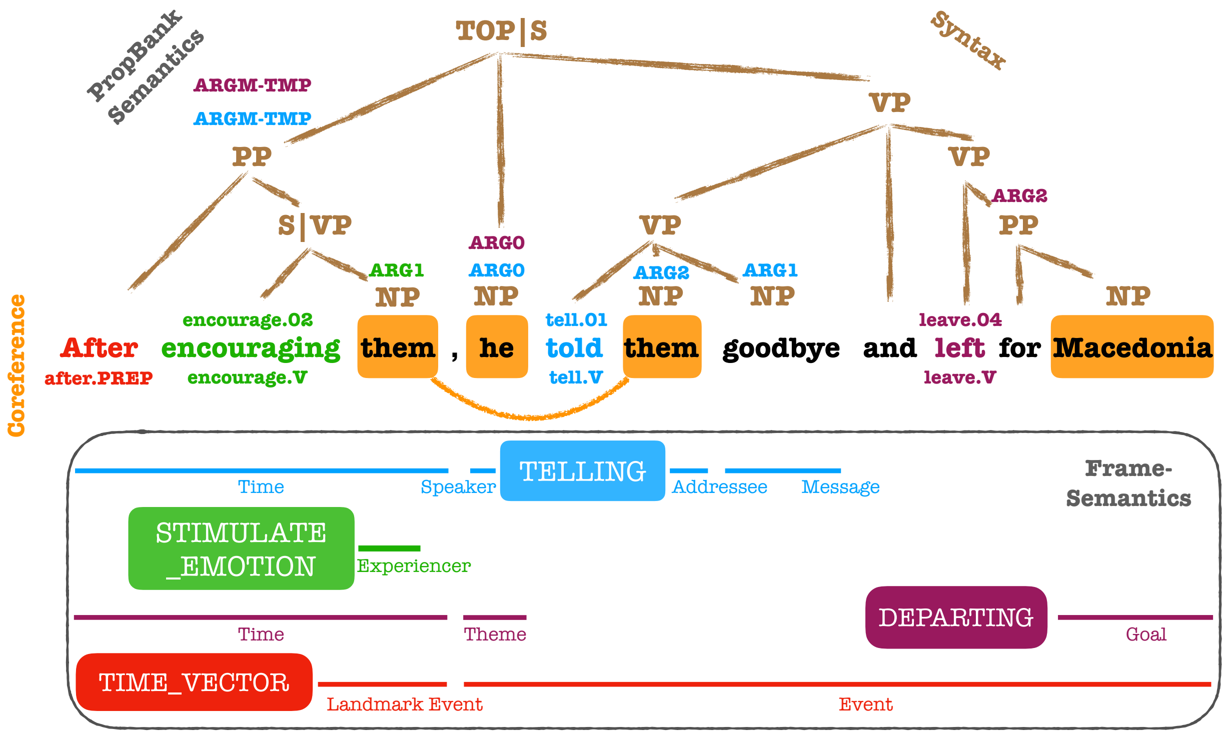


Figure 2.2: Syntax and semantics are closely related. The syntactic constituency tree is shown in brown above the sentence. Also shown alongside syntax are PropBank (Palmer et al., 2005) predicates and corresponding (color-coded) arguments; arguments are annotated on top of syntactic nodes, indicating a full overlap. Coreference chains (connecting references to the same entity) are shown in orange, fully overlapping with syntactic nodes. Color-coded FrameNet (Baker et al., 1998) frames are shown in layers below the sentence, each frame indicates an elementary event, and contains sentential spans as arguments to understand the event. Almost all frame-semantic arguments (except `Event`) are also syntactic constituents. The three different semantic structures also show considerable overlap with one another.

in reinforcement learning (Mnih et al., 2015), control and automation (Bojarski et al., 2016). Due to the presence of a single objective, the learned representations are biased only towards that objective. As a result, to achieve high performance, end-to-end architectures tend to be very complex. Transfer to other tasks, as well as generalization, suffers as a result (Glasmachers, 2017).

In NLP end-to-end methods which forgo explicit syntactic processing altogether have been applied to semantic structured prediction (Zhou and Xu, 2015; He et al., 2017; Lee et al., 2017; Peng et al., 2017). However, it has been shown that syntax still offers benefits to highly engineered end-to-end architectures (He et al., 2017; Yang and Mitchell, 2017).

Chapter 3

Joint Learning of Semantics and Entire Syntactic Trees

In this chapter, we focus on syntactic inductive biases that result from exposure to full trees.

We consider the task of dependency-based semantic role labeling (Surdeanu et al., 2008; Hajič et al., 2009). Semantic dependency structures highlight relationships between semantic predicates and their arguments. Semantic predicates are words or multi-word expressions in a sentence which denote the various events in the sentence, and are associated with arguments which fill in the roles of the different actors in the event. Edges between predicates and arguments are labeled with the specific role each argument plays. Figure 3.1 shows an example of different flavors semantic dependencies. In this chapter, we only consider semantic dependencies based on semantic role labeling annotations from PropBank and NomBank (Fig. 3.1a).

There exists a close correspondence between syntactic dependencies and SRL dependencies. Figure 3.2 illustrates this for an example sentence; we see one-to-one mappings between several semantic and syntactic dependency arcs. Because of this close correspondence, these structures can be trained jointly; knowing one might help predict the other. This calls for expert syntactic and semantic annotations on the same data, i.e. direct supervision of syntactic as well as semantic dependencies. Shared tasks from CoNLL in 2008 (Surdeanu et al., 2008) and 2009 (Hajič et al., 2009) were devoted to the task of prediction of syntactic and semantic dependencies; the focus in 2008 being on English and in 2009 on seven different languages.[1]

One solution is given by pipelines—predict syntactic dependencies, extract features from syntax, and use those to predict the semantic dependencies. Such pipelines are

[1]Catalan, Chinese, Czech, English, German, Japanese and Spanish.

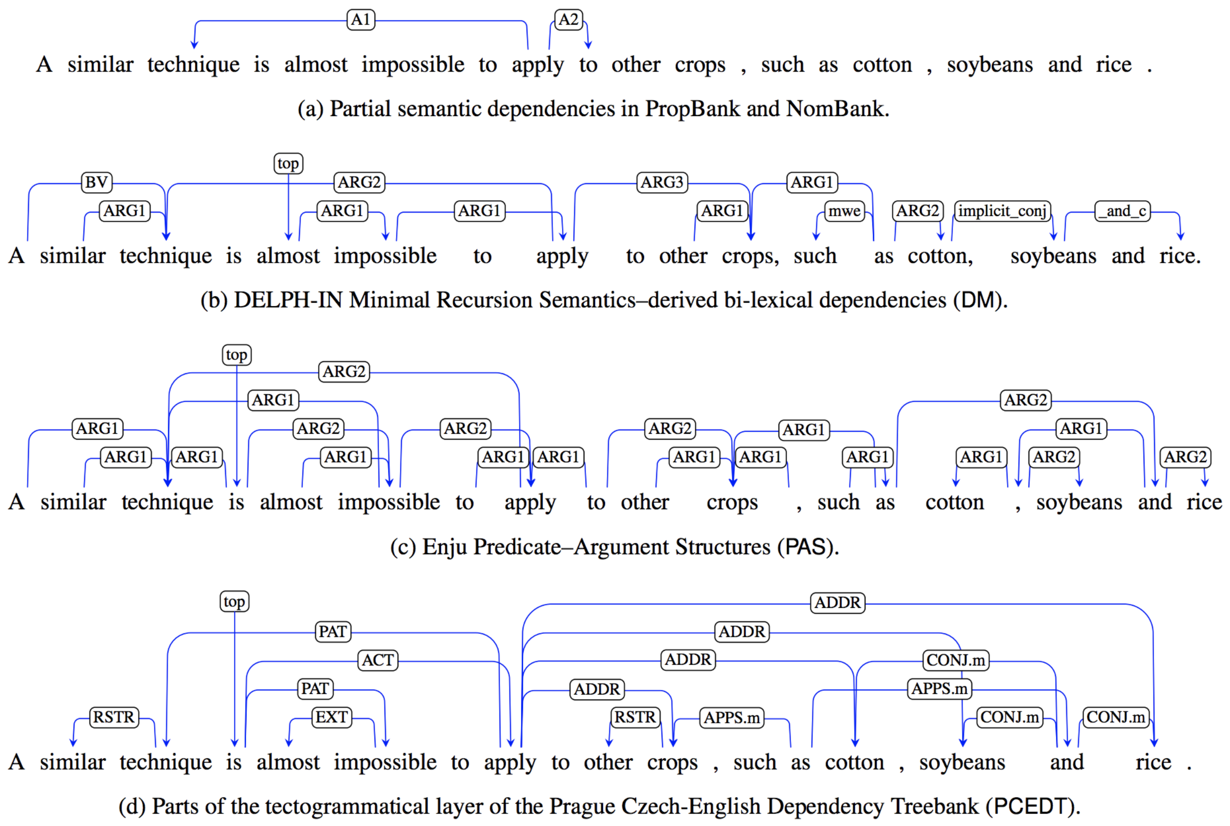
abundantly used since they make available the complete syntactic parse tree. They also enable arbitrarily-scoped syntactic features—such as the "path" between predicate and argument, proposed by Gildea and Jurafsky (2002)—for semantic analysis. Such features are a mainstay of high performance semantic role labeling systems (Roth and Woodsend, 2014; Lei et al., 2015; FitzGerald et al., 2015; Foland and Martin, 2015). However, such features are expensive to extract (Johansson, 2009; He et al., 2013). Moreover, pipelines are vulnerable to cascading errors—errors made during the syntactic parsing phase are bound to be problematic during the semantic parsing phase also.

Training Objective The solution we outline in this chapter is to *jointly* learn the entire syntactic dependency tree for a given sentence along with its semantic graph. Joint models have frequently been proposed as a way to avoid cascading errors in NLP pipelines; varying degrees of success have been attained for a range of joint syntactic-semantic analysis tasks (Sutton and McCallum, 2005; Henderson et al., 2008; Toutanova et al., 2008; Johansson, 2009; Lluís et al., 2013, *inter alia)*. Both structures can be combined to form a joint graph with distinct edges for syntactic and semantic relations. Joint models can potentially model the uncertainty of a syntactic decision—a feat pipelines cannot achieve.

Our parser draws from the algorithmic insights of the incremental structure building approach of Henderson et al. (2008), with two key differences. First, it learns representations for the parser's entire algorithmic state, not just the top items on the stack or the most recent parser states; in fact, it uses no expert-crafted features at all. Second, it uses entirely greedy inference rather than beam search. We leverage recent advances in representation learning that can bypass learning expensive features used in pipelines, discovering cheap alternatives available during a greedy parsing procedure. The specific advance we employ is the stack LSTM (Dyer et al., 2015), a neural network that continuously summarizes the contents of the stack data structures in which a transition-based parser's state is conventionally encoded. Stack LSTMs were shown to obviate many features used in syntactic dependency parsing; here we find them to do the same for joint syntactic-semantic dependency parsing. We believe this is an especially important finding for *greedy* models that cast parsing as a sequence of decisions made based on algorithmic state, where linguistic theory and researcher intuitions offer less guidance in feature design.

Syntactic Inductive Bias Our approach learns *intermediate continuous representations* of syntactic and semantic structure jointly. A composition function (§3.2.1) is used to create such representations, as parts of the joint graph are built incrementally. The representations are biased towards being both syntactically and semantically meaningful. Additionally, such intermediate continuous representations can bypass those expensive features used in pipelines.

Our model surpasses the performance of previous joint models on the CoNLL 2008


A1
A2
A similar technique is almost impossible to apply to other crops , such as cotton , soybeans and rice .
BV
ARG1
top
ARG2
ARG3
mwe
implicit_conj
_and_c
A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice.
A similar technique is almost impossible to apply to other crops , such as cotton , soybeans and rice
RSTR
PAT
ACT
EXT
ADDR
APPS.m
CONJ.m
A similar technique is almost impossible to apply to other crops , such as cotton , soybeans and rice .

(a) Partial semantic dependencies in PropBank and NomBank.

(b) DELPH-IN Minimal Recursion Semantics–derived bi-lexical dependencies (DM).

(c) Enju Predicate–Argument Structures (PAS).

(d) Parts of the tectogrammatical layer of the Prague Czech-English Dependency Treebank (PCEDT).

Figure 3.1: Different dependency semantic formalisms. Figure from Oepen et al. (2014) highlighting differences between PropBank-style dependencies and broad-coverage semantic dependencies. In this chapter, we deal only with the dependency SRL structures from CoNLL shared tasks from 2008-9 (shown in (a)). Dependency SRL structures are sparser—forming a forest of depth one trees, while other semantic formalisms form arbitrary graphs; predicates are predominantly verbal or nominal.

and 2009 tasks, without using expert-crafted, expensive features of the full syntactic parse. Specifically we show improvements over all prior joint models including Henderson et al. (2008) and variants (Gesmundo et al., 2009; Titov et al., 2009; Henderson et al., 2013) on the CoNLL 2008 and 2009 (English) shared tasks Hajič et al. (2009). Our parser's multilingual results are comparable to the top systems at CoNLL 2009. More importantly, we observed improvements over a baseline which predicted only the semantic graph, without any syntactic knowledge, showing that **access to syntax helped learn semantics**. This work was published as Swayamdipta et al. (2016).

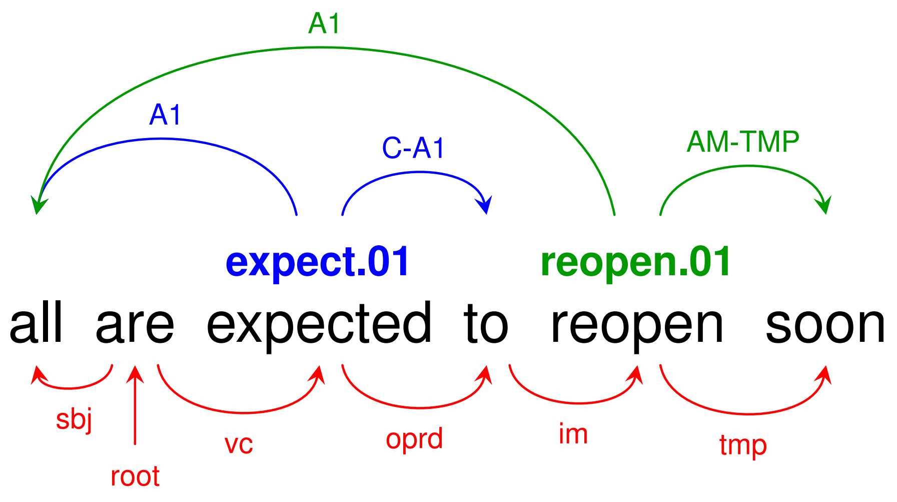


Figure 3.2: Syntactic and semantic role dependencies for a sentence from CoNLL shared tasks 2008-9. Syntactic dependencies are shown by red arcs below the sentence. Shown above the sentence, are semantic predicates, marked in boldface, and semantic dependencies, color-coded by their respective predicates. C- denotes continuation of argument A1. Correspondences between syntactic and semantic dependencies might be close (between *expected* and *to*) or not (between *reopen* and *all*).

3.1 Joint Syntactic and Semantic Dependency Parsing

We largely follow the transition-based, synchronized algorithm of Henderson et al. (2013) to predict joint parse structures. The input to the algorithm is a sentence annotated with part-of-speech tags. The output consists of a labeled syntactic dependency tree and a directed SRL graph, in which a subset of words in the sentence are selected as predicates, disambiguated to a sense, and linked by labeled, directed edges to their semantic arguments. Figure 3.2 shows an example.

3.1.1 Transition-Based Procedure

The two parses are constructed in a bottom-up fashion, incrementally processing words in the sentence from left to right. The state of the parsing algorithm at timestep t is represented by three stack data structures: a syntactic stack S_t, a semantic stack M_t—each containing partially built structures—and a buffer of input words B_t. Our algorithm also places partial syntactic and semantic parse structures onto the front of the buffer, so it is also implemented as a stack. Each arc in the output corresponds to a transition (or "action") chosen based on the current state; every transition modifies the state by updating S_t, M_t, and B_t to S_{t+1}, M_{t+1}, and B_{t+1}, respectively. While each state may license several valid actions, each action has a deterministic effect on the state of the algorithm.

Initially, S_0 and M_0 are empty, and B_0 contains the input sentence with the first word at the front of B and a special root symbol at the end.[2] Execution ends on iteration t such that B_t is empty and S_t and M_t contain only a single structure headed by root.

3.1.2 Transitions for Joint Parsing

There are separate sets of syntactic and semantic transitions; the former manipulate S and B, the latter M and B. All are formally defined in Table 3.1. The syntactic transitions are from the "arc-eager" algorithm of Nivre (2008). They include:

- S-Shift, which copies[3] an item from the front of B and pushes it on S.
- S-Reduce pops an item from S.
- S-Right(ℓ) creates a syntactic dependency. Let u be the element at the top of S and v be the element at the front of B. The new dependency has u as head, v as dependent, and label ℓ. u is popped off S, and the resulting structure, rooted at u, is pushed on S. Finally, v is copied to the top of S.
- S-Left(ℓ) creates a syntactic dependency with label ℓ in the reverse direction as S-Right. The top of S, u, is popped. The front of B, v, is replaced by the new structure, rooted at v.

The semantic transitions are similar, operating on the semantic stack.

- M-Shift removes an item from the front of B and pushes it on M.
- M-Reduce pops an item from M.
- M-Right(r) creates a semantic dependency. Let u be the element at the top of M and v, the front of B. The new dependency has u as head, v as dependent, and label r. u is popped off M, and the resulting structure, rooted at u, is pushed on M.
- M-Left(r) creates a semantic dependency with label r in the reverse direction as M-Right. The buffer front, v, is replaced by the new v-rooted structure. M remains unchanged.

Because SRL graphs allow a node to be a semantic argument of two parents—like *all*

[2]This works better for the arc-eager algorithm (Ballesteros and Nivre, 2013), in contrast to Henderson et al. (2013), who initialized with root at the buffer front.

[3]Note that in the original arc-eager algorithm (Nivre, 2008), Shift and Right-Arc actions *move* the item on the buffer front to the stack, whereas we only copy it (to allow the semantic operations to have access to it).

in the example in Figure 3.2—M-Left and M-Right do not remove the dependent from the semantic stack and buffer respectively, unlike their syntactic equivalents, S-Left and S-Right. We use two other semantic transitions from Henderson et al. (2013) which have no syntactic analogues:

- M-Swap swaps the top two items on M, to allow for crossing semantic arcs.
- M-Pred(p) marks the item at the front of B as a semantic predicate with the sense p, and replaces it with the disambiguated predicate.

The CoNLL 2009 corpus introduces semantic self-dependencies where many nominal predicates (from NomBank) are marked as their own arguments; these account for 6.68% of all semantic arcs in the English corpus. We introduce a new semantic transition, not in Henderson et al. (2013), to handle such cases:

- M-Self(r) adds a dependency, with label r between the item at the front of B and itself. The result replaces the item at the front of B.

Note that the syntactic and semantic transitions both operate on the same buffer, though they independently specify the syntax and semantics, respectively. In order to ensure that both syntactic and semantic parses are produced, the syntactic and semantic transitions are interleaved. Only syntactic transitions are considered until a transition is chosen that copies an item from the buffer front to the syntactic stack (either S-Shift or S-Right). The algorithm then switches to semantic transitions until a buffer-modifying transition is taken (M-Shift).[4] At this point, the buffer is modified and the algorithm returns to syntactic transitions. This implies that, for each word, its left-side syntactic dependencies are resolved before its left-side semantic dependencies. An example run of the algorithm is shown in Figure 3.3.

Constraints on Transitions

To ensure that the parser never enters an invalid state, the sequence of transitions is constrained, following Henderson et al. (2013). Actions that copy or move items from the buffer (S-Shift, S-Right and M-Shift) are forbidden when the buffer is empty. Actions that pop from a stack (S-Reduce and M-Reduce) are forbidden when that stack is empty. We disallow actions corresponding to the same dependency, or the same predicate to be repeated in the sequence. Repetitive M-Swap transitions are disallowed to avoid infinite swapping. Finally, as noted above, we restrict the parser to syntactic actions until it needs to shift an item from B to S, after which it can only execute semantic actions until it executes an M-Shift.

[4]Had we *moved* the item at the buffer front during the syntactic transitions, it would have been unavailable for the semantic transitions, hence we only *copy* it.

S_t	M_t	B_t	**Action**	S_{t+1}	M_{t+1}	B_{t+1}	**Dependency**
S	M	$(\mathbf{v}, v), B$	S-SHIFT	$(\mathbf{v}, v), S$	M	$(\mathbf{v}, v), B$	—
$(\mathbf{u}, u), S$	M	B	S-REDUCE	S	M	B	—
$(\mathbf{u}, u), S$	M	$(\mathbf{v}, v), B$	S-RIGHT(ℓ)	$(\mathbf{v}, v), (g_s(\mathbf{u}, \mathbf{v}, \mathbf{l}), u), S$	M	$(\mathbf{v}, v), B$	$\mathcal{S} \cup u \overset{\ell}{\to} v$
$(\mathbf{u}, u), S$	M	$(\mathbf{v}, v), B$	S-LEFT(ℓ)	S	M	$(g_s(\mathbf{v}, \mathbf{u}, \mathbf{l}), v), B$	$\mathcal{S} \cup u \overset{\ell}{\leftarrow} v$
S	M	$(\mathbf{v}, v), B$	M-SHIFT	S	$(\mathbf{v}, v), M$	B	—
S	$(\mathbf{u}, u), M$	B	M-REDUCE	S	M	B	—
S	$(\mathbf{u}, u), M$	$(\mathbf{v}, v), B$	M-RIGHT(r)	S	$(g_m(\mathbf{u}, \mathbf{v}, \mathbf{r}), u), M$	$(\mathbf{v}, v), B$	$\mathcal{M} \cup u \overset{r}{\to} v$
S	$(\mathbf{u}, u), M$	$(\mathbf{v}, v), B$	M-LEFT(r)	S	$(\mathbf{u}, u), M$	$(g_m(\mathbf{v}, \mathbf{u}, \mathbf{r}), v), B$	$\mathcal{M} \cup u \overset{r}{\leftarrow} v$
S	$(\mathbf{u}, u), (\mathbf{v}, v), M$	B	M-SWAP	S	$(\mathbf{v}, v), (\mathbf{u}, u), M$	B	—
S	M	$(\mathbf{v}, v), B$	M-PRED(p)	S	M	$(g_d(\mathbf{v}, \mathbf{p}), v), B$	—
S	M	$(\mathbf{v}, v), B$	M-SELF(r)	S	M	$(g_m(\mathbf{v}, \mathbf{v}, \mathbf{r}), v), B$	$\mathcal{M} \cup v \overset{r}{\leftrightarrow} v$

Table 3.1: Parser transitions along with the modifications to the stacks and the buffer resulting from each. Syntactic transitions are shown above, semantic below. Italic symbols denote symbolic representations of words and relations, and bold symbols indicate (learned) embeddings of words and relations; each element in a stack or buffer includes both symbolic and vector representations, either atomic or recursive. $\mathcal{S}$ represents the set of syntactic transitions, and $\mathcal{M}$ the set of semantic transitions.

Asymptotic runtime complexity of this greedy algorithm is linear in the length of the input, following the analysis by Nivre (2009).[5]

3.2 Statistical Model

The transitions in (§3.1.1) describe the execution paths our algorithm can take; like past work, we apply a statistical classifier to decide which transition to take at each timestep, given the current state. The novelty of our model is that it learns a finite-length vector representation of the state of the entire joint parser (S, M, and B) in order to make this decision.

Stack Long Short-Term Memory (LSTM)

LSTMs are recurrent neural networks equipped with specialized memory components in addition to a hidden state (Hochreiter and Schmidhuber, 1997; Graves, 2013) to model sequences. Stack LSTMs (Dyer et al., 2015) are LSTMs that allow for stack operations: *query*, *push*, and *pop*. A "stack pointer" is maintained which determines which cell in the LSTM provides the memory and hidden units when computing the new memory cell contents. *Query* provides a summary of the stack in a single fixed-length vector. *Push* adds an element to the top of the stack, resulting in a new summary. *Pop*, which does not correspond to a conventional LSTM operation, moves the stack pointer to the preceding timestep, resulting in a stack summary as it was before the popped item was observed. Implementation details (Dyer et al., 2015; Goldberg, 2016) and code have been made publicly available.[6]

Using stack LSTMs, we construct a representation of the algorithm state by decomposing it into smaller pieces that are combined by recursive function evaluations (similar to the way a list is built by a *concatenate* operation that operates on a list and an element). This enables information that would be distant from the "top" of the stack to be carried forward, potentially helping the learner.

3.2.1 Stack LSTMs for Joint Parsing

Our algorithm employs four stack LSTMs, one each for the S, M, and B data structures. Like Dyer et al. (2015), we use a fourth stack LSTM, A, for the history of actions—A is

[5]The analysis in (Nivre, 2009) does not consider SWAP actions. However, since we constrain the number of such actions, the linear time complexity of the algorithm stays intact.

[6]https://github.com/clab/lstm-parser

Transition	*S*	*M*	*B*	**Dependency**
	[]	[]	[all, are, expected, to, reopen, soon, root]	—
S-Shift	[all]	[]	[all, are, expected, to, reopen, soon, root]	—
M-Shift	[all]	[all]	[are, expected, to, reopen, soon, root]	—
S-Left(*sbj*)	[]	[all]	[are, expected, to, reopen, soon, root]	all $\xleftarrow{\text{sbj}}$ are
S-Shift	[are]	[all]	[are, expected, to, reopen, soon, root]	—
M-Shift	[are]	[all, are]	[expected, to, reopen, soon, root]	—
S-Right(*vc*)	[are, expected]	[all, are]	[expected, to, reopen, soon, root]	are $\xrightarrow{\text{vc}}$ expected
M-Pred(**expect.01**)	[are, expected]	[all, are]	[expected, to, reopen, soon, root]	—
M-Reduce	[are, expected]	[all]	[expected, to, reopen, soon, root]	—
M-Left(*A1*)	[are, expected]	[all]	[expected, to, reopen, soon, root]	all $\xleftarrow{\text{A1}}$ expect.01
M-Shift	[are, expected]	[all, expected]	[to, reopen, soon, root]	—
***S-Right(oprd)	[are, expected, to]	[all, expected]	[to, reopen, soon, root]	expected $\xrightarrow{\text{oprd}}$ to
M-Right(*C-A1*)	[are, expected, to]	[all, expected]	[to, reopen, soon, root]	expect.01 $\xrightarrow{\text{C-A1}}$ to
M-Reduce	[are, expected, to]	[all]	[to, reopen, soon, root]	—
M-Shift	[are, expected, to]	[all, to]	[reopen, soon, root]	—
S-Right(*im*)	[are, expected, to, reopen]	[all, to]	[reopen, soon, root]	to $\xrightarrow{\text{im}}$ reopen
M-Pred(**reopen.01**)	[are, expected, to, reopen]	[all, to]	[reopen, soon, root]	—
M-Reduce	[are, expected, to, reopen]	[all]	[reopen, soon, root]	—
M-Left(*A1*)	[are, expected, to, reopen]	[all]	[reopen, soon, root]	all $\xleftarrow{\text{A1}}$ reopen.01
M-Reduce	[are, expected, to, reopen]	[]	[reopen, soon, root]	—
M-Shift	[are, expected, to, reopen]	[reopen]	[soon, root]	—
S-Right(*tmp*)	[are, expected, to, reopen, soon]	[reopen]	[soon, root]	reopen $\xrightarrow{\text{tmp}}$ soon
M-Right(*AM-TMP*)	[are, expected, to, reopen, soon]	[reopen]	[soon, root]	reopen.01 $\xrightarrow{\text{AM-TMP}}$ soon
M-Reduce	[are, expected, to, reopen, soon]	[]	[soon, root]	—
M-Shift	[are, expected, to, reopen, soon]	[soon]	[root]	—
S-Reduce	[are, expected, to, reopen]	[soon]	[root]	—
S-Reduce	[are, expected, to]	[soon]	[root]	—
S-Reduce	[are, expected]	[soon]	[root]	—
S-Reduce	[are]	[soon]	[root]	—
S-Left(*root*)	[]	[soon]	[root]	are $\xleftarrow{\text{root}}$ root
S-Shift	[root]	[soon]	[root]	—
M-Reduce	[root]	[]	[root]	—
M-Shift	[root]	[root]	[]	—

Figure 3.3: Joint parser transition sequence for the sentence in Figure 3.2, "*all are expected to reopen soon*." Syntactic labels are in lower-case and semantic role labels are capitalized. *** marks the operation predicted in Figure 3.5.

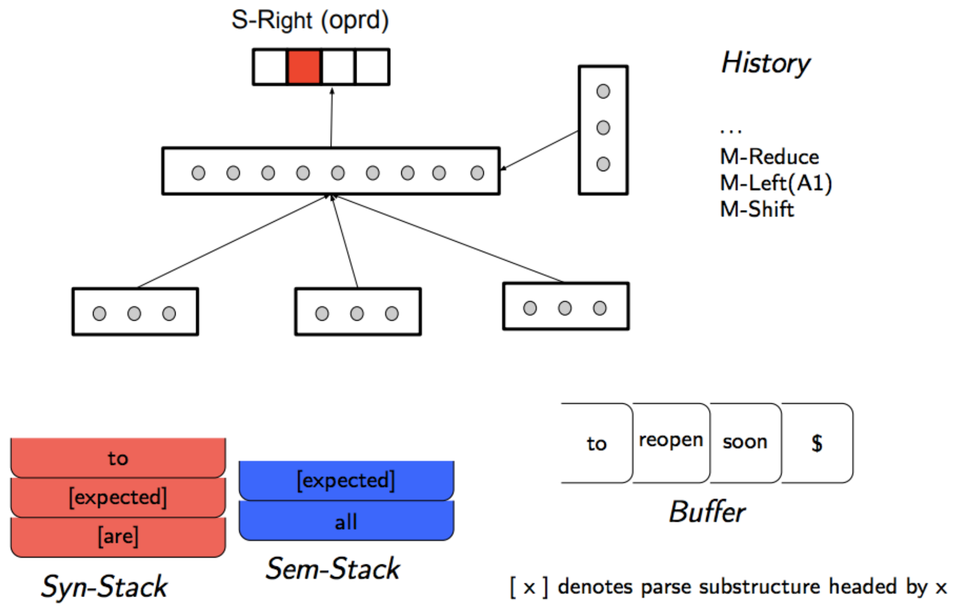


Figure 3.4: Stack LSTM-based architecture for joint parsing. The state illustrated corresponds to the ***-marked row in the example transition sequence in Fig. 3.3. Syntactic components are shown in red, and semantic in blue.

never popped from, only pushed to. Figure 3.4 illustrates the architecture. The algorithm's state at timestep t is encoded by the four vectors summarizing the four stack LSTMs, and this is the input to the classifier that chooses among the allowable transitions at that timestep.

Let $\mathbf{s}_t$, $\mathbf{m}_t$, $\mathbf{b}_t$, and $\mathbf{a}_t$ denote the summaries of S_t, M_t, B_t, and A_t, respectively. Let $\mathcal{A}_t = \text{Allowed}(S_t, M_t, B_t, A_t)$ denote the allowed transitions given the current stacks and buffer. The parser state at time t is given by a rectified linear unit (Nair and Hinton, 2010) in vector $\mathbf{y}_t$:

$$\mathbf{y}_t = \text{elementwisemax}\left\{\mathbf{0}, \mathbf{d} + \mathbf{W}[\mathbf{s}_t; \mathbf{m}_t; \mathbf{b}_t; \mathbf{a}_t]\right\}$$

where $\mathbf{W}$ and $\mathbf{d}$ are the parameters of the classifier. The transition selected at timestep t is

$$\begin{aligned} &\arg\max_{\tau \in \mathcal{A}_t} q_\tau + \boldsymbol{\theta}_\tau \cdot \mathbf{y}_t \qquad (3.1) \\ \equiv &\arg\max_{\tau \in \mathcal{A}_t} \text{score}(\tau; S_t, M_t, B_t, A_t) \end{aligned}$$

where $\boldsymbol{\theta}_\tau$ and q_τ are parameters for each transition type τ. Note that only allowed transitions are considered in the decision rule (see §3.1.2).

Composition Functions

To use stack LSTMs, we require vector representations of the elements that are stored in the stacks. Specifically, we require vector representations of atoms (words, possibly with part-of-speech tags) and parse fragments. Word vectors can be pretrained or learned directly; we consider a concatenation of both in our experiments; part-of-speech vectors are learned and concatenated to the same.

To obtain vector representations of parse fragments, we use neural networks which recursively compute representations of the complex structured output (Dyer et al., 2015). The tree structures here are always ternary trees, with each internal node's three children including a head, a dependent, and a label. The vectors for leaves are word vectors and vectors corresponding to syntactic and semantic relation types.

The vector for an internal node is a squashed ($\tanh$) affine transformation of its children's vectors. For syntactic and semantic attachments, respectively, the composition function is:

$$g_s(\mathbf{v}, \mathbf{u}, \mathbf{l}) = \tanh(\mathbf{Z}_s[\mathbf{v}; \mathbf{u}; \mathbf{l}] + \mathbf{e}_s) \tag{3.2}$$

$$g_m(\mathbf{v}, \mathbf{u}, \mathbf{r}) = \tanh(\mathbf{Z}_m[\mathbf{v}; \mathbf{u}; \mathbf{r}] + \mathbf{e}_m) \tag{3.3}$$

where $\mathbf{v}$ and $\mathbf{u}$ are vectors corresponding to atomic words or composed parse fragments; $\mathbf{l}$ and $\mathbf{r}$ are learned vector representations for syntactic and semantic labels respectively. Syntactic and semantic parameters are separated ($\mathbf{Z}_s$, $\mathbf{e}_s$ and $\mathbf{Z}_m$, $\mathbf{e}_m$, respectively).

Finally, for predicates, we use another recursive function to compose the word representation, $\mathbf{v}$ with a learned representation for the dismabiguated sense of the predicate, $\mathbf{p}$:

$$g_d(\mathbf{v}, \mathbf{p}) = \tanh(\mathbf{Z}_d[\mathbf{v}; \mathbf{p}] + \mathbf{e}_d) \tag{3.4}$$

where $\mathbf{Z}_d$ and $\mathbf{e}_d$ are parameters of the model. Note that, because syntactic and semantic transitions are interleaved, the fragmented structures are a blend of syntactic and semantic compositions. Figure 3.5 shows an example.

3.3 Experiments and Results

3.3.1 Experimental Setup

Our model is evaluated on the CoNLL shared tasks on joint syntactic and semantic dependency parsing in 2008 (Surdeanu et al., 2008) and 2009 (Hajič et al., 2009). The standard training, development and test splits of all datasets were used. Per the shared task

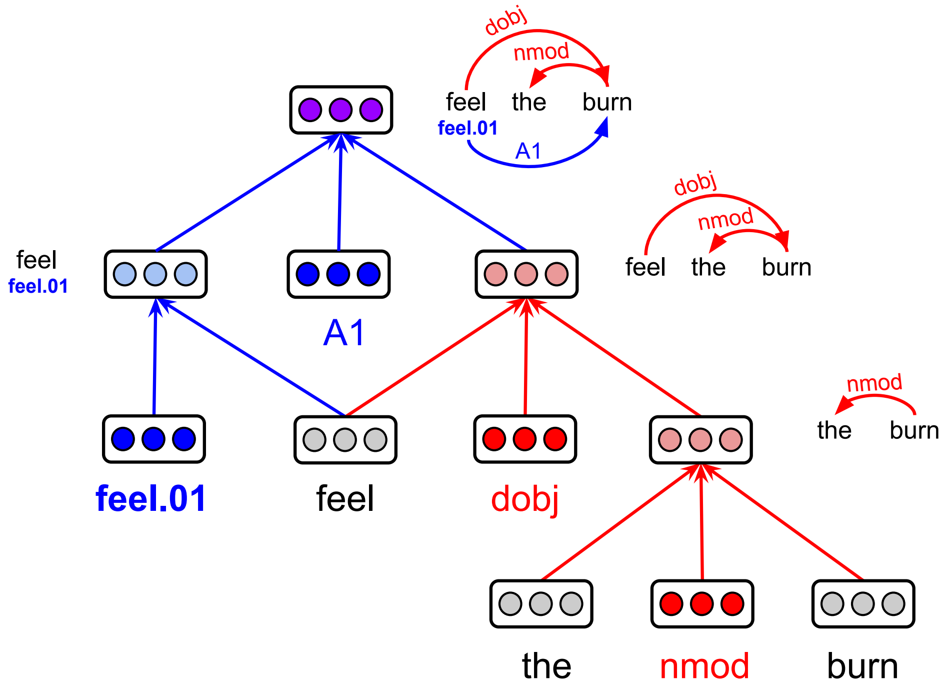


Figure 3.5: Illustration of the model composition architecture for a joint parse. Each node is associated with a vector. Vectors for the leaf nodes correspond to learned embeddings for tokens (grey), syntactic (red) and semantic (blue) labels. Internal vectors are obtained by recursive composition of the vectors for the head, dependent, and labels for a given dependency, their colors are a mix of the colors of their descendants. Syntactic dependencies and labels from the joint parse are shown in red, semantic in blue, alongside the parts of the composition network represented.

guidelines, automatically predicted POS tags and lemmas provided in the datasets were used for all experiments. As a preprocessing step, pseudo-projectivization of the syntactic trees (Nivre et al., 2007) was used, which allowed an accurate conversion of even the non-projective syntactic trees into syntactic transitions. However, the oracle conversion of semantic parses into transitions is not perfect despite using the M-Swap action, due to the presence of multiple crossing arcs.[7]

The standard evaluation metrics include the syntactic labeled attachment score (LAS), the semantic F_1 score on both in-domain (WSJ) and out-of-domain (Brown corpus) data, and their macro average (Macro F_1) to score joint systems. Because the task was defined somewhat differently in each year, each dataset is considered in turn.

3.3.2 CoNLL 2008

The CoNLL 2008 dataset contains annotations from the Penn Treebank (Marcus et al., 1993), PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004). The shared task evaluated systems on predicate identification in addition to predicate sense disambiguation and SRL.

To identify predicates, we trained a zero-Markov order bidirectional LSTM two-class classifier. As input to the classifier, we use learned representations of word lemmas and POS tags. This model achieves an F_1 score of 91.43% on marking words as predicates (or not).

Hyperparameters The input representation for a word consists of pretrained embeddings (size 100 for English, 80 for Chinese, 64 for German and Spanish), concatenated with additional learned word and POS tag embeddings (size 32 and 12, respectively). Learned embeddings for syntactic and semantic arc labels are of size 20 and predicates 100. Two-layer LSTMs with hidden state dimension 100 were used for each of the four stacks. The parser state $\mathbf{y_t}$ and the composition function g are of dimension 100. A dropout rate of 0.2 (Zaremba et al., 2014) was used on all layers at training time, tuned on the development data from the set of values $\{0.1, 0.2, 0.3, 1.0\}$. The learned representations for actions are of size 100, similarly tuned from $\{10, 20, 30, 40, 100\}$. Other hyperparameters have been set intuitively; careful tuning is expected to yield improvements (Weiss et al., 2015).

An initial learning rate of 0.1 for stochastic gradient descent was used and updated in every training epoch with a decay rate of 0.1 (Dyer et al., 2015). Training is stopped when the development performance does not improve for approximately 6–7 hours of elapsed

[7]For 1.5% of English sentences in the CoNLL 2009 English dataset, the transition sequence incorrectly encodes the gold-standard joint parse; details in Henderson et al. (2013).

	Labeled Accuracy Score	**Semantic F_1**	**Macro F_1**
joint models:			
Lluís and Màrquez (2008)	85.8	70.3	78.1
Henderson et al. (2008)	87.6	73.1	80.5
Johansson (2009)	86.6	77.1	81.8
Titov et al. (2009)	87.5	76.1	81.8
CoNLL 2008 best:			
#3: Zhao and Kit (2008)	87.7	76.7	82.2
#2: Che et al. (2008)	86.7	78.5	82.7
#2: Ciaramita et al. (2008)	87.4	78.0	82.7
#1: Johansson and Nugues (2008)	89.3	81.6	85.5
Joint (this work)	89.1	80.5	84.9

Table 3.2: Joint parsers: comparison on the CoNLL 2008 test (WSJ+Brown) set.

time. Experiments were run on a single thread on a CPU, with memory requirements of up to 512 MB.

From Table 3.2, we see that our joint model significantly outperforms the joint model of Henderson et al. (2008), from which our set of transitions is derived, showing the benefit of learning a representation for the entire algorithmic state. Several other joint learning models have been proposed (Lluís and Màrquez, 2008; Johansson, 2009; Titov et al., 2009) for the same task; our joint model surpasses the performance of all these models. The best reported systems on the CoNLL 2008 task are due to Johansson and Nugues (2008), Che et al. (2008), Ciaramita et al. (2008) and Zhao and Kit (2008), all of which pipeline syntax and semantics; our system's semantic and overall performance is comparable to these. We fall behind only Johansson and Nugues (2008), whose success was attributed to carefully designed global SRL features integrated into a pipeline of classifiers, making them asymptotically slower.

3.3.3 CoNLL 2009

Relative to the CoNLL 2008 task (above), the main change in 2009 is that predicates are pre-identified, and systems are only evaluated on predicate sense disambiguation (not identification). Hence, the bidirectional LSTM classifier is not used here. The preprocessing for projectivity, and the hyperparameter selection is the same as in (§3.3.2).

In addition to the joint approach described in the preceding sections, we experiment here with several variants:

Semantics-only: the set of syntactic transitions $\mathcal{S}$, the syntactic stack S, and the syntactic composition function g_s are discarded. As a result, the set of constraints on transitions is a subset of the full set of constraints in (§3.1.2). Effectively, this model does not use any syntactic features, similar to Collobert et al. (2011) and Zhou and Xu (2015). It provides a controlled test of the benefit of explicit syntax in a semantic parser.

Syntax-only: all semantic transitions in $\mathcal{M}$, the semantic stack M, and the semantic composition function g_m are discarded. S-Shift and S-Right now *move* the item from the front of the buffer to the syntactic stack, instead of copying. The set of constraints on the transitions is again a subset of the full set of constraints. This model is an arc-eager variant of Dyer et al. (2015), and serves to check whether semantic parsing degrades syntactic performance.

Hybrid: the semantics parameters are trained using automatically predicted syntax from the syntax-only model. At test time, only semantic parses are predicted. This setup bears similarity to other approaches which pipeline syntax and semantics, extracting features from the syntactic parse to help SRL. However, unlike other approaches, this model does not offer the entire syntactic tree for feature extraction, since only the partial syntactic structures present on the syntactic stack (and potentially the buffer) are visible at a given timestep. This model helps show the effect of joint prediction.

CoNLL 2009 Results (English)

All of our models (Syntax-only, Semantics-only, Hybrid and Joint) improve over Gesmundo et al. (2009) and Henderson et al. (2013), demonstrating the benefit of our entire-parser-state representation learner compared to the more locally scoped model. These results are shown in Table 3.3.

Given that syntax has consistently proven useful in SRL, we expected our Semantics-only model to underperform Hybrid and Joint, and it did. In the training domain, syntax and semantics benefit each other (Joint outperforms Hybrid). Out-of-domain (the Brown test set), the Hybrid pulls ahead, a sign that Joint overfits to WSJ. As a syntactic parser, our Syntax-only model performs slightly better than Dyer et al. (2015), who achieve 89.56 LAS on this task. Joint parsing is very slightly better still.

The overall performance of Joint is on par with the other winning participants at the CoNLL 2009 shared task (Zhao et al., 2009; Che et al., 2009; Gesmundo et al., 2009), falling behind only Zhao et al. (2009), who carefully designed language-specific features and used a series of pipelines for the joint task, resulting in an accurate but computationally expensive system.

Model	**LAS**	**Semantic F_1 (WSJ)**	**Semantic F_1 (Brown)**	**Macro F_1**
CoNLL'09 best:				
#3 Gesmundo et al. (2009)	88.79	83.24	70.65	86.03
#2 Che et al. (2009)	88.48	85.51	73.82	87.00
#1 Zhao et al. (2009)	89.19	86.15	74.58	87.69
This work:				
Syntax-only	89.83	-	-	-
Sem.-only	-	84.39	73.87	-
Hybrid	89.83	84.58	75.64	87.20
Joint	89.94	84.97	74.48	87.45
Pipelines:				
Roth and Woodsend (2014)	-	86.34	75.90	-
Lei et al. (2015)	-	86.58	75.57	-
Täckström et al. (2015)	-	87.30	75.50	-
FitzGerald et al. (2015)	-	87.80	75.50	-

Table 3.3: Comparison on the CoNLL 2009 English test set. The first block presents results of other models evaluated for both syntax and semantics on the CoNLL 2009 task. The second block presents our models. The third block presents the best published models, each using its own syntactic preprocessing.

State-of-the-art SRL systems (shown in the last block of Table 3.3) which use advances orthogonal to the contributions in this paper, perform better than our models. Many of these systems use expert-crafted features derived from full syntactic parses in a pipeline of classifiers followed by a global reranker (Björkelund et al., 2009; Björkelund et al., 2010; Roth and Woodsend, 2014); we have not used these features or reranking. Lei et al. (2015) use syntactic parses to obtain interaction features between predicates and their arguments and then compress feature representations using a low-rank tensor. Täckström et al. (2015) present an exact inference algorithm for SRL based on dynamic programming and their local and structured models make use of many syntactic features from a pipeline; our search procedure is greedy. Their algorithm is adopted by FitzGerald et al. (2015) for inference in a model that jointly learns representations from a combination of PropBank and FrameNet annotations; we have not experimented with extra annotations.

Our system achieves an end-to-end runtime of 177.6 ± 18 seconds to parse the CoNLL 2009 English test set on a single core. This is almost 2.5 times faster than the pipeline model of Lei et al. (2015) (439.9 ± 42 seconds) on the same machine.[8]

[8]See `https://github.com/taolei87/SRLParser`; unlike other state-of-the-art systems, this one is publicly available.

Language	Che et al. (2009) #1	Zhao et al. (2009) #2	Gesmundo et al. (2009) #3	Joint Ours
Catalan	81.84	**83.01**	82.66	82.40
Chinese	76.38	76.23	76.15	**79.27**
Czech	**83.27**	80.87	83.21	79.53
English	87.00	**87.69**	86.03	87.45
German	**82.44**	81.22	79.59	81.05
Japanese	**85.65**	85.28	84.91	80.91
Spanish	81.90	**83.31**	82.43	83.11
Average	82.64	82.52	82.14	81.96

Table 3.4: Comparison of macro F_1 scores on the multilingual CoNLL 2009 test set.

CoNLL 2009 (Multilingual)

We tested the joint model on the non-English CoNLL 2009 datasets, and the results demonstrate that it adapts easily—it is on par with the top three systems in most cases (Table 3.4). We note that our Chinese parser relies on pretrained word embeddings for its superior performance; without them (not shown), it was on par with the others. Japanese is a small-data case (4,393 training examples), illustrating our model's dependence on reasonably large training datasets.

We have not extended our model to incorporate morphological features, which are used by the systems to which we compare. Future work might incorporate morphological features where available; this could potentially improve performance, especially in highly inflective languages like Czech. An alternative might be to infer word-internal representations using character-based word embeddings, which was found beneficial for syntactic parsing (Ballesteros et al., 2015).

Other approaches to joint modeling, not considered in our experiments, are notable. Lluís et al. (2013) propose a graph-based joint model using dual decomposition for agreement between syntax and semantics, but do not achieve competitive performance on the CoNLL 2009 task. Lewis et al. (2015) proposed an efficient joint model for CCG syntax and SRL, which performs better than a pipelined model. However, their training necessitates CCG annotation; ours does not. Moreover, their evaluation metric rewards semantic dependencies regardless of where they attach within the argument span given by a PropBank constituent, making direct comparison to our evaluation infeasible. Krishnamurthy and Mitchell (2014) propose a joint CCG parsing and relation extraction model which improves over pipelines, but their task is different from ours. Li et al. (2010) also perform joint syntactic and semantic dependency parsing for Chinese, but do not report results on the CoNLL 2009 dataset.

There has also been an increased interest in models which use neural networks for

SRL. Collobert et al. (2011) proposed models which perform many NLP tasks without hand-crafted features. Though they did not achieve the best results on the constituent-based SRL task (Carreras and Màrquez, 2005), their approach inspired Zhou and Xu (2015), who achieved state-of-the-art results using deep bidirectional LSTMs. Our approach for dependency-based SRL is not directly comparable.

Our system's performance does not match that of the top expert-crafted feature-based systems (Zhao et al., 2009; Björkelund et al., 2010; Roth and Woodsend, 2014; Lei et al., 2015), systems which perform optimal decoding (Täckström et al., 2015), or of systems that exploit additional, differently-annotated datasets (FitzGerald et al., 2015). Many advances in those systems are orthogonal to our model, and we expect future work to achieve further gains by integrating them.

Because our system is very fast— with an end-to-end runtime of 177.6 ± 18 seconds to parse the CoNLL 2009 English test data on a single core—we believe it will be useful in practical settings. Our open-source implementation is available at `https://github.com/clab/joint-lstm-parser`.

3.4 Conclusion

We presented an incremental, greedy parser for joint syntactic and semantic dependency parsing. Our model surpasses the performance of previous joint models on the CoNLL 2008 and 2009 English tasks, without using expert-crafted, expensive features of the full syntactic parse.

Chapter 4

Multi-Task Training with Syntactic Substructures

Inspired by the usefulness of syntactic biases in learning semantic representations in Chapter 3, we will further explore this idea in the current chapter. Many semantic tasks involve labeling spans, including semantic role labeling (SRL; Gildea and Jurafsky, 2002) and coreference resolution (Ng, 2010) (tasks we consider in this chapter), as well as named entity recognition and some reading comprehension and question answering tasks (Rajpurkar et al., 2016). Here, we consider three different semantic tasks, each involving classification of spans of text as semantic arguments. The first task is frame-semantic role labeling, which involves labeling spans in a sentence as arguments to semantic frames from FrameNet (Baker et al., 1998; Fillmore, 1976). The second is PropBank span-based semantic role labeling, which is similar to the first task, but contains coarser-grained labels for arguments.[1] The final task is coreference resolution, which involves identifying spans in a document as named entities and chaining together coreferent entity spans. See Figure 4.1 for an example sentence with these three semantic annotations.

Also shown in Figure 4.1 is a phrase-syntactic parse. The key observation to be made here is that semantic argument spans for all three tasks almost always correspond to valid syntactic constituents (cf. PropBank; Palmer et al., 2005), making phrase-based syntax a natural choice for incorporating a syntactic bias. Classical approaches to span-based semantics employed syntactic span information as a feature (Xue and Palmer, 2004).

Our approach preserves most of the benefit of syntactic features without the accompanying computational cost of feature extraction or syntactic parsing. We incorporate the training objective of our syntax-free model into a multitask setting where the sec-

[1]There are many more differences, which we will highlight in §4.4.

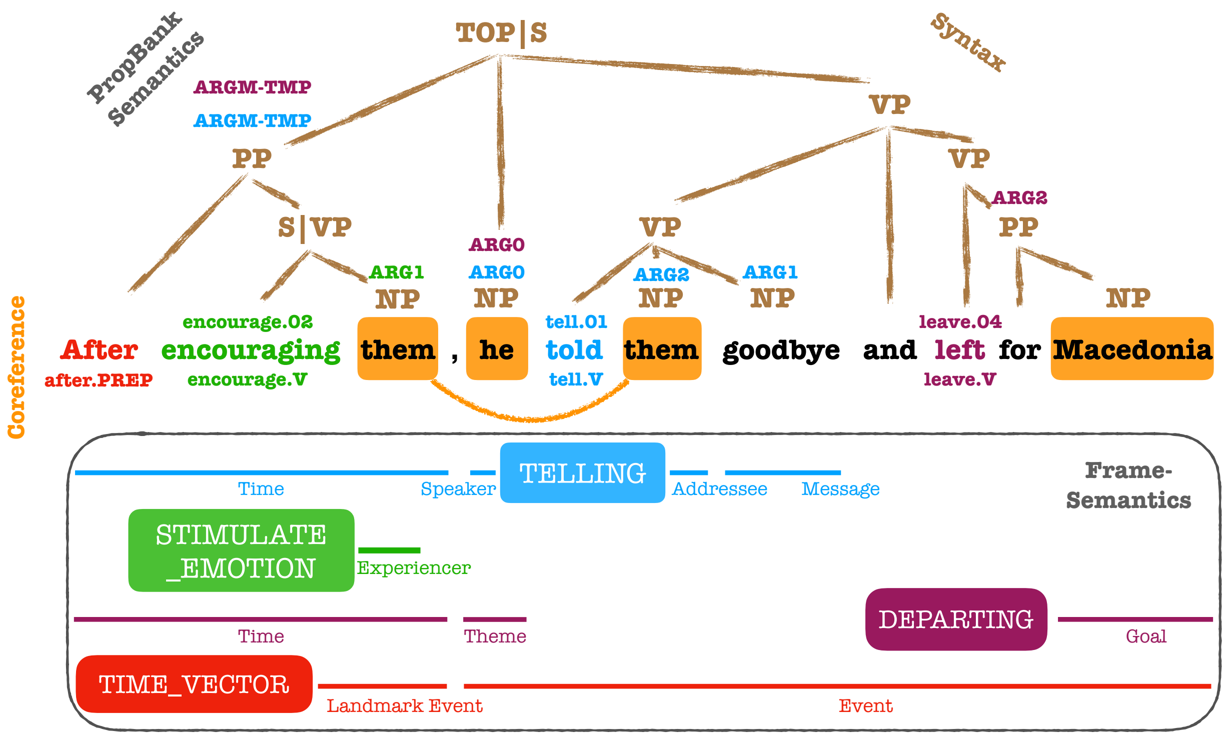


Figure 4.1: An example sentence with syntactic, PropBank and coreference annotations from OntoNotes, and author-annotated frame-semantic structures. PropBank SRL arguments and coreference mentions are annotated on top of syntactic constituents. All but one frame-semantic argument (Event) is a syntactic constituent. Targets evoke frames shown in the color-coded layers. FrameNet additionally annotates prepositional frames, such as one evoked by *"After"*.

ond task is unlabeled constituent identification (i.e., a separate binary decision for each span). This *syntactic scaffold*[2] task offers useful guidance to the frame-semantic model, leading to performance on par with our models that use syntactic features.

Training Objective We address the central question: is there a way for semantic analyzers to benefit from syntax without the computational cost of syntactic parsing? We propose a multitask learning approach to incorporate syntactic information into learned representations of neural semantics models (§4.1). Our approach, the **syntactic scaffold**, minimizes an auxiliary supervised loss function, derived from a syntactic treebank. We avoid the cost of training or executing a full syntactic parser, and at test time (i.e., runtime in applications) the semantic analyzer has no additional cost over a syntax-free baseline. Further, the method does not assume that the syntactic treebank overlaps the dataset for the primary task, which is in contrast to Chapter 3, and is a more practical setting for most tasks.

Syntactic Inductive Bias The goal is to steer the distributed, contextualized representations of words and spans toward accurate semantic *and* syntactic labeling. This is enabled by sharing representations of spans between the primary and the scaffolding tasks.

Since the scaffold task is not an end in itself, we relax the syntactic parsing problem to a collection of independent span-level predictions, with no constraint that they form a valid parse tree. This means we never need to run a syntactic parsing algorithm.

We propose strong syntax-free baselines, which were themselves state-of-the-art, using the strongest available neural network architectures for these tasks, integrating deep representation learning (He et al., 2017) and structured prediction at the level of spans (Kong et al., 2016). For SRL, the syntax-free baseline itself is a novel globally normalized structured conditional random field, which outperforms the previous state of the art.[3] Even more remarkably, annotations for syntax and semantics were not required on the same data, making this approach generalizable to other tasks. Our experiments demonstrate that the syntactic scaffold offers a substantial boost to state-of-the-art baselines for two SRL tasks (§4.4) and coreference resolution (§4.5). Syntactic scaffolds result in further improvements over prior work—3.6 absolute F_1 in FrameNet SRL, 1.1 absolute F_1 in PropBank SRL, and 0.6 F_1 in coreference resolution (averaged across three standard scores). Even though the tasks considered here concern span-based semantics, the methods discussed are general enough for any span-based task, which is poised to benefit from a different span-based task.

[2]We borrow the term *scaffolding* from developmental psychology (Wood et al., 1976) to describe a support task during learning that is eventually discarded.

[3]This holds regardless of whether the model is initialized with deep, contextualized embeddings (Peters et al., 2018a), an approach orthogonal to ours.

This work was presented in Swayamdipta et al. (2017) and (Swayamdipta et al., 2018b). Our code is open source and available at `https://github.com/swabhs/scaffolding`.

4.1 Syntactic Scaffolds

Multitask learning (Caruana, 1997) is a collection of techniques in which two or more tasks are learned from data with at least some parameters shared. A range of examples have recently been explored in NLP, showing in particular how neural architectures achieve better performance when learned for multiple tasks together (Collobert et al., 2011; Luong et al., 2015; FitzGerald et al., 2015; Chen et al., 2017; Peng et al., 2017; Hashimoto et al., 2017).

We assume there is only one task about whose performance we are concerned, denoted T_1 (in this chapter, T_1 is either frame-semantic role labeling or coreference resolution). We use the term "scaffold" to refer to a second task, T_2, that can be combined with T_1 during multitask learning. A scaffold task is *only* used during training; it holds no intrinsic interest beyond biasing the learning of T_1, and after learning is completed, the scaffold is discarded.

A **syntactic scaffold** is a task designed to steer the (shared) model toward awareness of syntactic structure. It could be defined through a parsing model that shares some parameters with T_1's model. Since syntactic parsing is costly, we use simpler syntactic prediction problems (discussed below) that do not produce whole trees.

As with multitask learning in general, we do not assume that the same data are annotated with outputs for T_1 and T_2. In this work, T_2 is defined using phrase-structure syntactic annotations from OntoNotes 5.0 (Weischedel et al., 2013; Pradhan et al., 2013). We experiment with two settings: one where the corpus for T_2 does not overlap with the training datasets for T_1 (frame-SRL) and the second where there is a complete overlap (coreference). This is a major advantage of our approach: it does not require any assumptions about or specification of the relationship between T_1 and T_2 output.

4.2 Related Work

We briefly contrast the syntactic scaffold with existing alternatives. For a more detailed discussion, see Chapter 2.

Pipelines. In a typical pipeline, T_1 and T_2 are separately trained, with the output of T_2 used to define the inputs to T_1 (Wolpert, 1992). Using syntax as T_2 in a pipeline is perhaps the most common approach for semantic structure prediction (Toutanova et al., 2008; Yang and Mitchell, 2017; Wiseman et al., 2016).[4] However, pipelines introduce the problem of cascading errors (T_2's mistakes affect the performance, and perhaps the training, of T_1; He et al., 2013). To date, remedies to cascading errors are so computationally expensive as to be impractical (e.g., Finkel et al., 2006). A syntactic scaffold is quite different from a pipeline since the output of T_2 is never explicitly used.

Latent variables. Another solution is to treat the output of T_2 as a (perhaps structured) latent variable. This approach obviates the need of supervision for T_2 and requires marginalization (or some approximation to it) in order to reason about the outputs of T_1. Syntax as a latent variable for semantics was explored by Zettlemoyer and Collins (2005) and Naradowsky et al. (2012). Apart from avoiding marginalization, the syntactic scaffold offers a way to use auxiliary syntactically-annotated data as direct supervision for T_2, and it need not overlap the T_1 training data.

Joint learning of syntax and semantics. The motivation behind joint learning of syntactic and semantic representations is that any one task is helpful in predicting the other (Lluís and Màrquez, 2008; Lluís et al., 2013; Henderson et al., 2013; Swayamdipta et al., 2016). This typically requires joint prediction of the outputs of T_1 and T_2, which tends to be computationally expensive at both training and test time.

Part of speech scaffolds. Similar to our work, there have been multitask models that use part-of-speech tagging as T_2, with transition-based dependency parsing (Zhang and Weiss, 2016) and CCG supertagging (Søgaard and Goldberg, 2016) as T_1. Both of the above approaches assumed parallel input data and used both tasks as supervision. Notably, we simplify our T_2, throwing away the structured aspects of syntactic parsing, whereas part-of-speech tagging has very little structure to begin with. While their approach results in improved token-level representations learned via supervision from POS tags, these must still be composed to obtain span representations. Instead, our approach learns span-level representations from phrase-type supervision directly, for semantic tasks. Additionally, these methods explore architectural variations in RNN layers for including supervision, whereas we focus on incorporating supervision with minimal changes to the baseline architecture. To the best of our knowledge, such simplified syntactic scaffolds have not been tried before.

Word embeddings. Our definition of a scaffold task *almost* includes stand-alone methods for estimating word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Pe-

[4]There has been some recent work on SRL which completely forgoes syntactic processing (Zhou and Xu, 2015), however it has been shown that incorporating syntactic information still remains useful (He et al., 2017).

ters et al., 2018a). After training word embeddings, the tasks implied by models like the skip-gram or ELMo's language model become irrelevant to the downstream use of the embeddings. A noteworthy difference is that, rather than pre-training, a scaffold is integrated directly into the training of T_1 through a multitask objective.

Multitask learning. Neural architectures have often yielded performance gains when trained for multiple tasks together (Collobert et al., 2011; Luong et al., 2015; Chen et al., 2017; Hashimoto et al., 2017). In particular, performance of semantic role labeling tasks improves when done jointly with other semantic tasks (FitzGerald et al., 2015; Peng et al., 2017, 2018a). Contemporaneously with this work, Hershcovich et al. (2018) proposed a multitask learning setting for universal syntactic dependencies and UCCA semantics (Abend and Rappoport, 2013). Syntactic scaffolds focus on a primary semantic task, treating syntax as an auxillary, eventually forgettable prediction task.

4.3 Syntactic Scaffold Model

We assume two sources of supervision: a corpus $\mathcal{D}_{\text{pr}}$ with instances $\boldsymbol{x}$ annotated for the primary task's outputs $\boldsymbol{y}$ (semantic role labeling or coreference resolution), and a treebank $\mathcal{D}_{\text{sc}}$ with sentences $\boldsymbol{x}$, each with a phrase-structure tree $\boldsymbol{z}$. Each task has an associated loss, and we seek to minimize the interpolation of task losses,

$$\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_{\text{pr}}} \mathcal{L}_{\text{pr}}(\boldsymbol{x},\boldsymbol{y}) + \delta \sum_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{D}_{\text{sc}}} \mathcal{L}_{\text{sc}}(\boldsymbol{x},\boldsymbol{z})$$

with respect to parameters, which are partially shared, where δ is a tunable hyperparameter.

In the rest of this section, we describe the scaffold task. The primary tasks are defined in Sections 4.4 and 4.5.

Each input is a sequence of tokens, $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, for some n. We refer to a *span* of contiguous tokens in the sentence as $\boldsymbol{x}_{i:j} = \langle x_i, x_{i+1}, \ldots, x_j \rangle$, for any $1 \leqslant i \leqslant j \leqslant n$. In our experiments we consider only spans up to a maximum length D, resulting in $O(nD)$ spans.

Supervision comes from a phrase-syntactic tree $\boldsymbol{z}$ for the sentence, comprising a syntactic category $z_{i:j} \in \mathcal{C}$ for every span $\boldsymbol{x}_{i:j}$ in $\boldsymbol{x}$ (many spans are given a NULL label). We experiment with different sets of labels $\mathcal{C}$ (§4.3.1).

In our model, every span $\boldsymbol{x}_{i:j}$ is represented by an embedding vector $\boldsymbol{v}_{i:j}$ (see details in §4.4.3). A distribution over the category assigned to $z_{i:j}$ is derived from $\boldsymbol{v}_{i:j}$:

$$p(z_{i:j} = c \mid \boldsymbol{x}_{i:j}) = \operatorname*{softmax}_{c} \boldsymbol{w}_c \cdot \boldsymbol{v}_{i:j}$$

where $\boldsymbol{w}_c$ is a parameter vector associated with category c. We sum the log loss terms for all the spans in a sentence to give the loss for the sentence:

$$\mathcal{L}_{\text{sc}}(\boldsymbol{x}, \boldsymbol{z}) = - \sum_{\substack{1 \leqslant i \leqslant j \leqslant n \\ j-i \leqslant D}} \log p(z_{i:j} \mid \boldsymbol{x}_{i:j}).$$

4.3.1 Labels for the Syntactic Scaffold Task

Different kinds of syntactic labels can be used for learning syntactically-aware span representations:

- **Constituent identity**: $\mathcal{C} = \{0, 1\}$; is a span a constituent, or not?
- **Non-terminal**: c is the phrase type of a span, including a NULL for non-constituents.
- **Non-terminal and parent**: c is the phrase type of a span, composed with the phrase type of its immediate ancestor. NULL is used for non-constituents, and for empty ancestors.
- **Common non-terminals**: Since a majority of semantic arguments and entity mentions are labeled with a small number of categories,[5] we experiment with a three-way classification among (i) noun phrase (or prepositional phrase, for frame SRL); (ii) any other category; and (iii) NULL.

In Figure 4.1, for the span "*encouraging them*", the constituent identity scaffold label is 1, the non-terminal label is `VP|S`, the non-terminal and parent label is `VP|S+par=PP`, and the common non-terminals label is set to `OTHER`.

4.4 Semantic Role Labeling

We contribute a new SRL model which contributes a strong baseline for experiments with syntactic scaffolds. The performance of this baseline itself is competitive with state-of-the-art methods (§5.4).

FrameNet. In the FrameNet lexicon (Baker et al., 1998), a *frame* represents a type of event, situation, or relationship, and is associated with a set of semantic roles, or argu-

[5] In the Ontonotes corpus, which includes both syntactic and semantic annotations, 44% of semantic arguments are noun phrases and 13% are prepositional phrases.

ments, called *frame elements*. A frame can be evoked by a word or phrase in a sentence, called a *target*. Each frame element of an evoked frame can then be realized in the sentence as a sentential span, called an *argument* (or it can be unrealized). Arguments for a given frame do not overlap. Frame elements can be classified as either core to the meaning of the frame or not.

PropBank. PropBank similarly disambiguates predicates and identifies argument spans. Targets are disambiguated to lexically specific *senses* rather than shared frames, and a set of generic roles is used for all targets, reducing the argument label space by a factor of 17. Most importantly, the arguments were annotated on top of syntactic constituents, directly coupling syntax and semantics. A detailed example for both formalisms is provided in Figure 4.1.

Semantic structure prediction is the task of identifying targets, labeling their frames or senses, and labeling all their argument spans in a sentence. Here we assume gold targets and frames, and consider only the SRL task.

Formally, a single input instance for argument identification consists of: an n-word sentence $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, a single target span $t = \langle t_{\text{start}}, t_{\text{end}} \rangle$, and its evoked frame, or sense, f. The argument labeling task is to produce a *segmentation* of the sentence: $\boldsymbol{s} = \langle s_1, s_2, \ldots, s_m \rangle$ for each input $\boldsymbol{x}$. A *segment* $s = \langle i, j, y_{i:j} \rangle$ corresponds to a labeled span of the sentence, where the label $y_{i:j} \in \mathcal{Y}_f \cup \{\textsc{null}\}$ is either a role that the span fills, or NULL if the span does not fill any role. In the case of PropBank, $\mathcal{Y}_f$ consists of all possible roles. The segmentation is constrained so that argument spans cover the sentence and do not overlap ($i_{k+1} = 1 + j_k$ for s_k; $i_1 = 1$; $j_m = n$). Segments of length 1 such that $i = j$ are allowed. A separate segmentation is predicted for each target annotation in a sentence.

4.4.1 Semi-CRF Model

In order to model the non-overlapping arguments of a given frame, we use a semi-Markov conditional random field (*semi-CRF*; Sarawagi et al., 2004). Semi-CRFs define a conditional distribution over labeled segmentations of an input sequence, and are globally normalized. A single frame's arguments can be neatly encoded as a labeled segmentation by giving the spans in between arguments a reserved NULL label. Semi-Markov models are more powerful than BIO tagging schemes, which have been used successfully for PropBank SRL (Collobert et al., 2011; Zhou and Xu, 2015, *inter alia*). The semi-Markov assumption allows scoring variable-length segments, rather than fixed-length label n-grams as under an $(n-1)$-order Markov assumption, while retaining exact inference and a linear runtime. [6]

Computing the marginal likelihood with a semi-CRF can be done using dynamic programming in $O(n^2)$ time (§4.4.2). By filtering out segments longer than D tokens, this can be reduced to $O(nD)$, and we use $D = 15$, pruning less than 5% of gold-standard arguments from the train and development data.

Given an input $\boldsymbol{x}$, a semi-CRF defines a conditional distribution $p(\boldsymbol{s} \mid \boldsymbol{x})$. Every segment $s = \langle i, j, y_{i:j} \rangle$ is given a real-valued score,

$$\psi(\langle i, j, y_{i:j} = r \rangle, \boldsymbol{x}_{i:j}) = \boldsymbol{w}_r \cdot \boldsymbol{v}_{i:j}.$$

where $\boldsymbol{v}_{i:j}$ is an embedding of the span (§4.4.3) and $\boldsymbol{w}_r$ is a parameter corresponding to its label. The score of the entire segmentation $\boldsymbol{s}$ is the sum of the scores of its segments:

$$\Psi(\boldsymbol{x}, \boldsymbol{s}) = \sum_{k=1}^{m} \psi(s_k, \boldsymbol{x}_{i_k:j_k}).$$

These scores are exponentiated and normalized to define the probability distribution. The sum-product variant of the semi-Markov dynamic programming algorithm is used to calculate the normalization term (required during learning). At test time, the max-product variant returns the most probable segmentation,

$$\hat{\boldsymbol{s}} = \arg\max_{\boldsymbol{s}} \Psi(\boldsymbol{s}, \boldsymbol{x}).$$

The parameters of the semi-CRF are learned to maximize a criterion related to the conditional log-likelihood of the gold-standard segments in the training corpus (§4.4.2). The learner evaluates and adjusts segment scores $\psi(s_k, \boldsymbol{x})$ for every span in the sentence, which in turn involves learning embedded representations for all spans (§4.4.3). The gradients of the objective with respect to the parameters could be computed using the forward-backward algorithm, but we rely on automatic differentiation of the dynamic program.

4.4.2 Softmax-Margin Objective

Typically CRF and semi-CRF models are trained to maximize a conditional log-likelihood objective. In early experiments, we found that incorporating a structured cost was beneficial; we do so by using a softmax-margin training objective (Gimpel and Smith, 2010),

[6]Relatedly, Täckström et al. (2015) introduced a dynamic program that allows direct modeling of variable-length segments as well as enforcing constraints such as certain roles being filled at most once. Its runtime is linear in the sentence length, but exponential in the number of roles. The semi-Markov CRF's inference algorithm is a relaxed special case of their method, with fewer constraints and without the exponential runtime constant.

a "cost-aware" variant of log-likelihood:

$$\mathcal{L}_{\text{pr}} = -\sum_{(\boldsymbol{x},\boldsymbol{s}^*)\in\mathcal{D}_{\text{pr}}} \log \frac{\exp \Psi(\boldsymbol{s}^*, \boldsymbol{x})}{Z(\boldsymbol{x})},$$

$$Z(\boldsymbol{x}, \boldsymbol{s}^*) = \sum_{\boldsymbol{s}} \exp\left\{\Psi(\boldsymbol{s}, \boldsymbol{x}) + \text{cost}(\boldsymbol{s}, \boldsymbol{s}^*)\right\}.$$

We design the cost function so that it factors by predicted span, in the same way Ψ does:

$$\text{cost}(\boldsymbol{s}, \boldsymbol{s}^*) = \sum_{s\in\boldsymbol{s}} \text{cost}(s, \boldsymbol{s}^*) = \sum_{s\in\boldsymbol{s}} \mathbb{I}(s \notin \boldsymbol{s}^*). \tag{4.1}$$

The softmax-margin criterion, like log-likelihood, is globally normalized over all of the exponentially many possible labeled segmentations. The following zeroth-order semi-Markov dynamic program (Sarawagi et al., 2004) efficiently computes the new partition function:

$$\alpha_j = \sum_{\substack{s=\langle i,j,y_{i:j}\rangle \\ j-i\leqslant D}} \alpha_{i-1} \exp\{\Psi(s, \boldsymbol{x}) + \text{cost}(s, \boldsymbol{s}^*)\},$$

where $Z = \alpha_n$, under the base case $\alpha_0 = 1$. Overall, the run time complexity is given by $O(nDL)$ where n is the length of the sentence, D is the maximum argument width considered and L the maximum number of labels for arguments.

The prediction under the model can be calculated using a similar dynamic program with the following recurrence where $\gamma_0 = 1$:

$$\gamma_j = \max_{\substack{s=\langle i,j,y_{i:j}\rangle \\ j-i\leqslant D}} \gamma_{i-1} \exp \Psi(s, \boldsymbol{x}).$$

The decoding problem uses the Viterbi algorithm which replaces the sums in the marginalization algorithm in Equation 5.1 with maximization operators and therefore runs in the same time. At test time, since the cost is unknown (since the gold standard is not known), we do not incorporate it in the maximization function.

Our model formulation enforces that arguments do not overlap. We do not enforce any other SRL constraints, such as non-repetition of core frame elements (Das et al., 2012).

4.4.3 Input Span Representation

This section describes the neural architecture used to obtain the span embedding, $\boldsymbol{v}_{i:j}$ corresponding to a span $\boldsymbol{x}_{i:j}$ and the target in consideration, $t = \langle t_{\text{start}}, t_{\text{end}}\rangle$. For the scaffold task, since the syntactic treebank does not contain annotations for semantic targets, we use the last verb in the sentence as a placeholder target, wherever target features are used. If there are no verbs, we simply use the first token in the sentence as a placeholder target. The parameters used to learn $\boldsymbol{v}$ are shared between the tasks.

First, we construct an embedding for the span using

- $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$: contextualized embeddings for the words at the span boundary (§4.4.3),
- $\boldsymbol{u}_{i:j}$: a span summary that pools over the contents of the span (§4.4.3), and
- $\boldsymbol{a}_{i:j}$: and a hand-engineered feature vector for the span (§4.4.3).

This embedding is then fed into a feed-forward layer to compute the span representation, $\boldsymbol{v}_{i:j}$.

Contextualized Token Embeddings

To obtain contextualized embeddings of each token in the input sequence, we run a bidirectional LSTM (Graves, 2012) with ℓ layers over the full input sequence. To indicate which token is a predicate, a linearly transformed one-hot embedding, $\boldsymbol{v}$ is used, following He et al. (2017). The input vector representing the token at position q in the sentence is the concatenation of a fixed pretrained embedding $\mathbf{x}_q$ and $\boldsymbol{v}_q$. When given as input to the bidirectional LSTM, this yields a hidden state vector $\boldsymbol{h}_q$ representing the qth token in the context of the sentence.

Span Summary

Tokens within a span might convey different amount of information necessary to label the span as a semantic argument. Following Lee et al. (2017), we use an attention mechanism (Bahdanau et al., 2014) to summarize each span. Each contextualized token in the span is passed through a feed-forward network to obtain a weight, normalized to give

$$\sigma_k = \operatorname*{softmax}_{i \leqslant k \leqslant j} \boldsymbol{w}^{\text{head}} \cdot \boldsymbol{h}_k,$$

where $\boldsymbol{w}^{\text{head}}$ is a learned parameter. The weights σ are then used to obtain a vector that summarizes the span,

$$\boldsymbol{u}_{i:j} = \textstyle\sum_{i \leqslant k \leqslant j; j-i<D} \sigma_k \cdot \boldsymbol{h}_k.$$

Span Features

Lastly we use the following three features for each span:

- width of the span in tokens (Das et al., 2014)
- distance (in tokens) of the span from the target (Täckström et al., 2015)
- position of the span with respect to the target (*before*, *after*, *overlap*) (Täckström et al., 2015)

Each of these features is encoded as a one-hot-embedding and then linearly transformed to yield a feature vector, $\boldsymbol{a}_{i:j}$.

4.5 Coreference Resolution

Coreference resolution is the task of determining clusters of mentions that refer to the same entity. Formally, the input is a document $\boldsymbol{x} = \langle x_1, x_2, \dots, x_n \rangle$ consisting of n words. The goal is to predict a set of clusters $\boldsymbol{c} = \langle c_1, c_2, \dots \rangle$, where each cluster $c = \langle s_1, s_2, \dots \rangle$ is a set of spans and each span $s = \langle i, j \rangle$ is a pair of indices such that $1 \leqslant i \leqslant j \leqslant n$.

As a baseline, we use the model of Lee et al. (2017), which we describe briefly in this section. This model decomposes the prediction of coreference clusters into a series of span classification decisions. Every span s predicts an antecedent $w_s \in \mathcal{Y}(s) = \{\textsc{null}, s_1, s_2, \dots s_m\}$. Labels s_1 to s_m indicate a coreference link between s and one of the m spans that precede it, and NULL indicates that s does not link to anything, either because it is not a mention or it is in a singleton cluster. The predicted clustering of the spans can be recovered by aggregating the predicted links.

Analogous to the frame-SRL model (§4.4), every span s is represented by an embedding $\boldsymbol{v}_s$, which is central to the model. For each span s and a potential antecedent $a \in \mathcal{Y}(s)$, pairwise coreference scores $\Psi(\boldsymbol{v}_s, \boldsymbol{v}_a, \phi(s, a))$ are computed via feedforward networks with the span embeddings as input. $\phi(s, a)$ are pairwise discrete features encoding the distance between span s and span a and metadata, such as the genre and speaker information. We refer the reader to Lee et al. (2017) for the details of the scoring function, since the focus of this work is on improving the span embeddings.

The scores from Ψ are normalized over the possible antecedents $\mathcal{Y}(s)$ of each span to induce a probability distribution for every span:

$$p(w_s = a) = \frac{\exp(\Psi(\boldsymbol{v}_s, \boldsymbol{v}_a, \phi(s, a)))}{\sum_{a' \in \mathcal{Y}(s)} \exp(\Psi(\boldsymbol{v}_s, \boldsymbol{v}_{a'}, \phi(s, a')))}$$

Learning involves minimizing the negative log-likelihood marginalized over the possibly correct antecedents:

$$\mathcal{L} = -\sum_{s \in \mathcal{D}} \log \sum_{a^* \in G(s) \cap \mathcal{Y}(s)} p(w_s = a^*)$$

where $\mathcal{D}$ is the set of spans in the training dataset, and $G(s)$ indicates the gold cluster of s if it belongs to one and $\{\text{NULL}\}$ otherwise.

To operate under reasonable computational requirements, inference under this model requires a two-stage beam search, which reduces the number of span pairs considered. We refer the reader to Lee et al. (2017) for details.

Input span representation. The input span embedding, $\boldsymbol{v}_s$ for coreference resolution and its syntactic scaffold follow the definition used in §4.4.3, with the key difference of using no target features. Since there is a complete overlap of input sentences between $\mathcal{D}_{\text{sc}}$ and $\mathcal{D}_{\text{pr}}$ as the coreference annotations are also from OntoNotes (Pradhan et al.), we reuse the $\boldsymbol{v}$ for the scaffold task. Additionally, instead of the entire document, each sentence in it is independently given as input to the bidirectional LSTMs.

4.6 Experiments and Results

We evaluate our models on the test set of FrameNet 1.5 for frame SRL and on the test set of OntoNotes for both PropBank SRL and coreference. For the syntactic scaffold in each case, we use syntactic annotations from OntoNotes 5.0 (Weischedel et al., 2013; Pradhan et al., 2013).[7]

Experimental Settings. We used GloVe embeddings (Pennington et al., 2014) for tokens in the vocabulary, with out of vocabulary words being initialized randomly. For frame-SRL, 300 dimensional embeddings were used, and kept fixed during training. For PropBank SRL, we used 100 dimensional embeddings which were updated during training. A 100-dimensional embedding is learned for indicating target positions, following Zhou and Xu (2015). Bidirectional LSTMs with highway connections (Srivastava et al., 2014) between 6 layers are used, each layer containing 300-dimensional hidden states. A dropout of 0.1 is applied to the LSTMs. The feed-forward networks are of dimension 150 and of depth 2, with rectified linear units (Nair and Hinton, 2010). A dropout of 0.2 is applied to the feed-forward networks.

We limit the maximum length of spans to $D = 15$ in FrameNet, resulting in oracle recall of 95% on the development set, and to 13 in Propbank, resulting in an oracle recall of 96%. An identical maximum span length is used for the scaffold task.

For the SRL scaffolds, we randomly sample instances from OntoNotes to match the size of the SRL data, and alternate between training an SRL batch and a scaffold batch. In

[7]`http://cemantix.org/data/ontonotes.html`

FrameNet, this amounts to downsampling OntoNotes. For PropBank SRL, this amounts to upsampling syntactic annotations from OntoNotes, since a sentence has a single syntactic tree, but could have multiple target annotations, each of which is a training instance.

The mixing ratio, δ is set to 1.0 (tuned across $\{0.1, 0.5, 1.0, 1.5\}$) for frame and PropBank SRL. We use Adam (Kingma and Ba, 2014) for optimization, at a learning rate of 0.001, and a minibatch of size 32. Our dynamic program formulation for loss computation and inference under the semi-CRF is also batched. To prevent exploding gradients, the 2-norm of the gradient is clipped to 1 before a gradient update (Graves, 2013). All models are trained for a maximum of 20 epochs, and stopped early based on dev F_1.

For the coreference model, we use the same hyperparameters and experimental settings from Lee et al. (2017). The only new hyperparameter needed for scaffolding is the mixing ratio, δ, which we set to 0.1 based on performance on the validation set.

We extended the `AllenNLP` library,[8] which is built on top of PyTorch.[9] Each experiment was run on a single TitanX GPU.

4.6.1 Frame SRL

Table 4.1 shows the performance of all the scaffold models on frame SRL with respect to prior work and a semi-CRF baseline (§4.4.1) without a syntactic scaffold. We follow the official evaluation from the SemEval shared task for frame-semantic parsing (Baker et al., 2007).

Prior work for frame SRL has relied on predicted syntactic trees, in two different ways: by using syntax-based rules to prune out spans of text that are unlikely to contain any frame's argument; and by using syntactic features in their statistical model (Das et al., 2014; Täckström et al., 2015; FitzGerald et al., 2015; Kshirsagar et al., 2015). Syntactic parsing comes at a computational cost, though, and syntactic filters are known to be too strict. Indeed, Täckström et al. (2015) found that filtering heuristics based on predicted dependencies bounded recall below 72.6%.

The best published results on FrameNet 1.5 are due to Yang and Mitchell (2017). In their sequential model (SEQ), they treat argument identification as a sequence-labeling problem using a deep bidirectional LSTM with a CRF layer. In their relational model (REL), they treat the same problem as a span classification problem. Finally, they introduce an ensemble to integerate both models, and use an integer linear program for inference satisfying SRL constraints. Though their model does not do any syntactic pruning,

[8]http://allennlp.org/
[9]http://pytorch.org/

it does use syntactic features for argument identification and labeling.[10]

Notably, all prior systems for frame SRL listed in Table 4.1 use a pipeline of syntax and semantics. Our semi-CRF baseline outperforms all prior work, without any syntax. This highlights the benefits of modeling spans and of global normalization.

Turning to scaffolds, even the most coarse-grained constituent identity scaffold improves the performance of our syntax-agnostic baseline. The nonterminal and nonterminal and parent scaffolds, which use more detailed syntactic representations, improve over this. The greatest improvements come from the scaffold model predicting common nonterminal labels (`NP` and `PP`, which are the most common syntactic categories of semantic arguments, vs. others): 3.6% absolute improvement in F_1 measure over prior work.

Contemporaneously with this work, Peng et al. (2018a) proposed a system for joint frame-semantic and semantic dependency parsing. They report results for joint frame and argument identification, and hence cannot be directly compared in Table 4.1. We evaluated their output for argument identification only; our semi-CRF baseline model exceeds their performance by 1 F_1, and our common nonterminal scaffold by 3.1 F_1.[11]

Model	**Prec.**	**Rec.**	$\boldsymbol{F_1}$
Kshirsagar et al. (2015)	66.0	60.4	63.1
Yang and Mitchell (2017) (Rel)	71.8	57.7	64.0
Yang and Mitchell (2017) (Seq)	63.4	66.4	64.9
†Yang and Mitchell (2017) (All)	70.2	60.2	65.5
Semi-CRF baseline	67.8	66.2	67.0
+ constituent identity	68.1	67.4	67.7
+ nonterminal and parent	68.8	68.2	68.5
+ nonterminal	69.4	68.0	68.7
+ common nonterminals	69.2	69.0	**69.1**

Table 4.1: Frame SRL results on the test set of FrameNet 1.5., using gold frames. Ensembles are denoted by †.

[10] Yang and Mitchell (2017) also evaluated on the full frame-semantic parsing task, which includes frame-SRL as well as identifying frames. Since our frame SRL performance improves over theirs, we expect that incorporation into a full system (e.g., using their frame identification module) would lead to overall benefits as well; this experiment is left to future work.

[11] This result is not reported in Table 4.1 since Peng et al. (2018a) used a preprocessing which renders the test set slightly larger — the difference we report is calculated using their test set.

Model	**Prec.**	**Rec.**	$\boldsymbol{F_1}$
Zhou and Xu (2015)	-	-	81.3
He et al. (2017)	81.7	81.6	81.7
He et al. (2018a)	83.9	73.7	82.1
Tan et al. (2018)	81.9	83.6	82.7
Semi-CRF baseline	84.8	81.2	83.0
+ common nonterminals	85.1	82.6	**83.8**

Table 4.2: PropBank span SRL results, using gold predicates, on CoNLL 2012 test. For fair comparison, we show only non-ensembled models.

Model	**MUC**			$\mathbf{B^3}$			$\mathbf{CEAF_{\phi_4}}$			**Avg.** $\boldsymbol{F_1}$
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	
Wiseman et al. (2016)	77.5	69.8	73.4	66.8	57.0	61.5	62.1	53.9	57.7	64.2
Clark and Manning (2016b)	79.9	69.3	74.2	71.0	56.5	63.0	63.8	54.3	58.7	65.3
Clark and Manning (2016a)	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2	65.7
Lee et al. (2017)	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
+ common nonterminals	78.4	74.3	76.3	68.7	62.9	65.7	62.9	60.2	61.5	**67.8**

Table 4.3: Coreference resolution results on the test set on the English CoNLL-2012 shared task. The average F_1 of MUC, B^3, and CEAF$_{\phi_4}$ is the main evaluation metric. For fair comparison, we show only non-ensembled models.

4.6.2 PropBank SRL

We use the OntoNotes data from the CoNLL shared task in 2012 (Pradhan et al., 2013) for Propbank SRL. Table 4.2 reports results using gold predicates.

Recent competitive systems for PropBank SRL follow the approach of Zhou and Xu (2015), employing deep architectures, and forgoing the use of any syntax. He et al. (2017) improve on those results, and in analysis experiments, show that constraints derived using syntax may further improve performance. Tan et al. (2018) employ a similar approach but use feed-forward networks with self-attention. He et al. (2018a) use a span-based classification to jointly identify and label argument spans.

Our syntax-agnostic semi-CRF baseline model improves on prior work (excluding `ELMo`), showing again the value of global normalization in semantic structure prediction. We obtain further improvement of 0.8 absolute F_1 with the best syntactic scaffold from the frame SRL task. This indicates that a syntactic inductive bias is beneficial even when using sophisticated neural architectures.

He et al. (2018a) also provide a setup where initialization was done with deep contextualized embeddings, `ELMo` (Peters et al., 2018a), resulting in 85.5 F_1 on the OntoNotes test set. The improvements from `ELMo` are methodologically orthogonal to syntactic scaffolds.

Since the datasets for learning PropBank semantics and syntactic scaffolds *completely* overlap, the performance improvement cannot be attributed to a larger training corpus (or, by extension, a larger vocabulary), though that might be a factor for frame SRL.

A syntactic scaffold can match the performance of a pipeline containing carefully extracted syntactic features for semantic prediction (Swayamdipta et al., 2017). This, along with other recent approaches (He et al., 2017, 2018b) show that syntax remains useful, even with strong neural models for SRL.

4.6.3 Coreference Resolution

We report the results on four standard scores from the CoNLL evaluation: MUC, B^3 and CEAF$_{\phi_4}$, and their average F_1 in Table 4.3. Prior competitive coreference resolution systems (Wiseman et al., 2016; Clark and Manning, 2016b,a) all incorporate synctactic information in a pipeline, using features and rules for mention proposals from predicted syntax.

Our baseline is the model from Lee et al. (2017), described in §4.5. Similar to the baseline model for frame SRL, and in contrast with prior work, this model does not use any syntax.

We experiment with the best syntactic scaffold from the frame SRL task. We used `NP`, `OTHER`, and NULL as the labels for the common nonterminals scaffold here, since coreferring mentions are rarely prepositional phrases. The syntactic scaffold outperforms the baseline by 0.6 absolute F_1. Contemporaneously, Lee et al. (2018) proposed a model which takes in account higher order inference and more aggressive pruning, as well as initialization with `ELMo` embeddings, resulting in 73.0 average F_1. All the above are orthogonal to our approach, and could be incorporated to potentially yield higher gains.

4.7 Discussion

To investigate the performance of the syntactic scaffold, we focus on the frame SRL results, where we observed the greatest improvement with respect to a non-syntactic baseline.

We consider a breakdown of the performance by the syntactic phrase types of the

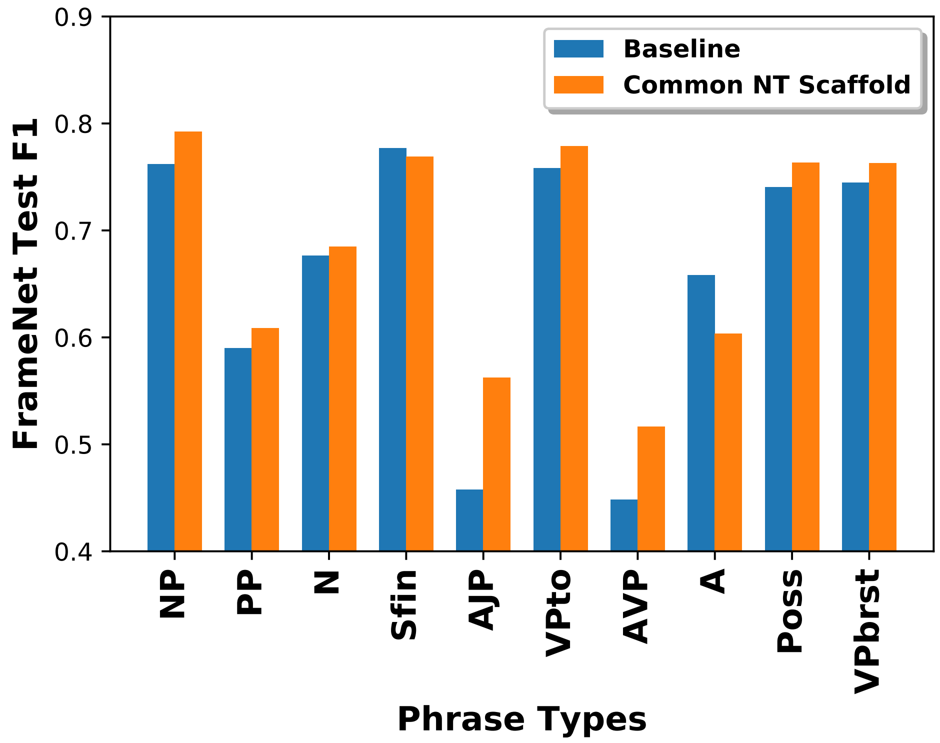


Figure 4.2: Performance breakdown by argument's phrase category, sorted left to right by frequency, for top ten phrase categories.

arguments, provided in FrameNet[12] in Figure 4.2. Not surprisingly, we observe large improvements in the common nonterminals used (`NP` and `PP`). However, the phrase type annotations in FrameNet do not correspond exactly to the OntoNotes phrase categories. For instance, FrameNet annotates non-maximal (`A`) and standard adjective phrases (`AJP`), while OntoNotes annotations for noun-phrases are flat, ignore the underlying adjective phrases. This explains why the syntax-agnostic baseline is able to recover the former while the scaffold is not.

Similarly, for frequent frame elements, scaffolding improves performance across the board, as shown in Fig. 4.3. The largest improvements come for `Theme` and `Goal`, which are predominantly realized as noun phrases and prepositional phrases. The scaffold also accounts for improvements in prediction of longer arguments, as is evidenced in Fig. 4.4 as well as arguments further away from the predicate, as in Fig. 4.5.

[12]We used FrameNet syntactic phrase annotations for analysis only, and not in our models, since they are annotated only for the gold arguments.

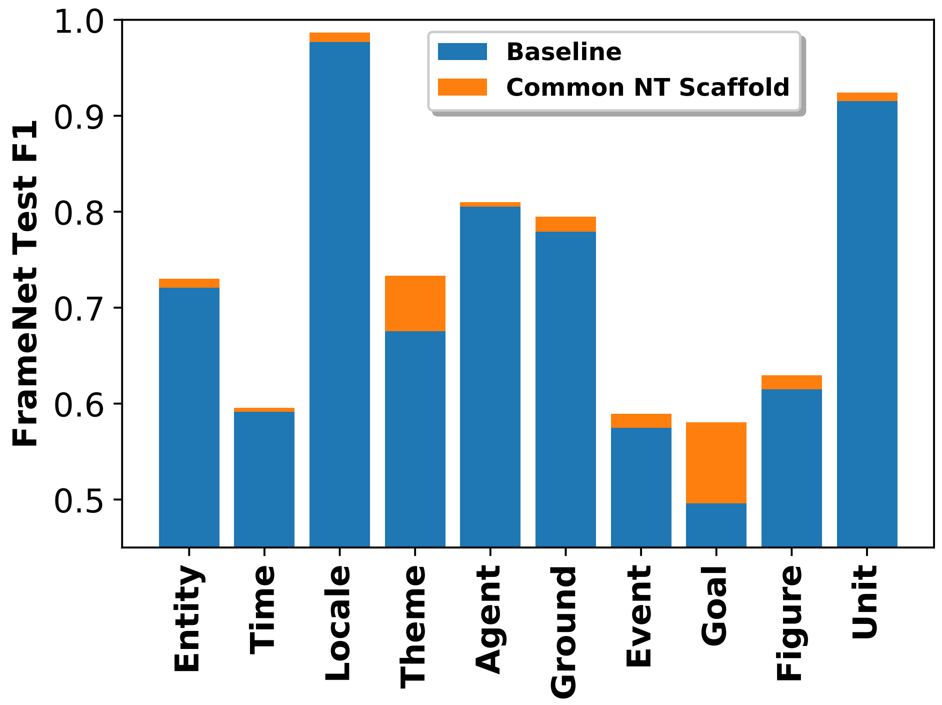


Figure 4.3: Performance breakdown by top ten frame element types, sorted left to right by frequency.

4.7.1 Effect of Contextualized Word Embeddings

The results from (§4.6.1) raise the question of how much improvement in performance could be attributed to using representations trained on more data. One excellent source of such representations are contextualized embeddings for word tokens obtained as a by product of training large language models (Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). We initialize our semi-CRF model with ELMo representations Peters et al. (2018a) to study this, and report results in Fig. 4.6. As expected, both the baseline and the scaffold models improve with ELMo embeddings, as these can capture much richer contextual information than bidirectional LSTMs trained on the target task alone. However, there still is a relative gain of 0.7 F_1 points with the noun and prepositional phrase scaffold, compared to the baseline, demonstrating that the improvements from ELMo are orthogonal. A similar finding was reported by Strubell et al. (2018).

4.8 Conclusion

We introduced syntactic scaffolds, a multitask learning approach to incorporate syntactic bias into semantic processing tasks. Unlike pipelines and approaches which jointly

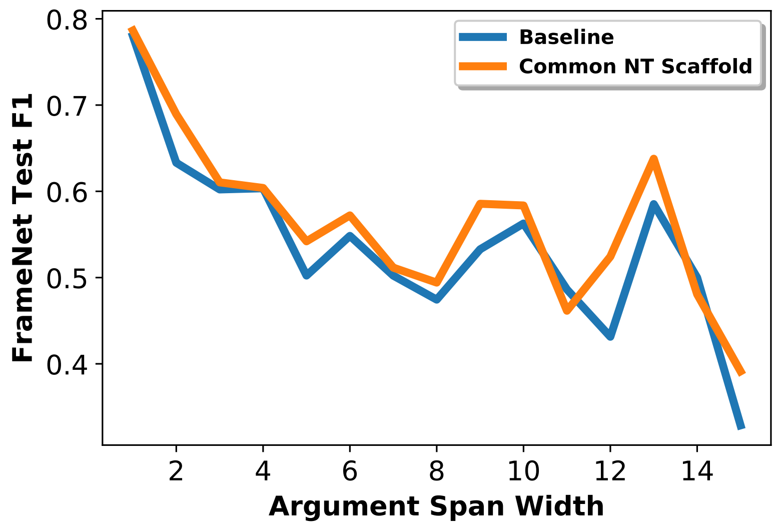


Figure 4.4: Performance of models on FN test by width of argument.

model syntax and semantics, no explicit syntactic processing is required at runtime. Our method improves the performance of competitive baselines for semantic role labeling on both FrameNet and PropBank, and for coreference resolution. While our focus was on span-based tasks, syntactic scaffolds could be applied in other settings (e.g., dependency and graph representations).

Semantic Scaffolds Moreover, scaffolds need not be syntactic; we can imagine, for example, *semantic* scaffolds being used to improve NLP applications with limited annotated data. Other kinds of span-based predictions, such as those needed in extractive question answering such as SQuAD (Rajpurkar et al., 2016) also frequently coincide with semantic arguments. Another direct application of span-based semantic scaffolds is given by causal structure learning, where sentences are divided into cause and effect subsequences, joined by causal relations (Dunietz et al., 2017). Such applications stand to still benefit from structural guidance, even as strong contextualized representations are employed; the advantages offered by either are orthogonal. It remains an open empirical question to determine the relative merits of different kinds of scaffolds and multi-task learners, and how they can be most productively combined.

One perspective on syntactic scaffolds is to view them as syntactic pretraining tasks.[13] Access to a large treebank would make it possible to pretrain representations with syntax. However, such large resources containing gold annotations are unavailable, but we could automatically predict syntactic parses on large datasets and use them for pretrain-

[13]On preliminary experiments we saw that training the scaffold model first, and then training the primary task model, did not offer much performance or run time benefit over training both tasks simultaneously. We hypothesize this might be because the primary and scaffold datasets are similarly sized.

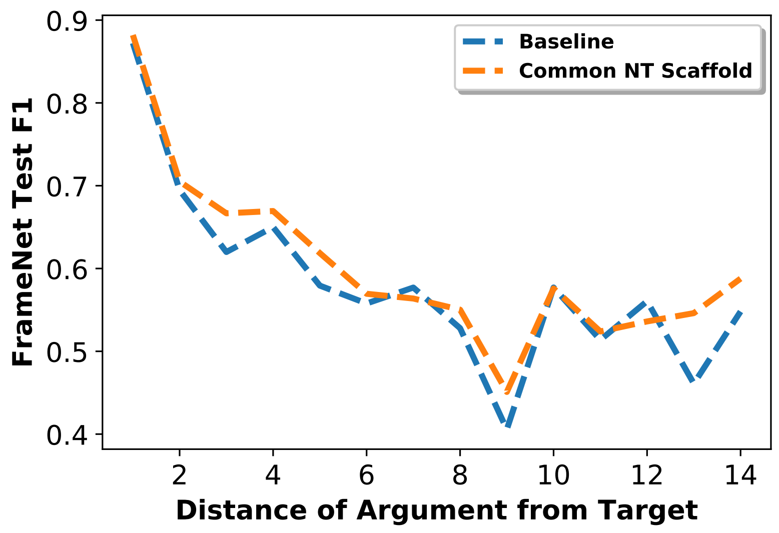


Figure 4.5: Performance of models on FN test by distance of argument from target.

ing, extending approaches suggested by Peters et al. (2018a). In the next chapter, we will see a method based on this idea.

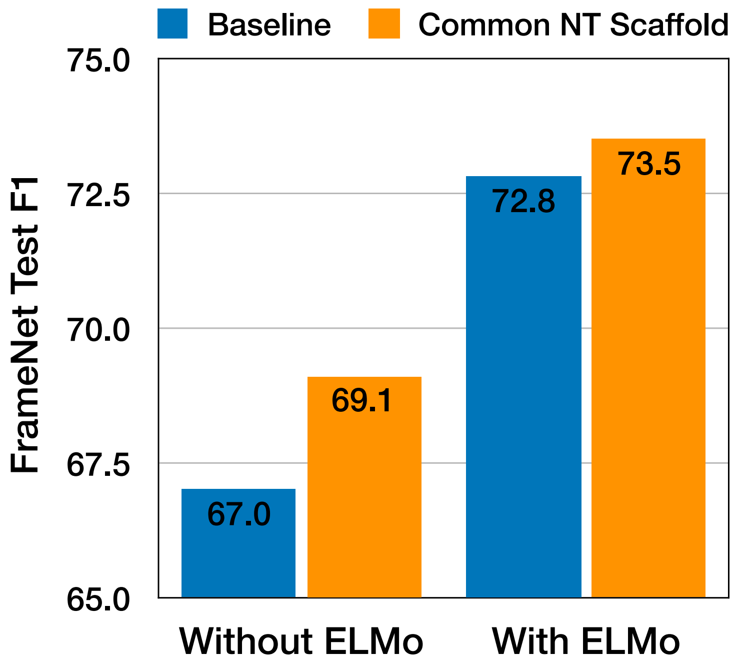


Figure 4.6: Effect of ELMo embeddings on Frame SRL, with and without scaffolding.

Chapter 5

Stage-Wise Pretraining with Shallow Syntax

In this chapter, we consider the role of syntactic inductive biases in language modeling. Specifically, we are interested in language modeling from the perspective of producing contextualized representations of word tokens (hereafter, CWR). These are dense, continuous vectors for word tokens *in context*[1] produced as a side-effect of language model estimation at a large scale (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018).[2] Language modeling, hence, serves as pretraining task, to obtain CWR, followed by the downstream task-specific training. As a result of being trained on large amounts of data, CWR such as ELMo (Peters et al., 2018a), OpenAI GPT (Radford et al., 2018) and BERT (Devlin et al., 2018) have been instrumental in improving performance of several downstream tasks, such as text classification, parsing and question answering. The larger the exposure to data, the more powerful the CWR are, as this allows them to model word tokens in a variety of contexts.

However, the large-scale requirement also urges the language models to be efficient, by making simplistic assumptions about linguistic structure. As a result, language modeling pretraining to produce CWR tends to treat language as *sequential*. Given that language is inherently hierarchical (Chapter 1), sequential biases are inappropriate for learning generalizations of language.[3] Generalizations hinge on access to the syntactic struc-

[1]Independent occurrences of the same word token may not correspond to the same CWR, since they might be in different contexts; CWR are therefore functions of the entire input sentence. In contrast, word vectors (Mikolov et al., 2013; Pennington et al., 2014) correspond to word types; even polysemous words (regardless of context) get the same word vector.

[2]Other tasks such machine translation (McCann et al., 2017) and sequence labeling (Clark et al., 2018) have also been used to obtain contextualized representations. However, the scale of supervision for these tasks is limited, hence semi-supervised methods are employed to match the performance of CWR from language modeling.

[3]Attention mechanisms (Bahdanau et al., 2014) have been employed in language modeling architec-

ture of language—access to this structure would enable the resulting cwr to be robust enough to handle new, unseen constructions. Syntactic language models (Dyer et al., 2016) have been well studied; see §5.1.2 for a detailed discussion.

On the other hand, despite having just a sequential bias, many recent studies have begun showing that some linguistic knowledge is implicitly encoded in cwr (Zhang and Bowman, 2018; Blevins et al., 2018). Goldberg (2019) found that BERT embeddings (Devlin et al., 2018) capture a surprisingly large amount of syntax. Tenney et al. (2019) and Liu et al. (2019) find that while cwr capture syntactic phenomena well, the same cannot be said for some semantic phenomena. Sequentially-biased language models seem to contain "enough" information for the downstream tasks. [4]

As we saw in previous chapters, downstream task performance may still benefit from *explicitly* injecting a syntactic inductive bias into model architectures. Kuncoro et al. (2018) argue for explicitly integrating syntactic inductive biases into models for increased awareness of syntactic phenomena such as number agreement. Strubell et al. (2018) show explicit syntactic integration is useful for semantic role labeling. However, high quality linguistic structure annotation at a large scale remains expensive—a trade-off needs to be made between the quality of the annotations and the computational expense of obtaining them.

Shallow syntactic structures (Abney, 1991; also called chunk sequences) offer a viable middle ground, by providing a flat, non-hierarchical approximation to phrase-syntactic trees (see Fig. 5.1 for an example). These structures can be obtained efficiently, and with high accuracy, using sequence labelers. Compared to phrase-syntactic parsers, chunkers can be extremely efficient.[5] In this chapter we consider shallow syntax to be a proxy for linguistic structure. While shallow syntactic chunks are almost as ubiquitous as part-of-speech tags in standard NLP pipelines (Jurafsky and Martin, 2009), their relative merits in the presence of cwrs remain unclear.

We investigate the role of these structures using two methods. First, we enhance the ELMo architecture (Peters et al., 2018b) to allow pretraining on predicted shallow syntactic parses, instead of just raw text, so that contextual embeddings make use of shallow syntactic context (§5.2). We pose the question of whether pretraining with syntax is useful for learning contextualized representations. Since a traditional language model pretraining objective does not consider chunked text, we propose a new objective

tures (Vaswani et al., 2017). While these may capture some longer range dependencies between words in a sentence, they do not have the capability of capturing the richness of linguistic structure, because of their limitation of considering word-word or bilexical relationships only. Moreover, without explicit knowledge of syntactic structure, they might only capture co-occurrence relationships between words.

[4]Sequential RNNs are theoretically powerful enough to recognize context-free languages (Siegelmann and Sontag, 1995).

[5]Chunking with CRFs can be executed in $O(nl^2)$ time, where n is the length of a sentence, and l is the number of labels in the tagging task. Constituency parsing takes $O(n^3G)$ time, where G is the size of the grammar.

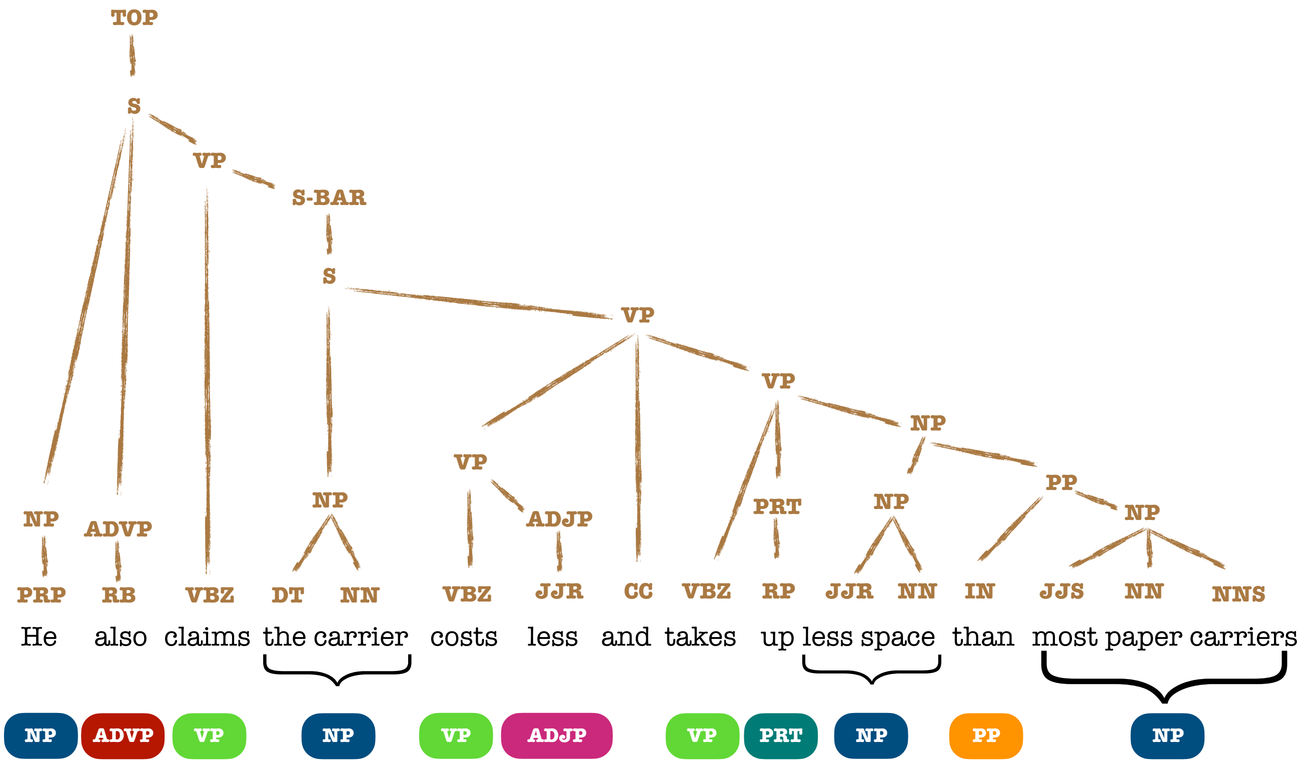


Figure 5.1: A sentence from WSJ Section 01 (Marcus et al., 1993) with its phrase-structure parse (shown above) and shallow syntactic (chunk) annotations, shown below. Following standard schemata, terminal nodes in the tree correspond to words in the sentence and preterminals to part-of-speech tags. Shallow syntactic annotations are derived from the full phrase-syntactic tree by a deterministic process (Tjong Kim Sang and Buchholz, 2000) to yield labeled *sequences* for the sentence. Unlike syntax, the structure is non-recursive and contains no overlapping tags. As a result of the deterministic process, some internal nodes (such as S) do not feature as shallow syntactic labels. Moreover, some spans, such as the single token span containing "and" get no tags. Shallow syntactic parsing, or chunking, is typically treated as a sequence labeling problem with a BIOUL encoding. The spans without tags get an O (outside) tag.

for estimating a model on *linguistically annotated* text, rather than raw text, and measure the effects on downstream tasks. For enabling efficient training, we initialize this model with pretrained embeddings from a bidirectional language model. Hence, we train on two objectives—a language modeling objective, followed by a data likelihood objective which conditions on a structured history; our approach involves a stage-wise pretraining. Our architecture is capable of consuming both word token representations as well as shallow syntactic labels using a hierarchy of encoders. This encoder facilitates a shallow syntactic inductive bias by producing an encoding which is structurally aware.

This also serves as an **analysis method** to determine whether syntactic pretraining is necessary. The reasoning behind our analysis is as follows: suppose that a syntactically endowed variant of CWR leads to improvements in accuracy of downstream consumer tasks (relative to baseline CWR). We could then infer that whatever implicit representation the baseline learned was less informative than the annotations.[6] Alternatively, if there is no benefit, then the relevant signal from the annotations about the task was *already* captured by baseline CWR.

Our second method involves classical addition of chunk features to CWR-infused architectures for four different downstream tasks (§5.3). Shallow syntactic information is obtained automatically using a highly accurate model (97% F_1 on standard benchmarks). In both settings, we observe only modest gains on three of the four downstream tasks relative to ELMo-only baselines (§5.4). We consistently find both to perform equally well, across four downstream tasks, including named entity recognition (coarse and fine-grained), phrase-structure parsing, and sentiment classification.

Recent work has probed the knowledge encoded in CWRs and found they capture a surprisingly large amount of syntax (Blevins et al., 2018; Liu et al., 2019; Tenney et al., 2019). We further examine the contextual embeddings obtained from the enhanced architecture and a shallow syntactic context, using black-box probes from Liu et al. (2019). Results on ten linguistic probes from Liu et al. (2019) confirm that performance is neither helped nor harmed by exposing the language model to explicit shallow syntax during pretraining, contributing strong evidence that baseline ELMo is already implicitly aware of shallow syntax. Our analysis indicates that our shallow-syntax-aware contextual embeddings do not transfer to linguistic tasks any more easily than ELMo embeddings ((§5.4.2)).

Overall, our experiments show while shallow syntax can be somewhat useful, ELMo-style pretraining discovers representations which make *additional* awareness of shallow syntax largely redundant. To summarize, the contributions in this chapter include:

- a general method for testing "knowledge" of CWR by training on annotated text (§5.2),
- strategies to make training large-scale models on annotated text feasible (§5.2.4),

[6]Levy and Goldberg (2014) observed gains from training non-contextual word embeddings from a syntactic dependency context.

and

- application of the method on four downstream tasks, as well as several probing tasks (§5.4).

5.1 Background

We discuss some background on CWR (§5.1.1), the relationship between language modeling and syntax (§5.1.2) and a brief discussion of shallow syntax (§5.1.3).

5.1.1 Language Models

Language models compute the probability of a piece of text $\boldsymbol{x}$, given its document context, or history. The probability of the entire document is given by

$$\begin{aligned} p(\boldsymbol{x}) &= p(x_1, \ldots x_T) \qquad (5.1) \\ &= \prod_{i=1}^{T} p(x_i | x_{1..i-1}) \\ &= \prod_{i=1}^{T} p(x_i \mid \boldsymbol{x}_{<i}) \qquad (5.2) \end{aligned}$$

Equation 5.2 shows the application of chain rule, and results in the factorization of the probability into components which estimate the probability of the subsequent word given the prefix. As is evident from above, there is no need of external supervision, hence language models are typically treated as unsupervised models of language. Perplexity, which measures how well the model predicts a sentence on a held-out test set is computed as $2^{-\frac{1}{T}\log p(\boldsymbol{x})}$; lower perplexity indicates better a language model.

Most modern language models are based on neural architectures such as recurrent neural nets (Mikolov et al., 2010), since these can score arbitrarily long histories, hence capturing distant contextual information. The context, or history, is updated from left to right in a sequential manner.[7]

$$p(x_i | x_{1..i-1}) = \prod_{i=1}^{T} \frac{\exp(\boldsymbol{v}_{x_i} \cdot \boldsymbol{h}_{i-1})}{\sum_{x \in \mathcal{V}} \exp(\boldsymbol{v}_x \cdot \boldsymbol{h}_{i-1})} \qquad (5.3)$$

[7]Bidirectional language models (Peters et al., 2018a) additionally update the future context by an additional objective which estimates parameters as it moves through a document in a right to left fashion.

Here, $\boldsymbol{v}_x$ is the vector associated with a word x, and $\boldsymbol{h}_t$ is the hidden state of the network after being updated with the first t words. Tractability is ensured by the ability of summarizing an entire history as a fixed-size representation, as we saw in Chapter 3.

Any sequence to sequence, encoder-decoder architecture can be used for language modeling. Popular architectures involve the transformer (Vaswani et al., 2017) which involves several feed-forward layers, followed by self-attention layers (Bahdanau et al., 2014). Character-based (Ling et al., 2015; Kim et al., 2016) and sub-word level features (Botha and Blunsom, 2014) are often used instead of words to capture morphological features as well as account for unknown vocabulary items.

5.1.2 Syntactic Language Models

While language models provide a statistical model of language, syntax provides a blueprint for the arrangement of words in a sentence—these two concepts are intimately related. As a result, there has been a long history of syntactially structured language models which estimate the joint probability of sentences and their underlying phrase-structure trees (Jelinek and Lafferty, 1991; Chelba and Jelinek, 2000; Roark, 2001; Emami and Jelinek, 2005; Dyer et al., 2016). The approaches of Roark (2001) and Dyer et al. (2016) involve generative models for top-down parsing and language modeling, the former with a grammar-based approach and the latter with a transition-based algorithm. In particular, Dyer et al. (2016) used a recurrent neural net architecture[8] as well as an explicit composition operator that represents completed constituents, serving as the syntactic inductive bias. Chelba and Jelinek (2000) and Emami and Jelinek (2005) proposed approaches for bottom-up shift reduce phrase-structure parsing and language modeling, primarily for speech recognition. While the above approaches model a richer linguistic structure given by the full parse tree, they need access to phrase-syntactic trees, and as a result of consuming the tree, can be expensive.

In contrast, hierarchical language models treat linguistic structure as latent (Chung et al., 2017; Chan et al., 2016; Wang et al., 2017; Buckman and Neubig, 2018). Most of these hierarchies correspond to text segmentations, as opposed to trees. Shen et al. (2018) show that the latent structures actually correspond to syntactic trees. As all latent variable approaches (§2.4), these can also be expensive to train. Most of these approaches involve backpropagation through a discrete variable (indicating segment boundary), and hence need to rely on approximate estimation techniques such as Gumbel softmax (Jang et al., 2017).

[8]LSTMs (Hochreiter and Schmidhuber, 1997), to be exact.

5.1.3 Shallow Syntax

Shallow syntax is a non-recursive simplification of phrase-syntactic trees, where the sentence is segmented into a sequence of non-overlapping syntactic *chunks*. Fig. 5.2 shows an example of a sentence with its phrase-syntactic parse as well as its shallow syntactic (partial) parse. Every token in the sentence is associated with a single chunk label, while it can be the descendant of several non-terminals in its tree. Tjong Kim Sang and Buchholz (2000) offered a rule-based transformation deriving non-overlapping chunks from phrase-structure trees as found in the Penn Treebank (Marcus et al., 1993). The process to find chunk boundaries involves finding the head of a constituent, including the span on the left of the head till the constituent boundary, and ignoring the rest of the span to the right. Since there are inconsistencies in the annotations of the Penn Treebank, there are similar inconsistencies in chunk boundary annotation (Tjong Kim Sang and Buchholz, 2000; Jurafsky and Martin, 2009).

Base phrase chunking is a cheap sequence-model–based alternative to full syntactic parsing. This task was addressed in the CoNLL 2000 shared task. The standard task definition includes eleven chunk labels, as shown in Table 5.1.

Label	Abbr.	% Occurrence
Noun Phrase	`NP`	51.7
Verb Phrase	`VP`	20.0
Prepositional Phrase	`PP`	19.8
Adverbial Phrase	`ADVP`	3.7
Subordinate Clause	`SBAR`	2.1
Adjective Phrase	`ADJP`	1.9
Verb Particles	`PRT`	0.5
Conjunctive Phrase	`CONJ`	0.06
Interjective Phrase	`INTJ`	0.03
List Marker	`LST`	0.01
Unlike Coordination Phrase	`UCP`	0.002

Table 5.1: Shallow syntactic chunk phrase types from CoNLL 2000 shared task (Tjong Kim Sang and Buchholz, 2000) and their occurrence percentage in the training data.

Chunking is useful for tasks such as information extraction where it might be sufficient to extract the meaning-bearing spans which correspond to chunk segments. Besides practical usages in applications, children are known to acquire language by learning elementary chunks of words first.[9]

[9] https://en.wikipedia.org/wiki/Language_acquisition

5.2 Pretraining with Shallow Syntactic Annotations

We briefly review language models for CWR, and then present a model architecture to obtain e**m**beddings from shallow **Syn**tactic **C**ontext (**mSynC**).

5.2.1 Language Models for Contextualized Representations

Traditional language models (§5.1.1) are estimated to maximize the likelihood of each word x_i given the words that precede it, $p(x_i \mid \boldsymbol{x}_{<i})$. A function f_{seq} is used for encoding the history of tokens $\boldsymbol{x}_{<i}$).

The token representation is context-independent (Jozefowicz et al., 2016; Melis et al., 2018; Merity et al., 2018) and is given by a CNN over character embeddings, followed by a linear projection to yield $\boldsymbol{v}_t$. This is input to the deep neural architecture, which results in contextualized representations for the tokens. A linear combination of all the hidden layers of the architecture, $\boldsymbol{h}_t$ is then used to predict the next word in the sequence via a cross-entropy loss, with negative sampling (Gutmann and Hyvärinen, 2012). We consider the architecture of Peters et al. (2018b), where transformers (Vaswani et al., 2017) were used as the encoder-decoder architecture.

Bidirectional language models (Peters et al., 2017) jointly estimate a left-to-right (forward) and right-to-left (backward) language model. Each objective uses the respective directional encoder, i.e. only forward encoder-decoders are used for the forward language model and vice-versa.

Adaptation to Downstream Tasks At test time, the language model is run over all the data for the downstream task (train, heldout and test). Resultant CWR are then used as input to the model for the downstream task. This approach is known as freezing, since the language model is frozen and not updated by fine-tuning to the downstream task data (Peters et al., 2019).

5.2.2 Training Objective

We extend the language modeling setting by considering a corpus that is annotated with shallow syntax. We propose to maximize the likelihood of each word conditioned on both the preceding words *and* their annotations. We associate with each word x_i three additional variables (denoted c_i): the indices of the beginning and end of the last completed chunk *before* x_i, and its label. For example, in Fig. 5.2, $c_4 = \langle 3, 3, \text{VP} \rangle$ for $x_4 = \text{the}$.

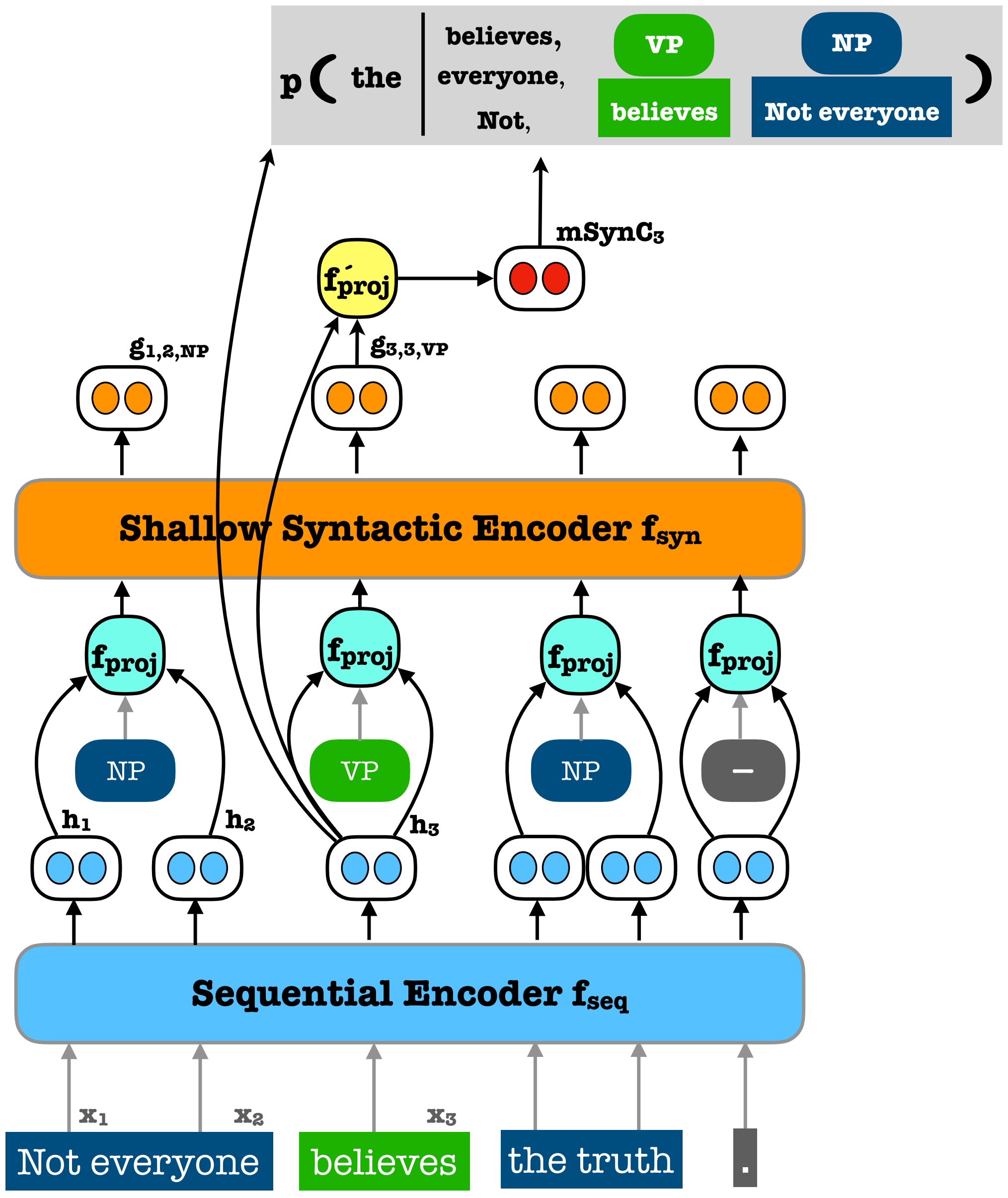


Figure 5.2: A sentence with its shallow syntactic (chunk) annotations, and the model architecture to handle such annotations. A sequential encoder converts the raw text into CWR (shown in blue). Observed shallow syntactic structure (chunk boundaries and labels) are combined with these CWR in a shallow syntactic encoder to get contextualized representations for chunks (shown in orange). CWR from both are passed through a projection layer to get mSynC embeddings (details shown only in some positions, for clarity), used both for computing the data likelihood, as shown, as well as in downstream tasks.

Chunks, $\boldsymbol{c}$ are only used as conditioning context via

$$p(x_i \mid \boldsymbol{x}_{<i}, \boldsymbol{c}_{\leqslant i}); \tag{5.4}$$

they are not predicted.[10] Because the $\boldsymbol{c}$ annotations depend on the entire sentence through the automatic chunker, conditioning each word's probability on any $\boldsymbol{c}_i$ means that our model is, strictly speaking, not a language model, and it can no longer be meaningfully evaluated using perplexity.

A right-to-left model is constructed analogously, conditioning on $\boldsymbol{c}_{\geqslant i}$ alongside $\boldsymbol{x}_{>i}$. Following Peters et al. (2018a), we use a joint objective maximizing data likelihood objectives in both directions, with shared softmax parameters.

5.2.3 Model Architecture

Our model uses two encoders: f_{seq} for encoding the sequential history ($\boldsymbol{x}_{<i}$), and f_{syn} for shallow syntactic (chunk) history ($\boldsymbol{c}_{\leqslant i}$). For both, we use transformers (Vaswani et al., 2017), which consist of large feedforward networks equipped with multiple self-attention mechanisms.[11]

As inputs to f_{seq}, we use a context-independent embedding, obtained from a CNN character encoder (Kim et al., 2016) for each token x_i. The outputs $\boldsymbol{h}_i$ from f_{seq} represent words in context.

Next, we build representations for (observed) chunks in the sentence by concatenating a learned embedding for the chunk label with $\boldsymbol{h}$'s for the boundaries and applying a linear projection (f_{proj}). The output from f_{proj} is input to f_{syn}, the shallow syntactic encoder, and results in contextualized chunk representations, $\boldsymbol{g}$. Note that the number of chunks in the sentence is less than or equal to the number of tokens.

Each $\boldsymbol{h}_i$ is now concatentated with $\boldsymbol{g}_{c_i}$, where $\boldsymbol{g}_{c_i}$ corresponds to c_i, the last chunk before position i. Finally, the output is given by $\mathbf{mSynC}_{\,i} = {f'}_{proj}(\boldsymbol{h}_i, \boldsymbol{g}_{c_i}) = \boldsymbol{W}^\top[\boldsymbol{h}_i; \boldsymbol{g}_{c_i}]$, where $\boldsymbol{W}$ is a model parameter. For training, $\mathbf{mSynC}_{\,i}$ is used to compute the probability of the next word, using a sampled softmax (Bengio et al., 2003). For downstream tasks, we use a learned linear weighting of all layers in the encoders to obtain a task-specific $\mathbf{mSynC}$, following Peters et al. (2018a).

[10]A different objective could consider predicting the next chunks, along with the next word. However, this chunker would have access to strictly less information than usual, since the entire sentence would no longer be available.

[11]While transformers are shown to have slightly lower performance on downstream tasks than RNNs for pretraining CWR (Peters et al., 2018b), this model trains almost twice as fast on two NVIDIA Tesla V100s, and is hence cost-effective.

5.2.4 Staged Parameter Updates

Training both the sequential encoder, f_{seq}, and the syntactic encoder, f_{syn}, jointly to maximize the data likelihood can be expensive, due to the large number of parameters involved. To reduce cost, we initialize our sequential CWR, $\boldsymbol{h}$, using *pretrained* embeddings from ELMo-transformer (Peters et al., 2018b). Once initialized as such, the encoder is fine-tuned to the data likelihood objective (§5.2.2). This results in a staged parameter update, which reduces training duration by a factor of 10 in our experiments. We discuss the empirical effect of this approach in §5.4.3.

5.3 Shallow Syntactic Features

Our second approach incorporates shallow syntactic information in downstream tasks via token-level chunk label embeddings. Task training (and test) data is automatically chunked, and chunk boundary information is passed into the task model via BIOUL encoding of the labels. We add randomly initialized chunk label embeddings to task-specific input encoders, which are then fine-tuned for task-specific objectives. This approach does not require a shallow syntactic encoder or chunk annotations for pretraining CWRS, only a chunker. Hence, this can more directly measure the impact of shallow syntax for a given task.[12]

5.4 Experiments and Results

Our experiments evaluate the effect of shallow syntax, via contextualization (**mSynC**, §5.2) and features (§5.3). We provide comparisons with four baselines—ELMo-transformer (Peters et al., 2018b), our reimplementation of the same, as well as two CWR-free baselines, with and without shallow syntactic features. Both ELMo-transformer and **mSynC** are trained on the 1B word benchmark corpus (Chelba et al., 2013); the latter also employs chunk annotations (§5.1.3) obtained automatically. We use a BiLSTM-CRF model for chunking (Lample et al., 2016; Peters et al., 2017), which achieves 97% F_1 on the CoNLL 2000 (Tjong Kim Sang and Buchholz, 2000) benchmark test set.[13]

Architecture and Implementation

[12]In contrast, in §5.2, the shallow-syntactic encoder itself, as well as predicted chunk quality on the large pretraining corpus could affect downstream performance.

[13]We trained on data from the CoNLL 2000 shared task, as well as the remaining sections (except §23 and §20) of the Penn Treebank. Chunks on PTB were obtained using the official script for chunk generation, `https://www.clips.uantwerpen.be/conll2000/chunking/`.

ELMo-transformer Our baseline pretraining model was a reimplementation of that given in Peters et al. (2018b). Hyperparameters were generally identical, but we trained on only 2 GPUs with (up to) 4,000 tokens per batch. This difference in batch size meant we used 6,000 warm up steps with the learning rate schedule of Vaswani et al. (2017).

mSynC The function f_{seq} is identical to the 6-layer biLM used in ELMo-transformer. f_{syn}, on the other hand, uses only 2 layers. The learned embeddings for the chunk labels have 128 dimensions and are concatenated with the two boundary $\boldsymbol{h}$ of dimension 512. Thus f_{proj} maps $1024 + 128$ dimensions to 512. Further, we did not perform weight averaging over several checkpoints.

Shallow Syntax The size of the shallow syntactic feature embedding was 50 across all experiments, initialized uniform randomly.

All model implementations are based on the `AllenNLP` library (Gardner et al., 2017).

5.4.1 Downstream Task Transfer

We employ four tasks to determine ELMo CWR awareness of shallow syntax. Following Peters et al. (2018a), we do not apply finetuning to task-specific architectures, allowing us to do a controlled comparison with ELMo CWR, in particular the transformer-based architecture from Peters et al. (2018b) (ELMo-transformer). Given the base architecture is identical across both settings, we can attribute any difference in performance to the incorporation of shallow syntax.

NER We use the English portion of the CoNLL 2003 dataset (Tjong Kim Sang and De Meulder, 2003), which provides named entity annotations on newswire data across four different entity types (`PER`, `LOC`, `ORG`, `MISC`). A bidirectional LSTM-CRF architecture (Lample et al., 2016) and a BIOUL tagging scheme are used.

Fine-grained NER The same architecture and tagging scheme from above is also used to predict fine-grained entity annotations from OntoNotes 5.0 (Weischedel et al., 2011). The 18 fine-grained NER labels in the dataset are shown in Table 5.2.

Phrase-structure parsing We use the standard Penn Treebank splits, and adopt the span-based model from Stern et al. (2017). Following their approach, we used predicted

Label	Description
`PERSON`	People, including fictional
`NORP`	Nationalities or religious or political groups
`FACILITY`	Buildings, airports, highways, bridges, etc.
`ORGANIZATION`	Companies, agencies, institutions, etc.
`GPE`	(Geo-Political Entity) Countries, cities, states
`LOCATION`	Non-GPE locations, mountain ranges, bodies of water
`PRODUCT`	Vehicles, weapons, foods, etc. (Not services)
`EVENT`	Named hurricanes, battles, wars, sports events, etc.
`WORK OF ART`	Titles of books, songs, etc.
`LAW`	Named documents made into laws
`LANGUAGE`	Any named language
`DATE`	Absolute or relative dates or periods
`TIME`	Times smaller than a day
`PERCENT`	Percentage (including "%")
`MONEY`	Monetary values, including unit
`QUANTITY`	Measurements, as of weight or distance
`ORDINAL`	"first", "second"
`CARDINAL`	Numerals that do not fall under another type

Table 5.2: Fine-grained NER labels from OntoNotes 5.0 (Weischedel et al., 2013).

part-of-speech tags from the Stanford tagger (Toutanova et al., 2003) for training and testing. About 51% of phrase-syntactic constituents align exactly with the predicted chunks used, with a majority being single-width noun phrases. Given that the rule-based procedure used to obtain chunks only propagates the phrase type to the head-word and removes all overlapping phrases to the right, this is expected. We did not employ jackknifing to obtain predicted chunks on PTB data; as a result there might be differences in the quality of shallow syntax annotations between the train and test portions of the data.

Task	Train	Heldout	Test
CoNLL 2003 NER (Tjong Kim Sang and De Meulder, 2003)	23,499	5,942	5,648
OntoNotes NER (Weischedel et al., 2013)	81,828	11,066	11,257
Penn TreeBank (Marcus et al., 1993)	39,832	1,700	2,416
Stanford Sentiment Treebank (Socher et al., 2013)	8,544	1,101	2,210

Table 5.3: Downstream Dataset Statistics.

Sentiment analysis We consider fine-grained (5-class) classification on Stanford Sentiment Treebank (Socher et al., 2013). The labels are `negative`, `somewhat_negative`, `neutral`, `positive` and `somewhat_positive`. Our model was based on the Biattentive Classification Network (McCann et al., 2017). We used all phrase lengths in the dataset for training, but test results are reported only on full sentences, following prior work.

Dataset statistics for all tasks is shown in Table 5.3. The first three of the four tasks above are *span-based*, so we would expect explicit shallow syntactic cues to provide useful signal for them. The sentiment classification task is included to test the impact of **mSynC** for text classification. All downstream model implementations use the defaults from the `AllenNLP` library.

	NER	**Fine-grained NER**	**Constit. Parsing**	**Sentiment Classif.**
Baseline (no CWR)	88.1 ± 0.27	78.5 ± 0.19	88.9 ± 0.05	51.6 ± 1.63
+ shallow syn. features	88.6 ± 0.22	78.9 ± 0.13	90.8 ± 0.14	51.1 ± 1.39
ELMo-transformer	91.1 ± 0.26	—	93.7	—
ELMo-transformer (our reimpl.)	91.5 ± 0.25	85.7 ± 0.08	94.1 ± 0.06	53.0 ± 0.72
+ shallow syn. features	91.6 ± 0.40	85.9 ± 0.28	94.3 ± 0.03	52.6 ± 0.54
Shallow syn. context (**mSynC**)	91.5 ± 0.19	85.9 ± 0.20	94.1 ± 0.07	53.0 ± 1.07

Table 5.4: Test-set performance of ELMo-transformer Peters et al. (2018b), our reimplementation, and **mSynC**, compared to baselines without CWR. Evaluation metric is F_1 for all tasks except sentiment, which reports accuracy. Reported results show the mean and standard deviation across 5 runs for coarse-grained NER and sentiment classification and 3 runs for other tasks.

	ELMo-transformer	**mSynC**
CCG	92.68	92.03
PTB POS Tagging	97.09	96.91
EWT POS Tagging	95.13	94.64
Chunking	92.18	96.89
Named Entity Recognition	81.21	79.98
Semantic Tagging	93.78	93.03
Grammar Error Detection	30.80	30.86
Prep. Role	72.81	70.83
Prep. Func.	82.24	82.67
Event Factuality	70.88	70.39

Table 5.5: Test performance of ELMo-Transformer vs. **mSynC** on several linguistic probes from Liu et al. (2019). In each case, performance of the best layer from the architecture is reported. Details on each metric can be found in Table 5.7.

	Model	**Fine-grained NER F_1**
end-to-end	ELMo	86.90 ± 0.11
	mSynC end-to-end	86.89 ± 0.04
initialized	**mSynC** frozen	87.36 ± 0.02
	mSynC fine-tuned	87.44 ± 0.07

Table 5.6: Validation F_1 for fine-grained NER across syntactic pretraining schemes, with mean and standard deviations across 3 runs.

Results are shown in Table 5.4. Consistent with previous findings, CWRs offer large improvements across all tasks. Though helpful to span-level task models without CWRs, shallow syntactic features offer little to no benefit to ELMo models. **mSynC**'s performance is similar. On sentiment classification, chunk features are slightly harmful on average (but variance is high); **mSynC** again performs similarly to ELMo-transformer. Overall, the performance differences across all tasks are small enough to infer that shallow syntax is not particularly helpful when using CWRs.

5.4.2 Linguistic Probes

To test whether **mSynC** learns shallow syntactic information, and how this compares to ELMo-transformer, we examine both embeddings for awareness of chunk information. In particular, we use the probes from Liu et al. (2019), which train linear models on frozen CWR for making predictions about linguistic (syntactic and semantic) properties of words and phrases. Unlike the downstream tasks in §5.4.1, there is minimal downstream task architecture, bringing into focus the transferrability of CWR.

As expected, on the probe for predicting chunk tags, **mSynC** achieves 96.9 F_1 vs 92.2 F_1 for ELMo-transformer, indicating that **mSynC** is indeed aware of shallow syntax. We further evaluated the CWR on 9 other probes—CCG supertagging (Hockenmaier and Steedman, 2007), PTB (Marcus et al., 1993) and Universal Dependencies (Silveira et al., 2014) POS tagging, NER (Tjong Kim Sang and De Meulder, 2003), grammar error detection (Yannakoudakis et al., 2011), semantic tagging (Bjerva et al., 2016), preposition supersense identification (Schneider et al., 2018), and event factuality detection (Rudinger et al., 2018). Metrics and references for each are summarized in Table 5.7. For more details, readers are referred to Liu et al. (2019).

Results in Table 5.5 show that again the performance of baseline ELMo-transformer and **mSynC** are similar, with **mSynC** doing slightly worse on 7 out of 9 tasks. Overall, the results further confirm that shallow syntax does not offer any benefits over ELMo-transformer.

5.4.3 Effect of Training Scheme

We test whether our staged parameter training (§5.2.4) is a viable alternative to an end-to-end training of both f_{syn} and f_{seq}. We make a further distinction between fine-tuning f_{seq} vs. not updating it all after initialization (frozen).

Downstream validation F_1 on fine-grained NER reported in Table 5.6 show that the end-to-end strategy lags behind the others, perhaps indicating the need to train longer than 10 epochs. However, a single epoch on the 1B-word benchmark takes 36 hours on 2 Tesla V100s, making this prohibitive. Interestingly, the frozen strategy, which takes the least amount of time to converge (24 hours on 1 Tesla V100), also performs almost as well as fine-tuning.

5.5 Conclusion

We find that exposing CWR-based models to shallow syntax, either through new CWR architectures or explicit pipelined features, has little effect on their performance, across several tasks. Linguistic probing also shows that CWRs aware of such structures do not improve task transferability. Both findings indicate that the shallow syntactic structural assumption is perhaps too strong to be useful, especially when pitted against contextualization.

Our generalized architecture and method can be extended to incorporate a larger variety of inductive biases, such as full syntactic trees, into pretraining language models. A modified pretraining could be used to predict shallow syntactic labels of the word, in addition to the word itself. We leave extending this approach to a masked language modeling objective used in BERT (Devlin et al., 2018) to future work.

Task	Dataset	Metric
CCG	CCGBank (Hockenmaier and Steedman, 2007)	Accuracy
PTB POS	PennTreeBank (Marcus et al., 1993)	Accuracy
EWT POS	Universal Dependencies (Silveira et al., 2014)	Accuracy
Chunk	CoNLL 2000 (Tjong Kim Sang and Buchholz, 2000)	F_1
NER	CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003)	F_1
Semantic Tagging	(Bjerva et al., 2016)	Accuracy
Grammar Error Detection	First Certificate in English (Yannakoudakis et al., 2011)	F_1
Preposition Supersense Role	STREUSLE 4.0 (Schneider et al., 2018)	Accuracy
Preposition Supersense Function	STREUSLE 4.0 (Schneider et al., 2018)	Accuracy
Event Fact.	UDS It Happened v2 (Rudinger et al., 2018)	Pearson R

Table 5.7: Dataset and metrics for each probing task from Liu et al. (2019), corresponding to Table 5.5.

Chapter 6

Conclusions and Future Work

We presented here three approaches to incorporate syntactic inductive biases for learning continuous representations of language. The motivation behind the need for such biases is that representation learning for language is aided by exposure to linguistic, particularly syntactic structure. Incorporation of syntax has the potential to help artificial language learners generalize to new, previously unseen data. We tried to address the challenges of incorporating a syntactic inductive bias and designing architectures which support such biases, across three settings:

- **Joint learning of entire syntactic trees** (Chapter 3, Swayamdipta et al. (2016)). We addressed the task of dependency-based semantic role labeling. Since these dependencies are structurally close to syntactic dependencies, we designed a syntactic inductive bias via a joint objective for predicting both structures. We used a transition-based incremental algorithm. An intermediate continuous representations summarized the entire parser history at each time step. A composition function synthesized syntactic and semantic substructures. This resulted in a better performance compared to prior joint modeling approaches across two different shared tasks and seven different languages.
- **Multitask Learning with Syntactic Substructures** (Chapter 4, Swayamdipta et al. (2017), Swayamdipta et al. (2018b)). We addressed three different span-based semantic structure prediction tasks. The substructures involved in each task had correspondences to syntactic constituent structures. A syntactic inductive bias was incorporated via a multitask objective to predict both the full semantic structure (primary task) as well as the syntactic substructures (scaffold task). Span embeddings trained towards scoring syntactic as well as semantic substructures were shared across both tasks. We reported improvements across strong baselines on all three tasks of FrameNet and Propbank SRL, and coreference resolution.
- **Stage-Wise Pretraining with Shallow Syntax** (Chapter 5) We analyzed how helpful shallow syntactic annotations might be to contextualized word representations.

A syntactic inductive bias was incorporated via a data likelihood objective which conditioned on both a sequential and a shallow syntactic history. A shallow-syntactic encoder was designed to consume both word tokens and base phrase chunks; hidden representations from which captured syntactic contextual information. The resultant contextualized representations performed on par with the baseline on several downstream tasks, as well as light-weight black box analyzers.

The primary difference between the three settings was the decreasing level of granularity of the syntactic representations used. We moved from full trees in Chapter 3 to overlapping phrase structures from the trees in Chapter 4 to non-overlapping phrases in Chapter 5. Syntactic inductive biases were most helpful in the first two settings, and in the last setting, did not hurt performance. There is perhaps a trade-off between using richer structures to induce biases and the efficiency of the approach itself.

While some of the tasks discussed achieved state-of-the-art performance, there exists a lot of room for improvement. There are obvious benefits of using representations pretrained on massive datasets, and carefully tuned hyperparameters. Perhaps taking into account more context than such models can capture, again via structures such as document structures, might lead to further benefits.

This thesis opens up several questions for future research. For each task, we (manually) selected the syntactic representation that is best suited for it. Approaches that instead learn to automatically predict the level of granularity of syntax might be more general. Another question this thesis raises is could we obtain performance benefits from incorporating more powerful structural encoders? Perhaps there is a need to move towards larger models which take into account different kinds of inductive biases all at the same time (Subramanian et al., 2018), instead of just syntactic ones. The above would involve building better structural decoders.

Structure Manipulation While this thesis has involved relying on available linguistic structure (gold or predicted) for training, similar principles might be instrumental in generating language which adheres to structural constraints. Linguistic structure can be manipulated to generate stylistic translations, for example, involving consistent use of certain lexical as well as syntactic constructions which do not occur in the source sentence. Continuous representations could be utilized to detect templates governing such translations, and bring about rephrasal of text, without altering the meaning of the utterance. Similarly, manipulation of semantic structures also opens up interesting possibilities, such as altering sentiment, polarity and stance. Given the background on learning continuous representations of structure in this thesis, these questions are interesting future directions.

Structure Learning from Large Crowd-Source Annotations One of the contributors to the success of deep learning is the availability of large datasets, where annotations are collected through crowd-sourcing. However, crowd-sourced annotations tend to accumulate artifacts (e.g. simple negations are used as a shortcut to generate contradictions). Deep models tend to learn from these artifacts, and ignore the actual reasoning—a finding we uncovered for some popular datasets (Gururangan et al., 2018). Given that crowd-sourcing is convenient, it is hard to get rid of annotation artifacts, which correspond to systematic, but shallow structural patterns. However, if a model can exploit these, it could also be trained to detect them. Based on this intuition, it might be possible to build models which do not just learn from the entire dataset but are able to identify instances which might be unreliable, downweight the quality of such instances, and ultimately learn to ignore them. Continuous latent variable models could be used to encode global information, such as style, which is a strong indicator of artifacts. Learning algorithms which iteratively subsample the training data, based on such global, structured information could be explored towards this.

Finally, the approaches discussed were applied to predominantly semantic structured prediction tasks, and used syntactic inductive biases, but are not limited to either. Semantic biases could be useful for capturing deeper phenomena as observed in discourse and pragmatics, which are essential in designing automated assistants and dialog systems. Other kinds of structure, such as knowledge graphs could be used instead of linguistic structure, to encourage knowledge-aware representations.

Bibliography

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proc. of ACL*, volume 1, pages 228–238. 5, 15, 41

Steven P Abney. 1991. Parsing by chunks. In *Principle-based parsing*, pages 257–278. Springer. 59

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. ArXiv:1409.0473. 46, 58, 63

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval'07 Task 19: Frame semantic structure extraction. In *Proc. of SemEval*. 49

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of ACL*. 5, 11, 15, 17, 36, 42

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proc. of EMNLP*. 34

Miguel Ballesteros and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics*, 39(1):5–13. 22

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (AMR) 1.0 specification. In *Proc. of EMNLP*. 5, 11, 15

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155. 67

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. *arXiv:1609.07053*. 72, 74

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proc. of COLING*. 12, 33, 35

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proc. of CoNLL*. 33

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. In *Proc. of ACL*. 59, 61

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort an Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to end learning for self-driving cars. 17

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907. 63

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. of EMNLP*. 2

Joan Bresnan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2015. *Lexical-functional syntax*, volume 16. John Wiley & Sons. 4

Jacob Buckman and Graham Neubig. 2018. Neural lattice language models. In *Proc. of ACL*. 63

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proc. of CoNLL*. 35

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1). 5, 15, 39

William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly. 2016. Latent sequence decompositions. In *Proc. of ICLR*. 63

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded syntactic and semantic dependency parsing system. In *Proc. of CoNLL*. 31

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proc. of CoNLL*. 32, 33, 34

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech & Language*, 14(4):283–332. 6, 63

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. ArXiv:1312.3005. 68

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. ArXiv:1704.07556. 15, 39, 41

Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague. 2, 4

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *Proc. of ICLR*. 63

Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008. DeSRL: A linear-time semantic role labeling system. In *Proc. of CoNLL*. 31

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proc. of EMNLP*. 58

Kevin Clark and Christopher D Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proc. of EMNLP*. 51, 52

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proc. of ACL*. 51, 52

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537. 1, 15, 32, 35, 39, 41, 43

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proc. of LREC*. 6

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Proc. of NeurIPS*, pages 3079–3087. 58

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56. 6, 12, 47, 49

Dipanjan Das, AndrÃľ F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proc. of *SEM*. 45

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. ArXiv:1810.04805. 2, 6, 14, 58, 59, 73

Jesse Dunietz, Lori S. Levin, and Jaime G. Carbonell. 2017. Automatically tagging constructions of causation and their slot-fillers. *Transactions of the Association for Computational Linguistics*, 5:117–133. 55

Chris Dyer. 2016. Should neural network architecture reflect linguistic structure? 2

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*. 19, 25, 28, 30, 32

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL-HLT*. 6, 59, 63

Ahmad Emami and Frederick Jelinek. 2005. A neural syntactic language model. *Machine learning*, 60(1-3):195–227. 6, 63

Charles J. Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32. 36

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proc. of EMNLP*. 14, 40

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of EMNLP*. 12, 15, 19, 33, 35, 39, 41, 49

William R. Foland and James Martin. 2015. Dependency based semantic role labeling using convolutional neural networks. In *Proc. of *SEM*. 19

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F.

Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform. ArXiv:1803.07640. 69

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proc. of CoNLL*. 20, 32, 33, 34

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288. 19, 36

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proc. of ACL*. 6, 12

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proc. of NAACL*. 44

Tobias Glasmachers. 2017. Limits of end-to-end learning. Abs/1704.08305. 17

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420. 25

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. ArXiv:1901.05287. 59

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. 1

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer. 46

Alex Graves. 2013. Generating sequences with recurrent neural networks. ArXiv:1308.0850. 1, 25, 49

Normunds Gruzitis and Guntis Barzdins. 2016. The role of CNL and AMR in scalable abstractive summarization for multilingual media monitoring. 11

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *Proc. of NAACL*. 2, 77

Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13. 65

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CoNLL*. 18, 20, 28

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proc. of EMNLP*. 15, 39, 41

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *Proc. of EMNLP*. 14, 19, 40

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proc. of ACL*. 51

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whatâĂŹs next. In *Proc. of ACL*. 17, 38, 40, 46, 51, 52

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proc. of ACL*. 52

Marti A Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64. 2

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proc. of CoNLL*. 14, 19, 20, 31

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multi-lingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4). 14, 20, 21, 22, 23, 30, 32, 40

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*. 15, 41

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8). 25, 63

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of ccg derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3). 72, 74

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proc. of ACL*. 54, 58

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with Gumbel-softmax. In *Proc. of ICLR*. 63

Frederick Jelinek and John D. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–353. 6, 63

Richard Johansson. 2009. Statistical bistratal dependency parsing. In *Proc. of EMNLP*. 14, 19, 31

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with PropBank and NomBank. In *Proc. of CoNLL*. 31

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*. 65

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc. 59, 64

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proc. of AAAI*, AAAI'16. 63, 67

Diederik P. Kingma and Jimmy Ba. 2014. ADAM: A method for stochastic optimization.

ArXiV:1412.6980. 49

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Segmental Recurrent Neural Networks. In *Proc. of ICLR*. 38

Jayant Krishnamurthy and Tom M. Mitchell. 2014. Joint syntactic and semantic parsing with combinatory categorial grammar. In *Proc. of ACL*. 34

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 1

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proc. of NAACL*. 49, 50

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proc. of ACL*. 59

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*. 68, 69

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436. 1

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proc. of EMNLP*. 17, 46, 47, 48, 49, 51, 52

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proc. of ACL*. 52

Tao Lei, Yuan Zhang, Lluís Màrquez i Villodre, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proc. of NAACL*. 19, 33, 35

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373. 1

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proc. of ACL*. 61

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proc. of EMNLP*. 34

Junhui Li, Guodong Zhou, and Hwee Tou Ng. 2010. Joint syntactic and semantic parsing of Chinese. In *Proc. of ACL*. 34

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proc. of NAACL*. 63

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proc.*

of NAACL-HLT. 59, 61, 71, 72, 74

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *Transactions of the ACL*, 1:219–230. 14, 19, 34, 40

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *Proc. of CoNLL*. 14, 31, 40

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. ArXiv:1511.06114. 1, 15, 39, 41

Christopher D Manning. 2015. Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707. 2

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330. 3, 9, 14, 30, 60, 64, 70, 72, 74

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proc. of NeurIPS*, pages 6294–6305. 58, 71

David McClosky. 2010. Any domain parsing: automatic domain adaptation for natural language parsing. *Brown University*. 14

GÃąbor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *Proc. of ICLR*. 65

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. 65

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *Proc. of NAACL*. 30

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. ArXiv:1301.3781. 1, 40, 58

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of InterSpeech*. 62

Tom M Mitchell. 1980. The need for biases in learning generalizations. 5

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529. 1, 17

Phoebe Mulcaire, Swabha Swayamdipta, and Noah Smith. 2018. Polyglot semantic role labeling. In *Proc. of ACL*. 15

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proc. of ICML*. 27, 48

Jason Naradowsky, Sebastian Riedel, and David A. Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proc. of EMNLP*. 16, 40

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years.

In *Proc. of ACL*. 36

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553. 22

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proc. of ACL*. 25

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135. 30

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*, pages 63–72. 5, 20

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106. 4, 5, 11, 15, 17, 30, 36

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*. 15, 17, 39, 41

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A Smith. 2018a. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL*. 15, 41, 50

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A Smith. 2018b. Learning joint semantic parsers from disjoint data. In *Proc. of NAACL-HLT*. 16

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*. 1, 40, 48, 58

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*. 65, 68

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proc. of NAACL-HLT*. 6, 14, 38, 40, 52, 54, 56, 58, 62, 67, 69

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proc. of EMNLP*, pages 1499–1509. 59, 65, 67, 68, 69, 71

Matthew Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. 65

Steven Pinker. 1993. Language acquisition. *Foundations of Cognitive Neuroscience*, pages 359–399. 5

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proc. of CoNLL*. 39, 48, 51

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen

Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proc. of EMNLP-CoNLL*. 48

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287. 6, 12, 14

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. 2, 6, 14, 54, 58

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of ACL*. 2, 36, 55

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488. 5

Brian Roark. 2001. Probabilistic Top-Down parsing and language modeling. *Computational Linguistics*, 27(2):249–276. 6, 63

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *Proc. of EMNLP*. 2, 19, 33, 35

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018. Neural models of factuality. In *Proc. of ACL*. 72, 74

Jenny R Saffran. 2003. Statistical language learning: Mechanisms and constraints. *Current directions in psychological science*, 12(4):110–114. 2

Sunita Sarawagi, William W Cohen, et al. 2004. Semi-Markov conditional random fields for information extraction. In *Proc. of NeurIPS*, volume 17. 43, 45

Nathan Schneider, Jena D Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive supersense disambiguation of english prepositions and possessives. *arXiv:1805.04905*. 72, 74

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks. ArXiv prepint arXiv:1810.09536. 63

H.T. Siegelmann and E.D. Sontag. 1995. On the computational power of neural nets. *J. Comput. Syst. Sci.*, 50(1):132–150. 59

Natalia Silveira, Timothy Dozat, Marie-Catherine De Marneffe, Samuel R Bowman, Miriam Connor, John Bauer, and Christopher D Manning. 2014. A gold standard dependency corpus for english. In *LREC*, pages 2897–2904. 4, 72, 74

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*. 70, 71

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*. 40

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1). 48

Mark Steedman. 2000. Information structure and the syntax-phonology interface. *Linguistic Inquiry*, 31(4):649–689. 6

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proc. of ACL*. 69

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proc. of EMNLP*. 54, 59

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multitask learning. 76

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of CoNLL*. 11, 18, 28

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NeurIPS*. 1

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proc. of CoNLL*. 14, 19

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with Stack LSTMs. In *Proc. of CoNLL*. 8, 14, 20, 40, 75

Swabha Swayamdipta, Ankur P Parikh, and Tom Kwiatkowski. 2018a. Multi-mention learning for reading comprehension with neural cascades. In *Proc. of ICLR*. 12, 16

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold. ArXiV:1706.09528. 9, 39, 52, 75

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A Smith. 2018b. Syntactic scaffolds for semantic structures. In *Proc. of EMNLP*. 9, 39, 75

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the ACL*, 3:29–41. 2, 33, 35, 44, 47, 49

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proc. of AAAI*. 51

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *Proc. of ICLR*. 59, 61

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proc. of IJCAI*. 20, 31

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. of CoNLL*. 60, 64, 68, 74

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of NAACL*. Association for Computational Linguistics. 69, 70, 72, 74

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191. 12, 14, 19, 40

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of NAACL*. 70

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*, pages 5998–6008. 59, 63, 65, 67, 69

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NeurIPS*, pages 2773–2781. 2

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. In *Proc. of ICML*. 63

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. OntoNotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*. 39, 48, 70

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. OntoNotes Release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*. 69

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. of ACL*. 30

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proc. of EMNLP*. 1

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2017. Learning to parse from a semantic objective: It works. is it syntax? ArXiv:1709.01121. 2

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. In *Proc. of NAACL*. 12, 40, 51, 52

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259. 12, 39

David Wood, Jerome S. Bruner, and Gail Ross. 1976. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2). 38

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of EMNLP*. 12, 36

Bishan Yang and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proc. of EMNLP*. 12, 17, 40, 49, 50

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proc. of ACL*. 72, 74

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. ArXiv:1409.2329. 30

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI*. 16, 40

Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proc. of EMNLP*. 59

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*. 40

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proc. of CoNLL*. 32, 33, 34, 35

Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proc. of CoNLL*. 31

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of ACL*. 2, 17, 32, 35, 40, 43, 48, 51