Learning Joint Semantic Parsers from Disjoint Data

Hao Peng[♢] **Sam Thomson**[♣] **Swabha Swayamdipta**[♣] **Noah A. Smith**[♢]
[♢] Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA
[♣] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
{hapeng,nasmith}@cs.washington.edu, {sthomson,swabha}@cs.cmu.edu

Abstract

We present a new approach to learning semantic parsers from multiple datasets, even when the target semantic formalisms are drastically different, and the underlying corpora do not overlap. We handle such "disjoint" data by treating annotations for unobserved formalisms as latent structured variables. Building on state-of-the-art baselines, we show improvements both in frame-semantic parsing and semantic dependency parsing by modeling them jointly. Our code is open-source and available at https://github.com/Noahs-ARK/NeurboParser.

1 Introduction

Semantic parsing aims to automatically predict formal representations of meaning underlying natural language, and has been useful in question answering (Shen and Lapata, 2007), text-to-scene generation (Coyne et al., 2012), dialog systems (Chen et al., 2013) and social-network extraction (Agarwal et al., 2014), among others. Various formal meaning representations have been developed corresponding to different semantic theories (Fillmore, 1982; Palmer et al., 2005; Flickinger et al., 2012; Banarescu et al., 2013). The distributed nature of these efforts results in a set of annotated resources that are similar in spirit, but not strictly compatible. A major axis of structural divergence in semantic formalisms is whether based on *spans* (Baker et al., 1998; Palmer et al., 2005) or *dependencies* (Surdeanu et al., 2008; Oepen et al., 2014; Banarescu et al., 2013; Copestake et al., 2005, *inter alia*). Depending on application requirements, either might be most useful in a given situation.

Learning from a union of these resources seems promising, since more data almost always translates into better performance. This is indeed the case for two prior techniques—parameter sharing (FitzGerald et al., 2015; Kshirsagar et al., 2015), and joint decoding across multiple formalisms using cross-task factors that score combinations of substructures from each (Peng et al., 2017). Parameter sharing can be used in a wide range of multitask scenarios, when there is no data overlap or even any similarity between the tasks (Collobert and Weston, 2008; Søgaard and Goldberg, 2016). But techniques involving joint decoding have so far only been shown to work for parallel annotations of dependency-based formalisms, which are structurally very similar to each other (Lluís et al., 2013; Peng et al., 2017). Of particular interest is the approach of Peng et al., where three kinds of semantic graphs are jointly learned on the same input, using parallel annotations. However, as new annotation efforts cannot be expected to use the same original texts as earlier efforts, the utility of this approach is limited.

We propose an extension to Peng et al.'s formulation which addresses this limitation by considering **disjoint resources**, each containing only a single kind of annotation. Moreover, we consider **structurally divergent** formalisms, one dealing with semantic spans and the other with semantic

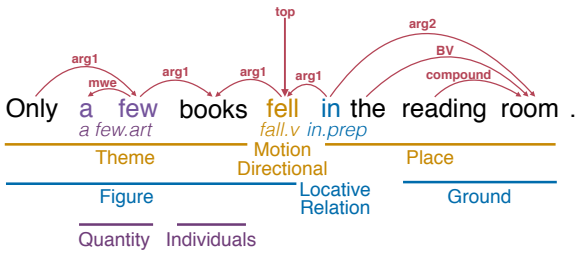

Figure 1: An example sentence from the FrameNet 1.5 corpus, shown with an author-annotated DM semantic dependency graph (above) and frame-semantic annotation (below). Two more gold frames (and their arguments) have been omitted for space.

dependencies. We experiment on frame-semantic parsing (Gildea and Jurafsky, 2002; Das et al., 2010), a span-based semantic role labeling (SRL) task (§2.1), and on a dependency-based minimum recursion semantic parsing (DELPH-IN MRS, or DM; Flickinger et al., 2012) task (§2.2). See Figure 1 for an example sentence with gold FrameNet annotations, and author-annotated DM representations.

Our joint inference formulation handles missing annotations by treating the structures that are not present in a given training example as latent variables (§3).[1] Specifically, semantic dependencies are treated as a collection of latent variables when training on FrameNet examples.

Using this latent variable formulation, we present an approach for relating spans and dependencies, by explicitly scoring affinities between pairs of potential spans and dependencies. Because there are a huge number of such pairs, we limit our consideration to only certain pairs—our design is inspired by the head rules of Surdeanu et al. (2008). Further possible span-dependency pairs are pruned using an ℓ_1-penalty technique adapted from sparse structure learning (§5). Neural network architectures are used to score frame-semantic structures, semantic dependencies, as well as cross-task structures (§4).

To summarize, our contributions include:

- using a latent variable formulation to extend cross-task scoring techniques to scenarios where datasets do not overlap;
- learning cross-task parts across structurally divergent formalisms; and
- using an ℓ_1-penalty technique to prune the space of cross task parts.

Our approach results in a new state-of-the-art in frame-semantic parsing, improving prior work by 0.8% absolute F_1 points (§6), and achieves competitive performance on semantic dependency parsing. Our code is available at `https://github.com/Noahs-ARK/NeurboParser`.

2 Tasks and Related Work

We describe the two tasks addressed in this work—frame-semantic parsing (§2.1) and semantic dependency parsing (§2.2)—and discuss how their structures relate to each other (§2.3).

2.1 Frame-Semantic Parsing

Frame-semantic parsing is a span-based task, under which certain words or phrases in a sentence evoke semantic **frames**. A **frame** is a group of events, situations, or relationships that all share the same set of participant and attribute types, called **frame elements** or **roles**. Gold supervision for frame-semantic parses comes from the FrameNet lexicon and corpus (Baker et al., 1998).

Concretely, for a given sentence, $\mathbf{x}$, a frame-semantic parse $\mathbf{y}$ consists of:

- a set of **targets**, each being a short span (usually a single token[2]) that evokes a frame;
- for each target t, the **frame** f that it evokes; and
- for each frame f, a set of non-overlapping **argument** spans in the sentence, each argument $a = (i, j, r)$ having a start token index i, end token index j and role label r.

The lemma and part-of-speech tag of a target comprise a **lexical unit** (or **LU**). The FrameNet ontology provides a mapping from an LU ℓ to the set of possible frames it could evoke, $\mathcal{F}_\ell$. Every frame $f \in \mathcal{F}_\ell$ is also associated with a set of roles, $\mathcal{R}_f$ under this ontology. For example, in Figure 1, the LU *"fall.v"* evokes the frame MOTION_DIRECTIONAL. The roles THEME and PLACE (which are specific to MOTION_DIRECTIONAL), are filled by the spans *"Only a few books"* and *"in the reading room"* respectively. LOCATIVE_RELATION has other roles (PROFILED_REGION, ACCESSIBILITY, DEIXIS, etc.) which are not realized in this sentence.

In this work, we assume gold targets and LUs are given, and parse each target independently, following the literature (Johansson and Nugues, 2007; FitzGerald et al., 2015; Yang and Mitchell, 2017; Swayamdipta et al., 2017, *inter alia*). Moreover, following Yang and Mitchell (2017), we perform frame and argument identification jointly. Most prior work has enforced the constraint that a role may be filled by at most one argument span, but following Swayamdipta et al. (2017) we do not impose this constraint, requiring only that arguments for the same target do not overlap.

[1] Following past work on support vector machines with latent variables (Yu and Joachims, 2009), we use the term "latent variable," even though the model is not probabilistic.

[2] 96.5% of targets in the training data are single tokens.

2.2 Semantic Dependency Parsing

Broad-coverage **semantic dependency parsing** (**SDP**; Oepen et al., 2014, 2015, 2016) represents sentential semantics with labeled bilexical dependencies. The SDP task mainly focuses on three semantic formalisms, which have been converted to dependency graphs from their original annotations. In this work we focus on only the DELPHIN MRS (**DM**) formalism.

Each semantic dependency corresponds to a labeled, directed edge between two words. A single token is also designated as the `top` of the parse, usually indicating the main predicate in the sentence. For example in Figure 1, the left-most arc has head *"Only"*, dependent *"few"*, and label `arg1`. In semantic dependencies, the head of an arc is analogous to the target in frame semantics, the destination corresponds to the argument, and the label corresponds to the role. The same set of labels are available for all arcs, in contrast to the frame-specific roles in FrameNet.

2.3 Spans vs. Dependencies

Early semantic role labeling was span-based (Gildea and Jurafsky, 2002; Toutanova et al., 2008, *inter alia*), with spans corresponding to syntactic constituents. But, as in syntactic parsing, there are sometimes theoretical or practical reasons to prefer dependency graphs. To this end, Surdeanu et al. (2008) devised heuristics based on syntactic head rules (Collins, 2003) to transform PropBank (Palmer et al., 2005) annotations into dependencies. Hence, for PropBank at least, there is a very direct connection (through syntax) between spans and dependencies.

For many other semantic representations, such a direct relationship might not be present. Some semantic representations are designed as graphs from the start (Hajič et al., 2012; Banarescu et al., 2013), and have no gold alignment to spans. Conversely, some span-based formalisms are not annotated with syntax (Baker et al., 1998; He et al., 2015),[3] and so head rules would require using (noisy and potentially expensive) predicted syntax.

Inspired by the head rules of Surdeanu et al. (2008), we design cross-task parts, without relying on gold or predicted syntax (which may be either unavailable or error-prone) or on heuristics.

3 Model

Given an input sentence $\mathbf{x}$, and target t with its LU ℓ, denote the set of valid frame-semantic parses (§2.1) as $\mathcal{Y}(\mathbf{x}, t, \ell)$, and valid semantic dependency parses as $\mathcal{Z}(\mathbf{x})$.[4] We learn a parameterized function S that scores candidate parses. Our goal is to *jointly* predict a frame-semantic parse and a semantic dependency graph by selecting the highest scoring candidates:

$$(\hat{\mathbf{y}}, \hat{\mathbf{z}}) = \underset{(\mathbf{y},\mathbf{z}) \in \mathcal{Y}(\mathbf{x},t,\ell) \times \mathcal{Z}(\mathbf{x})}{\arg\max} S(\mathbf{y}, \mathbf{z}, \mathbf{x}, t, \ell). \quad (1)$$

The overall score S can be decomposed into the sum of frame SRL score S_{f}, semantic dependency score S_{d}, and a cross-task score S_{c}:

$$S(\mathbf{y}, \mathbf{z}, \mathbf{x}, t, \ell) = S_{\text{f}}(\mathbf{y}, \mathbf{x}, t, \ell) + S_{\text{d}}(\mathbf{z}, \mathbf{x}) + S_{\text{c}}(\mathbf{y}, \mathbf{z}, \mathbf{x}, t, \ell). \quad (2)$$

S_{f} and S_{c} require access to the target and LU, in addition to $\mathbf{x}$, but S_{d} does not. For clarity, we omit the dependence on the input sentence, target, and lexical unit, whenever the context is clear. Below we describe how each of the scores is computed based on the individual **parts** that make up the candidate parses.

Frame SRL score. The score of a frame-semantic parse consists of

- the score for a predicate part, $s_f(p)$ where each predicate is defined as a combination of a target t, the associated LU, ℓ, and the frame evoked by the LU, $f \in \mathcal{F}_\ell$;
- the score for argument parts, $s_f(a)$, each associated with a token span and semantic role from $\mathcal{R}_f$.

Together, this results in a set of frame-semantic parts of size $O(n^2 \, |\mathcal{F}_\ell| \, |\mathcal{R}_f|)$.[5] The score for a frame semantic structure $\mathbf{y}$ is the sum of local scores of parts in $\mathbf{y}$:

$$S_{\text{f}}(\mathbf{y}) = \sum_{y_i \in \mathbf{y}} s_{\text{f}}(y_i). \quad (3)$$

The computation of s_{f} is described in §4.2.

[3] In FrameNet, phrase types of arguments and their grammatical function in relation to their target have been annotated. But in order to apply head rules, the *internal* structure of arguments (or at least their semantic heads) would also require syntactic annotations.

[4] For simplicity, we consider only a single target here; handling of multiple targets is discussed in §6.

[5] With pruning (described in §6) we reduce this to a number of parts linear in n. Also, $|\mathcal{F}_\ell|$ is usually small (averaging 1.9), as is $|\mathcal{R}_f|$ (averaging 9.5).

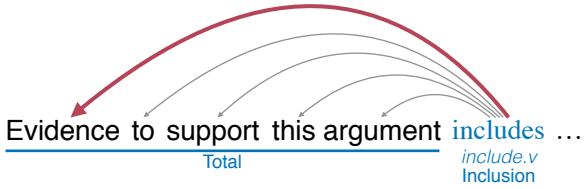

Figure 2: An example of cross-task parts from the FrameNet 1.5 development set. We enumerate all unlabeled semantic dependencies from the first word of the target (*includes*) to any token inside the span. The red bolded arc indicates the prediction of our model.

Semantic dependency score. Following Martins and Almeida (2014), we consider three types of parts in a semantic dependency graph: semantic heads, unlabeled semantic arcs, and labeled semantic arcs. Analogous to Equation 3, the score for a dependency graph $\mathbf{z}$ is the sum of local scores:

$$S_{\mathrm{d}}(\mathbf{z}) = \sum_{z_j \in \mathbf{z}} s_{\mathrm{d}}(z_j), \tag{4}$$

The computation of s_{d} is described in §4.3.

Cross task score. In addition to task-specific parts, we introduce a set $\mathcal{C}$ of cross-task parts. Each cross-task part relates an argument part from $\mathbf{y}$ to an *unlabeled* dependency arc from $\mathbf{z}$. Based on the head-rules described in §2.3, we consider unlabeled arcs from the target to any token inside the span.[6] Intuitively, an argument in FrameNet *would* be converted into a dependency from its target to the semantic head of its span. Since we do not know the semantic head of the span, we consider all tokens in the span as potential modifiers of the target. Figure 2 shows examples of cross-task parts. The cross-task score is given by

$$S_{\mathrm{c}}(\mathbf{y}, \mathbf{z}) = \sum_{(y_i, z_j) \in (\mathbf{y} \times \mathbf{z}) \cap \mathcal{C}} s_{\mathrm{c}}(y_i, z_j). \tag{5}$$

The computation of s_{c} is described in §4.4.

In contrast to previous work (Lluís et al., 2013; Peng et al., 2017), where there are parallel annotations for all formalisms, our input sentences contain only one of the two—either the span-based frame SRL annotations, or semantic dependency graphs from DM. To handle missing annotations, we treat semantic dependencies $\mathbf{z}$ as latent when decoding frame-semantic structures.[7] Because the DM dataset we use does not have target annotations, we do not use latent variables for frame semantic structures when predicting semantic dependency graphs. The parsing problem here reduces to

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z} \in \mathcal{Z}} S_{\mathrm{d}}(\mathbf{z}), \tag{6}$$

in contrast with Equation 1 .

4 Parameterizations of Scores

This section describes the parametrization of the scoring functions from §3. At a very high level: we learn contextualized token and span vectors using a bidirectional LSTM (biLSTM; Graves, 2012) and multilayer perceptrons (MLPs) (§4.1); we learn lookup embeddings for LUs, frames, roles, and arc labels; and to score a part, we combine the relevant representations into a single scalar score using a (learned) low-rank multilinear mapping. Scoring frames and arguments is detailed in §4.2, that of dependency structures in §4.3, and §4.4 shows how to capture interactions between arguments and dependencies. All parameters are learned jointly, through the optimization of a multitask objective (§5).

Tensor notation. The **order** of a tensor is the number of its dimensions—an order-2 tensor is a matrix and an order-1 tensor is a vector. Let $\otimes$ denote tensor product; the tensor product of two order-2 tensors $\mathcal{A}$ and $\mathcal{B}$ yields an order-4 tensor where $(\mathcal{A} \otimes \mathcal{B})_{i,j,k,l} = \mathcal{A}_{i,j}\mathcal{B}_{k,l}$. We use $\langle \cdot, \cdot \rangle$ to denote inner products.

4.1 Token and Span Representations

The representations of tokens and spans are formed using biLSTMs followed by MLPs.

Contextualized token representations. Each token in the input sentence $\mathbf{x}$ is mapped to an embedding vector. Two LSTMs (Hochreiter and Schmidhuber, 1997) are run in opposite directions over the input vector sequence. We use the concatenation of the two hidden representations at each position i as a contextualized word embedding for each token:

$$\mathbf{h}_i = \left[\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i\right]. \tag{7}$$

[6] Most targets are single-words (§2.1). For multi-token targets, we consider only the first token, which is usually content-bearing.

[7] Semantic dependency parses over a sentence are not constrained to be identical for different frame-semantic targets.

Span representations. Following Lee et al. (2017), span representations are computed based on boundary word representations and discrete length and distance features. Concretely, given a target t and its associated argument $a = (i, j, r)$ with boundary indices i and j, we compute three features $\boldsymbol{\phi}_t(a)$ based on the length of a, and the distances from i and j to the start of t. We concatenate the token representations at a's boundary with the discrete features $\boldsymbol{\phi}_t(a)$. We then use a two-layer $\tanh$-MLP to compute the span representation:

$$\mathbf{g}^{\text{span}}(i, j) = \text{MLP}^{\text{span}}\left([\mathbf{h}_i; \mathbf{h}_j; \boldsymbol{\phi}_t(a)]\right). \quad (8)$$

The target representation $\mathbf{g}^{\text{tgt}}(t)$ is similarly computed using a separate MLP^{tgt}, with a length feature but no distance features.

4.2 Frame and Argument Scoring

As defined in §3, the representation for a predicate part incorporates representations of a target span, the associated LU and the frame evoked by the LU. The score for a predicate part is given by a multilinear mapping:

$$\mathbf{g}^{\text{pred}}(f) = \mathbf{g}^{\text{fr}}(f) \otimes \mathbf{g}^{\text{tgt}}(t) \otimes \mathbf{g}^{\text{lu}}(\ell) \quad (9a)$$

$$s_{\text{f}}(p) = \left\langle \boldsymbol{\mathcal{W}}, \mathbf{g}^{\text{pred}}(f) \right\rangle, \quad (9b)$$

where $\boldsymbol{\mathcal{W}}$ is a low-rank order-3 tensor of learned parameters, and $\mathbf{g}^{\text{fr}}(f)$ and $\mathbf{g}^{\text{lu}}(\ell)$ are learned lookup embeddings for the frame and LU.

A candidate argument consists of a span and its role label, which in turn depends on the frame, target and LU. Hence the score for argument part, $a = (i, j, r)$ is given by extending definitions from Equation 9:

$$\mathbf{g}^{\text{arg}}(a) = \mathbf{g}^{\text{span}}(i, j) \otimes \mathbf{g}^{\text{role}}(r), \quad (10a)$$

$$s_{\text{f}}(a) = \left\langle \boldsymbol{\mathcal{W}} \otimes \boldsymbol{\mathcal{U}}, \mathbf{g}^{\text{pred}}(f) \otimes \mathbf{g}^{\text{arg}}(a) \right\rangle, \quad (10b)$$

where $\boldsymbol{\mathcal{U}}$ is a low-rank order-2 tensor of learned parameters and $\mathbf{g}^{\text{role}}(r)$ is a learned lookup embedding of the role label.

4.3 Dependency Scoring

Local scores for dependencies are implemented with two-layer $\tanh$-MLPs, followed by a final linear layer reducing the represenation to a single scalar score. For example, let $u = i \rightarrow j$ denote an unlabeled arc (ua). Its score is:

$$\mathbf{g}^{\text{ua}}(u) = \text{MLP}^{\text{ua}}\left([\mathbf{h}_i; \mathbf{h}_j]\right) \quad (11a)$$

$$s_{\text{d}}(u) = \mathbf{w}^{\text{ua}} \cdot \mathbf{g}^{\text{ua}}(u), \quad (11b)$$

where $\mathbf{w}^{\text{ua}}$ is a vector of learned weights. The scores for other types of parts are computed similarly, but with separate MLPs and weights.

4.4 Cross-Task Part Scoring

As shown in Figure 2, each cross-task part c consists of two first-order parts: a frame argument part a, and an unlabeled dependency part, u. The score for a cross-task part incorporates both:

$$s_{\text{c}}(c) = \left\langle \boldsymbol{\mathcal{W}} \otimes \boldsymbol{\mathcal{U}} \otimes \boldsymbol{\mathcal{V}}, \mathbf{g}^{\text{pred}}(f) \otimes \mathbf{g}^{\text{arg}}(a) \otimes \mathbf{w}^{\text{ua}} \otimes \mathbf{g}^{\text{ua}}(u) \right\rangle, \quad (12)$$

where $\boldsymbol{\mathcal{V}}$ is a low-rank order-2 tensor of parameters. Following previous work (Lei et al., 2014; Peng et al., 2017), we construct the parameter tensors $\boldsymbol{\mathcal{W}}$, $\boldsymbol{\mathcal{U}}$, and $\boldsymbol{\mathcal{V}}$ so as to upper-bound their ranks.

5 Training and Inference

All parameters from the previous sections are trained using a max-margin training objective (§5.1). For inference, we use a linear programming procedure, and a sparsity-promoting penalty term for speeding it up (§5.2).

5.1 Max-Margin Training

Let $\mathbf{y}^*$ denote the gold frame-semantic parse, and let $\delta(\mathbf{y}, \mathbf{y}^*)$ denote the cost of predicting $\mathbf{y}$ with respect to $\mathbf{y}^*$. We optimize the latent structured hinge loss (Yu and Joachims, 2009), which gives a subdifferentiable upper-bound on δ:

$$\mathcal{L}(\mathbf{y}^*) = \max_{(\mathbf{y}, \mathbf{z}) \in \mathcal{Y} \times \mathcal{Z}} \{S(\mathbf{y}, \mathbf{z}) + \delta(\mathbf{y}, \mathbf{y}^*)\} - \max_{\mathbf{z} \in \mathcal{Z}} \{S(\mathbf{y}^*, \mathbf{z})\}. \quad (13)$$

Following Martins and Almeida (2014), we use a weighted Hamming distance as the cost function, where, to encourage recall, we use costs 0.6 for false negative predictions and 0.4 for false positives. Equation 13 can be evaluated by applying the same max-decoding algorithm twice—once with cost-augmented inference (Crammer et al., 2006), and once more keeping $\mathbf{y}^*$ fixed. Training then aims to minimize the average loss over all training instances.[8]

Another potential approach to training a model on disjoint data would be to marginalize out the

[8] We do not use latent frame structures when decoding semantic dependency graphs (§3). Hence, the loss reduces to structured hinge (Tsochantaridis et al., 2004) when training on semantic dependencies.

latent structures and optimize the conditional log-likelihood (Naradowsky et al., 2012). Although max-decoding and computing marginals are both NP-hard in general graphical models, there are more efficient off-the-shelf implementations for approximate max-decoding, hence, we adopt a max-margin formulation.

5.2 Inference

We formulate the maximizations in Equation 13 as 0–1 integer linear programs and use AD^3 to solve them (Martins et al., 2011). We only enforce a non-overlapping constraint when decoding FrameNet structures, so that the argument identification subproblem can be efficiently solved by a dynamic program (Kong et al., 2016; Swayamdipta et al., 2017). When decoding semantic dependency graphs, we enforce the determinism constraint (Flanigan et al., 2014), where certain labels may appear on at most one arc outgoing from the same token.

Inference speedup by promoting sparsity. As discussed in §3, even after pruning, the number of within-task parts is linear in the length of the input sentence, so the number of cross-task parts is quadratic. This leads to potentially very slow inference. We address this problem by imposing an ℓ_1 penalty on the cross-task part scores:

$$\mathcal{L}(\mathbf{y}^*) + \lambda \sum_{(y_i, z_j) \in \mathcal{C}} \left| s_c(y_i, z_j) \right|, \qquad (14)$$

where λ is a hyperparameter, set to 0.01 as a practical tradeoff between efficiency and development set performance. Whenever the score for a cross-task part is driven to zero, that part's score no longer needs to be considered during inference. It is important to note that by promoting sparsity this way, we do not prune out any candidate solutions. We are instead encouraging fewer terms in the scoring function, which leads to smaller, faster inference problems even though the space of feasible parses is unchanged.

The above technique is closely related to a line of work in estimating the structure of sparse graphical models (Yuan and Lin, 2007; Friedman et al., 2008), where an ℓ_1 penalty is applied to the inverse covariance matrix in order to induce a smaller number of conditional dependencies between variables. To the best of our knowledge, we are the first to apply this technique to the output of neural scoring functions. Here, we are interested in learning sparse graphical models only because they result in faster inference, not because we have any *a priori* belief about sparsity. This results in roughly a $14\times$ speedup in our experiments, without any significant drop in performance.

	Train	Exemplars	Dev.	Test
FN 1.5	17,143	153,952	2,333	4,457
FN 1.7	19,875	192,460	2,308	6,722
DM id	33,961	-	1,692	1,410
DM ood	-	-	-	1,849

Table 1: Number of instances in datasets.

6 Experiments

Datasets. Our model is evaluated on two different releases of FrameNet: FN 1.5 and FN 1.7,[9] using splits from Swayamdipta et al. (2017). Following Swayamdipta et al. (2017) and Yang and Mitchell (2017), each target annotation is treated as a separate training instance. We also include as training data the exemplar sentences, each annotated for a single target, as they have been reported to improve performance (Kshirsagar et al., 2015; Yang and Mitchell, 2017). For semantic dependencies, we use the English DM dataset from the SemEval 2015 Task 18 closed track (Oepen et al., 2015).[10] DM contains instances from the WSJ corpus for training and both in-domain (id) and out-of-domain (ood) test sets, the latter from the Brown corpus.[11] Table 1 summarizes the sizes of the datasets.

Baselines. We compare FN performance of our joint learning model (FULL) to two baselines:

BASIC: A single-task frame SRL model, trained using a structured hinge objective.

NOCTP: A joint model without cross-task parts. It demonstrates the effect of sharing parameters in word embeddings and LSTMs (like in FULL). It does not use latent semantic dependency structures, and aims to minimize the sum of training losses from both tasks.

We also compare semantic dependency parsing performance against the single task model by Peng

[9] https://FN.icsi.berkeley.edu/fndrupal/

[10] http://sdp.delph-in.net/. The closed track does not have access to any syntactic analyses. The impact of syntactic features on SDP performance is extensively studied in Ribeyre et al. (2015).

[11] Our FN training data does not overlap with the DM test set. We remove the 3 training sentences from DM which appear in FN test data.

Model	Prec.	Rec.	F_1
Roth	72.2	68.0	70.0
Täckström	75.4	65.8	70.3
FitzGerald	74.8	65.5	69.9
FitzGerald (10×)	75.0	67.3	70.9
open-SESAME	71.0	67.8	69.4
open-SESAME (5×)	71.2	70.5	70.9
Yang and Mitchell (REL)	77.1	68.7	72.7
†*Yang and Mitchell (ALL)	78.8	74.5	76.6
†This work (FULL)	80.4	73.5	76.8
†This work (FULL, 2×)	**80.4**	**74.7**	**77.4**
†This work (BASIC)	79.2	71.7	75.3
†This work (NOCTP)	76.9	74.8	75.8

Table 2: FN 1.5 full structure extraction test performance. † denotes the models jointly predicting frames and arguments, and other systems implement two-stage pipelines and use the algorithm by Hermann et al. (2014) to predict frames. $K\times$ denotes a product-of-experts ensemble of K models. *Ensembles a sequential tagging CRF and a relational model. Bold font indicates best performance among all systems.

et al. (2017), denoted as NeurboParser (BASIC). To ensure fair comparison with our FULL model, we made several modifications to their implementation (§6.3). We observed performance improvements from our reimplementation, which can be seen in Table 5.

Pruning strategies. For frame SRL, we discard argument spans longer than 20 tokens (Swayamdipta et al., 2017). We further pretrain an unlabeled model and prune spans with posteriors lower than $1/n^2$, with n being the input sentence length. For semantic dependencies, we generally follow Martins and Almeida (2014), replacing their feature-rich pruner with neural networks. We observe that $O(n)$ spans/arcs remain after pruning, with around 96% FN development recall, and more than 99% for DM.[12]

6.1 Empirical Results

FN parsing results. Table 2 compares our full frame-semantic parsing results to previous systems. Among them, Täckström et al. (2015) and Roth (2016) implement a two-stage pipeline and use the method from Hermann et al. (2014) to predict frames. FitzGerald et al. (2015) uses the same pipeline formulation, but improves the frame identification of Hermann et al. (2014) with better syntactic features. open-SESAME (Swayamdipta et al., 2017) uses predicted frames from FitzGerald et al. (2015), and improves argument identification using a softmax-margin segmental RNN. They observe further improvements from product of experts ensembles (Hinton, 2002).

The best published FN 1.5 results are due to Yang and Mitchell (2017). Their relational model (REL) formulates argument identification as a sequence of local classifications. They additionally introduce an ensemble method (denoted as ALL) to integrate the predictions of a sequential CRF. They use a linear program to jointly predict frames and arguments at test time. As shown in Table 2, our single-model performance outperforms their REL model, and is on par with their ALL model. For a fair comparison, we build an ensemble (FULL, 2×) by separately training two models, differing only in random seeds, and averaging their part scores. Our ensembled model outperforms previous best results by 0.8% absolute.

Table 3 compares our frame identification results with previous approaches. Hermann et al. (2014) and Hartmann et al. (2017) use distributed word representations and syntax features. We follow the FULL LEXICON setting (Hermann et al., 2014) and extract candidate frames from the offi-

Model	All	Ambiguous
Hartmann	87.6	-
Yang and Mitchell	88.2	-
Hermann	88.4	73.1
†This work (BASIC)	89.2	76.3
†This work (NOCTP)	89.2	76.4
†This work (FULL)	89.9	77.7
†This work (FULL, 2×)	**90.0**	**78.0**

Table 3: Frame identification accuracy on the FN 1.5 test set. *Ambiguous* evaluates only on lexical units having more than one possible frames. † denotes joint frame and argument identification, and bold font indicates best performance.[13]

[12] On average, around $0.8n$ argument spans, and $5.7n$ unlabeled dependency arcs remain after pruning.

[13] Our comparison to Hermann et al. (2014) is based on their updated version: http://www.aclweb.org/anthology/P/P14/P14-1136v2.pdf. Ambiguous frame identification results by Yang and Mitchell (2017) and Hartmann et al. (2017) are 75.7 and 73.8. Their ambiguous lexical unit sets are different from the one extracted from the official frame directory, and thus the results are not comparable to those in Table 3.

	Full Structure			Frame Id.	
Model	**Prec.**	**Rec.**	F_1	**All**	**Amb.**
BASIC	78.0	72.1	75.0	88.6	76.6
NOCTP	79.8	72.4	75.9	88.5	76.3
FULL	**80.2**	**72.9**	**76.4**	**89.1**	**77.5**

Table 4: FN 1.7 full structure extraction and frame identification test results. Bold font indicates best performance. FN 1.7 test set is an extension of FN 1.5 test, hence the results here are not comparable to those reported in Table 2.

Model	**id F_1**	**ood F_1**
NeurboParser (BASIC)	89.4	84.5
NeurboParser (FREDA3)	90.4	85.3
NeurboParser (BASIC, reimpl.)	90.0	84.6
This work (NOCTP)	89.9	85.2
This work (FULL)	90.5	85.9
This work (FULL, 2×)	**91.2**	**86.6**

Table 5: Labeled parsing performance in F_1 score for DM semantic dependencies. *id* denotes in-domain WSJ test data, and *ood* denotes out-of-domain brown corpus test data. Bold font indicates best performance.

cial directories. The *Ambiguous* setting compares lexical units with more than one possible frames. Our approach improves over all previous models under both settings, demonstrating a clear benefit from joint learning.

We observe similar trends on FN 1.7 for both full structure extraction and for frame identification only (Table 4). FN 1.7 extends FN 1.5 with more consistent annotations. Its test set is different from that of FN 1.5, so the results are not directly comparable to Table 2. We are the first to report frame-semantic parsing results on FN 1.7, and we encourage future efforts to do so as well.

Semantic dependency parsing results. Table 5 compares our semantic dependency parsing performance on DM with the baselines. Our reimplementation of the BASIC model slightly improves performance on in-domain test data. The NOCTP model ties parameters from word embeddings and LSTMs when training on FrameNet and DM, but does not use cross-task parts or joint prediction. NOCTP achieves similar in-domain test performance, and improves over BASIC on out-of-domain data. By jointly predicting FrameNet structures and semantic dependency graphs, the FULL model outperforms the baselines by more than 0.6% absolute F_1 scores under both settings.

Previous state-of-the-art results on DM are due to the joint learning model of Peng et al. (2017), denoted as NeurboParser (FREDA3). They adopted a multitask learning approach, jointly predicting three different parallel semantic dependency annotations. Our FULL model's in-domain test performance is on par with FREDA3, and improves over it by 0.6% absolute F_1 on out-of-domain test data. Our ensemble of two FULL models achieves a new state-of-the-art in both in-domain and out-of-domain test performance.

		Rel. Err. (%)	
Operation	**Description**	BASIC	FULL
Frame error	Frame misprediction.	11.3	11.1
Role error	Matching span with incorrect role.	12.6 (5.2)	13.4 (5.9)
Span error	Matching role with incorrect span.	11.4	12.3
Arg. error	Predicted argument does not overlap with any gold span.	18.6	22.4
Missing arg.	Gold argument does not overlap with any predicted span.	43.5	38.0

Table 6: Percentage of errors made by BASIC and FULL models on the FN 1.5 development set. Parenthesized numbers show the percentage of role errors when frame predictions are correct.

6.2 Analysis

Error type breakdown. Similarly to He et al. (2017), we categorize prediction errors made by the BASIC and FULL models in Table 6. Entirely missing an argument accounts for most of the errors for both models, but we observe fewer errors by FULL compared to BASIC in this category. FULL tends to predict more arguments in general, including more incorrect arguments.

Since candidate roles are determined by frames, frame and role errors are highly correlated. Therefore, we also show the role errors when frames are correctly predicted (parenthesized numbers in the second row). When a predicted argument span matches a gold span, predicting the semantic role is less challenging. Role errors account for only around 13% of all errors, and half of them are due to mispredictions of frames.

Performance by argument length. Figure 3 plots dev. precision and recall of both BASIC and FULL against binned argument lengths. We ob-

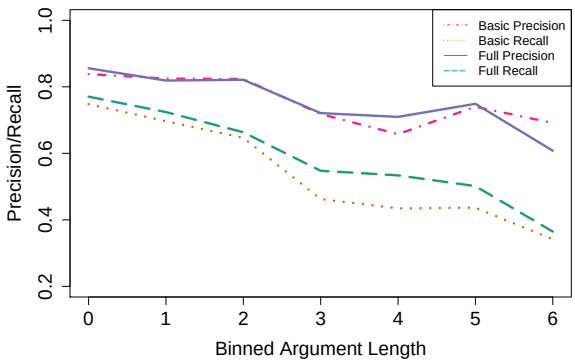

Figure 3: FN 1.5 development precision and recall of BASIC and FULL by different argument lengths. Length ℓ is binned to $\lfloor \log_{1.6} \ell \rfloor$, and precision/recall values are smoothed with `loess`, with a smoothing parameter of 0.1.

serve two trends: (a) FULL tends to predict longer arguments (averaging 3.2) compared to BASIC (averaging 2.9), while keeping similar precision;[14] (b) recall improvement in FULL mainly comes from arguments longer than 4.

6.3 Implementation Details

Our implementation is based on DyNet (Neubig et al., 2017).[15] We use predicted part-of-speech tags and lemmas using NLTK (Bird et al., 2009).[16]

Parameters are optimized with stochastic subgradient descent for up to 30 epochs, with ℓ_2 norms of gradients clipped to 1. We use 0.33 as initial learning rate, and anneal it at a rate of 0.5 every 10 epochs. Early stopping is applied based on FN development F_1. We apply logarithm with base 2 to all discrete features, e.g., $\log_2(d+1)$ for distance feature valuing d. To speed up training, we randomly sample a 35% subset from the FN exemplar instances each epoch.

Hyperparameters. Each input token is represented as the concatenation a word embedding vector, a learned lemma vector, and a learned vector for part-of speech, all updated during training. We use 100-dimensional `GloVe` (Pennington et al., 2014) to initialize word embeddings. We apply word dropout (Iyyer et al., 2015) and randomly replace a word w with a special `UNK` symbol with probability $\frac{\alpha}{1+\#(w)}$, with $\#(w)$ being the count of w in the training set. We follow the default parameters initialization procedure by DyNet, and an ℓ_2 penalty of 10^{-6} is applied to all weights. See Table 7 for other hyperparameters.

Hyperparameter	Value
Word embedding dimension	100 (32)
Lemma embedding dimension	50 (16)
POS tag embedding dimension	50 (16)
MLP dimension	100 (32)
Tensor rank r	100 (32)
BiLSTM layers	2 (1)
BiLSTM dimensions	200 (64)
α for word dropout	1.0 (1.0)

Table 7: Hyperparameters used in the experiments. Parenthesized numbers indicate those used by the pretrained pruners.

Modifications to Peng et al. (2017). To ensure fair comparisons, we note two implementation modifications to Peng et al.'s basic model. We use a more recent version (2.0) of the DyNet toolkit, and we use 50-dimensional lemma embeddings instead of their 25-dimensional randomly-initialized learned word embeddings.

7 Conclusion

We presented a novel multitask approach to learning semantic parsers from disjoint corpora with structurally divergent formalisms. We showed how joint learning and prediction can be done with scoring functions that explicitly relate spans and dependencies, even when they are never observed together in the data. We handled the resulting inference challenges with a novel adaptation of graphical model structure learning to the deep learning setting. We raised the state-of-the-art on DM and FrameNet parsing by learning from both, despite their structural differences and non-overlapping data. While our selection of factors is specific to spans and dependencies, our general techniques could be adapted to work with more combinations of structured prediction tasks. We have released our implementation at `https://github.com/Noahs-ARK/NeurboParser`.

Acknowledgments

We thank Kenton Lee, Luheng He, and Rowan Zellers for their helpful comments, and the anonymous reviewers for their valuable feedback. This work was supported in part by NSF grant IIS-1562364.

[14] Average gold span length is 3.4 after discarding those longer than 20.

[15] `https://github.com/clab/dynet`

[16] `http://www.nltk.org/`

References

Apoorv Agarwal, Sriramkumar Balasubramanian, Anup Kotalwar, Jiehan Zheng, and Owen Rambow. 2014. Frame semantic tree kernels for social network extraction from text. In *Proc. of EACL*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. ACL*.

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

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. " O'Reilly Media, Inc.".

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Proc. of ASRU-IEEE*.

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. ICML*.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language & Computation* 3(4):281–332.

Bob Coyne, Alex Klapheke, Masoud Rouhizadeh, Richard Sproat, and Daniel Bauer. 2012. Annotation tools and knowledge representation for a text-to-scene system. In *Proc. of COLING*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR* 7:551–585.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proc. of NAACL*.

Charles Fillmore. 1982. Frame semantics. *Linguistics in the morning calm* pages 111–137.

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

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. ACL*.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: A dynamically annotated treebank of the Wall Street Journal. In *Proc. of TLT*. pages 85–96.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9(3):432–441.

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

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

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proc. of LREC*.

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. 2017. Out-of-domain FrameNet semantic role labeling. In *Proc. of EACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proc. of ACL*.

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

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proc. of ACL*.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Richard Johansson and Pierre Nugues. 2007. LTH: Semantic structure extraction using nonprojective dependency trees. In *Proc. of SemEval*.

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

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

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

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proc. ACL*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *TACL* 1:219–230.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proc. of SemEval*.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

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

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*.

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*.

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

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

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

Corentin Ribeyre, Éric Villemonte De La Clergerie, and Djamé Seddah. 2015. Because Syntax does Matter: Improving Predicate-Argument Structures Parsing Using Syntactic Features. In *Proc. of NAACL*.

Michael Roth. 2016. Improving frame semantic parsing via dependency path embeddings. In *Book of Abstracts of the 9th International Conference on Construction Grammar*.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proc. of EMNLP-CoNLL*.

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

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*.

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.

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

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

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

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

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proc. of ICML*.

Ming Yuan and Yi Lin. 2007. Model selection and estimation in the gaussian graphical model. *Biometrika* 94(1):19–35.