Multitask Parsing Across Semantic Representations

Daniel Hershcovich[1,2] **Omri Abend**[2] **Ari Rappoport**[2]

[1]The Edmond and Lily Safra Center for Brain Sciences
[2]School of Computer Science and Engineering
Hebrew University of Jerusalem
`{danielh,oabend,arir}@cs.huji.ac.il`

Abstract

The ability to consolidate information of different types is at the core of intelligence, and has tremendous practical value in allowing learning for one task to benefit from generalizations learned for others. In this paper we tackle the challenging task of improving semantic parsing performance, taking UCCA parsing as a test case, and AMR, SDP and Universal Dependencies (UD) parsing as auxiliary tasks. We experiment on three languages, using a uniform transition-based system and learning architecture for all parsing tasks. Despite notable conceptual, formal and domain differences, we show that multitask learning significantly improves UCCA parsing in both in-domain and out-of-domain settings. Our code is publicly available.[1]

1 Introduction

Semantic parsing has arguably yet to reach its full potential in terms of its contribution to downstream linguistic tasks, partially due to the limited amount of semantically annotated training data. This shortage is more pronounced in languages other than English, and less researched domains.

Indeed, recent work in semantic parsing has targeted, among others, Abstract Meaning Representation (AMR; Banarescu et al., 2013), bilexical Semantic Dependencies (SDP; Oepen et al., 2016) and Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013). While these schemes are formally different and focus on different distinctions, much of their semantic content is shared (Abend and Rappoport, 2017).

Multitask learning (MTL; Caruana, 1997) allows exploiting the overlap between tasks to ef-

fectively extend the training data, and has greatly advanced with neural networks and representation learning (see §2). We build on these ideas and propose a general transition-based DAG parser, able to parse UCCA, AMR, SDP and UD (Nivre et al., 2016). We train the parser using MTL to obtain significant improvements on UCCA parsing over single-task training in (1) in-domain and (2) out-of-domain settings in English; (3) an in-domain setting in German; and (4) an in-domain setting in French, where training data is scarce.

The novelty of this work is in proposing a general parsing and learning architecture, able to accommodate such widely different parsing tasks, and in leveraging it to show benefits from learning them jointly.

2 Related Work

MTL has been used over the years for NLP tasks with varying degrees of similarity, examples including joint classification of different arguments in semantic role labeling (Toutanova et al., 2005), and joint parsing and named entity recognition (Finkel and Manning, 2009). Similar ideas, of parameter sharing across models trained with different datasets, can be found in studies of domain adaptation (Blitzer et al., 2006; Daume III, 2007; Ziser and Reichart, 2017). For parsing, domain adaptation has been applied successfully in parser combination and co-training (McClosky et al., 2010; Baucom et al., 2013).

Neural MTL has mostly been effective in tackling formally similar tasks (Søgaard and Goldberg, 2016), including multilingual syntactic dependency parsing (Ammar et al., 2016; Guo et al., 2016), as well as multilingual (Duong et al., 2017), and cross-domain semantic parsing (Herzig and Berant, 2017; Fan et al., 2017).

Sharing parameters with a low-level task has

---

[1] http://github.com/danielhers/tupa

shown great benefit for transition-based syntactic parsing, when jointly training with POS tagging (Bohnet and Nivre, 2012; Zhang and Weiss, 2016), and with lexical analysis (Constant and Nivre, 2016; More, 2016). Recent work has achieved state-of-the-art results in multiple NLP tasks by jointly learning the tasks forming the NLP standard pipeline using a single neural model (Collobert et al., 2011; Hashimoto et al., 2017), thereby avoiding cascading errors, common in pipelines.

Much effort has been devoted to joint learning of syntactic and semantic parsing, including two CoNLL shared tasks (Surdeanu et al., 2008; Hajič et al., 2009). Despite their conceptual and practical appeal, such joint models rarely outperform the pipeline approach (Lluís and Màrquez, 2008; Henderson et al., 2013; Lewis et al., 2015; Swayamdipta et al., 2016, 2017).

Peng et al. (2017a) performed MTL for SDP in a closely related setting to ours. They tackled three tasks, annotated over the same text and sharing the same formal structures (bilexical DAGs), with considerable edge overlap, but differing in target representations (see §3). For all tasks, they reported an increase of 0.5-1 labeled F_1 points. Recently, Peng et al. (2018) applied a similar approach to joint frame-semantic parsing and semantic dependency parsing, using disjoint datasets, and reported further improvements.

3 Tackled Parsing Tasks

In this section, we outline the parsing tasks we address. We focus on representations that produce full-sentence analyses, i.e., produce a graph covering all (content) words in the text, or the lexical concepts they evoke. This contrasts with "shallow" semantic parsing, primarily semantic role labeling (SRL; Gildea and Jurafsky, 2002; Palmer et al., 2005), which targets argument structure phenomena using flat structures. We consider four formalisms: UCCA, AMR, SDP and Universal Dependencies. Figure 1 presents one sentence annotated in each scheme.

Universal Conceptual Cognitive Annotation. UCCA (Abend and Rappoport, 2013) is a semantic representation whose main design principles are ease of annotation, cross-linguistic applicability, and a modular architecture. UCCA represents the semantics of linguistic utterances as directed acyclic graphs (DAGs), where terminal (childless) nodes correspond to the text tokens, and non-
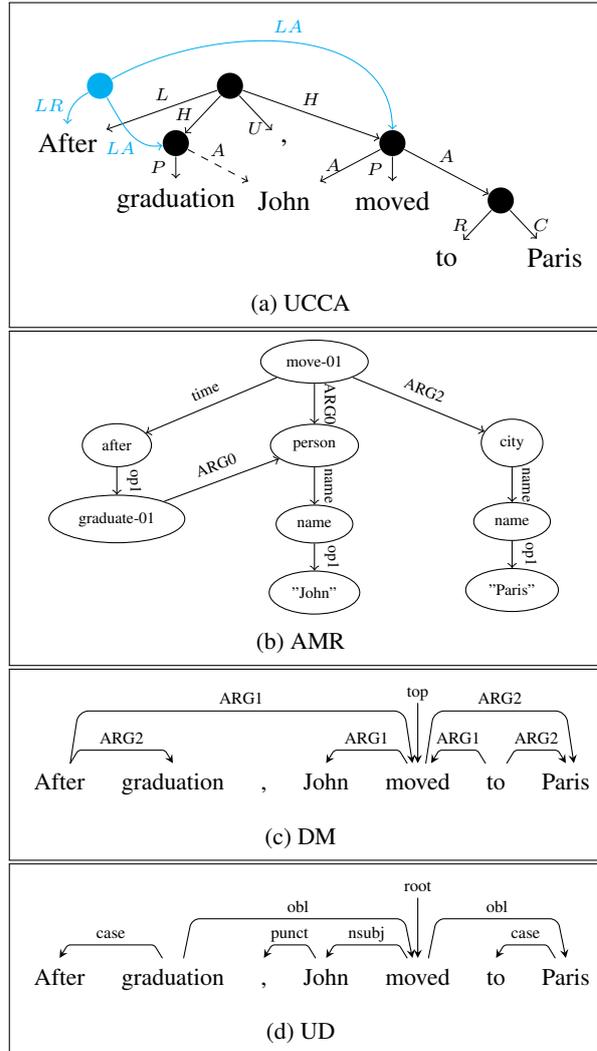

Figure 1: Example graph for each task. Figure 1a presents a UCCA graph. The dashed edge is remote, while the blue node and its outgoing edges represent inter-Scene linkage. Pre-terminal nodes and edges are omitted for brevity. Figure 1b presents an AMR graph. Text tokens are not part of the graph, and must be matched to concepts and constants by alignment. Variables are represented by their concepts. Figure 1c presents a DM semantic dependency graph, containing multiple roots: "After", "moved" and "to", of which "moved" is marked as *top*. Punctuation tokens are excluded from SDP graphs. Figure 1d presents a UD tree. Edge labels express syntactic relations.

terminal nodes to semantic units that participate in some super-ordinate relation. Edges are labeled, indicating the role of a child in the relation the parent represents. Nodes and edges belong to one of several *layers*, each corresponding to a "module" of semantic distinctions. UCCA's *foundational layer* (the only layer for which annotated data exists) mostly covers predicate-argument structure, semantic heads and inter-Scene relations.

UCCA distinguishes *primary* edges, corresponding to explicit relations, from *remote* edges

(appear dashed in Figure 1a) that allow for a unit to participate in several super-ordinate relations. Primary edges form a tree in each layer, whereas remote edges enable reentrancy, forming a DAG.

Abstract Meaning Representation. AMR (Banarescu et al., 2013) is a semantic representation that encodes information about named entities, argument structure, semantic roles, word sense and co-reference. AMRs are rooted directed graphs, in which both nodes and edges are labeled. Most AMRs are DAGs, although cycles are permitted.

AMR differs from the other schemes we consider in that it does not anchor its graphs in the words of the sentence (Figure 1b). Instead, AMR graphs connect variables, concepts (from a pre-defined set) and constants (which may be strings or numbers). Still, most AMR nodes are alignable to text tokens, a tendency used by AMR parsers, which align a subset of the graph nodes to a subset of the text tokens (concept identification). In this work, we use pre-aligned AMR graphs.

Despite the brief period since its inception, AMR has been targeted by a number of works, notably in two SemEval shared tasks (May, 2016; May and Priyadarshi, 2017). To tackle its variety of distinctions and unrestricted graph structure, AMR parsers often use specialized methods. Graph-based parsers construct AMRs by identifying concepts and scoring edges between them, either in a pipeline fashion (Flanigan et al., 2014; Artzi et al., 2015; Pust et al., 2015; Foland and Martin, 2017), or jointly (Zhou et al., 2016). Another line of work trains machine translation models to convert strings into linearized AMRs (Barzdins and Gosko, 2016; Peng et al., 2017b; Konstas et al., 2017; Buys and Blunsom, 2017b). Transition-based AMR parsers either use dependency trees as pre-processing, then mapping them into AMRs (Wang et al., 2015a,b, 2016; Goodman et al., 2016), or use a transition system tailored to AMR parsing (Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017). We differ from the above approaches in addressing AMR parsing using the same general DAG parser used for other schemes.

Semantic Dependency Parsing. SDP uses a set of related representations, targeted in two recent SemEval shared tasks (Oepen et al., 2014, 2015), and extended by Oepen et al. (2016). They correspond to four semantic representation schemes, referred to as DM, PAS, PSD and CCD, representing predicate-argument relations between content words in a sentence. All are based on semantic formalisms converted into bilexical dependencies—directed graphs whose nodes are text tokens. Edges are labeled, encoding semantic relations between the tokens. Non-content tokens, such as punctuation, are left out of the analysis (see Figure 1c). Graphs containing cycles have been removed from the SDP datasets.

We use one of the representations from the SemEval shared tasks: DM (DELPH-IN MRS), converted from DeepBank (Flickinger et al., 2012), a corpus of hand-corrected parses from LinGO ERG (Copestake and Flickinger, 2000), an HPSG (Pollard and Sag, 1994) using Minimal Recursion Semantics (Copestake et al., 2005).

Universal Dependencies. UD (Nivre et al., 2016, 2017) has quickly become the dominant dependency scheme for syntactic annotation in many languages, aiming for cross-linguistically consistent and coarse-grained treebank annotation. Formally, UD uses bilexical trees, with edge labels representing syntactic relations between words.

We use UD as an auxiliary task, inspired by previous work on joint syntactic and semantic parsing (see §2). In order to reach comparable analyses cross-linguistically, UD often ends up in annotation that is similar to the common practice in semantic treebanks, such as linking content words to content words wherever possible. Using UD further allows conducting experiments on languages other than English, for which AMR and SDP annotated data is not available (§7).

In addition to basic UD trees, we use the *enhanced++* UD graphs available for English, which are generated by the Stanford CoreNLP converters (Schuster and Manning, 2016).[2] These include additional and augmented relations between content words, partially overlapping with the notion of remote edges in UCCA: in the case of control verbs, for example, a direct relation is added in enhanced++ UD between the subordinated verb and its controller, which is similar to the semantic schemes' treatment of this construction.

4 General Transition-based DAG Parser

All schemes considered in this work exhibit reentrancy and discontinuity (or non-projectivity), to varying degrees. In addition, UCCA and AMR

---

[2] http://github.com/stanfordnlp/CoreNLP

contain non-terminal nodes. To parse these graphs, we extend TUPA (Hershcovich et al., 2017), a transition-based parser originally developed for UCCA, as it supports all these structural properties. TUPA's transition system can yield any labeled DAG whose terminals are anchored in the text tokens. To support parsing into AMR, which uses graphs that are not anchored in the tokens, we take advantage of existing alignments of the graphs with the text tokens during training (§5).

First used for projective syntactic dependency tree parsing (Nivre, 2003), transition-based parsers have since been generalized to parse into many other graph families, such as (discontinuous) constituency trees (e.g., Zhang and Clark, 2009; Maier and Lichte, 2016), and DAGs (e.g., Sagae and Tsujii, 2008; Du et al., 2015). Transition-based parsers apply *transitions* incrementally to an internal state defined by a buffer B of remaining tokens and nodes, a stack S of unresolved nodes, and a labeled graph G of constructed nodes and edges. When a terminal state is reached, the graph G is the final output. A classifier is used at each step to select the next transition, based on features that encode the current state.

4.1 TUPA's Transition Set

Given a sequence of tokens $w_1, \ldots, w_n$, we predict a rooted graph G whose terminals are the tokens. Parsing starts with the root node on the stack, and the input tokens in the buffer.

The TUPA transition set includes the standard SHIFT and REDUCE operations, NODE$_X$ for creating a new non-terminal node and an X-labeled edge, LEFT-EDGE$_X$ and RIGHT-EDGE$_X$ to create a new primary X-labeled edge, LEFT-REMOTE$_X$ and RIGHT-REMOTE$_X$ to create a new remote X-labeled edge, SWAP to handle discontinuous nodes, and FINISH to mark the state as terminal.

Although UCCA contains nodes without any text tokens as descendants (called *implicit units*), these nodes are infrequent and only cover 0.5% of non-terminal nodes. For this reason we follow previous work (Hershcovich et al., 2017) and discard implicit units from the training and evaluation, and so do not include transitions for creating them.

In AMR, implicit units are considerably more common, as any unaligned concept with no aligned descendents is implicit (about 6% of the nodes). Implicit AMR nodes usually result from alignment errors, or from abstract concepts which
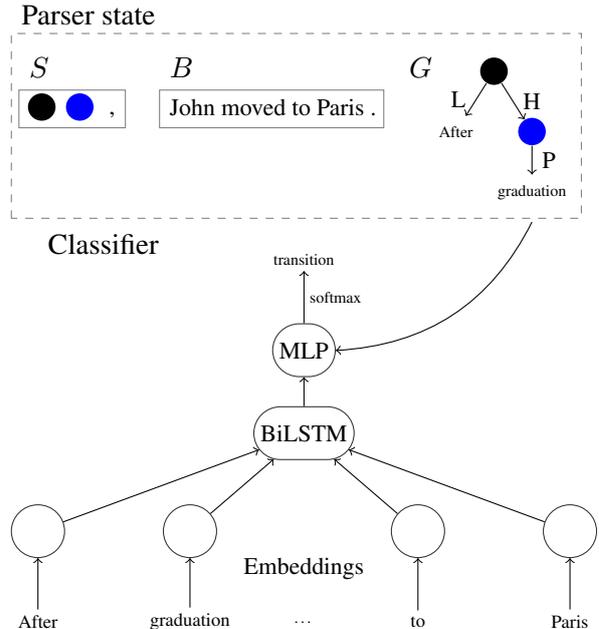

Figure 2: Illustration of the TUPA model, adapted from Hershcovich et al. (2017). Top: parser state. Bottom: BiLTSM architecture.

have no explicit realization in the text (Buys and Blunsom, 2017a). We ignore implicit nodes when training on AMR as well. TUPA also does not support node labels, which are ubiquitous in AMR but absent in UCCA structures (only edges are labeled in UCCA). We therefore only produce edge labels and not node labels when training on AMR.

4.2 Transition Classifier

To predict the next transition at each step, we use a BiLSTM with embeddings as inputs, followed by an MLP and a softmax layer for classification (Kiperwasser and Goldberg, 2016). The model is illustrated in Figure 2. Inference is performed greedily, and training is done with an oracle that yields the set of all optimal transitions at a given state (those that lead to a state from which the gold graph is still reachable). Out of this set, the actual transition performed in training is the one with the highest score given by the classifier, which is trained to maximize the sum of log-likelihoods of all optimal transitions at each step.

Features. We use the original TUPA features, representing the words, POS tags, syntactic dependency relations, and previously predicted edge labels for nodes in specific locations in the parser state. In addition, for each token we use embeddings representing the one-character prefix, three-character suffix, shape (capturing orthographic
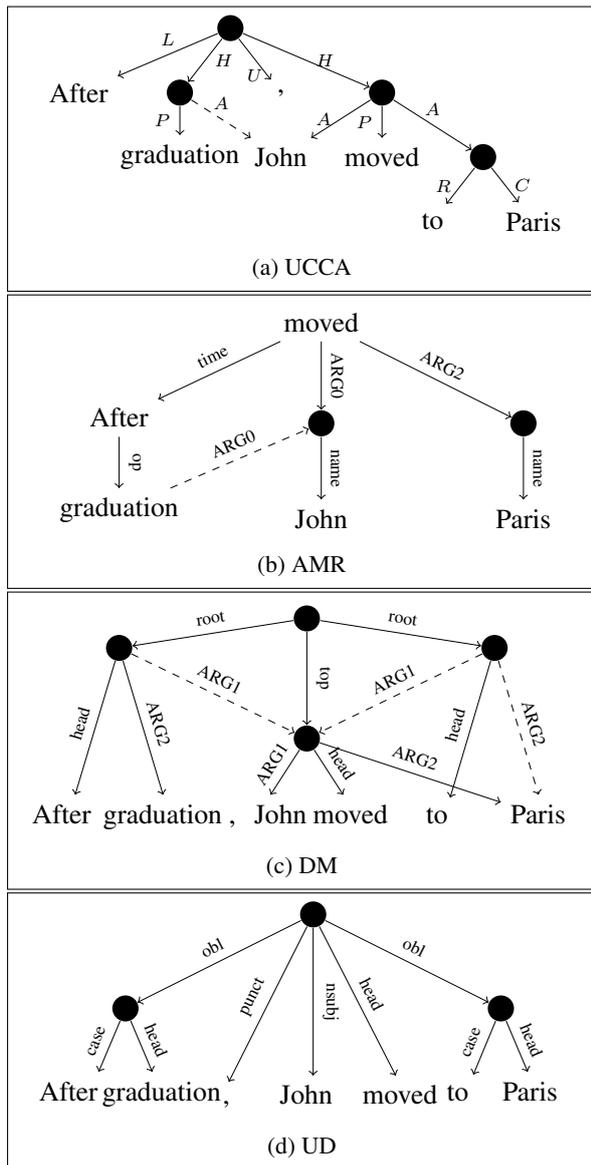
Figure 3: Graphs from Figure 1, after conversion to the unified DAG format (with pre-terminals omitted: each terminal drawn in place of its parent). Figure 3a presents a converted UCCA graph. Linkage nodes and edges are removed, but the original graph is otherwise preserved. Figure 3b presents a converted AMR graph, with text tokens added according to the alignments. Numeric suffixes of *op* relations are removed, and names collapsed. Figure 3c presents a converted SDP graph (in the DM representation), with intermediate non-terminal *head* nodes introduced. In case of reentrancy, an arbitrary reentrant edge is marked as remote. Figure 3d presents a converted UD graph. As in SDP, intermediate non-terminals and *head* edges are introduced. While converted UD graphs form trees, enhanced++ UD graphs may not.

features, e.g., "Xxxx"), and named entity type,[3] all provided by spaCy (Honnibal and Montani, 2018).[4] To the learned word vectors, we concatenate the 250K most frequent word vectors from

fastText (Bojanowski et al., 2017),[5] pre-trained over Wikipedia and updated during training.

Constraints. As each annotation scheme has different constraints on the allowed graph structures, we apply these constraints separately for each task. During training and parsing, the relevant constraint set rules out some of the transitions according to the parser state. Some constraints are task-specific, others are generic. For example, in UCCA, a terminal may only have one parent. In AMR, a concept corresponding to a Prop-Bank frame may only have the core arguments defined for the frame as children. An example of a generic constraint is that stack nodes that have been swapped should not be swapped again.[6]

5 Unified DAG Format

To apply our parser to the four target tasks (§3), we convert them into a unified DAG format, which is inclusive enough to allow representing any of the schemes with very little loss of information.[7]

The format consists of a rooted DAG, where the tokens are the terminal nodes. As in the UCCA format, edges are labeled (but not nodes), and are divided into *primary* and *remote* edges, where the primary edges form a tree (all nodes have at most one primary parent, and the root has none). Remote edges enable reentrancy, and thus together with primary edges form a DAG. Figure 3 shows examples for converted graphs. Converting UCCA into the unified format consists simply of removing linkage nodes and edges (see Figure 3a), which were also discarded by Hershcovich et al. (2017).

Converting bilexical dependencies. To convert DM and UD into the unified DAG format, we add a pre-terminal for each token, and attach the pre-terminals according to the original dependency edges: traversing the tree from the root down, for each head token we create a non-terminal parent with the edge label *head*, and add the node's dependents as children of the created non-terminal node (see Figures 3c and 3d). Since DM allows multiple roots, we form a single root node, whose
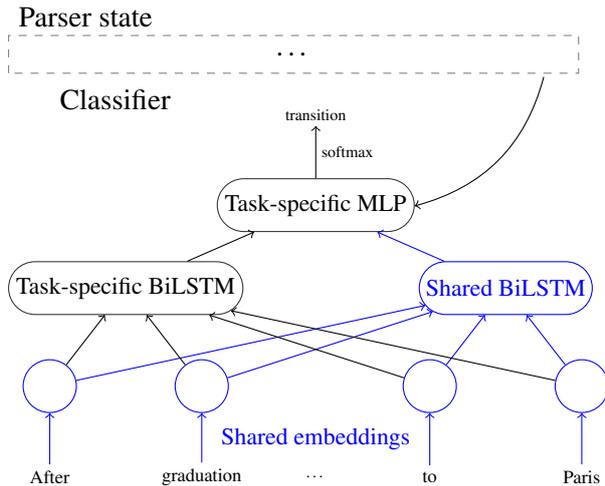
Figure 4: MTL model. Token representations are computed both by a task-specific and a shared BiLSTM. Their outputs are concatenated with the parser state embedding, identical to Figure 2, and fed into the task-specific MLP for selecting the next transition. Shared parameters are shown in blue.

children are the original roots. The added edges are labeled *root*, where top nodes are labeled *top* instead. In case of reentrancy, an arbitrary parent is marked as primary, and the rest as remote (denoted as dashed edges in Figure 3).

Converting AMR. In the conversion from AMR, node labels are dropped. Since alignments are not part of the AMR graph (see Figure 3b), we use automatic alignments (see §7), and attach each node with an edge to each of its aligned terminals.

Named entities in AMR are represented as a subgraph, whose *name*-labeled root has a child for each token in the name (see the two *name* nodes in Figure 1b). We collapse this subgraph into a single node whose children are the name tokens.

6 Multitask Transition-based Parsing

Now that the same model can be applied to different tasks, we can train it in a multitask setting. The fairly small training set available for UCCA (see §7) makes MTL particularly appealing, and we focus on it in this paper, treating AMR, DM and UD parsing as auxiliary tasks.

Following previous work, we share only some of the parameters (Klerke et al., 2016; Søgaard and Goldberg, 2016; Bollmann and Søgaard, 2016; Plank, 2016; Braud et al., 2016; Martínez Alonso and Plank, 2017; Peng et al., 2017a, 2018), leaving task-specific sub-networks as well. Concretely, we keep the BiLSTM used by TUPA for the main task (UCCA parsing), add a BiLSTM that is shared

across all tasks, and replicate the MLP (feedforward sub-network) for each task. The BiLSTM outputs (concatenated for the main task) are fed into the task-specific MLP (see Figure 4). Feature embeddings are shared across tasks.

Unlabeled parsing for auxiliary tasks. To simplify the auxiliary tasks and facilitate generalization (Bingel and Søgaard, 2017), we perform unlabeled parsing for AMR, DM and UD, while still predicting edge labels in UCCA parsing. To support unlabeled parsing, we simply remove all labels from the EDGE, REMOTE and NODE transitions output by the oracle. This results in a much smaller number of transitions the classifier has to select from (no more than 10, as opposed to 45 in labeled UCCA parsing), allowing us to use no BiLSTMs and fewer dimensions and layers for task-specific MLPs of auxiliary tasks (see §7). This limited capacity forces the network to use the shared parameters for all tasks, increasing generalization (Martínez Alonso and Plank, 2017).

7 Experimental Setup

We here detail a range of experiments to assess the value of MTL to UCCA parsing, training the parser in single-task and multitask settings, and evaluating its performance on the UCCA test sets in both in-domain and out-of-domain settings.

Data. For UCCA, we use v1.2 of the English Wikipedia corpus (*Wiki*; Abend and Rappoport, 2013), with the standard train/dev/test split (see Table 1), and the *Twenty Thousand Leagues Under the Sea* corpora (*20K*; Sulem et al., 2015), annotated in English, French and German.[8] For English and French we use 20K v1.0, a small parallel corpus comprising the first five chapters of the book. As in previous work (Hershcovich et al., 2017), we use the English part only as an out-of-domain test set. We train and test on the French part using the standard split, as well as the German corpus (v0.9), which is a pre-release and still contains a considerable amount of noisy annotation. Tuning is performed on the respective development sets.

For AMR, we use LDC2017T10, identical to the dataset targeted in SemEval 2017 (May and Priyadarshi, 2017).[9] For SDP, we use the DM representation from the SDP 2016 dataset (Oepen

---

[8] http://github.com/huji-nlp/ucca-corpora
[9] http://catalog.ldc.upenn.edu/LDC2017T10

	English						French						German					
	# tokens			# sentences			# tokens			# sentences			# tokens			# sentences		
	train	dev	test	train	dev	test	train	dev	test	train	dev	test	train	dev	test	train	dev	test
UCCA																		
Wiki	128444	14676	15313	4268	454	503												
20K			12339			506	10047	1558	1324	413	67	67	79894	10059	42366	3429	561	2164
AMR	648950			36521														
DM	765025			33964														
UD	458277			17062			899163			32347			268145			13814		

Table 1: Number of tokens and sentences in the training, development and test sets we use for each corpus and language.

et al., 2016).[10] For Universal Dependencies, we use all English, French and German treebanks from UD v2.1 (Nivre et al., 2017).[11] We use the enhanced++ UD representation (Schuster and Manning, 2016) in our English experiments, henceforth referred to as UD^{++}. We use only the AMR, DM and UD training sets from standard splits.

While UCCA is annotated over Wikipedia and over a literary corpus, the domains for AMR, DM and UD are blogs, news, emails, reviews, and Q&A. This domain difference between training and test is particularly challenging (see §9). Unfortunately, none of the other schemes have available annotation over Wikipedia text.

Settings. We explore the following settings: (1) in-domain setting in English, training and testing on Wiki; (2) out-of-domain setting in English, training on Wiki and testing on 20K; (3) French in-domain setting, where available training dataset is small, training and testing on 20K; (4) German in-domain setting on 20K, with somewhat noisy annotation. For MTL experiments, we use unlabeled AMR, DM and UD^{++} parsing as auxiliary tasks in English, and unlabeled UD parsing in French and German.[12] We also report baseline results training only the UCCA training sets.

Training. We create a unified corpus for each setting, shuffling all sentences from relevant datasets together, but using only the UCCA development set F_1 score as the early stopping criterion. In each training epoch, we use the same number of examples from each task—the UCCA training set size. Since training sets differ in size, we sample this many sentences from each one. The model is implemented using DyNet (Neubig et al., 2017).[13]

Hyperparameter	Single	Multitask		
		Main	Aux	Shared
Pre-trained word dim.	300			300
Learned word dim.	200			200
POS tag dim.	20			20
Dependency relation dim.	10			10
Named entity dim.	3			3
Punctuation dim.	1			1
Action dim.	3			3
Edge label dim.	20	20		
MLP layers	2	2	1	
MLP dimensions	50	50	50	
BiLSTM layers	2	2		2
BiLSTM dimensions	500	300		300

Table 2: Hyperparameter settings. Middle column shows hyperparameters used for the single-task architecture, described in §4.2, and right column for the multitask architecture, described in §6. **Main** refers to parameters specific to the main task—UCCA parsing (task-specific MLP and BiLSTM, and edge label embedding), **Aux** to parameters specific to each auxiliary task (task-specific MLP, but no edge label embedding since the tasks are unlabeled), and **Shared** to parameters shared among all tasks (shared BiLSTM and embeddings).

Hyperparameters. We initialize embeddings randomly. We use dropout (Srivastava et al., 2014) between MLP layers, and recurrent dropout (Gal and Ghahramani, 2016) between BiLSTM layers, both with $p = 0.4$. We also use word ($\alpha = 0.2$), tag ($\alpha = 0.2$) and dependency relation ($\alpha = 0.5$) dropout (Kiperwasser and Goldberg, 2016).[14] In addition, we use a novel form of dropout, *node dropout*: with a probability of 0.1 at each step, all features associated with a single node in the parser state are replaced with zero vectors. For optimization we use a minibatch size of 100, decaying all weights by 10^{-5} at each update, and train with stochastic gradient descent for N epochs with a learning rate of 0.1, followed by AMSGrad (Sashank J. Reddi, 2018) for N epochs with $\alpha = 0.001, \beta_1 = 0.9$ and $\beta_2 = 0.999$. We use $N = 50$ for English and German, and $N = 400$ for French. We found this training strategy better than using only one of the optimization methods,

---

	Primary			Remote		
	LP	LR	LF	LP	LR	LF
English (in-domain)						
HAR17	74.4	72.7	73.5	47.4	51.6	49.4
Single	74.4	72.9	73.6	53	50	51.5
AMR	74.7	72.8	73.7	48.7★	51.1	49.9
DM	75.7★	73.9★	74.8★	54.9	53	**53.9**
UD^{++}	75★	73.2	74.1★	49	52.7	50.8
AMR + DM	75.6★	73.9★	74.7★	49.9	53	51.4
AMR + UD^{++}	74.9	72.7	73.8	47.1	50	48.5
DM + UD^{++}	75.9★	73.9★	**74.9★**	48	54.8	51.2
All	75.6★	73.1	74.4★	50.9	53.2	52

Table 3: Labeled precision, recall and F_1 (in %) for primary and remote edges, on the **Wiki** test set. ★ indicates significantly better than *Single*. HAR17: Hershcovich et al. (2017).

	Primary			Remote		
	LP	LR	LF	LP	LR	LF
English (out-of-domain)						
HAR17	68.7	68.5	68.6	38.6	18.8	25.3
Single	69	69	69	41.2	19.8	26.7
AMR	69.5	69.5	69.5	42.9	20.2	27.5
DM	70.7★	70.7★	70.7★	42.7	18.6	25.9
UD^{++}	69.6	69.8★	69.7	41.4	22	28.7
AMR + DM	70.7★	70.2★	70.5★	45.8	19.4	27.3
AMR + UD^{++}	70.2★	69.9★	70★	45.1	21.8	29.4
DM + UD^{++}	70.8★	70.3★	70.6★	41.6	21.6	28.4
All	71.2★	70.9★	**71★**	45.1	22	**29.6**
French (in-domain)						
Single	68.2	67	67.6	26	9.4	13.9
UD	70.3	70★	**70.1★**	43.8	13.2	20.3
German (in-domain)						
Single	73.3	71.7	72.5	57.1	17.7	27.1
UD	73.7★	72.6★	**73.2★**	61.8	24.9★	**35.5★**

Table 4: Labeled precision, recall and F_1 (in %) for primary and remote edges, on the **20K** test sets. ★ indicates significantly better than *Single*. HAR17: Hershcovich et al. (2017).

similar to findings by Keskar and Socher (2017). We select the epoch with the best average labeled F_1 score on the UCCA development set. Other hyperparameter settings are listed in Table 2.

Evaluation. We evaluate on UCCA using labeled precision, recall and F_1 on primary and remote edges, following previous work (Hershcovich et al., 2017). Edges in predicted and gold graphs are matched by terminal yield and label. Significance testing of improvements over the single-task model is done by the bootstrap test (Berg-Kirkpatrick et al., 2012), with $p < 0.05$.

8 Results

Table 3 presents our results on the English in-domain Wiki test set. MTL with all auxiliary tasks and their combinations improves the primary F_1 score over the single task baseline. In most settings the improvement is statistically significant. Using all auxiliary tasks contributed less than just

DM and UD^{++}, the combination of which yielded the best scores yet in in-domain UCCA parsing, with 74.9% F_1 on primary edges. Remote F_1 is improved in some settings, but due to the relatively small number of remote edges (about 2% of all edges), none of the differences is significant. Note that our baseline single-task model (*Single*) is slightly better than the current state-of-the-art (HAR17; Hershcovich et al., 2017), due to the incorporation of additional features (see §4.2).

Table 4 presents our experimental results on the 20K corpora in the three languages. For English out-of-domain, improvements from using MTL are even more marked. Moreover, the improvement is largely additive: the best model, using all three auxiliary tasks (*All*), yields an error reduction of 2.9%. Again, the single-task baseline is slightly better than HAR17.

The contribution of MTL is also apparent in French and German in-domain parsing: 3.7% error reduction in French (having less than 10% as much UCCA training data as English) and 1% in German, where the training set is comparable in size to the English one, but is noisier (see §7). The best MTL models are significantly better than single-task models, demonstrating that even a small training set for the main task may suffice, given enough auxiliary training data (as in French).

9 Discussion

Quantifying the similarity between tasks. Task similarity is an important factor in MTL success (Bingel and Søgaard, 2017; Martínez Alonso and Plank, 2017). In our case, the main and auxiliary tasks are annotated on different corpora from different domains (§7), and the target representations vary both in form and in content.

To quantify the domain differences, we follow Plank and van Noord (2011) and measure the L1 distance between word distributions in the English training sets and 20K test set (Table 5). All auxiliary training sets are more similar to 20K than Wiki is, which may contribute to the benefits observed on the English 20K test set.

As a measure of the formal similarity of the different schemes to UCCA, we use unlabeled F_1 score evaluation on both primary and remote edges (ignoring edge labels). To this end, we annotated 100 English sentences from Section 02 of the Penn Treebank Wall Street Journal (PTB WSJ). Anno-

	20K	AMR	DM	UD
Wiki	1.047	0.895	0.913	0.843
20K		0.949	0.971	0.904
AMR			0.757	0.469
DM				0.754

Table 5: L1 distance between dataset word distributions, quantifying domain differences in English (low is similar).

	Primary			Remote		
	UP	UR	UF	UP	UR	UF
AMR	53.8	15.6	24.2	7.3	5.5	6.3
DM	65	49.2	56	7.4	65.9	13.3
UD^{++}	82.7	84.6	83.6	12.5	12.7	12.6

Table 6: Unlabeled F_1 scores between the representations of the same English sentences (from PTB WSJ), converted to the unified DAG format, and annotated UCCA graphs.

tation was carried out by a single expert UCCA annotator, and is publicly available.[15] These sentences had already been annotated by the AMR, DM and PTB schemes,[16] and we convert their annotation to the unified DAG format.

Unlabeled F_1 scores between the UCCA graphs and those converted from AMR, DM and UD^{++} are presented in Table 6. UD^{++} is highly overlapping with UCCA, while DM less so, and AMR even less (cf. Figure 3).

Comparing the average improvements resulting from adding each of the tasks as auxiliary (see §8), we find AMR the least beneficial, UD^{++} second, and DM the most beneficial, in both in-domain and out-of-domain settings. This trend is weakly correlated with the formal similarity between the tasks (as expressed in Table 6), but weakly negatively correlated with the word distribution similarity scores (Table 5). We conclude that other factors should be taken into account to fully explain this effect, and propose to address this in future work through controlled experiments, where corpora of the same domain are annotated with the various formalisms and used as training data for MTL.

AMR, SDP and UD parsing. Evaluating the full MTL model (*All*) on the unlabeled auxiliary tasks yielded 64.7% unlabeled Smatch F_1 (Cai and Knight, 2013) on the AMR development set, when using oracle concept identification (since the auxiliary model does not predict node labels), 27.2% unlabeled F_1 on the DM development set, and

4.9% UAS on the UD development set. These poor results reflect the fact that model selection was based on the score on the UCCA development set, and that the model parameters dedicated to auxiliary tasks were very limited (to encourage using the shared parameters). However, preliminary experiments using our approach produced promising results on each of the tasks' respective English development sets, when treated as a single task: 67.1% labeled Smatch F_1 on AMR (adding a transition for implicit nodes and classifier for node labels), 79.1% labeled F_1 on DM, and 80.1% LAS F_1 on UD. For comparison, the best results on these datasets are 70.7%, 91.2% and 82.2%, respectively (Foland and Martin, 2017; Peng et al., 2018; Dozat et al., 2017).

10 Conclusion

We demonstrate that semantic parsers can leverage a range of semantically and syntactically annotated data, to improve their performance. Our experiments show that MTL improves UCCA parsing, using AMR, DM and UD parsing as auxiliaries. We propose a unified DAG representation, construct protocols for converting these schemes into the unified format, and generalize a transition-based DAG parser to support all these tasks, allowing it to be jointly trained on them.

While we focus on UCCA in this work, our parser is capable of parsing any scheme that can be represented in the unified DAG format, and preliminary results on AMR, DM and UD are promising (see §9). Future work will investigate whether a single algorithm and architecture can be competitive on all of these parsing tasks, an important step towards a joint many-task model for semantic parsing.

---

[15] http://github.com/danielhers/wsj

[16] We convert the PTB format to UD^{++} v1 using Stanford CoreNLP, and then to UD v2 using Udapi: http://github.com/udapi/udapi-python.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*, pages 228–238.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proc. of ACL*, pages 77–89.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *TACL*, 4:431–444.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proc. of EMNLP*, pages 1699–1710.

Miguel Ballesteros and Yaser Al-Onaizan. 2017. AMR parsing using stack-LSTMs. In *Proc. of EMNLP*, pages 1269–1275.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the Linguistic Annotation Workshop*.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proc. of SemEval*, pages 1143–1147.

Eric Baucom, Levi King, and Sandra Kübler. 2013. Domain adaptation for parsing. In *Proc. of RANLP*, pages 56–64.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proc. of EMNLP-CoNLL*, pages 995–1005.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proc. of EACL*, pages 164–169.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*, pages 120–128.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. of EMNLP-CoNLL*, pages 1455–1465.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bidirectional lstms and multi-task learning. In *Proc. of COLING*, pages 131–139.

Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of RST discourse parsers. In *Proc. of COLING*, pages 1903–1913.

Jan Buys and Phil Blunsom. 2017a. Oxford at SemEval-2017 task 9: Neural AMR parsing with pointer-augmented attention. In *Proc. of SemEval*, pages 914–919.

Jan Buys and Phil Blunsom. 2017b. Robust incremental neural semantic graph parsing. In *Proc. of ACL*, pages 1215–1226.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*, pages 748–752.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):41–75.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Matthieu Constant and Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *Proc. of ACL*, pages 161–171.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proc. of LREC*, pages 591–600.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2):281–332.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proc. of EACL*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*, pages 256–263.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proc. of CoNLL*, pages 20–30.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proc. of SemEval*, pages 927–931.

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip Cohen, and Mark Johnson. 2017. Multilingual semantic parsing and code-switching. In *Proc. of CoNLL*, pages 379–389.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proc. of Workshop on Representation Learning for NLP*, pages 48–56.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proc. of NAACL-HLT*, pages 326–334.

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. of ACL*, pages 1426–1436.

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

William Foland and James H. Martin. 2017. Abstract Meaning Representation parsing using LSTM recurrent neural networks. In *Proc. of ACL*, pages 463–472.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *Proc. of ACL*, pages 1–11.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2016. Exploiting multi-typed treebanks for parsing with deep multi-task learning. *CoRR*, abs/1606.01161.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štepánek, 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*, pages 1–18.

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*, pages 1923–1933.

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

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL*, pages 1127–1138.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proc. of ACL*, pages 623–628.

Matthew Honnibal and Ines Montani. 2018. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Nitish Shirish Keskar and Richard Socher. 2017. Improving generalization performance by switching from Adam to SGD. *CoRR*, abs/1712.07628.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proc. of NAACL-HLT*, pages 1528–1533.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proc. of ACL*, pages 146–157.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proc. of EMNLP*, pages 1444–1454.

Xavier Lluís and Lluís Màrquez. 2008. A joint model for parsing syntactic and semantic dependencies. In *Proc. of CoNLL*, pages 188–192.

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proc. of Workshop on Discontinuous Structures in NLP*, pages 47–57.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In *Proc. of EACL*, pages 44–53.

Jonathan May. 2016. SemEval-2016 task 8: Meaning representation parsing. In *Proc. of SemEval*, pages 1063–1073.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract Meaning Representation parsing and generation. In *Proc. of SemEval*, pages 536–545.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proc. of NAACL-HLT*, pages 28–36.

Amir More. 2016. Joint morpho-syntactic processing of morphologically rich languages in a transition-based framework. Master's thesis, The Interdisciplinary Center, Herzliya.

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. *CoRR*, abs/1701.03980.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT*, pages 149–160.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà M, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phng Lê Hng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Niko-

laev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresova. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proc. of LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*, pages 915–926.

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

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

Hao Peng, Sam Thomson, and Noah A. Smith. 2017a. Deep multitask learning for semantic dependency parsing. In *Proc. of ACL*, pages 2037–2048.

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

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017b. Addressing the data sparsity issue in neural AMR parsing. In *Proc. of EACL*, pages 366–375.

Barbara Plank. 2016. Keystroke dynamics as signal for shallow syntactic parsing. In *Proc. of COLING*, pages 609–619.

Barbara Plank and Gertjan van Noord. 2011. Effective measures of domain similarity for parsing. In *Proc. of ACL-HLT*, pages 1566–1576.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into Abstract Meaning Representation using syntax-based machine translation. In *Proc. of EMNLP*, pages 1143–1154.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proc. of COLING*, pages 753–760.

Sanjiv Kumar Sashank J. Reddi, Satyen Kale. 2018. On the convergence of Adam and beyond. *ICLR*.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proc. of LREC*. ELRA.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*, pages 231–235.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A French-English case study. In *Proc. of S2MT*, pages 11–22.

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*, pages 159–177.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proc. of CoNLL*, pages 187–197.

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

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proc. of ACL*, pages 589–596.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 task 8: An extended transition-based AMR parser. In *Proc. of SemEval*, pages 1173–1178.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proc. of ACL*, pages 857–862.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proc. of NAACL*, pages 366–375.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*, pages 1557–1566.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proc. of IWPT*, pages 162–171.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proc. of EMNLP*, pages 680–689.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proc. of CoNLL*, pages 400–410.