Surface Statistics of an Unknown Language Indicate How to Parse It

Dingquan Wang and **Jason Eisner**
Department of Computer Science, Johns Hopkins University
{wdd,jason}@cs.jhu.edu

Abstract

We introduce a novel framework for delexicalized dependency parsing in a new language. We show that useful features of the target language can be extracted automatically from an unparsed corpus, which consists only of gold part-of-speech (POS) sequences. Providing these features to our neural parser enables it to parse sequences like those in the corpus. Strikingly, our system has no supervision in the target language. Rather, it is a multilingual system that is trained end-to-end on a variety of *other* languages, so it learns a feature extractor that works well. We show experimentally across multiple languages: (1) Features computed from the unparsed corpus improve parsing accuracy. (2) Including thousands of synthetic languages in the training yields further improvement. (3) Despite being computed from *unparsed* corpora, our learned task-specific features beat previous work's interpretable typological features that require *parsed* corpora or expert categorization of the language. Our best method improved attachment scores on held-out test languages by an average of 5.6 percentage points over past work that does not inspect the unparsed data (McDonald et al., 2011), and by 20.7 points over past "grammar induction" work that does not use training languages (Naseem et al., 2010).

1 Introduction

Dependency parsing is one of the core natural language processing tasks. It aims to parse a given sentence into its dependency tree: a directed graph of labeled syntactic relations between words. *Supervised* dependency parsers—which are trained using a "treebank" of known parses in the target language—have been very successful (McDonald, 2006; Nivre, 2008; Kiperwasser and Goldberg, 2016). By contrast, the progress of *unsupervised* dependency parsers has been slow, and they have apparently not been used in any downstream NLP systems (Mareček, 2016). An unsupervised parser does not have access to a treebank, but only to a corpus of unparsed sentences in the target language.

Unsupervised parsing has been studied for decades. The most common approach is *grammar induction* (Lari and Young, 1990; Carroll and Charniak, 1992; Klein and Manning, 2004). Grammar induction induces an explicit grammar from the unparsed corpus, such as a probabilistic context-free grammar (PCFG), and uses that to parse sentences of the language. This approach has encountered two major difficulties:

- **Search error:** Most formulations of grammar induction involve optimizing a highly non-convex objective function such as likelihood. The optimization is typically NP-hard (Cohen and Smith, 2012), and approximate local search methods tend to get stuck in local optima.

- **Model error:** Likelihood does not correlate well with parsing accuracy anyway (Smith, 2006, Figure 3.2). Likelihood optimization seeks latent trees that help to predict the observed sentences, but these unsupervised trees may use a non-standard syntactic analysis or even be optimized to predict non-syntactic properties such as topic. We seek a *standard syntactic analysis*—what Smith (2006) calls the MATCHLINGUIST task.

We address both difficulties by using a *supervised* learning framework—one whose objective function is easier to optimize and explicitly tries to match linguists' standard syntactic analyses.

Our approach is inspired by Wang and Eisner (2017), who use an unparsed but tagged corpus to predict the fine-grained syntactic typology of a language. For example, they may predict that about 70% of the direct objects fall to the right of the verb. Their system is trained on a large number of (unparsed corpus, true typology) pairs, each representing a different language. With this training, it can generalize to predict typology from the unparsed corpus of a new language. Our approach is similar except that we predict parses rather than just a typology. In both cases, the system is trained to optimize a task-specific quality measure. The system's parameterization can be chosen to simplify optimization (strikingly, the training objective could even be made *convex* by using a conditional random field architecture) and/or to incorporate linguistically motivated features.

The positive results of Wang and Eisner (2017) demonstrate that there are indeed surface clues to syntactic structure in the input corpus, at least if it is POS-tagged (as in their work and ours). However, their method only found global typological information: it did not establish *which* 70% of the direct objects fell to the right of their verbs, let alone identify which nouns were in fact direct objects of which verbs. That requires a token-level analysis of each sentence, which we undertake in this paper. Again, the basic idea is that instead of predicting interpretable typological properties of a language as Wang and Eisner (2017) did, we will predict a language-specific version of the scoring function that a parser uses to choose among various actions or substructures.

2 Unsupervised Parsing with Supervised Tuning

Our fundamental question is whether gold part-of-speech (POS) sequences carry useful information about the syntax of a language.[1] As we will show, the answer is yes, and the information can be extracted and used to obtain actual parses.

This is the same question that has been implicitly asked by previous papers in the unsupervised parsing tradition (see §5). Unsupervised parsing of gold POS sequences is an artificial task, to be sure.[2] Nonetheless, it is a starting point for more ambitious settings that would learn from words and real-world grounding (with or without the POS tags). Even this starting point has proved surprisingly difficult over decades of research, so it has not been clear whether the POS sequences even contain the necessary information.

Yet this task—like others that engineers, linguists, or human learners might face—might be solvable with general knowledge about the distribution of human languages. An experienced linguist can sometimes puzzle out the structure of a new language. The reader may be willing to guess a parse for the gold POS sequence VERB DET NOUN ADJ DET NOUN. After all, adjectives usually attach to nouns (Naseem et al., 2010), and the adjective in this example seems to attach to the first noun—not to the second, since determiners usually fall at the edge of a noun phrase. Meanwhile, the sequence's sole verb is apparently followed by two noun phrases, which suggests either VSO (verb-subject-object) or VOS order—and VSO is a good guess as it is more common (Dryer and Haspelmath, 2013). Observing a corpus of additional POS sequences might help resolve the question of whether this language is primarily VSO or VOS, for example, by guessing that short noun phrases in the corpus (for example, unmodified pronouns) are more often subjects.

Thus, we propose to solve the task by training a kind of "artificial linguist" that can do such analysis on corpora of new languages.

This is a general approach to developing an unsupervised method for a specific type of dataset: tune its structure and hyperparameters so that it works well on actual datasets *of that sort*, and then apply it to *new* datasets. For example, consider clustering—the canonical unsupervised problem. What constitutes a useful cluster depends on the type of data and the application. Basu et al. (2013) develop a text clustering system specifically to aid teachers. Their "Powergrading" system can group all the student-written answers to a *novel* question, having been trained on human judgments of answer similarity for *other* questions. Their novel questions are analogous to our novel languages: their unsupervised system is specifically tailored to match teachers' semantic similarity judgments within any corpus of student answers, just as ours is tailored to match linguists' syntactic judgments within any corpus of human-language POS sequences. Other NLP work on supervised tuning of

[1] We also include an experiment on noisy POS sequences.

[2] It is clearly not the task setting faced by human language learners. Nor is it a plausible engineering setting: a language with gold POS sequences often also has at least a small treebank of gold parses, or at least parallel text in a language from which noisy parses can be noisily projected (Agić et al., 2016). There is also no practical reason to consider POS tags without their attached words.

unsupervised learners includes strapping (Eisner and Karakos, 2005; Karakos et al., 2007), which tunes with the help of both real and synthetic datasets, just as we will (§3).

Are such systems really "unsupervised"? Yes, in the sense that they are able to discover desirable structure in a *new* dataset. Unsupervised learners are normally crafted using assumptions about the data domain. Their structure and hyperparameters may have been manually tuned to produce pleasing results for typical datasets in that domain. In the domain of POS corpora, we simply scale up this practice to *automatically* tune a large set of parameters, which later guide our system's search for linguist-approved structure on each new human-language dataset. Our system should be regarded as "supervised" if the examples are taken to be entire languages: after all, we train it to map unlabeled corpora to usefully labeled corpora. But once trained, it is "unsupervised" if the examples are taken to be the sentences within a given corpus: by analyzing the corpus, our system figures out how to map sentences of *that* language to parses, without any labeled examples in that language.

3 Data

We use two datasets in our experiment:

UD: Universal Dependencies version 1.2 (Nivre et al., 2015) A collection of 37 dependency treebanks of 33 languages, tokenized and annotated with a common set of POS tags and dependency relations.[3] In principle, our trained system could be applied to predict UD-style dependency relations in any tokenized natural-language corpus with UD-style POS tags.

GD: Galactic Dependencies version 1.0 (Wang and Eisner, 2016) A collection of dependency treebanks for 53,428 synthetic languages (of which we will use a subset). A GD treebank is generated by starting with some UD treebank and stochastically permuting the child subtrees of nouns and/or verbs to match their orders in other UD treebanks. For example, one of the GD treebanks reflects what the English UD treebank might have looked like if English had been both VSO (like Irish) and postpositional (like Japanese). This typologically diverse collection of resource-rich synthetic languages aims to propel the development of NLP systems that can handle diverse natural languages, such as multilingual parsers and taggers.

3.1 Why Synthetic Training Languages?

We hope for our system to do well, on average, at matching real linguist-parsed corpora of real human languages. We therefore tune its parameters $\mathbf{\Theta}$ on such treebanks. UD provides training examples *actually* drawn from that distribution $\mathcal{D}$ over treebanks—but alas, rather few. Thus to better estimate the expected performance of $\mathbf{\Theta}$ under $\mathcal{D}$, we follow Wang and Eisner (2017) and augment our training data with GD's *synthetic* treebanks.

Ideally we would have sampled these synthetic treebanks from a careful estimate $\hat{\mathcal{D}}$ of $\mathcal{D}$: for example, the mean of a Bayesian posterior for $\mathcal{D}$, derived from prior assumptions and UD evidence. However, such adventurous "extrapolation" of unseen languages would have required actually constructing such an estimate $\hat{\mathcal{D}}$—which would embody a distribution over semantic content and a full theory of universal grammar! The GD treebanks were derived more simply and more conservatively by "interpolation" among the actual UD corpora. They combine observed parse trees (which provide *attested* semantic content) with stochastic word order models trained on observed languages (which attempt to mimic *attested* patterns for presenting that content). GD's sampling distribution $\hat{\mathcal{D}}$ still offers moderately varied synthetic datasets, which remain moderately realistic, as they are limited to phenomena observed in UD.

As Wang and Eisner (2016) pointed out, synthetic examples have been used in many other supervised machine learning settings. A common technique is to exploit invariance: if real image $\mathbf{z}$ should be classified as a cat, then so should a *rotated* version of image $\mathbf{z}$. Our technique is the same! We assume that if real corpus $\mathbf{u}$ should be parsed as having certain dependencies among the word tokens, then so should a version of corpus $\mathbf{u}$ in which those tokens have been *systematically permuted* in a *linguistically plausible* way.[4] This is analogous to how rotation sytematically transforms the image (rotating all pixels through the *same* angle) in a physically plausible way (as real objects do rotate relative to the camera). The systematicity is needed to ensure that the task on syn-

[3] While it might have been preferable to use the expanded and revised UD version 2.0, we wished to compare fairly with GD 1.0, which is based on UD 1.2.

[4] Another example is back-translation.

thetic data is feasible. In our case, the synthetic corpus then provides *many* sentences that have been similarly permuted, which may *jointly* provide enough clues to guess the word order of this synthetic language (for example, VSO vs. VOS in §2) and thus recover the dependencies. See Wang and Eisner (2018, §2) for related discussion.

With enough good synthetic languages to use for training, even nearest-neighbor could be an effective method. That is, one could obtain the parser for a test corpus simply by copying the trained parser for the most similar training corpus (under some metric). Wang and Eisner (2016) explored this approach of "single-source transfer" from synthetic languages. Yet with only thousands of synthetic languages, perhaps no *single* training corpus is sufficiently similar.[5] To draw on patterns in *many* training corpora to figure out how to parse the test corpus, we will train a single parser that can handle all of the training corpora (Ammar et al., 2016), much as we trained our typological classifier in earlier work (Wang and Eisner, 2017).

4 Task Formulation

An unsupervised parser for language ℓ is built without any gold parse trees for ℓ. However, we assume a corpus $\mathbf{u}$ of *unparsed* but POS-tagged sentences of ℓ is available. From $\mathbf{u}$, we will extract statistics $\mathrm{T}(\mathbf{u})$ that are informative about the syntactic structure of ℓ, to guide us in parsing POS-tagged sentences of ℓ.

Overall, our approach is to train a "language-agnostic" parser—one that does not know what language ℓ it is parsing in. It produces a parse tree $\hat{y} = \mathrm{Parse}_{\boldsymbol{\Theta}}(x; \mathbf{u})$ from a sentence x, constructing $\mathrm{T}(\mathbf{u})$ as an intermediate quantity that carries (for example) typological information about ℓ. The parameters $\boldsymbol{\Theta}$ are shared by all languages, and determine how to construct and use T. To learn them, we will allow ℓ to range over training languages, and then test our ability to parse when ℓ ranges over novel test languages.

Our $\mathrm{Parse}_{\boldsymbol{\Theta}}(x; \mathbf{u})$ system has two stages. First it uses a neural network to compute $\mathrm{T}(\mathbf{u}) \in \mathbb{R}^m$, a vector that represents the typological properties of ℓ and resembles the *language embedding* of Ammar et al. (2016). Then it parses sentence x while taking $\mathrm{T}(\mathbf{u})$ as an additional input. We will give details of these two components in §6 and §7.

We assume in this paper that the input sentence x is given as a POS sequence: that is, our parser is *delexicalized*. This spares us from also needing language-specific lexical parameters associated with the specific vocabulary of each language, a problem that we leave to future work.

We will choose our universal parameter values by minimizing an estimate of their expected loss,

$$\hat{\boldsymbol{\Theta}} = \operatorname*{argmin}_{\boldsymbol{\Theta}} \operatorname*{mean}_{\ell \in \mathcal{L}_{\texttt{train}}} \mathrm{Loss}(\boldsymbol{\Theta}; \mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)}, \mathbf{u}^{(\ell)}) \tag{1}$$

where $\mathcal{L}_{\texttt{train}}$ is a collection of training languages (ideally drawn IID from the distribution $\mathcal{D}$ of possible human languages) for which some syntactic information is available. Specifically, each training language ℓ has a treebank $(\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)})$, where $\mathbf{x}^{(\ell)}$ is a collection of POS-tagged sentences whose correct dependency trees are given by $\mathbf{y}^{(\ell)}$. Each ℓ also has an unparsed corpus $\mathbf{u}^{(\ell)}$ (possibly equal to $\mathbf{x}^{(\ell)}$ or containing $\mathbf{x}^{(\ell)}$). We can therefore define the parser's loss on training language ℓ as

$$\begin{aligned} &\mathrm{Loss}(\boldsymbol{\Theta}; \mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)}, \mathbf{u}^{(\ell)}) \qquad (2) \\ &= \operatorname*{mean}_{(x,y) \in (\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)})} \mathrm{loss}(\underbrace{\mathrm{Parse}_{\boldsymbol{\Theta}}(x; \mathbf{u}^{(\ell)})}_{\hat{y}}, y) \end{aligned}$$

where $\mathrm{loss}(\ldots)$ is a task-specific per-sentence loss (defined in §8.1) that evaluates the parser's output $\hat{y}$ on sentence x against x's correct tree y.

5 Related Work

5.1 Per-Language Learning

Many papers rely on some *universal learning procedure* to determine $\mathrm{T}(\mathbf{u})$ (see §4) for a target language. For example, $\mathrm{T}(\cdot)$ may be the Expectation-Maximization (EM) algorithm, yielding a PCFG $\mathrm{T}(\mathbf{u})$ that fully determines a CKY parser (Carroll and Charniak, 1992; Klein and Manning, 2004). Since EM and CKY are fixed algorithms, this approach has no trainable parameters.

Grammar induction tries to turn an unsupervised corpus into a generative grammar. The approach of the previous paragraph is often modified to reduce model error or search error (§1). To reduce model error, many papers have used dependency grammar, with training objectives that incorporate notions like lexical attraction (Yuret, 1998) and grammatical bigrams (Paskin, 2001, 2002). The dependency model with valence (DMV) (Klein and Manning, 2004) was the first

[5] Wang and Eisner (2018) do investigate synthesis "on demand" of a permuted training corpus that is as similar as possible to the test corpus.

method to beat a simple right-branching heuristic. Headden III et al. (2009) and Spitkovsky et al. (2012) made the DMV more expressive by considering higher-order valency or punctuation. To reduce search error, strategies for eliminating or escaping local optima have included convexified objectives (Wang et al., 2008; Gimpel and Smith, 2012), smart initialization (Klein and Manning, 2004; Mareček and Straka, 2013), search bias (Smith and Eisner, 2005, 2006; Naseem et al., 2010; Gillenwater et al., 2010), branch-and-bound search (Gormley and Eisner, 2013), and switching objectives (Spitkovsky et al., 2013).

Unsupervised parsing (which is also our task) tries to turn the same corpus directly into a treebank, without necessarily finding a grammar. We discuss some recent milestones here. Grave and Elhadad (2015) propose a transductive learning objective for unsupervised parsing, and a convex relaxation of it. (Jiang et al. (2017) combined that work with grammar induction.) Martínez Alonso et al. (2017) create an unsupervised dependency parser that is formally similar to ours in that it uses cross-linguistic knowledge as well as statistics computed from a corpus of POS sequences in the target language. However, its cross-linguistic knowledge is hand-coded: namely, the set of POS-to-POS dependencies that are allowed by the UD annotation scheme, and the typical directions for some of these dependencies. The only corpus statistic extracted from $\mathbf{u}$ is whether `ADP-NOMINAL` or `NOMINAL-ADP` bigrams are more frequent,[6] which distinguishes prepositional from postpositional languages. The actual parser starts by identifying the head word as the most "central" word according to a PageRank (Page et al., 1999) analysis of the graph of candidate edges, and proceeds by greedily attaching words of decreasing PageRank at lower depths in the tree.

5.2 Multi-Language Learning

This approach parses a "target" language using the treebanks of other resource-rich languages as "source" languages. There are two main variants.

Memory-based. This method trains a supervised parsing model on each source treebank. It uses these (delexicalized) source-language models to help parse the target sentence, favoring sources that are similar to the target language. A common similarity measure (Rosa and Žabokrtský, 2015a) considers the probability of the target language's POS-corpus $\mathbf{u}$ under a trigram language model of source-language POS sequences.

Single-source transfer (SST) (Rosa and Žabokrtský, 2015a; Wang and Eisner, 2016) simply uses the parser for the *most similar* source treebank. Multi-source transfer (MST) (Rosa and Žabokrtský, 2015a) parses the target POS sequence with *each* of the source parsers, and then combines these parses into a consensus tree using the Chu-Liu-Edmonds algorithm (Chu, 1965; Edmonds, 1967). As a faster variant, model interpolation (Rosa and Žabokrtský, 2015b) builds a consensus *model* for the target language (via a weighted average of source models' parameters), rather than a consensus *parse* for each target sentence separately.

Memory-based methods require storing models for all source treebanks, which is expensive when we include thousands of GD treebanks (§3).

Model-based. This method trains a single *language-agnostic model*. McDonald et al. (2011) train a delexicalized parser on the concatenation of all source treebanks, achieving a large gain over grammar induction. This parser can learn universals such as the preference for determiners to attach to nouns (which was hard-coded by Naseem et al. (2010)). However, it is expected to parse a sentence x without being told the language ℓ or even a corpus $\mathbf{u}$, possibly by guessing properties of the language from the configurations it encounters in the single sentence x alone.

Further gains were achieved (Naseem et al., 2012; Täckström et al., 2013b; Zhang and Barzilay, 2015; Ammar et al., 2016) by *providing* the parser with about 10 typological properties of x's language—for example, whether direct objects generally fall to the right of the verb—as listed in the World Atlas of Linguistic Structures (Dryer and Haspelmath, 2013).

However, relying on WALS raises some issues. (1) The unknown language might not be in WALS.[7] (2) Some typological features are missing for some languages. (3) All the WALS features are categorical values, which loses useful information about tendencies (for example, how often the canonical word order is violated). (4) Not all WALS features are useful—only 56 of them pertain to word order, and only 8 of those have been

[6] In our notation of §6.1, below, this asks whether $\sum_{t\in\{\text{NOUN,PRON,PROPN}\}} \pi^{w}_{t|\text{ADP}}$ is greater for $w = 1$ or $w = -1$.

[7] 2,679 out of about 7,000 world languages are in WALS.

used in past work. (5) With a richer parser (a stack LSTM dependency parser), WALS features do not appear to help at all on unknown languages (Ammar et al., 2016, footnote 30).

5.3 Exploiting Parallel Data

Some other work on generalizing from source to target languages assumes the availability of source-target parallel data, or bitext. Two uses:

Induction of multilingual word embeddings. Similar to universal POS tags, multilingual word embeddings serve as a universal representation that bridges the lexical differences among languages. Guo et al. (2016) proposed two approaches: (1) Training a variant of the skip-gram model (Mikolov et al., 2013) by using bilingual sets of context words. (2) Generating the embedding of each target word by averaging the embeddings of the source words to which it is aligned.

Annotation projection. Given aligned bitext, one can generate an approximate parse for a target sentence by "projecting" the parse tree of the corresponding source sentence. A target-language parser can then be trained from these approximate parses. The idea was originally proposed by Yarowsky et al. (2001), and then applied to dependency parsing on low-resource languages (Hwa et al., 2005; Ganchev et al., 2009; Smith and Eisner, 2009; Tiedemann, 2014, *inter alia*). McDonald et al. (2011) extend this approach to multiple source languages by *projected transfer*. Later work in this vein mainly tries to improve the approximate parses, including translating the source treebanks into the target language with an off-the-shelf machine translation system (Tiedemann et al., 2014), augmenting the trees with weights (Agić et al., 2016), and using only partial trees with high-confidence alignments (Rasooli and Collins, 2015, 2017; Lacroix et al., 2016).

5.4 Situating Our Work

Our own approach can be categorized as model-based multi-language learning with no parallel text or target-side supervision. However, we also analyze an unparsed corpus $\mathbf{u}$ of the target language, as the per-language systems of §5.1 do. Our analysis of $\mathbf{u}$ does not produce a specialized target grammar or parser, but only extracts a target vector $\mathrm{T}(\mathbf{u})$ to be fed to the language-agnostic parser. The analyzer is trained jointly with the parser, over many languages.

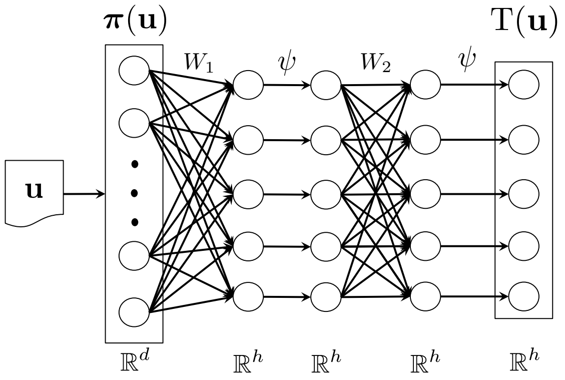

Figure 1: A 2-layer typology component. The bias vectors ($\mathbf{b}_W$) are suppressed for readability.

6 The Typology Component

Wang and Eisner (2017) extract typological properties of a language from its POS-tagged corpus $\mathbf{u}$, in effect predicting syntactic structure from superficial features. Like them, we compute a hidden layer $\mathrm{T}(\mathbf{u})$ using a standard multilayer perceptron architecture, for example,

$$\mathrm{T}(\mathbf{u}) = \psi(W\boldsymbol{\pi}(\mathbf{u}) + \mathbf{b}_W) \in \mathbb{R}^h \qquad (3)$$

where $\boldsymbol{\pi}(\mathbf{u}) \in \mathbb{R}^d$ is the surface features of $\mathbf{u}$, $W \in \mathbb{R}^{h \times d}$ maps $\boldsymbol{\pi}(\mathbf{u})$ into a h-dimensional space, $\mathbf{b}_W \in \mathbb{R}^h$ is a bias vector, and ψ is an element-wise activation function. While equation (3) has only 1 layer, we explore versions with from 0 to 3 layers (where $\mathrm{T}(\mathbf{u}) = \boldsymbol{\pi}(\mathbf{u})$ in the 0-layer case). A 2-layer version is shown in Figure 1. The number of layers is chosen by cross-validation, as are h and the ψ function.

6.1 Design of the Surface Features $\boldsymbol{\pi}(\mathbf{u})$

To define $\boldsymbol{\pi}(\mathbf{u})$, we used development data to select the following fast but effective subset of the features proposed by Wang and Eisner (2017).

Hand-engineered features. Given a token j in a sentence, let its **right window** R_j be the sequence of POS tags $p_{j+1}, \ldots, p_{j+w}$ (padding the sentence as needed with # symbols). w is the **window size**. Define $g^w(t \mid j) \in [0, 1]$ to be the fraction of words in R_j tagged with t. Now, given a corpus $\mathbf{u}$, define

$$\pi^w_t = \operatorname*{mean}_j g^w(t \mid j), \quad \pi^w_{t|s} = \operatorname*{mean}_{j:T_j=s} g^w(t \mid j)$$

where j ranges over tokens of $\mathbf{u}$. The **unigram prevalence** π^w_t measures the frequency of t overall, while the **bigram prevalence** $\pi^w_{t|s}$ measures the frequency with which t can be found to the left of an average s tag (in a window of size w). For each of these quantities, we have a corresponding mirror-image quantity (denoted by negating w) by computing it on a reversed version of the corpus.

The final hand-engineered $\boldsymbol{\pi}(\mathbf{u})$ includes:

- π_t^w, for each tag type t and each $w \in \{1, 3, 8, 100\}$. This quantity measures how frequently t appears in $\mathbf{u}$.

- $\pi_{t|s}^w // \pi_t^w$ and $\pi_{t|s}^{-w} // \pi_t^{-w}$, for each tag type pair s, t and each $w \in \{1, 3, 8, 100\}$. We define $x//y = \min(x/y, 1)$ to bound the feature values for better generalization. Notice that if $w = 1$, the log of $\pi_{t|s}^w / \pi_t^w$ is the bigram pointwise mutual information. Each matched pair of these quantities is intuitively related to the word order typology—for example, if `ADP`s are more likely to have closely following than closely preceding `NOUN`s ($\pi_{\texttt{NOUN|ADP}}^w // \pi_{\texttt{NOUN}}^w > \pi_{\texttt{NOUN|ADP}}^{-w} // \pi_{\texttt{NOUN}}^{-w}$), the language is more likely to be prepositional than postpositional.

Neural features. In contrast, our neural features automatically learn to extract arbitrary predictive configurations. As Figure 2 shows, we encode each POS-tagged sentence $u_i \in \mathbf{u}$ using a recurrent neural network, which reads one-hot POS embeddings from left to right, then outputs its final hidden state vector $\mathbf{f}_i$ as the encoding. The final neural $\boldsymbol{\pi}(\mathbf{u})$ is the average encoding of all sentences (average-pooling): that is, the average of all sentence-level configurations. We specifically use a gated recurrent unit (GRU) network (Cho et al., 2014). The GRU is jointly trained with all other parameters in the system so that it focuses on detecting word-order properties of $\mathbf{u}$ that are useful for parsing.

7 The Parsing Architecture

To construct $\text{Parse}(x; \mathbf{u})$, we can extend any statistical parsing architecture $\text{Parse}(x)$ to be sensitive to $\text{T}(\mathbf{u})$. For our experiments, we extend the delexicalized graph-based implementation of the BIST parser (Kiperwasser and Goldberg, 2016)—an arc-factored dependency model with neural context features extracted by a bidirectional LSTM. This recent parser was the state of the art when it was published.

Given a POS-sentence x and a corpus $\mathbf{u}$, our parser first computes an unlabeled projective tree

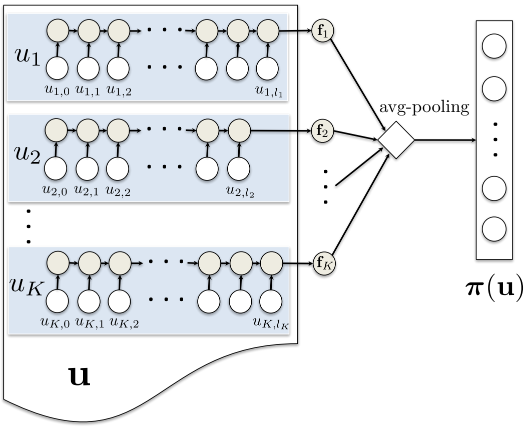
u_1
$u_{1,0}$ $u_{1,1}$ $u_{1,2}$
u_{1,l_1}
u_2
$u_{2,0}$ $u_{2,1}$ $u_{2,2}$
u_{2,l_2}
u_K
$u_{K,0}$ $u_{K,1}$ $u_{K,2}$
u_{K,l_K}
$\mathbf{u}$
f_1
f_2
f_K
avg-pooling
$\boldsymbol{\pi}(\mathbf{u})$

Figure 2: Computing the neural feature vector $\boldsymbol{\pi}(\mathbf{u})$.

$$\underset{y \in \mathcal{Y}(x)}{\text{argmax}} \; \text{score}(x, y; \mathbf{u}) \tag{4}$$

where, letting a range over the arcs in tree y,

$$\text{score}(x, y; \mathbf{u}) = \sum_{a \in y} s(\phi(a; x, \mathbf{u})) \tag{5}$$

With this definition, the argmax in (4) is computed efficiently by the algorithm of Eisner (1996).

$s(\cdot)$ is a neural scoring function on vectors,

$$s(\phi(\cdots)) = \mathbf{v} \tanh(V \phi(\cdots) + \mathbf{b}_V) \tag{6}$$

where V is a matrix, $\mathbf{b}_V$ is a bias vector, and $\mathbf{v}$ is a vector, all being parameters in $\boldsymbol{\Theta}$.

The function $\phi(a; x, \mathbf{u})$ extracts the feature vector of arc a given x and $\mathbf{u}$. BIST scores unlabeled arcs, so a denotes a pair (i, j)—the indices of the parent and child, respectively. We define

$$\phi(a; x, \mathbf{u}) = [\text{B}(x, i; \text{T}(\mathbf{u})); \text{B}(x, j; \text{T}(\mathbf{u}))] \tag{7}$$

which concatenates contextual representations of tokens i and j. $\text{B}(x, i)$ is itself a concatenation of the hidden states of a left-to-right LSTM and a right-to-left LSTM (Graves, 2012) when each has read sentence x up through word i (really POS tag i). These LSTM parameters are included in $\boldsymbol{\Theta}$.

The POS tags in x are provided to the LSTMs as one-hot vectors. Crucially, $\text{T}(\mathbf{u})$ is also provided to the LSTM at each step, as shown in Figure 3.

After selecting the best tree via equation (4), we use each arc's ϕ vector again to predict its label. This yields the labeled tree $\hat{y} = \text{Parse}_{\boldsymbol{\Theta}}(x; \mathbf{u})$.

The only extension that this makes to BIST is

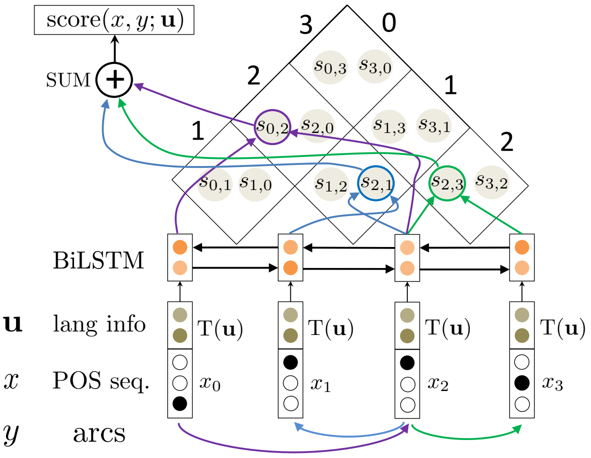

Figure 3: The architecture of the delexicalized graph-based BIST parser with the introduction of $\mathrm{T}(\mathbf{u})$, where $s_{i,j}$ in each cell is the arc score $s(\phi(a; x, \mathrm{T}(\mathbf{u}))$ from equation (6). The root of the tree is always position 0, where x_0 is a distinguished "root" symbol that is prepended to the input sentence.

to supply $\mathrm{T}(\mathbf{u})$ to the BiLSTM.[8] This extension is not a significant slowdown at test time, since $\mathrm{T}(\mathbf{u})$ only needs to be computed once per test language, not once per test sentence. Since $\mathrm{T}(\mathbf{u})$ can be computed for any novel language at test time, this differs from the "many languages, one parser" architecture (Ammar et al., 2016), in which a test-time language must have been seen at training time or at least must have known WALS features.

Product of experts. We also consider a variant of the function (6) for scoring arc a, namely

$$\lambda s_{\mathrm{H}}(a) + (1 - \lambda) s_{\mathrm{N}}(a) \qquad (8)$$

where $s_{\mathrm{H}}(a)$ and $s_{\mathrm{N}}(a)$ are the scores produced by separately trained systems using, respectively, the hand-engineered and neural features from §6.1. Hyperparameter $\lambda \in [0, 1]$ is tuned on dev data.

8 Training the System

8.1 Training Objective

We exactly follow the training method of Kiperwasser and Goldberg (2016), who minimize a structured max-margin hinge loss (Taskar et al., 2004; McDonald et al., 2005; LeCun et al., 2007). We want the correct tree y to beat each tree y' by a margin equal to the number of errors in y' (we count spurious edges). Formally, $\mathrm{loss}(x, y; \mathbf{u})$ is given by

$$\max(0, -\mathrm{score}(x, y; \mathbf{u}) + \max_{y'} \Big(\underbrace{\mathrm{score}(x, y'; \mathbf{u})}_{\text{model score}} + \underbrace{\sum_{a \in y'} \mathbb{1}_{a \notin y}}_{\text{precision error}} \Big)) \qquad (9)$$

where a ranges over the arcs of a tree y, and $\mathbb{1}_{a \notin y}$ is an indicator that is 1 if $a \notin y$. Thus, this loss function is high if there exists a tree y' that has a high score relative to y yet low precision.[9]

The training algorithm makes use of loss-augmented inference (Taskar et al., 2005), a variant on the ordinary inference of (4). The most violating tree y' (in the $\max_{y'}$ above) is computed again by an arc-factored dependency algorithm (Eisner, 1996), where the score of any candidate arc a is $s(\phi(a; x, \mathbf{u})) + \mathbb{1}_{a \notin y}$.

Actually, the above method would only train the score function to predict the correct *unlabeled* tree as above (since a ranges over unlabeled arcs as before). In practice, we also jointly train the labeler to predict the correct labels on the gold arcs, using a separate hinge-loss objective. Because these two components share parameters through $\phi(a; x, \mathbf{u})$, this is a multi-task learning problem.

8.2 Training Algorithm

Augment training data. Unlike ordinary NLP problems whose training examples are sentences, each training example in equation (1) is an entire language. Unfortunately, UD (§3) only provides a few dozen languages—presumably not enough to generalize well to novel languages. We therefore augment our training dataset $\mathcal{L}_{\texttt{train}}$ with thousands of synthetic languages from the GD dataset (§3), as already discussed in §3.1.

Stochastic gradient descent (SGD).[10] Treating each language as a single large example during training would lead to slow SGD steps. Instead, we take our SGD examples to be individual sentences, by regarding equations (1)–(2) together as

[8] An alternative would be to concatenate $\mathrm{T}(\mathbf{u})$ with the representation computed by the BiLSTM. This gets empirically worse results, probably because the BiLSTM does not have advance knowledge of language-specific word order as it reads the sentence. We also tried an architecture that does both, with no notable improvement.

[9] Formally, for this loss function to be used in equation (2), we must interpret Parse_{Θ} in that equation as returning a forest of scored parses, not just a single parse.

[10] More precisely, we use Adam (Kingma and Ba, 2015), a popular variant of SGD. The parameters $\mathbf{\Theta}$ are initialized by "Xavier initialization" (Glorot and Bengio, 2010).

an objective averaged over sentences. Each example $(x, y, \mathbf{u})$ is sampled hierarchically, by first drawing a language ℓ from $\mathcal{L}_{\texttt{train}}$ and setting $\mathbf{u} = \mathbf{u}^{(\ell)}$, then drawing the sentence (x, y) uniformly from $(\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)})$. We train using mini-batches of 100 sentences; each mini-batch can mix many languages.

Encourage real languages. To sample ℓ from $\mathcal{L}_{\texttt{train}}$, we first flip a coin with weight $\beta \in [0, 1]$ to choose "real" vs. "synthetic," and then sample uniformly within that set. Why? The test sentences will come from real languages, so the synthetic languages are out-of-domain. Including them reduces variance but increases bias. We raise β to keep them from overwhelming the real languages.

Sample efficiently. The sentences (x, y) are stored in different files by language. To reduce disk accesses, we do not visit a file on each sample. Rather, for each language ℓ, we maintain in memory a *subset* of $(\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)})$, obtained by reservoir sampling. Samples from $(\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)})$ are drawn sequentially from this "chunk," and when it is used up we fetch a new chunk. We also maintain $\mathbf{u}^{(\ell)}$ and the hand-engineered features from $\boldsymbol{\pi}(\mathbf{u}^{(\ell)})$ in memory.

9 Experiments

9.1 Basic Setup

Our data split follows that of Wang and Eisner (2017), as shown in Table 2,[11] which has 18 training languages (20 treebanks) and 17 test languages. All hyperparameters are tuned via 5-fold cross-validation on the 20 training treebanks—that is, we evaluate each fold (4 treebanks) using the model trained on the remaining folds (16 treebanks). However, a model trained on a treebank of language ℓ is never evaluated on another treebank of language ℓ. We selected the hyperparameters that maximized the average unlabeled attachment score (UAS) (Kübler et al., 2009), which is the evaluation metric that is reported by most previous work on unsupervised parsing. We also report labeled attachment score (LAS).[12]

When augmenting the data, the 16 training treebanks are "mixed and matched" to get GD treebanks for $16 \times 17 \times 17 = 4624$ additional synthetic training languages (Wang and Eisner, 2016, §5).

The next sections analyze these cross-validation results. Finally, §9.8 will evaluate on 15 *previously unseen* languages (excluding Latin and Finnish$_{\text{ftb}}$) with our model trained on all 18 training languages (20 treebanks for UD, plus $20 \times 21 \times 21 = 8840$ when adding GD) with the hyperparameters that achieved the best average unlabeled attachment score during cross-validation.

The UD and GD corpora provide a train/dev/test split of each treebank, denoted as $(\mathbf{x}_{train}, \mathbf{y}_{train})$, $(\mathbf{x}_{dev}, \mathbf{y}_{dev})$ and $(\mathbf{x}_{test}, \mathbf{y}_{test})$. Throughout this paper, for both training and testing languages, we take $(\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)}) = (\mathbf{x}_{train}, \mathbf{y}_{train})$. We take $\mathbf{u}^{(\ell)}$ to consist of all $\mathbf{x}_{train}$ sentences with ≤ 40 tokens.

9.2 Comparison Among Architectures

Table 1 shows the cross-validation parsing results over different systems discussed so far. For each architecture, we show the best average unlabeled attachment score (the UAS column) chosen by cross-validation, and the corresponding labeled attachment score (the LAS column). In brief, the main sources of improvement are twofold:

Synthetic languages. We observe that +GD consistently outperforms UD across all architectures. It even helps with the baseline system that we tried, which simply *ignores the target corpus* $\mathbf{u}^{(\ell)}$. In that system (similar to McDonald et al. (2011)), the BiLSTM may still manage to extract ℓ-specific information from the single sentence $x \in \mathbf{x}^{(\ell)}$ that it is parsing.[13] The additional GD training languages apparently help it learn to do so in a way that generalizes to new languages.

[11] However, as we are interested in transfer to *unseen* languages, our Table 2 follows the principle of Eisner and Wang (n.d.) and does not test on the Finnish$_{\text{ftb}}$ or Latin treebanks because other treebanks of those languages appeared in training data. Specifically, Latin$_{\text{itt}}$ and Latin$_{\text{proiel}}$ fall in the same training folds as French and Italian, respectively. For the same reason, Table 2 does not show cross-validation development results on these Latin treebanks—nor on the Ancient Greek$_{\text{grc}}$ and Ancient Greek$_{\text{grc\_proiel}}$ treebanks, which fall in the same training folds as Czech and Danish, respectively.

[12] When reporting LAS and when studying the labeling errors in §9.7, we would ideally have first re-tuned our system to optimize LAS via cross-validation. Unfortunately, these potentially improved LAS results would have required months of additional computation. The optimal hyperparameters may not be very different, however, since UAS and LAS rose and fell together when we varied other training conditions in Figures 4–6.

[13] That is, our baseline system has learned a single parser that can handle a cross-linguistic variety of POS sequences (cf. McDonald et al., 2011; Ammar et al., 2016, section 4.2), just as the reader was able to parse VERB DET NOUN ADJ DET NOUN in §2.

		UAS		LAS	
	System	UD	+GD	UD	+GD
	SST	**66.22***	**65.70**	**50.40**	**50.54**
	Baseline	63.95	**67.97**	48.46	**52.78**
	H	64.83	**69.41**	49.41	**53.63**
	N	65.30	**70.06**	49.43	**54.19**
	H;N	65.26	**69.62**	49.67	**53.68**
	H+N	67.34*	**70.65***	52.02*	**55.18***
oracle features	$\mathrm{T_D}$	65.94	**70.01***	49.77	**53.43**
	$\mathrm{T_W}$	64.84	**69.75**	49.30	**53.79**

Table 1: Average parsing results over 16 languages, computed by 5-fold cross-validation. We compare training on real languages only (the UD column) versus augmenting with synthetic languages at $\beta = 0.2$ (the +GD column). Baseline is the ablated system that omits $\mathrm{T}(\mathbf{u})$ (§9.2). SST is the single-source transfer approach (§5.2). H and N use only hand-engineered features or neural features, while H;N defines $\boldsymbol{\pi}(\mathbf{u})$ to concatenate both (§6.1) and H+N is the product-of-experts model (§7). $\mathrm{T_D}$ and $\mathrm{T_W}$ that incorporate oracle knowledge of the target-language syntax (§9.4). For each comparison between UD and +GD, we boldface the better (higher) result, or both if they are not significantly different (paired permutation test over languages with $p < 0.05$). In each column, we star the best result as well as all results that are not significantly worse.

To better understand the trend, we study how the performance varies when more synthetic languages are used. As shown in Figure 4, when $\beta = 1$, all the training languages are sampled from real languages. By gradually increasing the proportion of GD languages (reducing β from §8.2), the baseline UAS increases dramatically from 63.95 to 67.97. However, if all languages are uniformly sampled ($\beta = \frac{16}{4624+16} \approx 0.003$) or only synthetic languages are used ($\beta = 0$), the UAS falls back slightly to 67.42 or 67.36. The best β value is 0.2, which treats each real language as $\frac{0.2/16}{0.8/4624} \approx 72$ times more helpful than each synthetic language, yet 80% of the training data is contributed by synthetic languages. $\beta = 0.2$ was also optimal for the non-baseline systems in Table 1.

Unparsed corpora. The systems that exploit unparsed corpora consistently outperform the baseline system in both the UD and +GD conditions. To investigate, we examine the impact of reducing $\mathbf{u}^{(\ell)}$ when parsing a held-out language ℓ. We used the system in row N and column +GD of Table 1, which was trained on full-sized $\mathbf{u}$ corpora. When testing on a held-out language ℓ, we compute $\mathrm{T}(\mathbf{u}^{(\ell)})$ using only a random size-t subset of $\mathbf{u}^{(\ell)}$. As shown in Figure 5, the system does not need a very large unparsed corpus—most of the benefit is obtained by $t = 256$. Nonetheless, a larger corpus always achieves a better and more stable performance.

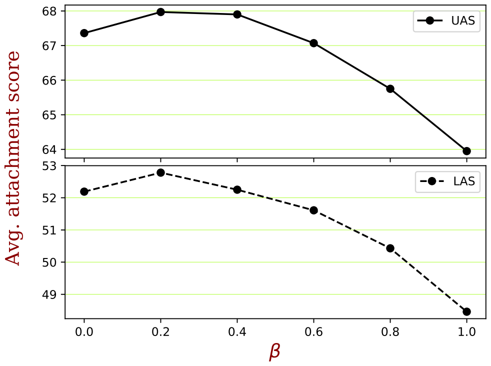

Figure 4: Effect of β. The UAS and LAS (y-axis) of the baseline system as a function of β (x-axis).

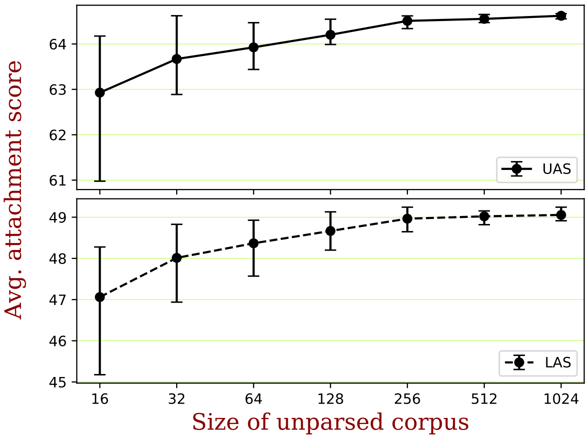

Figure 5: Effect of the size $|\mathbf{u}^{(\ell)}|$ of the unparsed corpus. The y-axis represents the cross-validation UAS and LAS scores, averaged over the 7 languages that have $|\mathbf{u}^{(\ell)}| \geq 9000$ sentences, when using only a subset of the sentences from $\mathbf{u}^{(\ell)}$. Using all of $\mathbf{u}^{(\ell)}$ would achieve 64.61 UAS and 49.04 LAS. The plot shows the average over 10 runs with different random subsets; the error bars indicate the 10th to the 90th percentile of those runs. The 7 languages are Finnish (Finnic), Norwegian (Germanic), Dutch (Germanic), Czech (Slavic), German (Germanic), Hindi (Indic), and English (Germanic).

9.3 Comparison to SST

Besides Baseline, another directly comparable approach is SST (§5.2). As shown in Table 1, SST gives a stronger baseline on the UD column—as good as H+N. However, this advantage does not carry over to the +GD column, meaning that SST cannot exploit the extra training data. Wang and Eisner (2016, Figure 5) already found that GD languages provide diminishing benefit to SST as more UD languages get involved.[14] For H+N, however, the extra GD languages do help to identify the truly useful surface patterns in $\mathbf{u}$.

We also considered trying model interpolation (Rosa and Žabokrtský, 2015b). Unfortunately, as mentioned in §5.2, this method is impractical with GD languages, because it requires storing 4624 (§9.1) additional local models. Nonetheless, we can estimate an "upper bound" on how well the interpolation might do. Our upper bound is SST where an oracle is used to choose the source language; Rosa and Žabokrtský (2015b) found that in practice, this does better than interpolation. This approximate upper bound is 68.03 of UAS and 52.10 of LAS, neither of which is significantly better than H+N on UD, but both of which are significantly outperformed by H+N on +GD.

9.4 Oracle Typology vs. Our Learned $\mathrm{T}(\mathbf{u})$

The results in Table 1 demonstrate that we learned to extract features $\mathrm{T}(\mathbf{u})$, from the unparsed target corpus $\mathbf{u}$, that improve the baseline parser. We consider replacing $\mathrm{T}(\mathbf{u})$ by an oracle that has access to the true syntax of the target language. We consider two different oracles, T_{D} and T_{W}.

T_{D} is the *directionalities* typology that was studied by Liu (2010) and used as a training target by Wang and Eisner (2017). Specifically, $\mathrm{T}_{\mathrm{D}} \in [0,1]^{57}$ is a vector of the directionalities of each type of dependency relation; it specifies what fraction of direct objects fall to the right of the verb, and so on.[15] In principle, this should be very helpful for parsing, but it must be extracted from a treebank, which is presumably unavailable for unknown languages.

We also consider T_{W}—the WALS features—as the typological classification given by linguists. This resembles the previous multi-language learning approaches (Naseem et al., 2012; Täckström et al., 2013b; Zhang and Barzilay, 2015; Ammar et al., 2016) that exploited the WALS features. In particular, we use 81A, 82A, 83A, 85A, 86A, 87A, 88A and 89A—a union of WALS features used by those works. In order to derive the WALS features for a synthetic GD language, we first copy the features from its substrate language (Wang and Eisner, 2016). We then replace the 81A, 82A, 83A features—which concern the order between verbs and their dependents—by those of its $\mathtt{V}$-superstrate language[16] (if any). We replace 85A, 86A, 87A, 88A and 89A—which concern the order between nouns and their dependents—by those of its $\mathtt{N}$-superstrate language (if any).

As a pleasant surprise, we find that our best system (H+N) is competitive with both oracle methods. It outperforms both of them on both UAS and LAS, and the improvements are significant and substantial in 3 of these 4 cases. Our parser has learned to extract information $\mathrm{T}(\mathbf{u})$ that is not only cheap (no treebank needed), but also at least as useful as "gold" typology for parsing.

9.5 Selected Hyperparameter Settings

For the rest of the experiments, we use the H+N system, as it wins under cross-validation on both UD and +GD (Table 1). This is a combination via (8) of the best H system and the best N system under cross-validation, with the mixture hyperparameter λ also chosen by cross-validation.

For both UD and +GD, cross-validation selected 125 as the sizes of the LSTM hidden states and 100 as the sizes of the hidden layers for scoring arcs (the length of $\mathbf{v}$ in equation (6)).

Hyperparameters for UD. The H system computes $\mathrm{T}(\mathbf{u})$ with a 1-layer network (as in equation (3)), with hidden size $h = 128$ and $\psi = \tanh$ as the activation function. For the N system, $\mathrm{T}(\mathbf{u})$ is a 1-layer network with hidden size $h = 64$ and $\psi = \mathrm{sigmoid}$ as the activation function. The size of the hidden state of GRU as shown in Figure 2 is 128. The mixture weight for the final H+N system is $\lambda = 0.5$.

Hyperparameters for +GD. The H system computes $\mathrm{T}(\mathbf{u})$ with a 2-layer network (as shown in Figure 1), with $h = 128$ and $\psi = \mathrm{sigmoid}$ for both hidden layers. For N, $\mathrm{T}(\mathbf{u})$ is a 1-layer network with hidden size $h = 64$ and $\psi = \mathrm{sigmoid}$. The size of the hidden state of GRU is 256. Both H and N set $\beta = 0.2$ (see §8.2). The mixture weight for the final H+N system is $\lambda = 0.4$.

9.6 Performance on Noisy Tag Sequences

We test our trained system in a more realistic scenario where both $\mathbf{u}$ and x for held-out languages

[14] The number of real treebanks in our cross-validation setting is 16, greater than the 10 in Wang and Eisner (2016).

[15] The directionality of a relation a in language ℓ is given by $\frac{\mathrm{count}_\ell(\overset{a}{\rightarrow})}{\mathrm{count}_\ell(a)}$, where $\mathrm{count}_\ell(\overset{a}{\rightarrow})$ is the count of a-relations that point from left to right, and $\mathrm{count}_\ell(a)$ is the count of all a-relations.

[16] The language whose word order model is used to permute the dependents of the verbs. See Wang and Eisner (2016) for details.

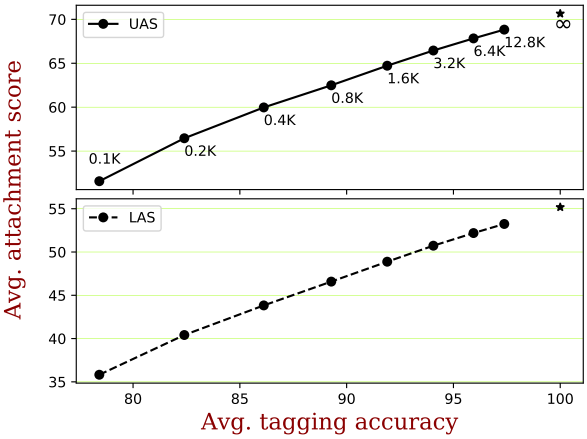

Figure 6: Performance on noisy input over 16 training languages. Each dot is an experiment annotated by the number of sentences used to train the tagger. (The rightmost "∞" point uses gold tags instead of a tagger, which is the result from Table 1.) The x-axis gives the average accuracy of the trained RDRPOSTagger. The y-axis gives the average parsing performance.

consist of noisy POS tags rather than gold POS tags. Following Wang and Eisner (2016, Appendix B), at test time, the gold POS tags in a corpus are replaced by a noisy version produced by the RDRPOSTagger (Nguyen et al., 2014) trained on a subset of the original gold-tagged corpus.[17] Figure 6 shows a linear relationship between the performance of our best model (H+N with +GD) and the noisiness of the POS tags, which is controlled by altering the amount of training data. With only 100 training sentences, the performance suffers greatly—the UAS drops from 70.65 to 51.57. Nonetheless, even this is comparable to Naseem et al. (2010) on *gold* POS tags, which yields a UAS of 50.00. That system was the first grammar induction approach to exploit knowledge of the distribution of natural languages, and remained state-of-the-art (Noji et al., 2016) until the work of Mareček (2016) and Martínez Alonso et al. (2017).

9.7 Analysis by Dependency Relation Type

Figure 7 breaks down the results by dependency relation type—showing that using $\mathbf{u}$ and synthetic data improves results *almost across the board.*

We also notice large differences between labeled and unlabeled F1 scores for some relations, especially rarer ones. In other words, the system mislabels the arcs that it correctly recovers. (Remember from §9.2 that the hyperparameters were selected to maximize unlabeled scores (UAS) rather than labeled (LAS).)

Figure 8 gives the label confusion matrix. While the dark NONE column shows that arcs of each type are often missed altogether (recall errors), the dark diagonal shows that they are usually labeled correctly if found. That said, it is relatively common to confuse the different labels for nominal dependents of verbs (`nsubj`, `dobj`, `nmod`). We suspect that lexical information could help sort out these roles via distributional semantics. Some other mistakes arise from discrepancies in the annotation scheme. For example, `neg` can be easily confused with `advmod`, as some languages (for example, Spanish) use ADV instead of PART for negations.

9.8 Final Evaluation on Test Data

In all previous sections, we evaluated on the 16 languages in the training set by cross-validation. For the final test, we combine all the 20 treebanks and train the system with the hyperparameters given in §9.5, then test on the 15 unseen test languages. Table 2 displays results on these 15 test languages (top) as well as the cross-validation results on the 16 languages (bottom).

We see that we improve significantly over baseline on almost every language. Indeed, on the test languages, +T($\mathbf{u}$) improves both UAS and LAS by $>$ 3.5 percentage points on average. The improvement grows to $>$ 5.6 if we augment the training data as well (+GD, meaning +T(u)+GD).

One disappointment concerns the *added* benefit on the **LAS** of +GD over just +T($\mathbf{u}$): while this data augmentation helped significantly on nearly every one of the 16 development languages, it produced less consistent improvements on the test languages and hurt some of them. We suspect that this is because we tuned the hyperparameters to maximize UAS, not LAS (§9.2). As a result, while the *average* benefit across our 15 test languages was fairly large, this sample was not large enough to establish that it was significantly greater from 0, that is, that future test languages would also see an improvement from data augmentation.

We also notice that there seems to be a small difference between the pattern of results on development versus test languages. This may simply reflect overfitting to the development languages, but we also note that the test languages (chosen by

[17] Another way to get noisy tags, as a reviewer notes, would have been to use a cross-lingual POS tagger designed for low-resource settings (Täckström et al., 2013a; Kim et al., 2017).

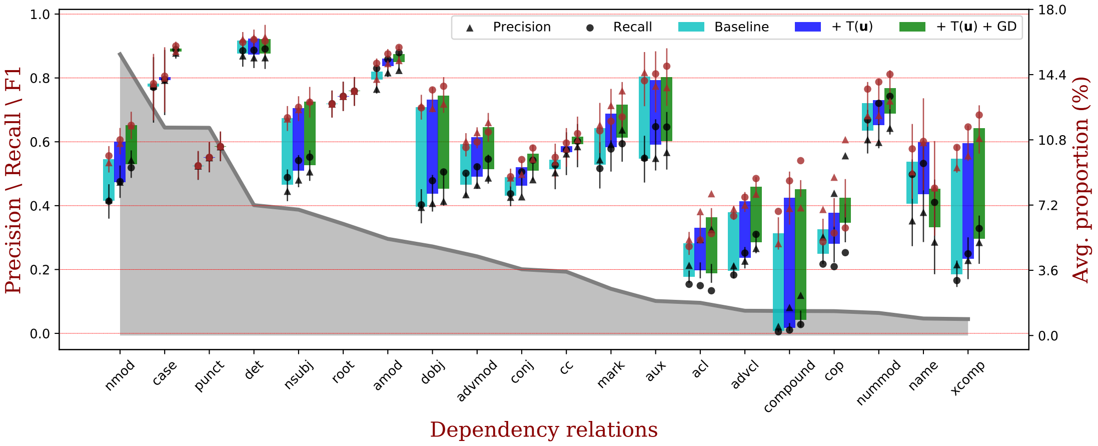

Figure 7: Evaluation by dependency relation type, showing an equal-weighted average of the 16 development languages. Each vertical bar spans the range from labeled F1 (bottom) to unlabeled F1 (top), with error bars given by bootstrap resampling over the 16 languages. Precision and recall are also indicated. The pattern is that F1, precision, and recall—both labeled and unlabeled—are improved over baseline when we exploit unlabeled corpora (+T($\mathbf{u}$)), and improved again when we augment training data (+T($\mathbf{u}$)+GD). The relations are sorted by their average gold proportion in the 16 languages, shown by the gray area and right vertical axis. For example, nmod is the most common relation, accounting for 15.5% of all arcs. Altogether, the 20 most frequent relations (shown here) account for 94% of the arcs.

Wang and Eisner (2016)) tended to have considerably smaller unparsed corpora $\mathbf{u}$, so there may be a domain mismatch problem. To ameliorate this problem, one could include training examples with versions of $\mathbf{u}$ that are truncated to lengths seen in test data (cf. Figure 5). One could also include the size $|\mathbf{u}|$ explicitly in $\mathrm{T}(\mathbf{u})$.

10 Conclusion and Future Work

We showed how to build a "language-agnostic" delexicalized dependency parser that can better parse sentences of an unknown language by exploiting an unparsed (but POS-tagged) corpus of that language. Unlike grammar induction, which estimates a PCFG from the unparsed corpus, we train a neural network to extract a feature vector from the unparsed corpus that helps a subsequent neural parser. By end-to-end training on the treebanks of many languages (optionally including synthetic languages), our neural network can extract linguistic information that helps neural dependency parsing.

Variants of our architecture are possible. In future work, the neural parser could use *attention* to look at individual relevant sentences of $\mathbf{u}$, which are posited to be *triggers* in some theories of child grammar acquisition (Gibson and Wexler, 1994; Frank and Kapur, 1996). We could also try injecting $\mathrm{T}(\mathbf{u})$ into the neural parser by means other than concatenating it with the input POS embeddings. We might also consider parsing architectures other than BIST, such as the LSTM-Minus architecture for scoring spans (Cross and Huang, 2016), or the recent attention-based arc-factored model (Dozat and Manning, 2017). Finally, our approach is applicable to tasks other than dependency parsing, such as constituent parsing or semantic parsing—if suitable treebanks are available for many training languages.

For applied uses, it would be interesting to combine the *unsupervised* techniques of this paper with *low-resource* techniques that make use of some annotated or parallel data in the target language. It would also be interesting to include further synthetic languages that have been modified to better resemble the actual target languages, using the method of (Wang and Eisner, 2018).

It is important to relax the delexicalized assumption. As shown in §9.6, the performance of our system relies heavily on the gold POS tags, which are presumably not available for unknown languages. What is available is lexical information—which has proved to be very important for supervised parsing, and should help unsupervised parsers as well. As discussed in §9.7, some errors seem easily fixable by considering word distributions. In the future, we will explore ways to extend our cross-linguistic parser to work with word sequences rather than POS sequences, perhaps by learning a cross-language word representation that is shared among training and test languages (Ruder et al., 2017).

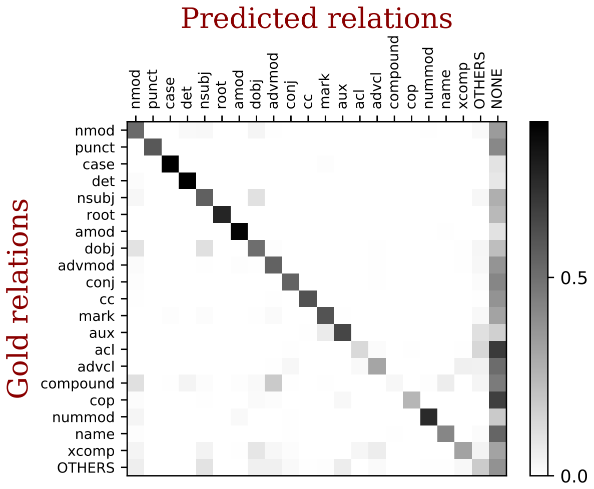

Figure 8: The confusion matrix of our parser, as an equal-weight average over 16 development languages. Each row is normalized to sum to 1 and represents a frequent gold relation. For example, the `nsubj` row shows how well we recovered the gold `nsubj` arcs; the (`nsubj`, `dobj`) entry shows $p(\text{predicted} = \texttt{dobj} \mid \text{gold} = \texttt{nsubj})$, which measures the fraction of `nsubj` relations that are recovered but mislabeled as `dobj`. The diagonal represents correct arcs: where dark, it indicates high labeled recall for that relation. The final column represents gold arcs that were not recovered with any label: where dark, it indicates low unlabeled recall for that relation. We show the top 20 relations sorted by gold frequency.

One takeaway message from this work is contained in our title. Surface statistics of a language—mined from the surface part-of-speech order—provide clues about how to find the underlying syntactic dependencies. Chomsky (1965) imagined that such clues might be exploited by a Language Acquisition Device, so it is interesting to know that they do exist.

Another takeaway message is that synthetic training languages are useful for NLP. Using synthetic examples in training is a way to encourage a system to be invariant to superficial variation. We created synthetic languages by varying the surface structure in a way that "should" preserve the deep structure. This allows our trained system to be invariant to variation in surface structure, just as object recognition wants to be invariant to an image's angle or lighting conditions (§3.1).

Our final takeaway goes beyond language: one can treat unsupervised structure discovery as a supervised learning problem. As §§1–2 discussed, this approach inherits the advantages of supervised learning. Training may face an easier optimization landscape, and we can train the system to find the *specific* kind of structure that we desire, using any features that we think may be discriminative.

	UAS			LAS		
Language	B	+T($\mathbf{u}$)	+GD	B	+T($\mathbf{u}$)	+GD
Basque	49.89	54.34	**57.59**	27.07	31.46	**35.32**
Croatian	65.03	**67.78**	**68.65**	48.68	52.29	**53.68**
Greek	65.91	68.37	**70.46**	50.1	56.73	**57.89**
Hebrew	62.58	**66.27**	65.3	49.71	**53.29**	52.08
Hungarian	58.5	64.13	**70.02**	42.85	47.73	**49.99**
Indonesian	55.22	**64.63**	**65.36**	39.46	**47.63**	**48.38**
Irish	58.58	61.51	**62.21**	39.06	40.75	**42.36**
Japanese	54.97	**60.41**	58.4	37.57	**40.6**	37.86
Slavonic	**68.79**	**71.13**	**71.54**	40.03	**43.95**	**44.12**
Persian	40.38	34.2	**57.25**	30.06	24.6	**47.14**
Polish	72.15	76.85	**78.28**	50.08	54.85	**58.15**
Romanian	66.55	69.69	**71.18**	50.9	53.42	**55.17**
Slovenian	72.21	76.06	**78.62**	57.09	61.48	**64.1**
Swedish	72.26	**75.32**	**73.89**	55.35	**58.42**	52.39
Tamil	51.59	**57.53**	**57.91**	28.39	**37.81**	32.52
Avg.	60.97	64.55	**67.11**	43.09	**47.00**	**48.74**
Arabic	45.75	49.32	**53.83**	36.4	40.39	**44.14**
Danish	66.71	**68.41**	**68.4**	52.24	**54.49**	**54.67**
Norwegian	68.35	**70.89**	**71.22**	52.33	**56.01**	**56.37**
Czech	64.31	68.77	**72.42**	50.19	55.16	**57.95**
Estonian	72.67	79.88	**81.67**	42.81	51.32	**52.57**
Portuguese	70.48	73.47	**74.83**	60.85	63.18	**64.96**
German	62.18	63.62	**66.52**	48.44	49.46	**53.51**
Gothic	63.23	66.72	**70.75**	39.1	42.6	**45.17**
Italian	75.9	79.24	**80.57**	65.46	68.8	**70.0**
Bulgarian	77.57	79.53	**83.66**	55.83	57.65	**61.47**
Finnish$_{\text{fi}}$	53.73	58.03	**60.44**	34.68	39.55	**43.15**
French	74.57	76.88	**79.34**	64.1	66.83	**68.48**
Dutch	59.63	**62.58**	60.31	45.84	**48.28**	47.98
English	61.66	63.99	**65.9**	47.61	51.43	**53.13**
Hindi	35.84	40.74	**62.45**	18.63	21.65	**41.12**
Spanish	70.65	75.36	**78.03**	60.8	65.45	**68.23**
All Avg.	62.51	65.99	**68.94**	45.86	49.59	**52.07**

Table 2: Data splits and final evaluation on the 15 test languages (top), along with cross-validation results on the 16 development languages (bottom) grouped by 5 folds (separated by dashed lines). For languages with multiple treebanks, we identify them by subscripts. We use "Slavonic" for Old Church Slavonic. Column B is the baseline that doesn't use $\mathrm{T}(\mathbf{u})$ (McDonald et al., 2011). $+\mathrm{T}(\mathbf{u})$ is our H+N system, and +GD is that system when the training data is augmented with synthetic languages. In comparing among these three systems, we boldface the highest score as well as all scores that are not significantly worse (paired permutation test, $p < 0.05$). If a row is an average over many sentences of a single language, then each paired datapoint is a *sentence*, so a significant improvement should generalize to new sentences. But if a row is an average, then each paired datapoint is a *language* (as in Table 1), so a significant improvement should generalize to new languages.

Acknowledgements This material is based upon work supported by the U.S. National Science Foundation under Grants No. 1423276 and 1718846. We are grateful to the state of Maryland for providing indispensable computing resources via the Maryland Advanced Research Computing Center (MARCC). Thanks to Kiperwasser and Goldberg (2016) for releasing their Dynet parsing code, which was the basis for our reimplementation in PyTorch. We thank the Argo lab members for useful discussions. Finally, we thank TACL action editor Marco Kuhlmann and the anonymous reviewers for high-quality suggestions.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association of Computational Linguistics*, 4:431–444.

Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. 2013. Powergrading: A clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1:391–402.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Working Notes of the Workshop on Statistically-Based NLP Techniques*, pages 1–13.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Shay B. Cohen and Noah A. Smith. 2012. Empirical risk minimization for probabilistic grammars: Sample complexity and hardness of learning. *Computational Linguistics*, 38(3):479–526.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the International Conference on Learning Representations*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology. http://wals.info/.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345.

Jason Eisner and Damianos Karakos. 2005. Bootstrapping without the boot. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 395–402.

Jason Eisner and Dingquan Wang. No date. On evaluating cross-lingual generalization. To be posted on arXiv.

Robert Frank and Shyam Kapur. 1996. On the use of triggers in parameter setting. *Linguistic Inquiry*, 27:623–660.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–377.

Edward Gibson and Kenneth Wexler. 1994. Triggers. *Linguistic Inquiry*, 25(3):407–454.

Jennifer Gillenwater, Kuzman Ganchev, João Gra ca, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 194–199.

Kevin Gimpel and Noah A. Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 577–581.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256.

Matthew Gormley and Jason Eisner. 2013. Nonconvex global optimization for latent-variable models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 444–454.

Edouard Grave and Noémie Elhadad. 2015. A convex and feature-rich discriminative approach to dependency grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1375–1384.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2734–2740.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2017. Combining generative and discriminative approaches to unsupervised dependency parsing via dual decomposition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1689–1694.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Carey E. Priebe. 2007. Cross-instance tuning of unsupervised document clustering algorithms. In *Human Language Technologies: Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for POS tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association of Computational Linguistics*, 4:313–327.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063.

Karim Lari and Steve J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4(1):35–56.

Yann LeCun, Sumit Chopra, Raia Hadsell, and Fu Jie Huang. 2007. A tutorial on energy-based learning. In Gökhan Bakır, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan, editors, *Predicting Structured Data*. MIT Press.

Haitao Liu. 2010. Dependency direction as a means of word-order typology: A method based on dependency treebanks. *Lingua*, 120(6):1567–1578.

David Mareček. 2016. Delexicalized and minimally supervised parsing on universal dependencies. In *Statistical Language and Speech Processing: 4th International Conference, SLSP 2016, Pilsen, Czech Republic, October 11-12, 2016, Proceedings*, pages 30–42, Cham.

David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 281–290.

David Mareček. 2016. Twelve years of unsupervised dependency parsing. In *Proceedings of the 16th ITAT Conference Information Technologies—Applications and Theory*, pages 56–62.

Héctor Martínez Alonso, Željko Agić, Barbara Plank, and Anders Søgaard. 2017. Parsing universal dependencies without training. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 230–240.

Ryan McDonald. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 629–637.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244.

Dat Quoc Nguyen, Dai Quoc Nguyen, Dang Duc Pham, and Son Bao Pham. 2014. RDRPOSTagger: A ripple down rules-based part-of-speech tagger. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 17–20.

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

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel

Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.

Hiroshi Noji, Yusuke Miyao, and Mark Johnson. 2016. Using left-corner parsing to encode universal structural constraints in grammar induction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 33–43.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the Web. Technical Report 1999-66, Stanford InfoLab.

Mark A. Paskin. 2001. Cubic-time parsing and learning algorithms for grammatical bigram.

Mark A. Paskin. 2002. Grammatical bigrams. In *Advances in Neural Information Processing Systems*, pages 91–97.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association for Computational Linguistics*, 5:279–293.

Rudolf Rosa and Zdeněk Žabokrtský. 2015a. KL_{cpos^3} – a language similarity measure for delexicalized parser transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 243–249.

Rudolf Rosa and Zdeněk Žabokrtský. 2015b. Mstparser model interpolation for multi-source delexicalized transfer. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 71–75.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *Computing Research Repository*, arXiv:1706.04902.

David A. Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 822–831.

Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University.

Noah A. Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Grammatical Inference Applications*, pages 73–82.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the International Conference on Computational Linguistics and the Association for Computational Linguistics*, pages 569–576.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2012. Three dependency-and-boundary models for grammar induction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 688–698.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1983–1995.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013a. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013b. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 896–903.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140.

Dingquan Wang and Jason Eisner. 2016. The Galactic Dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association of Computational Linguistics*, 4:491–505. Data available at `https://github.com/gdtreebank/gdtreebank`.

Dingquan Wang and Jason Eisner. 2017. Fine-grained prediction of syntactic typology: Discovering latent structure with *supervised* learning. *Transactions of the Association for Computational Linguistics*, 5:147–161.

Dingquan Wang and Jason Eisner. 2018. Synthetic data made to order: The case of parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1325–1337, Brussels.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *Proceedings of the 46th Annual Meeting of Annual Meeting of the Association for Computational Linguistics and the 2008 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 532–540.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*.

Deniz Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, Massachusetts Institute of Technology.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867.