Parse Imputation for Dependency Annotations

Jason Mielens[1] **Liang Sun**[2] **Jason Baldridge**[1]

[1]Department of Linguistics
The University of Texas at Austin
{jmielens,jbaldrid}@utexas.edu

[2]Department of Mechanical Engineering
The University of Texas at Austin
sally722@utexas.edu

Abstract

Syntactic annotation is a hard task, but it can be made easier by allowing annotators flexibility to leave aspects of a sentence underspecified. Unfortunately, partial annotations are not typically directly usable for training parsers. We describe a method for imputing missing dependencies from sentences that have been partially annotated using the Graph Fragment Language, such that a standard dependency parser can then be trained on all annotations. We show that this strategy improves performance over not using partial annotations for English, Chinese, Portuguese and Kinyarwanda, and that performance competitive with state-of-the-art unsupervised and weakly-supervised parsers can be reached with just a few hours of annotation.

1 Introduction

Linguistically annotated data is produced for many purposes in many contexts. It typically requires considerable effort, particularly for language documentation efforts in which tooling, data, and expertise in the language are scarce. The challenge presented by this scarcity is compounded when doing deeper analysis, such as syntactic structure, which typically requires greater expertise and existing tooling. In such scenarios, unsupervised approaches are a tempting strategy. While the performance of unsupervised dependency parsing has improved greatly since Klein and Manning's (2004) Dependency Model with Valence (DMV), state-of-the-art unsupervised parsers still perform well below supervised approaches (Martins et al., 2010; Spitkovsky et al., 2012; Blunsom and Cohn, 2010). Additionally, they typically require large amounts of raw data. While this is not a problem for some languages, many of the world's languages do not have a clean, digitized corpus available.[1] For instance, the approach of Naseem et al. (2010) is unsupervised in the sense that it requires no dependency annotations, but it still makes use of the raw version of the full Penn Treebank. The approach of Marecek et al. (2013) requires extra unlabeled texts to estimate parameters.

Another strategy is to exploit small amounts of supervision or knowledge. Naseem et al. (2010) use a set of universal dependency rules and obtain substantial gains over unsupervised methods in many languages. Spitkovsky et al. (2010b; 2011) use web mark-up and punctuation as additional annotations. Alternatively, one could try to obtain actual dependency annotations cheaply. We use the Graph Fragment Language (GFL), which was created with the goal of making annotations easier for experts and possible for novices (Schneider et al., 2013; Mordowanec et al., 2014). GFL supports partial annotations, so annotators can omit obvious dependencies or skip difficult constructions. The ability to focus on portions of a sentence frees the annotator to target constituents and dependencies that maximize information that will be most useful for machine-learned parsers. For example, Hwa (1999) found higher-level sentence constituents to be more informative for learning parsers than lower-level ones.

To support this style of annotation while getting the benefit from partial annotations, we develop a two-stage parser learning strategy. The first stage completes the partial GFL annotations by adapting a Gibbs tree sampler (Johnson et al., 2007; Sun et al., 2014). The GFL annotations constrain the tree sampling space by using both dependencies and the constituent boundaries they express. The system performs missing dependency arc imputation using Gibbs sampling – we refer to this approach

[1]In fact, standardized writing systems have yet to be adopted for some languages.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1385–1394,
Beijing, China, July 26-31, 2015.

as the Gibbs Parse Completer[2] (GPC). The second stage uses the full dependencies output by the GPC to train Turbo Parser (Martins et al., 2010), and evaluation is done with this trained model on unseen sentences. In simulation experiments for English, Chinese and Portuguese, we show that the method gracefully degrades when applied to training corpora with increasing percentages of the gold training dependencies removed. We also do actual GFL annotations for those languages plus Kinyarwanda, and show that using the GPC to fill in the missing dependencies after two hours of annotation enables Turbo Parser to obtain 2-6% better absolute performance than when it has to throw incomplete annotations out. Furthermore, the gains are even greater with less annotation time and it never hurts to use the GPC—so an annotation project can pursue a partial annotation strategy without undermining the utility of the work for parser training.

This strategy has the further benefit of needing only a small number of sentences—in our case, under 100 sentences annotated in a 2-4 hour window. Furthermore, it relies on no outside tools or corpora other than a part-of-speech tagger; a resource that can be built with two hours of annotation time (Garrette and Baldridge, 2013).

2 Data

Data sources We use four languages from three language families in an effort to both verify the cross-linguistic applicability of our approach, accounting for variations in linguistic properties, as well as to attempt to realistically simulate a real-world, low-resource environment. Our data comes from English (ENG), Chinese (CHI), Portuguese (POR), and Kinyarwanda (KIN).

For ENG we use the Penn Treebank (Marcus et al., 1993), converted into dependencies by the standard process. Section 23 was used as a test set, and a random sample of sentences from sections 02-21 were selected for annotation with GFL as described below and subsequently used as the minimal training set. For CHI we use the Chinese Treebank (CTB5) (Xue et al., 2005), also converted to dependencies. The testing set consisted of files 1-40/900-931, and the sentences presented for GFL annotation were randomly sampled from files 81-899. The POR data is from

[2]The software, instructions, and data are available at http://www.github.com/jmielens/gpc-acl-2015

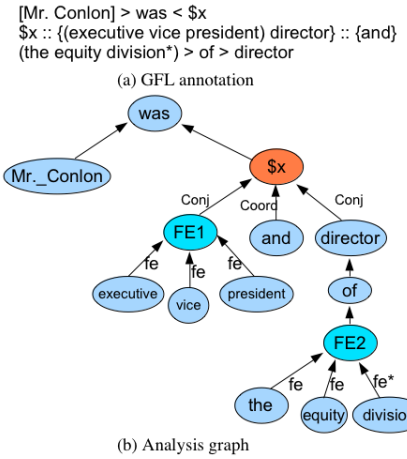


Figure 1: GFL example for *Mr. Conlon was executive vice president and director of the equity division*.

the CoNLL-X Shared Task on Multilingual Dependency Parsing and is derived from the Bosque portion of the Floresta sintá(c)tica corpus (Afonso et al., 2002), using the standard provided splits for training and testing. The KIN data is a corpus consisting of transcripts of testimonies by survivors of the Rwandan genocide, provided by the Kigali Genocide Memorial Center – this data is described by Garrette and Baldridge (2013).

GFL annotation We use a small number of sentences annotated using the Graph Fragment Language (GFL), a simple ASCII markup language for dependency grammar (Schneider et al., 2013). Unlike traditional syntactic annotation strategies requiring trained annotators and great effort, rapid GFL annotations can be collected from annotators who have minimal training. Kong et al. (2014) demonstrate the feasibility of training a dependency parser based on a GFL-annotated corpus of English tweets.

An example of GFL is shown in Figure 1: (a) is the GFL markup itself and (b) is a graphical representation of the dependencies it encodes. Figure 1 specifies several dependencies: *of* is a dependent of *director*, *executive vice president* and *director* are conjuncts and *and* is the coordinator. However, the complete internal structure of the phrase *the equity division* remains unspecified, other than *division* being marked as the head (via an asterisk).[3] Finally, *Mr. Conlon* in square brackets indicates it is a multiword expression.

[3]The graphical representation shows both of these as FE nodes, for *fudge expression*, indicating they are grouped together but otherwise underspecified.

CFG Rule	EVG distribution	Description
$S \rightarrow Y_H$	$P(root = H)$	The head of the sentence is H
$Y_H \rightarrow L_H R_H$	-	Split-head representation
$L_H \rightarrow H_L$	$P(STOP \vert dir = L, head = H, val = 0)$	H has no left children
$L_H \rightarrow L_H^1$	$P(CONT \vert dir = L, head = H, val = 0)$	H has at least one left child
$L'_H \rightarrow H_L$	$P(STOP \vert dir = L, head = H, val = 1)$	H has no more left children
$L'_H \rightarrow L_H^1$	$P(CONT \vert dir = L, head = H, val = 1)$	H has other left children
$L_H^1 \rightarrow Y_A L'_H$	$P(ArgA \vert dir = L, head = H, val = 1)$	A is a left child of H

Table 1: The CFG-DMV grammar schema from Klein and Manning (2004). Note that in these rules H and A are parts-of-speech. For brevity, we omit the portion of the grammar that handles the right-hand arguments since they are symmetric to the left. Valency (val) can take the value 1 (we have made attachments in the direction (dir) d) or 0 (not).

	CHI	ENG	KIN	POR
Sentences Annotated	24	34	69	63
Tokens Annotated	820	798	988	1067
Fully Specified Sentences	4	15	31	20

Table 2: Two Hour GFL Annotation Statistics

Kong et al. (2014) stipulate that the GFL annotations in their corpus must be fully-specified. They are thus unable to take advantage of such underspecified sentences, and we address that limitation in this paper. From the GFL annotations we can extract and deduce dependency arcs and constraints (see Section 3.2 for full details) in order to guide the Gibbs sampling process.

Time-bounded annotation As described in Section 1, a primary goal of this work was to consider the time in which a useful number of dependency tree annotations might be collected, such as might be required during the initial phase of a language documentation project or corpus build. To this end our annotators were operating under a strict two hour time limit. We also collected two further hours for English.

The annotators were instructed to annotate as many sentences as possible in the two hours, and that they should liberally use underspecification, especially for particularly difficult sequences in a given sentence. This was done to facilitate the availability of partial annotations for experimentation. All of the annotators had some previous experience providing GFL annotations, so no training period was needed. Annotation was done in 30-minute blocks, to provide short breaks for the annotators and so that learning curves could be generated. Each language was annotated by a single annotator. The ENG and CHI annotators were native speakers of their annotation language, while the POR and KIN annotators were non-native though proficient speakers.

The annotators achieved rates of 400-500 tokens/hr, whereas we find rates of 150-200 tokens/hr more typical when annotators are asked to fully specify. Requiring full specification also introduces more errors in cases of annotator uncertainty.

Table 2 shows the size of the GFL corpora that were created. Typically, over 50% of the sentences were not fully specified—the partial annotations provided in these are useless to Turbo Parser unless the missing dependencies are imputed.

3 Gibbs Parse Completer (GPC)

3.1 Gibbs sampler for CFG-DMV Model

CFG-DMV model The GPC is based on the DMV model, a generative model for the unsupervised learning of dependency structures (Klein and Manning, 2004). We denote the input corpus as $\boldsymbol{\omega} = (\boldsymbol{\omega^1}, \cdots, \boldsymbol{\omega^N})$, where each $\boldsymbol{\omega^s}$ is a sentence consisting of words and in a sentence $\boldsymbol{\omega}$, word ω_i has an corresponding part-of-speech tag τ_i. We denote the set of all words as V_ω and the set of all parts-of-speech as V_τ. We use the part-of-speech sequence as our terminal strings, resulting in an unlexicalized grammar. Dependencies can be formulated as split head bilexical context free grammars (CFGs) (Eisner and Satta, 1999) and these bilexical CFGs require that each terminal τ_i in sentence $\boldsymbol{\omega}$ is represented in a split form by two terminals, with labels marking the left and right heads ($\tau_{i,L}$, $\tau_{i,R}$). Henceforth, we denote $\boldsymbol{w} = w_{0,n}$ as our terminals in the split-form of sentence $\boldsymbol{\omega}$ (e.g., the terminals for *the dog walks* are $DT_L\ DT_R\ NN_L\ NN_R\ V_L\ V_R$). Table 1 shows the grammar rules for the DMV model, from Klein and Manning (2004).

Require: A is parent node of binary rule; $w_{i,k}$ is a valid span of terminals and $i+1<k$
function TREESAMPLER(A, i, k)
for $i<j<k$ and pair of child nodes of A:B, C **do**
$P(j,B,C) = \frac{\theta^w_{A\to BC} c(i,j)c(j,k)\cdot p_{B,i,j}\cdot p_{C,j,k}}{p_{A,i,k}}$
end for Sample $j*, B*, C*$ from multinomial distribution for (j,B,C) with probabilities calculated above
return $j*, B*, C*$
end function

Algorithm 1: Sampling split position and rule to expand parent node.

Gibbs sampler The split-head representation encodes dependencies as a CFG. This enables the use of a Gibbs sampler algorithm for estimating PCFGs (Johnson et al., 2007; Sun et al., 2014), and it is straightforward to incorporate constraints from partial annotations into this sampler. To do this, we modified the tree-sampling step to incorporate constraints derived from GFL annotations and thereby impute the missing dependencies.

Given a string $\boldsymbol{w} = (w_1, \cdots w_n)$, we define a span of $\boldsymbol{w}$ as $w_{i,k} = (w_{i+1}, \cdots, w_k)$, so that $\boldsymbol{w} = w_{0,n}$. As introduced in Pereira and Schabes (1992), a bracketing $\mathcal{B}$ of $\boldsymbol{w}$ is a finite set of spans on $\boldsymbol{w}$ satisfying the requirement that no two spans in a bracketing may overlap unless one span contains the other. For each sentence $\boldsymbol{w} = w_{0,n}$ we define the auxiliary function for each span $w_{i,j}$, $0 \le i < j \le n$:

$$c(i,j) = \begin{cases} 1 & \text{if span } w_{i,j} \text{ is valid for } \mathcal{B}; \\ 0 & \text{otherwise.} \end{cases}$$

Here one span is valid for $\mathcal{B}$ if it doesn't cross any brackets. Section 3.2 describes how to derive bracketing information from GFL annotations and how to determine if a span $w_{i,j}$ is valid or not. Note that for parsing a corpus without any annotations and constraints, $c(i,j) = 1$ for any span, and the algorithm is equivalent to the Gibbs sampler in Sun et al. (2014).

There are two parts to the tree-sampling. The first constructs an inside table as in the Inside-Outside algorithm for PCFGs and the second selects the tree by recursively sampling productions from top to bottom. Consider a sentence $\boldsymbol{w}$, with sub-spans $w_{i,k} = (w_{i+1}, \cdots, w_k)$. Given θ^w (modified rule probabilities θ given constraints of sentence $\boldsymbol{w}$, see Section 3.2), we construct the inside table with entries $p_{A,i,k}$ for each nonterminal and each span $w_{i,k}$: $0 \le i < k \le n$. We introduce

Require: $Arcs$ is the set of all directed arcs extracted from annotation for sentence $\boldsymbol{w}$
function RULEPROB-SENT($\boldsymbol{w}, \theta, Arcs$)
$\theta^w = \theta$
for each directed arc $w_i < w_j$ **do**
if $i<j$ **then**
for nonterminal A $\neq L_{\tau_j}$ **do**
$\theta^w_{A\to\beta} = 0$ if β contains Y_{τ_i}
end for
else
for nonterminal A $\neq R_{\tau_j}$ **do**
$\theta^w_{A\to\beta} = 0$ if β contains Y_{τ_i}
end for
end if
end for
return θ^w
end function

Algorithm 2: Modifying Rule Probabilities for w to ensure parse tree contains all directed arcs.

$c(i,j)$ into the calculation of inside probabilities:

$$p_{A,i,k} = c(i,k)\cdot \sum_{A\to BC\in R} \sum_{i<j<k} \theta^w_{A\to BC}\cdot p_{B,i,j}\cdot p_{C,j,k} \quad (1)$$

Here, $p_{A,i,k} = P_{G_A}(w_{i,k} \mid \theta^w)$ is the probability that terminals i through k were produced by the non-terminal A, $A \to BC \in R$ are possible rules to expand A. The inside table is computed recursively using Equation 1.

The resulting inside probabilities are then used to generate trees from the distribution of all valid trees of the sentence. The tree is generated from top to bottom recursively with the function $TreeSampler$ defined in Algorithm 1, which introduces $c(i,j)$ into the sampling function from Sun et al. (2014).

3.2 Constraints derived from GFL

We exploit one dependency constraint and two constituency constraints from partial GFL annotations.

Dependency rule Directed arcs are indicated with angle brackets pointing from the dependent to its head, e.g. *black* > *cat*. Once we have a directed arc annotation, say $\omega_i > \omega_j$, if $i<j$, which means word j has a left child, we must have rule $L^1_{\tau_j} \to Y_{\tau_i} L'_{\tau_j}$ in our parse tree (similarly if $i>j$, we have $R^1_{\tau_j} \to R'_{\tau_j} Y_{\tau_i}$ in our parse tree), where τ_i, τ_j are parts-of-speech for ω_i and ω_j. We enforce this by modifying the rule probabilities for sample sentence $\boldsymbol{w}$ to ensure that any sampled tree contains all specified arcs.

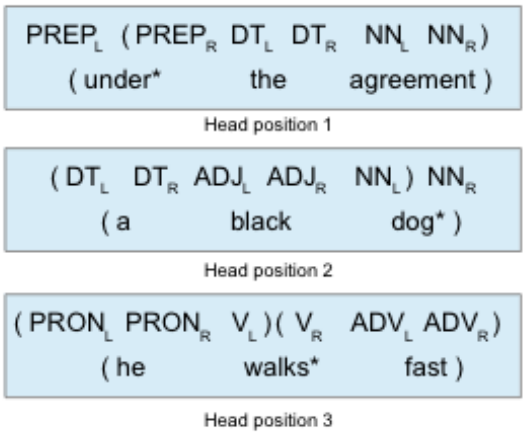


Figure 2: Generating brackets for known head

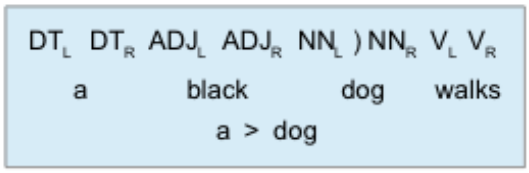


Figure 3: Generating half brackets

Brackets GFL allows annotators to group words with parenthesis, which provides an explicit indicator of constituent brackets. Even when the internal structure is left underspecified (e.g. *(the equity division*) in Figure 1 (a)*, the head is usually marked with *, and we can use this to infer sub-constituents. Given such a set of parentheses and the words inside them, we generate brackets over the split-head representations of their parts-of-speech, based on possible positions of the head. Figure 2 shows how to generate brackets for three situations: the head is the leftmost word, rightmost word, or is in a medial position. For example, the first annotation indicates that *under* is the head of *under the agreement*, and the rest of words are right descendants of *under*. This leads to the bracketing shown over the split-heads.

Half brackets We can also derive one-sided half brackets from dependency arcs by assuming that dependencies are projective. For example, in Figure 3, the annotation $a > dog$ specifies that *dog* has a left child *a*, so we know that there is a right bracket before the right-head of *dog*. Thus, we can detect invalid spans using the half brackets; if a span starts after *a* and ends after *dog*, this span is invalid because it would result in crossing brackets. This half bracketing is a unique advantage provided by the split-head representation. The details of this algorithm are shown in Algorithm 3.

Require: $Arcs$ is the set of all directed arcs extracted for sentence, $w_{a,b}$ is a span to detect
function DETECTINVALIDSPAN($a, b, Arcs$)
  for each directed arc $\omega_i < \omega_j$ **do**
    if $i < j$ **then**
      if $a < 2i - 1 < b < 2j$ **then**
        $c(a, b) = 0$
      end if
    else
      if $2j - 2 < a < 2i - 1 < b$ **then**
        $c(i, j) = 0$
      end if
    end if
  end for
  return $c(a, b)$
end function

Algorithm 3: Detect whether one span is invalid given all directed arcs.

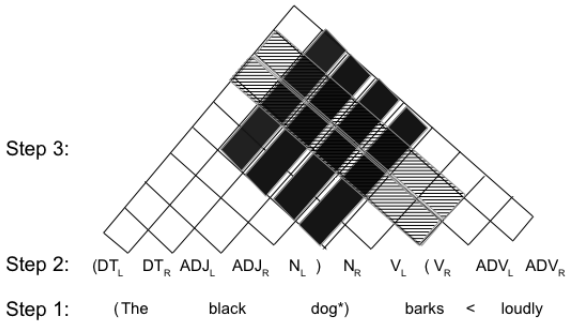


Figure 4: Process of generating brackets and detecting invalid spans.

We use both half bracket and full bracket information, $\mathcal{B}$, to determine whether a span is valid. We set $c(i, j) = 0$ for all spans over $\boldsymbol{w}$ detected by Algorithm 3 and violating $\mathcal{B}$. Then, in the sampling scheme, we'll only sample parse trees that satisfy these underlying constraints.

Figure 4 shows the resulting blocked out spans in the chart based on both types of brackets for the given partial annotation, which is Step 1 of the process. *The black dog* is a constituent with *dog* marked as its head, so we generate a full bracket over the terminal string in Step 2. Also, *barks* has a right child *loudly*; this generates a half bracket before V_R. In Step 3, the chart in Figure 4 represents all spans over terminal symbols. The cells in black are invalid spans based on the full bracket, and the hatched cells are invalid spans based on the half bracket.

4 Results

Experiments There are two points of variation to consider in empirical evaluations of our ap-

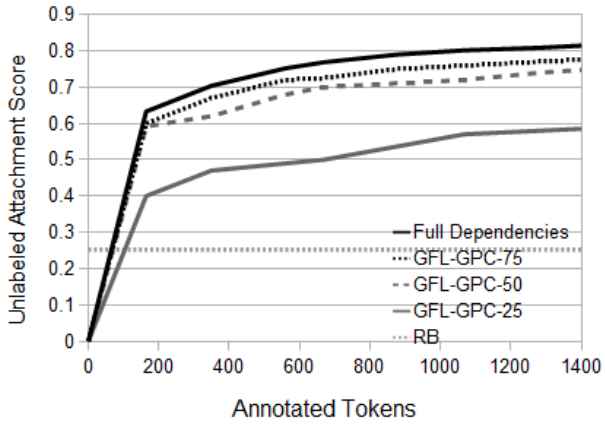


Figure 5: English oracle and degradation results

Language	ENG	CHI	POR
RB	25.0	11.6	27.0
GFL-GPC-25	58.7	33.5	60.2
GFL-GPC-50	75.0	46.1	71.4
GFL-GPC-75	77.8	50.1	73.7
Full	81.6	56.2	78.1

Table 3: Results with simulated partial annotations, GFL-GPC-X indicates X percent of dependencies were retained.

proach. The first is the effectiveness of the GPC in imputing missing dependencies and the second is the effectiveness of the GFL annotations themselves. Of particular note with respect to the latter is the reasonable likelihood of divergence between the annotator and the corpus used for evaluation—for example, how coordination is handled and whether subordinate verbs are dependents or heads of auxiliary verbs. To this end, we perform simulation experiments that remove increasing portions of gold dependencies from a training corpus to understand imputation performance and annotation experiments to evaluate the entire pipeline in a realistically constrained annotation effort.

In that regard, one thing to consider are the part-of-speech tags used by the unlexicalized GPC. These do not come for free, so rather than ask annotators to provide them, the raw sentences to be annotated were tagged automatically. For English and Kinyarwanda, we used taggers trained with resources built in under two hours (Garrette and Baldridge, 2013), so these results are actually constrained to the GFL annotation time plus two hours. Such taggers were not available for Chinese or Portuguese, so the Stanford tagger (Toutanova et al., 2003) was used instead.

After imputing missing dependencies, the GPC outputs fully sentences that are used to train TurboParser (Martins et al., 2010). In all cases, we compare to a right-branching baseline (RB). Although comparing to a random baseline is more typical of imputation experiments, a right-branching baseline provides a stronger initial comparison. For the GFL annotation experiments, we use two additional baselines. The first is simply to use the sentences with full annotations and drop any incomplete ones (GFL-DROP). The second is to make any partial annotations usable by assuming a right-branching completion (GFL-RBC).

Simulated partial annotations Figure 5 shows the learning curve with respect to number of annotated tokens when retaining 100%, 75%, 50% and 25% of gold-standard training dependencies and using the GPC to impute the removed ones. With both 75% and 50% retained, performance degrades gracefully. It is substantially lower for 25%, but the curve is steeper than the others, indicating it is on track to catch up. Nonetheless, one recommendation from these results is that it probably makes sense to start with a small number of fully annotated sentences and then start mixing in partially annotated ones.

Table 3 shows the attachment scores obtained for English, Chinese, and Portuguese with varying proportions of dependencies removed for the GPC to impute.[4] English and Portuguese hold up well with 75% and 50% retained, while Chinese drops more precipitously, and 25% leads to substantial reductions in performance for all.

Note that these simulations indicate that, given an equivalent number of total annotated arcs, using the GPC is more beneficial than requiring annotators to fully specify annotations. Imputing fifty percent of the dependency arcs from sentences containing 1000 tokens is typically more effective by a few points than using the full gold-standard arcs from sentences containing 500 tokens. Actually, this simulation is too generous to complete annotations in that it leaves out consideration of the time and effort required to obtain those 100% full gold-standard arcs: it is often a small part of a sentence that consumes the most effort when full annotation is required. Additionally, these simulation experiments randomly removed dependencies while humans tend to annotate higher-level con-

[4]These are based on the same sentences used in the next section's GFL annotation experiments for each language.

Eval Length	< 10	< 20	all
GFL-DROP (4hr)	54.5	55.0	52.6
GFL-GPC (4hr)	60.1	**61.8**	55.1
Blunsom and Cohn, 2010	67.7	–	**55.7**
Naseem et al., 2010	**71.9**	50.4	–

Table 4: English results compared to previous unsupervised and weakly-supervised methods.

Language	KIN	CHI	POR
RB	52.6	11.6	27.0
GFL-DROP (2hr)	64.4	36.7	59.8
GFL-GPC (2hr)	64.5	38.8	65.0

Table 5: Non-English results summary

stituents and leave internal structure (e.g. of noun phrases) underspecified. Given Hwa's (1999) findings, we expect non-random partial annotations to better serve as a basis for imputation.

GFL annotations We conducted three sets of experiments with GFL annotations, evaluating on sentences of all lengths, less than 10 words, and less than 20 words. This was done to determine the types of sentences that our method works best on and to compare to previous work that evaluates on sentences of different lengths.

Table 4 shows how our results on ENG compare to others. Blunsom and Cohn (2010) represent state-of-the-art unsupervised results for all lengths, while Naseem et al. (2010) was chosen as a previous weakly-supervised approach. GFL-GPC achieves similar results on the 'all lengths' criterion as Blunsom and Cohn and substantially outperforms Naseem et al. on sentences less than 20 words. Our poor performance on short sentences is slightly surprising, and may result from an uneven length distribution in the sentences selected for annotation—we have only 3 training sentences less than 10 words—as discussed by Spitkovsky et al. (2010a). To correct this problem, both long and short sentences should be included to construct a more representative sample for annotation.

We did not expect GFL-RBC to perform so similarly to RB. It is possible that the relatively large number of under-specified sentences led to the right-branching quality of GFL-RBC dominating, rather than the more informative GFL annotations.

The results of the ENG annotation session can be seen in Figure 6a. GFL-GPC is quite strong even at thirty minutes, with only seven sentences annotated. GFL-DROP picks up substantially at the end; this may be in part explained by the fact that the last block contained many short sentences, which provide greater marginal benefit to GFL-DROP than to GFL-GPC.

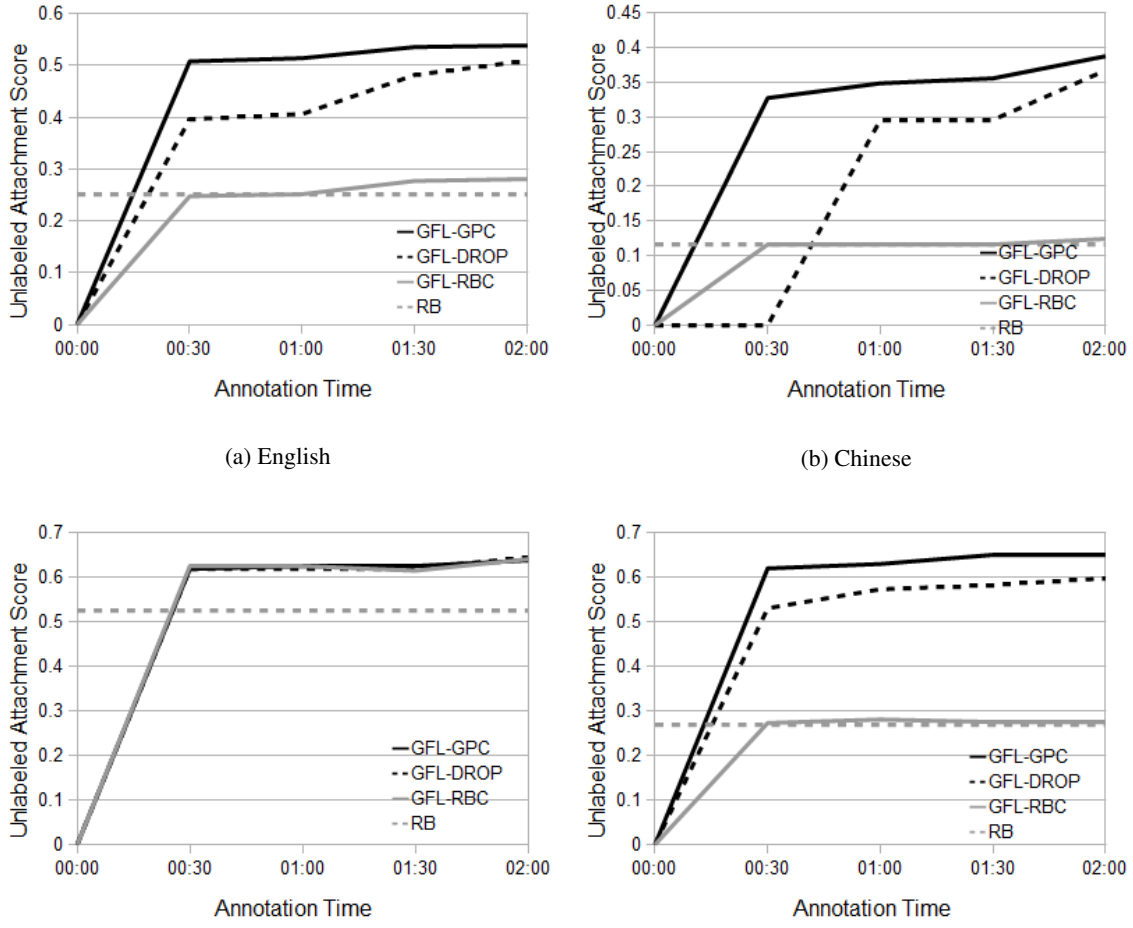
The learning curves for the other languages can be seen in Figures 6b-6d, with a summary available in Table 5. Like ENG, CHI and POR both show clear wins for the GPC strategy. Of particular note is that the CHI annotations contained many fewer fully-completed sentences (4) than the ENG annotations (15). This somewhat addresses the question raised by the 25% retention simulation experiments—the GPC method improves results over dropping partial annotations. The POR results show a consistent strong win for GPC throughout.

The KIN results in Figure 6c exhibit a pattern unlike the other languages; specifically, the KIN data has a very high right-branching baseline (RB in figures) and responds nearly identically for all of the more informed methods. Upon investigation, this appears to be an artifact of the data used in KIN evaluation plus domain adaptation issues. The gold data consists of transcribed natural speech, whereas the training data consists of sentences extracted from the Kinyarwanda Wikipedia.

All of the learning curves display a large initial jump after the first round of annotations. This is encouraging for approaches that use annotated sentences: just a small number of examples provide tremendous benefit, regardless of the strategy employed.

Error analysis The primary errors seen on an analysis of the GPC-completed sentences varies somewhat between languages. The ENG data contains many short sentences, consisting often of a few words and a punctuation mark. Part of the GFL convention is that the annotator is free to annotate punctuation as part of the sentence or instead view it as extra-linguistic and drop the punctuation from the annotation. Often the punctuation in the ENG data went unannotated, with the result being that the final parse model is not particularly good at handling these types of sentences when encountered in the test set.

Specific constructions like coordination and possession also suffer a similar issue in that annotators (and corpora) varied slightly on how they were handled. Thus, some languages like CHI contained many of a particular type of error due to mismatches in the conventions of the annotator and corpus. Issues like this could have been avoided by a longer training period prior to anno-



(a) English

(b) Chinese

(c) Kinyarwanda

(d) Portuguese

Figure 6: GPC results by annotation time for eval sentences of all lengths.

tation, although were this a real annotation project, there would be no existing corpus to compare to at first. This brings up a more basic question of evaluation - one of usability versus representational norm matching. It is likely that the GFL annotations (and thus the models trained on them) diverge from the gold standard in what amount to annotation conventions rather than substantive linguistic divergences. To evaluate more fully or fairly, we would need test sets produced by the same set of annotators or an external, task-based evaluation that uses the dependencies as in input.

5 Conclusions

We have described a modeling strategy that takes advantage of a Gibbs sampling algorithm for CFG parsing plus constraints obtained from partial annotations to build dependency parsers. This strategy's performance improves on that of a parser built only on the available complete annotations. In doing so, our approach supports annotation efforts that use GFL to obtain guidance from non-expert human annotators and allow any annotator to put in less effort than they would to do complete annotations.

We find that a remarkably small amount of supervised data can rival existing unsupervised methods. While unsupervised methods have been considered an attractive option for low-resource parsing, they typically rely on large quantities of clean, raw sentences. Our method uses less than one hundred sentences, so in a truly low-resource scenario, it has the potential to require much less total effort. For instance, a single native speaker could easily both generate and annotate the sentences required for our method in a few hours, while the many thousands of raw sentences needed for state-of-the-art unsupervised methods could

take much longer to assemble if there is no existing corpus. This also means our method would be useful for getting in-domain training data for domain adaptation for parsers.

Finally, our method has the ability to encode both universal grammar and test-language grammar as a prior. This would be done by replacing the uniform prior used in this paper with a prior favoring those grammar rules during the updating-rule-probabilities phase of the GPC, and would essentially have the effect of weighting those grammar rules.

Acknowledgments

This work was supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533

References

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. "Floresta sintá(c)tica": a Treebank for Portuguese. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, pages 1698–1703. LREC.

Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213. Association for Computational Linguistics.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 457–464. Association for Computational Linguistics.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *HLT-NAACL*, pages 138–147. Citeseer.

Rebecca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 73–79. Association for Computational Linguistics.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 139–146.

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

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, to appear*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

David Marecek and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *ACL (1)*, pages 281–290.

André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics.

Michael T. Mordowanec, Nathan Schneider, Chris Dyer, and Noah A Smith. 2014. Simplified dependency annotations with GFL-Web. *ACL 2014*, page 121.

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. Association for Computational Linguistics.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

Nathan Schneider, Brendan OConnor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under) specifying dependency syntax without overloading annotators. *LAW VII & ID*, page 51.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759. Association for Computational Linguistics.

Valentin I Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010b. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1278–1287. Association for Computational Linguistics.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 19–28. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Liang Sun, Jason Mielens, and Jason Baldridge. 2014. Parsing low-resource languages using Gibbs sampling for PCFGs with latent annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238.