Incrementally Learning a Dependency Parser to Support Language Documentation in Field Linguistics

Morgan Ulinski* **Julia Hirschberg*** **Owen Rambow†**
*Department of Computer Science
†Center for Computational Learning Systems
Columbia University
New York, NY, USA
{mulinski@cs,julia@cs,rambow@ccls}.columbia.edu

Abstract

We present experiments in incrementally learning a dependency parser. The parser will be used in the WordsEye Linguistics Tools (WELT) (Ulinski et al., 2014a; Ulinski et al., 2014b) which supports field linguists documenting a language's syntax and semantics. Our goal is to make syntactic annotation faster for field linguists. We have created a new parallel corpus of descriptions of spatial relations and motion events, based on pictures and video clips used by field linguists for elicitation of language from native speaker informants. We collected descriptions for each picture and video from native speakers in English, Spanish, German, and Egyptian Arabic. We compare the performance of MSTParser (McDonald et al., 2006) and MaltParser (Nivre et al., 2006) when trained on small amounts of this data. We find that MaltParser achieves the best performance. We also present the results of experiments using the parser to assist with annotation. We find that even when the parser is trained on a single sentence from the corpus, annotation time significantly decreases.

1 Introduction

Although languages have appeared and disappeared throughout history, today languages are facing extinction at an unprecedented pace. Over 40% of the estimated 7,000 languages in the world are at risk of disappearing (Alliance for Linguistic Diversity, 2013). Efforts to document languages become even more important with the increasing rate of extinction. Bird (2009) emphasizes a particular need to make use of computational linguistics during fieldwork. One way the WordsEye Linguistics Tools will address this issue is by providing field linguists with the means to easily document syntax, something that is largely missing from existing documentation tools. We model our tools for documenting syntax on the tools for documenting morphology in SIL FieldWorks Language Explorer (FLEx) (SIL FieldWorks, 2014), one of the most widely-used toolkits for field linguists.

An important part of FLEx is its "linguist-friendly" morphological parser (Black and Simons, 2006), which is fully integrated into lexicon development and interlinear text analysis. The parser is constructed "stealthily," in the background, and can help a linguist by predicting glosses and morphological analyses for interlinear texts. In the same way, WELT will provide tools for specifying the syntax of sentences in the form of dependency structures and use them to train a parser in the background. The parser will provide predictions for new sentences, and, as these are corrected and approved by the linguist, they are added to the training data and the parser is incrementally improved.

This paper makes three contributions. First, we introduce a new corpus of English, Spanish, German, and Egyptian Arabic descriptions of spatial relations and motion events, which we have annotated with dependency structures and other linguistic information. We focused on spatial relations and motion because one of the other functions of WELT will be to assist field linguists with elicitation of spatial language and documentation of spatial and motion semantics. We are making the corpus available to the public. Second, we compare the performance of two existing dependency parsing packages, MSTParser (McDonald et al., 2006) and MaltParser (Nivre et al., 2006), using incrementally increasing amounts of

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 440–449, Osaka, Japan, December 11-17 2016.

this training data. Third, we show that using a parser trained on small amounts of data can assist with dependency annotation.

In Section 2, we discuss related work. In Section 3, we describe the new publicly available corpus. In Section 4, we describe the parsing experiments and discuss the results. Section 5 discusses initial experiments with nonlexical models. We discuss the annotation experiments and results in Section 6. We conclude in Section 7.

2 Related Work

There have been a number of investigations into multilingual dependency parsing. For example, Nivre et al. (2007b) presents detailed results for 11 languages using the arc-eager deterministic parsing algorithm included in MaltParser. However, results are reported only for the parser trained on the full training set and would not generalize to situations where training data is limited. Likewise, the 2006 and 2007 CoNLL shared tasks of multilingual dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007a) relied on the existence of ample training data for the languages being investigated. Our work differs in that we are interested in the performance of a dependency parser trained on very little data.

Duong et al. (2015) approach dependency parsing for a low-resource language as a domain adaptation task; a treebank in a high-resource language is considered out-of-domain, and a much smaller treebank in a low-resource language is considered in-domain. They jointly train a neural network dependency parser to model the syntax of both the high-resource and the low-resource language. In this paper, we focus on the alternate approach of training directly on small amounts of data.

Guo et al. (2015) also investigate inducing dependency parsers for low-resource languages using training data from high-resource languages. They focus on lexical features, which are not directly transferable among languages, and propose the use of distributed feature representations instead of discrete lexical features. Lacroix et al. (2016) describe a method for transferring dependency parsers across languages by projecting annotations across word alignments and learning from the partially annotated data. However, both of these methods rely on large amounts of (unannotated) data in the target language in order to learn the word embeddings and alignments. It is unclear how well these approaches would work in the context of an endangered language where large amounts of unannotated text will not be available.

Our work also differs from the above because our goal is to incorporate a parser into tools for field linguists studying endangered languages. Currently, there are limited options for creating a syntactic parser for an endangered language. The *ParGram* project (ParGram, 2013) aims to produce wide coverage grammars for a variety of languages, but doing so requires knowledge both of the LFG formalism and the XLE development platform (Crouch et al., 2011). It is unlikely that a field linguist would have the grammar engineering skills necessary to create a grammar in this way. Similarly, the LinGO Grammar Matrix (Bender et al., 2002) is a framework for creating broad-coverage HPSG grammars. The Grammar Matrix facilitates grammar engineering by generating "starter grammars" for a language from the answers to a typological questionnaire. Extending a grammar beyond the prototype, however, does require extensive knowledge of HPSG, making this tool more feasibly used by computational linguists than by field linguists. Our work differs from both ParGram and the Grammar Matrix because we do not require the field linguist to master a particular grammar formalism. Instead, linguists will create a syntactic parser simply by labeling individual sentences, a procedure that builds easily upon an existing workflow that already includes annotating sentences with morphological information.

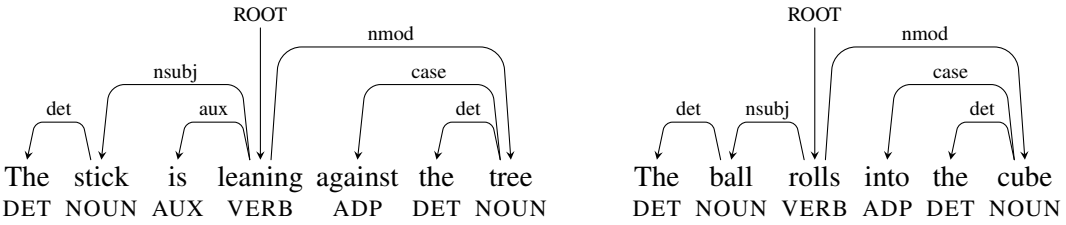
3 Corpus

To obtain a corpus that is similar to sentences that field linguists would probably analyze using WELT, we started with two stimulus kits used by field linguists to study spatial and motion language: the Picture Series for Positional Verbs (Ameka et al., 1999) and the Motion Verb Stimulus Kit (Levinson, 2001). For each picture and video clip, we elicited a one-sentence description from native speakers of English, Spanish, German, and Egyptian Arabic. We chose languages covering a range of linguistic phenomena. For example, German uses morphological case, and Spanish and Arabic both use clitics. In future work, we hope to add languages from other language families, including Chinese and Korean. Our languages

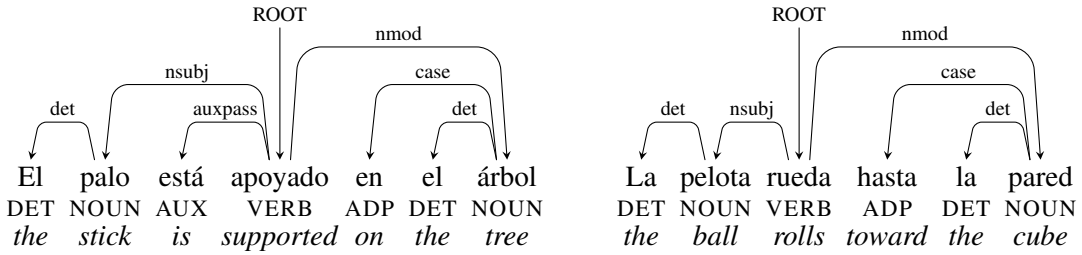
are high-resource languages because we needed to have access to linguistically trained native speakers in order to create the gold standard; however, we did not actually use any additional resources for these languages in our experiments, and we believe they can therefore stand in for low-resource languages.

	# sents	Avg. len.	# words	# lemmas	# pos tags	# features	# labels
English	163	7.21	152	135	12	26	20
Spanish	165	8.51	180	149	12	30	20
German	157	7.52	217	175	13	32	19
Arabic	158	10.04	174	117	10	37	15

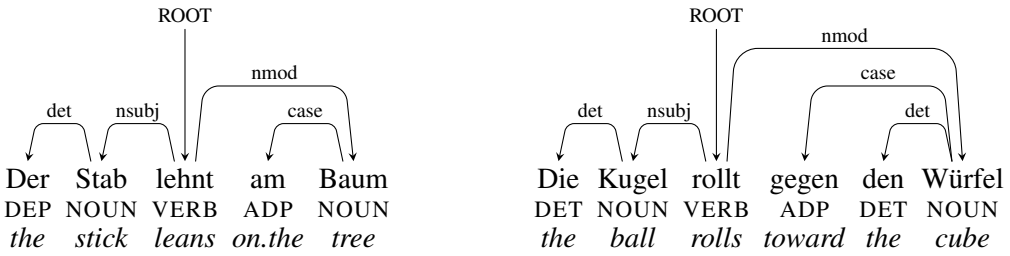
Table 1: Summary of each language in the corpus



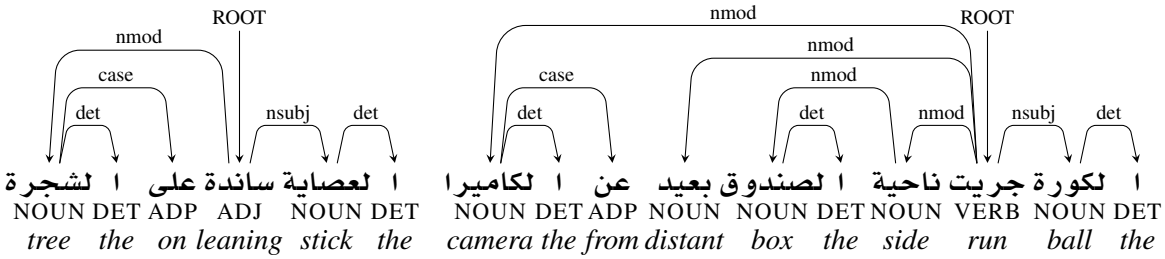
(a) English



(b) Spanish



(c) German



(d) Arabic

Figure 1: Example dependency structures

We start out by tokenizing each sentence; for Spanish and Arabic, this step includes splitting off the clitics. We then annotated each token with its lemma, morphological information, part of speech, syntactic head, and dependency label. For consistency across languages, we used part of speech tags, morphological features, and dependency labels from the Universal Dependencies project (Nivre et al., 2016) and attempted to follow the universal guidelines as closely as possible. The total number of sentences, aver-

age sentence length, and number of unique words, lemmas, part of speech tags, morphological features, and dependency labels for each language is shown in Table 1. Note the sentence count varies slightly for each language; this is because for some of the pictures and videos, the native informant gave us several possible descriptions. English and German have very similar average sentence lengths; average lengths in Spanish and Arabic are higher. German had the largest vocabulary; English had the smallest vocabulary. All languages used similar numbers of part of speech tags and dependency labels, except Arabic which used fewer of both. Arabic had the largest number of morphological features, and English the smallest. Some example sentences with dependency labels are shown in Figure 1.

4 Parsing Experiments and Results

We used four methods of training a dependency parser on our data: MSTParser (McDonald et al., 2006), two configurations of MaltParser (Nivre et al., 2006), and a baseline. All experiments used 5-fold cross validation. For each of the four training methods, we trained on a subset of the train fold ranging from 1 sentence to 100 sentences. We tested on the full test fold, and then averaged the accuracy across the five folds. Results are shown in Table 2. Arc accuracy requires selecting the correct head for a token; Lbl accuracy requires selecting the correct dependency label; Both requires that both head and dependency label are correct. The highest accuracy in each row for each metric (Arc, Lbl, Both) is shown in bold.

The baseline is determined by assigning the majority dependency label from the train data. Heads are selected using left or right attachment, whichever is more common in the train data. For most of the training sets, we did left attachment and assigned `det` as the dependency label, and the baseline usually remained constant across all train sizes. For German with train size = 2, one of the folds had a majority of right attachment, which resulted in a slight decrease in baseline accuracy. Likewise, for Arabic with train size = 1 and train size = 2, one fold used right attachment, resulting in a decrease in arc accuracy for those rows. The Arabic label accuracy was much more variable, since `nmod` was the majority label about half of the time. The Arabic baseline used for each train size and train fold is shown in Table 3.

The first parser we tested was MSTParser (McDonald et al., 2006; McDonald et al., 2005), which uses a two-stage approach to parsing: an unlabeled parser and a separate edge labeler. The parser works by finding a maximum spanning tree; the label sequence is then found using Viterbi's algorithm. MSTParser uses a combination of lexical, part of speech, and morphological features; we did not modify the default feature set.

Train size	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
1	det	det	nmod **(right)**	nmod	det
2	det	det	nmod **(right)**	det	nmod
5	det	det	det	nmod	nmod
10	nmod	det	det	det	nmod
25	nmod	det	nmod	nmod	nmod
50	nmod	det	det	nmod	nmod
100	det	det	det	nmod	nmod

Table 3: Arabic baseline: majority label per fold; if not otherwise indicated, the default attachment is left

We next tested MaltParser (Nivre et al., 2006), which implements a variety of deterministic parsing algorithms. A dependency structure is derived using features of the current parser state to predict the next action. Parser state is represented by a stack of partially processed tokens and a list of remaining input tokens. We tested two algorithms: Nivre arc-eager and Nivre arc-standard. The arc-eager algorithm adds arcs to the dependency tree as soon as the head and dependent are available; the arc-standard algorithm requires that the dependent already be complete with respect to its own dependents. We used the default feature sets for each of the algorithms. Like MSTParser, the feature set includes a combination of lexical, part of speech, and morphological features; MaltParser also adds dependency features (arcs and labels) from the current parser state.

Even with only one training sentence, both MSTParser and MaltParser performed well above the baseline. MaltParser consistently achieved higher accuracy than MSTParser for all languages and train sizes, especially when predicting the dependency labels. The performance of the arc-eager algorithm vs.

Train size	Baseline			MSTParser			MaltParser (Nivre arc-eager)			MaltParser (Nivre arc-standard)		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
1	0.452	0.325	0.318	0.669	0.495	0.437	0.720	0.785	0.699	**0.741**	**0.793**	**0.708**
2	0.452	0.325	0.318	0.730	0.702	0.643	0.798	**0.831**	**0.769**	**0.801**	0.826	0.764
5	0.452	0.325	0.318	0.794	0.780	0.723	**0.852**	**0.860**	**0.822**	0.817	0.843	0.789
10	0.452	0.325	0.318	0.826	0.787	0.743	**0.872**	**0.880**	**0.846**	0.829	0.856	0.804
25	0.452	0.325	0.318	0.872	0.830	0.798	**0.935**	**0.925**	**0.902**	0.878	0.901	0.855
50	0.452	0.325	0.318	0.930	0.884	0.865	**0.950**	**0.942**	**0.919**	0.920	0.925	0.897
100	0.452	0.325	0.318	0.939	0.913	0.896	**0.961**	**0.965**	**0.946**	0.945	0.951	0.926

(a) English

Train size	Baseline			MSTParser			MaltParser (Nivre arc-eager)			MaltParser (Nivre arc-standard)		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
1	0.397	0.273	0.271	0.454	0.329	0.311	0.504	0.570	0.478	**0.533**	**0.588**	**0.493**
2	0.397	0.273	0.271	0.568	0.444	0.397	0.600	0.650	**0.558**	**0.605**	**0.663**	**0.558**
5	0.397	0.273	0.271	0.662	0.608	0.541	**0.753**	0.779	**0.713**	0.752	**0.786**	0.702
10	0.397	0.273	0.271	0.758	0.729	0.662	0.797	0.836	0.773	**0.810**	**0.838**	**0.777**
25	0.397	0.273	0.271	0.837	0.813	0.770	**0.890**	**0.905**	**0.865**	0.868	0.887	0.843
50	0.397	0.273	0.271	0.880	0.861	0.817	**0.921**	**0.937**	**0.905**	0.910	0.930	0.895
100	0.397	0.273	0.271	0.923	0.898	0.871	**0.947**	**0.959**	**0.935**	0.932	0.947	0.919

(b) Spanish

Train size	Baseline			MSTParser			MaltParser (Nivre arc-eager)			MaltParser (Nivre arc-standard)		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
1	0.446	0.286	0.273	0.575	0.407	0.353	0.643	0.680	0.594	**0.655**	**0.685**	**0.595**
2	0.446	0.269	0.234	0.631	0.494	0.435	0.737	0.738	0.657	**0.749**	**0.770**	**0.676**
5	0.446	0.286	0.273	0.753	0.634	0.585	**0.794**	0.800	**0.732**	0.781	**0.820**	0.730
10	0.446	0.286	0.273	0.770	0.676	0.634	**0.819**	**0.848**	**0.782**	0.810	0.844	0.770
25	0.446	0.286	0.273	0.820	0.751	0.707	**0.883**	**0.896**	**0.854**	0.836	0.895	0.819
50	0.446	0.286	0.273	0.850	0.797	0.758	**0.914**	**0.936**	**0.899**	0.896	0.935	0.879
100	0.446	0.286	0.273	0.908	0.845	0.816	**0.942**	0.953	**0.931**	0.916	**0.954**	0.908

(c) German

Train size	Baseline			MSTParser			MaltParser (Nivre arc-eager)			MaltParser (Nivre arc-standard)		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
1	0.358	0.253	0.181	0.623	0.491	0.434	**0.650**	**0.707**	**0.611**	0.617	0.681	0.571
2	0.358	0.254	0.184	**0.712**	0.656	0.598	0.704	0.738	0.646	0.687	**0.760**	**0.654**
5	0.424	0.237	0.171	0.787	0.747	0.694	0.842	0.861	0.799	**0.844**	**0.871**	**0.811**
10	0.424	0.235	0.168	0.864	0.808	0.768	0.902	0.907	0.869	**0.906**	**0.923**	**0.882**
25	0.424	0.194	0.062	0.920	0.858	0.827	**0.941**	**0.939**	**0.917**	0.930	0.938	0.909
50	0.424	0.216	0.114	0.948	0.888	0.869	0.954	0.958	0.938	**0.957**	**0.965**	**0.941**
100	0.424	0.237	0.171	0.962	0.912	0.897	**0.975**	0.972	**0.961**	0.973	**0.973**	0.957

(d) Arabic

Table 2: Accuracy of each parsing method.

the arc-standard algorithm seems to vary by language and train size. For English, Spanish, and German, the arc-standard algorithm has higher performance on small training sets, while the arc-eager algorithm becomes superior as more training data is available. Results are more mixed for the Arabic data.

5 Initial Experiments with Nonlexical Models

One of the goals of WELT will be to encourage sharing of data and analyses among field linguists. Since stimulus packs (such as the picture series and video series that we used to create our corpus) are commonly reused across many languages, it would be helpful if a parser trained on a fully-annotated version of the data for one language could be used by a field linguist just starting out with another, potentially similar, language. To that end, we performed an initial experiment to see whether a parser trained on one language could be applied successfully to the other languages in our corpus. To test this, we used MaltParser (arc-eager algorithm) to train a parser on English data, using only nonlexical features: part of speech, morphological tags, and dependency labels/arcs. We then applied this model to the other three languages. Results are shown in Table 4. For this experiment, we used all 163 sentences from the English corpus.

Comparing these results to those in Table 2, we see that, for Spanish, the English nonlexical model performs similarly to MaltParser trained on 5 Spanish sentences. For German, the English nonlexical model performs similarly to MaltParser trained on 5-10 German sentences. For Arabic, the English nonlexical model has lower accuracy than MaltParser trained on a single Arabic sentence. This suggests that a simple nonlexical model such as this one may only be useful for linguists using WELT if an annotated corpus in a *related* language is available.

	Arc	Label	Arc+Label
Spanish	0.767	0.816	0.719
German	0.808	0.840	0.773
Arabic	0.629	0.713	0.567

Table 4: Accuracy of (English) nonlexical model applied to other languages

6 Annotation Experiments and Results

To test whether our parser would help with annotation, we performed annotation experiments using the English data. We timed how long it took an annotator to label a sentence, when the sentence is preprocessed in one of four ways. In the first method, a baseline assigns a flat structure and the dependency label `det` to all nodes. The other methods use MaltParser (Nivre arc-eager algorithm) trained on 1, 5, or 25 sentences to provide an initial parse for the annotator to correct.

Annotators labeled 5 trees for each parsing method, for a total of 20 trees. To ensure each of the four sets of 5 contained sentences with similar syntactic complexity, the sentences were chosen as follows. Each parsing method was assigned 1 sentence of each of 5 lengths: 7 words, 8 words, 9 words, 10-11 words, and 12-14 words. These were randomly selected from among all sentences of the required length. The 20 sentences were then presented to the annotator in random order. To keep the experiment consistent, all annotators labeled the same 20 sentences, in the same order.

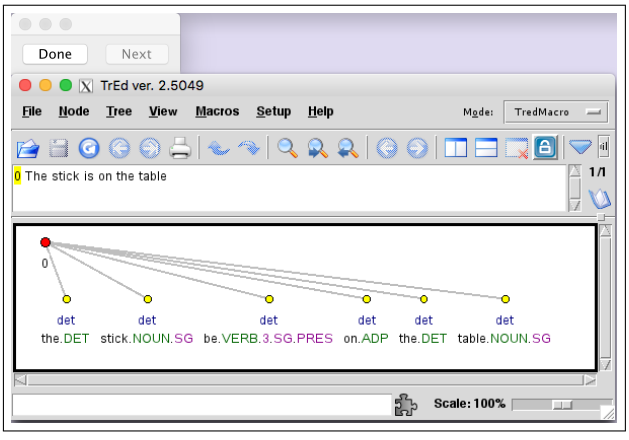


Figure 2: Screenshot of annotation software

Three annotators participated in the experiment. The first was the first author of this paper. She is an expert annotator, very familiar with the universal guidelines for dependency annotation and the annotation software. The other two annotators were undergraduate students who participated in a brief training session to familiarize them with the desired analysis and the software. They were given ref-

erence materials showing sample trees covering a variety of syntactic phenomena including: auxiliaries, copulas, coordination, secondary predication, and subordinate clauses. They were permitted to refer to this material throughout the annotation task. Before participating in the annotation task, they annotated 10 additional trees for practice.

The software used for annotation was Tree Editor (TrEd) (Pajas and Štěpánek, 2008) with a simple Java wrapper that handled opening files in TrEd and keeping track of annotation time. A screenshot of the setup is shown in Figure 2. Upon pressing the Next button, the annotator was shown the next tree in TrEd and the program recorded the start time. When the annotator finished labeling a tree, they saved the file in TrEd and pressed the Done button. The wrapper program closed the current file in TrEd and recorded the end time.

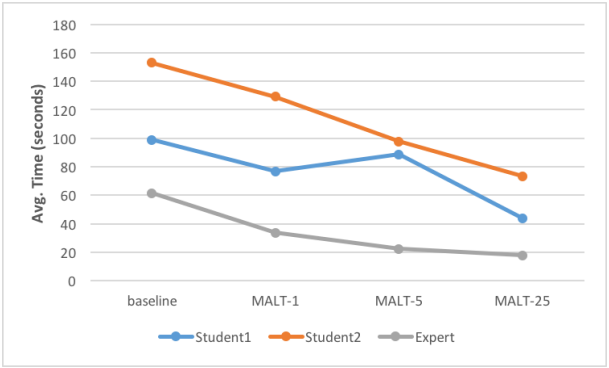


Figure 3: Annotation time across all sentence sizes

A graph showing the average time (across all sentence lengths) it took each annotator to label a tree for each of the four parsing types is shown in Figure 3. A detailed listing of the time each annotator took for each sentence size is shown in Table 5. A graph showing the average time (across annotators) for each sentence size is shown in Figure 4(a). All times are given in seconds. The accuracy of MaltParser (arc-eager) on sentences of different lengths is shown in Figure 4(b).

Sent. size	Baseline	Malt-1	Malt-5	Malt-25
7	87.0	54.9	14.4	14.3
8	73.0	83.6	65.1	17.7
9	85.9	67.2	61.7	43.3
10-11	144.3	70.6	65.7	86.6
12-14	104.5	107.5	237.1	57.3
Avg	98.9	76.8	88.8	43.8

(a) Student1

Sent. size	Baseline	Malt-1	Malt-5	Malt-25
7	132.7	54.0	23.6	26.3
8	97.8	98.9	24.3	30.4
9	90.8	153.0	96.9	102.4
10-11	203.4	66.2	139.9	138.9
12-14	240.8	274.1	203.7	68.7
Avg	153.1	129.2	97.7	73.4

(b) Student2

Sent. size	Baseline	Malt-1	Malt-5	Malt-25
7	59.2	25.4	8.1	8.7
8	44.7	42.0	8.6	9.2
9	52.3	29.7	21.4	21.0
10-11	75.8	34.9	11.2	36.8
12-14	75.5	36.0	62.0	12.8
Avg	61.5	33.6	22.3	17.7

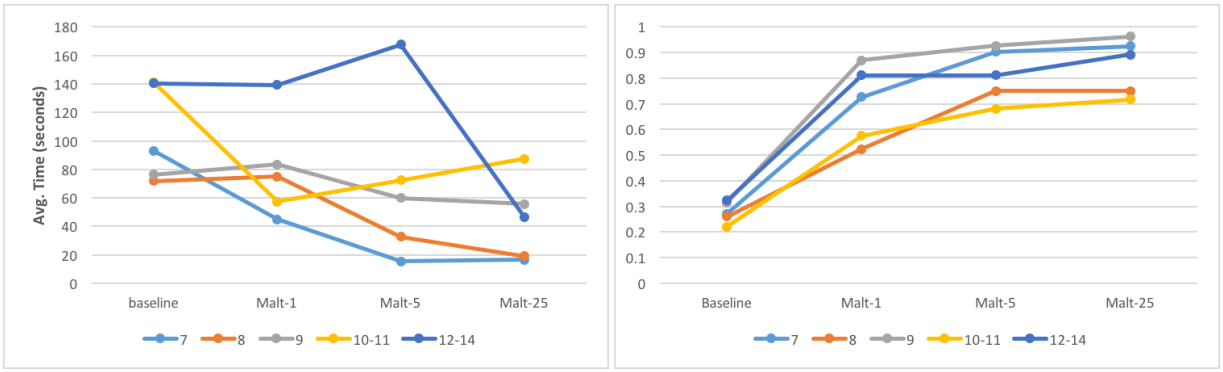
(c) Expert

Sent. size	Baseline	Malt-1	Malt-5	Malt-25
7	93.0	44.8	15.4	16.5
8	71.8	74.8	32.7	19.1
9	76.3	83.3	60.0	55.6
10-11	141.1	57.3	72.3	87.5
12-14	140.3	139.2	167.6	46.3
Avg	104.5	79.9	69.6	45.0

(d) Average

Table 5: Time (seconds) to annotate each sentence

Results vary slightly across annotators, but it is clear that, even when training on a single sentence, annotation time is improved. Average annotation time decreases from 104.5 seconds for the baseline parse to 79.9 seconds for the parser trained on one sentence. Using the parser trained on 5 sentences, average annotation time decreases again to 69.6 seconds. Using the parser trained 25 sentences, we see a decrease in annotation time to an average of 45 seconds. Statistical significance testing was done with a paired t-test. Significant decreases in annotation time are: between the baseline and 1 training sentence ($p = 0.034$), between the baseline and 5 training sentences ($p = 0.015$), between the baseline



(a) (b)

Figure 4: Graphs of (a) annotation time for each sentence size, across all annotators (b) parser accuracy on each sentence size

and 25 training sentences ($p = 2.51\text{e}{-5}$), and between 1 training sentence and 25 training sentences ($p = 0.018$).

There are several reasons to account for the fact that training on a single sentence significantly decreases annotation time. MaltParser works by predicting steps in a derivation, so one sentence actually translates into more than one data point. With only one sentence, the parser can learn that a determiner should be the left child of a noun, or that a noun should be the left child of the root predicate. Having these dependencies already attached reduces the work the annotator must do compared to a completely flat structure. In addition, our corpus consists only of descriptions of spatial relations and motion events, so we expect a much more limited range of grammatical constructs than one finds in other treebanks.

For very short sentences (length 7), the graph in Figure 4(b) shows a clear downward trend as the amount of training data increases. For sentences of length 8-9, we do not see improvement with one training sentence, but annotation time begins to decrease substantially when there are 5 training sentences. For longer sentences, the downward trend is less clear. This makes sense, since we can expect to find a wider range of syntactic structures in a longer sentence, and parser performance on these will require that a similar structure was seen in the train set. For sentences length 10-11, there is a substantial drop in annotation time from baseline to 1 training sentence, at which point it seems to plateau. For sentences length 12-14, average annotation time does not decrease until we have 25 training sentences.

Parser	**Student1**			**Student2**			**Expert**		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
Baseline	0.951	1.000	0.951	1.000	1.000	1.000	1.000	1.000	1.000
Malt-1	0.971	1.000	0.971	1.000	1.000	1.000	1.000	1.000	1.000
Malt-5	0.930	0.950	0.905	0.980	1.000	0.980	0.980	1.000	0.980
Malt-25	0.962	0.982	0.962	1.000	1.000	1.000	1.000	1.000	1.000

Table 6: Annotation accuracy

One concern with using a parser to assist with annotation is whether there will be any effect on overall accuracy. When presented with a mostly-correct parse, will annotators be able to see all the errors and fix them? The accuracy of the annotators for each of the four parsing types is shown in Table 6. We do see a drop in accuracy for all annotators when training on 5 sentences, especially for Student1. However, this decrease is very slight for both Student2 and Expert. We suspect there may have been several difficult sentences in this set; all annotators made errors on the sentence of length 10, and Student1 had particularly low accuracy (0.625) on the sentence of length 8.

7 Summary and Future Work

We have reported results for incrementally training a dependency parser across four languages. Our results show that such a parser can improve on baseline performance even when trained on a single

sentence, making our method particularly useful in the documentation of endangered and low-resource languages. We found that MaltParser achieved the highest accuracy overall; the arc-standard algorithm seems preferable for very small training sizes and arc-eager for slightly larger training sizes. We found that using a parser to predict each sentence before annotation did significantly improve annotation time, without a substantial decrease in accuracy.

The results of our experiments demonstrate that it is possible to extend FieldWorks' "stealthy" approach to learning a morphological parser into the realm of syntax. By providing a way to assign dependency structures to sentences, WELT will allow field linguists to incorporate syntax into language documentation. The incrementally trained parser will reduce their workload by letting them correct errors in a dependency structure rather than starting from scratch. This method of syntactic documentation does not limit the field linguist to a particular syntactic theory. We chose to use the universal labels and analyses in our corpus, but WELT users will have complete control over assignment of heads and choice of dependency labels. The only requirement is that they are consistent across sentences.

In future work, we will experiment with other parsers, such as TurboParser (Martins et al., 2010), Mate (Bohnet, 2010), and Easy-First (Goldberg and Elhadad, 2010). Furthermore, we will continue to investigate methods of re-using existing parsers and dependency annotations with new languages (see Section 5); specifically, we will investigate more effective methods of adapting existing parsers to other languages. For example, we will investigate how to combine a non-lexical model with a lexical model obtained from a small number of target language sentences. We will also investigate ways for linguists to directly specify syntactic properties that can be used by the parser, similar to the way FLEx converts morphological properties specified by users into formal rules compatible with the underlying parser. Finally, we plan to try our WELT system in actual fieldwork.

Acknowledgements

We would like to thank Bob Coyne, Victor Soto, Daniel Bauer, and Heba Elfardy for their help creating and annotating the parallel corpus. We would like to thank the anonymous reviewers for their comments that helped us improve the paper.

References

Alliance for Linguistic Diversity. 2013. The Endangered Languages Project. `http://www.endangeredlanguages.com/`.

F. Ameka, C. de Witte, and D. Wilkins. 1999. Picture series for positional verbs: Eliciting the verbal component in locative descriptions. In D. Wilkins, editor, *Manual for the 1999 Field Season*, pages 48–54. Max Planck Institute for Psycholinguistics.

E. Bender, D. Flickinger, and S. Oepen. 2002. The Grammar Matrix. In J. Carroll, N. Oostdijk, and R. Sutcliffe, editors, *Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

S. Bird. 2009. Natural language processing and linguistic fieldwork. *Computational Linguistics*, 35(3):469–474.

H.A. Black and G.F. Simons. 2006. The SIL FieldWorks Language Explorer approach to morphological parsing. In *Computational Linguistics for Less-studied Languages: Texas Linguistics Society 10*, Austin, TX, November.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 89–97, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164.

D. Crouch, M. Dalrymple, R. Kaplan, T. King, J. Maxwell, and P. Newman. 2011. XLE Documentation. `http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html`.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. A neural network model for low-resource universal dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 339–348, Lisbon, Portugal, September. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 742–750, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China, July. Association for Computational Linguistics.

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, San Diego, California, June. Association for Computational Linguistics.

Stephen C. Levinson. 2001. Motion verb stimul, version 2. In Stephen C. Levinson and N. J. Enfield, editors, *Manual for the field season 2001*, pages 9–13. Max Planck Institute for Psycholinguistics.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 34–44, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 216–220, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proc. of LREC-2006*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Petr Pajas and Jan Štěpánek. 2008. Recent advances in a feature-rich framework for treebank annotation. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 673–680, Stroudsburg, PA, USA. Association for Computational Linguistics.

ParGram. 2013. ParGram/ParSem. `http://pargram.b.uib.no/`.

SIL FieldWorks. 2014. SIL FieldWorks. `http://fieldworks.sil.org`.

Morgan Ulinski, Anusha Balakrishnan, Daniel Bauer, Bob Coyne, Julia Hirschberg, and Owen Rambow. 2014a. Documenting endangered languages with the WordsEye Linguistics Tool. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 6–14.

Morgan Ulinski, Anusha Balakrishnan, Bob Coyne, Julia Hirschberg, and Owen Rambow. 2014b. WELT: Using graphics generation in linguistic fieldwork. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 49–54.