The Dissertation Committee for Jason David Mielens

certifies that this is the approved version of the following dissertation:

# Supervision for Syntactic Parsing of Low-Resource Languages

**Committee:**

Jason Baldridge, Supervisor

Katrin Erk

Ray Mooney

Chris Dyer

John Beavers

# Supervision for Syntactic Parsing of Low-Resource Languages

by

**Jason David Mielens, B.S., M.A.**

**Dissertation**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**Doctor of Philosophy**

The University of Texas at Austin

May 2016

# Acknowledgments

I would like to thank my co-authors: Jason Baldridge, Liang Sun, and Dan Garrette. Also, thank you to the many people who offered insights and suggestions for my work, along with assisting in various annotation efforts: Grant DeLozier, Jim Evans, Kyle Jerro, and all the members of the Computational Linguistics Tea Group.

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, as well as a Carlota Smith Fellowship award.

# Supervision for Syntactic Parsing of Low-Resource Languages

Jason David Mielens, Ph.D.
The University of Texas at Austin, 2016

Supervisor: Jason Baldridge

Developing tools for doing computational linguistics work in low-resource scenarios often requires creating resources from scratch, especially when considering highly specialized domains or languages with few existing tools or research. Due to practical considerations in project costs and sizes, the resources created in these circumstances are often different from large-scale resources in both quantity and quality, and working with these resources poses a distinctly different set of challenges than working with larger, more established resources.

There are different approaches to handling these challenges, including many variations aimed at reducing or eliminating the annotations needed to train models for various tasks. This work considers the task of low-resource syntactic parsing, and looks at the relative benefits of different methods of supervision. I will argue here that the benefits of doing some amount of supervision almost always outweigh the costs associated with doing that annotation; unsupervised or minimally supervised methods are often surpassed with surprisingly small amounts of supervision.

This work is primarily concerned with identifying and classifying sources of supervision that are both useful and practical in low-resource scenarios, along with analyzing the performance of systems that make use of these different supervision

sources and the behaviors of the minimally trained annotators that provide them. Additionally, I demonstrate several cases where linguistic theory and computational performance are directly connected. Maintaining a focus on the linguistic side of computational linguistics can provide many benefits, especially when working with languages where the correct analysis for various phenomena may still be very much unsettled.

# Table of Contents

# Chapter 1

# Introduction

In the field of Computational Linguistics, as in Computer Science generally, there has been a dramatically increased availability of large-scale natural language corpora in recent years. Corpora that were considered large even a few years ago are now thought of as too small to produce meaningful results on certain tasks. In large part, this explosion in data is the result of an ever-increasing supply of natural language data easily available for collection on the Internet – both professionally created articles and personal, socially-published data like that of Twitter and blogs.

Simultaneously, a major concern of linguists today is the disappearance of many of the world's languages and the efforts associated with both documenting and preserving these languages. Increasingly, the data demands of modern computational techniques mean that computational linguists often have little to offer to those researchers and end-users who want practical, helpful tools for either researching or working with these low-resource languages, despite the relative prevalence of small datasets that do exist for the languages in question [Bird, 2009]. While the proliferation of 'big data' and its associated methods has been tremendously useful

for English and the handful of other languages with available data at that scale, the focus on these techniques is marginalizing many of the world's languages.

In response to the problems introduced by 'big English', this work explicitly aims to focus on the properties of small-scale, quickly collected, noisy corpora and the practical matters associated with developing and evaluating tools and techniques for them and the need for supervision sources that are realistically providable for these corpora. This process brings many challenges that simply do not exist when working with large-scale, well-established corpora for widely-spoken languages. For instance, a basic tenet of this work is the hypothesis that small corpora cannot be adequately modeled by simply scaling down large, standardized corpora. When small-scale corpora are considered in the literature, this is often the approach that is taken, however there is substantial evidence to suggest that this is a relatively poor approximation for a real-world small corpus; everything from the types of annotators used to the time invested in their creation conspires to make these small corpora substantially different from their more mature versions.

It is becoming clear, particularly through results using semi-supervised methods, that there is often no substitute for the linguistic knowledge of experts and speakers, even in small amounts. Increasingly, supervised techniques are proving better than unsupervised techniques when working with small corpora, even given the added expenses of obtaining those annotations if necessary [Garrette et al., 2013]. This somewhat counter-intuitive point has informed the basic structure of the research presented here, where techniques that are capable of functioning with low volumes of supervision are identified, and then the results of those systems are subsequently analyzed to determine ways of potentially improving and directing the annotation process to focus on those aspects that seem to be most crucial.

The central thesis of this dissertation is that directly supervised techniques can provide a superior level of performance relative to either wholly unsupervised or indirectly supervised techniques, even in the face of limited, noisy annotations. To support this thesis, I present results primarily from the domain of syntactic parsing, where in particular I focus on the benefits of adopting a partial annotation strategy in order to open the annotation process to a wider potential annotator base.

This dissertation has three primary goals. First, to demonstrate that introducing a higher level of linguistic sophistication and awareness into computational linguistics models can provide benefits both in terms of actual performance as well as theoretical benefits. Although this often means introducing more complex units of annotation that can potentially complicate the annotation process, I provide results both from experimental settings and the literature that show that efficient annotation setups and smart model choices can minimize the costs associated with moving to this increased level of sophistication. A second goal of this dissertation is to consider the benefits of working in a partial annotation environment specifically for low-resource settings. I describe adaptations of multiple syntactic parsers that are capable of utilizing partial annotations as a training source and show that the penalty these parsers pay for operating on partial data is relatively minimal compared to the benefits that partial annotations provide with respect to the annotation process. A final goal is to consider the types of annotators and annotations that are most useful in these scenarios, and to discuss the natural variations that exist between annotators and the consequences that variation has for evaluation in the context of a gold standard set of annotations.

## 1.1 Low-Resource Languages

The primary target of the techniques described in this dissertation are languages that are commonly described as low-resource languages. In truth, the vast majority of the world's languages could be accurately described as being low-resource from the point-of-view of computational linguistics. Which is to say that there is simply not much annotated data available for use in training models for tools like parsers or part-of-speech taggers, and in most cases there isn't even a very large amount of raw data available either – the language's digitized presence is just overall small. For the most part, tools for natural language processing have focused on languages like English and other Indo-European languages for the simple fact that at the time that computational linguistics gained popularity, they were the only languages with a significant digital presence.

However, increasingly, speakers of smaller languages are gaining increased access to technology and are bringing their native languages with them. Indeed, it has been pointed out that the times we are living in represent a potentially unique situation in world history: that we live in a time of both rapidly expanding access to technology and rapid loss of cultural and linguistic heritage [Bird, 2009]. The development of tools and techniques that are well-suited for use with the small datasets of these languages, then, should be of great interest to those researchers who would seek to preserve linguistic diversity and allow more efficient and effective use of technology in the communities of speakers of these low-resource languages.

It should be noted that low-resource languages need not be endangered in a linguistic sense, or even spoken in small communities. For instance, Javanese is an Austronesian language spoken by 82+ million people, yet it lacks all but the most basic of resources for natural language processing. As another example, consider the

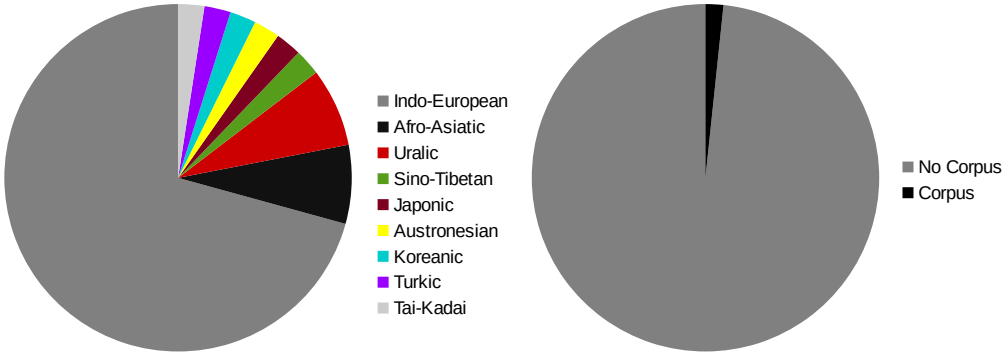



Figure 1.1: Basic statistics of syntactic corpus coverage by language. Sourced from a literature survey containing around 100 corpora.

case of Cebuano and Jan Edmund Carlson described by Oard (2003) in the context of the 'Surprise Language Exercises'—a series of experiments in building language technologies for previously unanticipated languages. When the exercise began and Cebuano (spoken in the Philippines by 20 million people) was announced as the target language, nearly all of the participating teams contacted a single researcher–Jan Edmund Carlson–because he was working on essentially the only Cebuano-English resource available.

Often, this scramble for scarce resources is even more difficult than it was for Cebuano; as described by Ethnologue[1], nearly 99% of languages have fewer speakers and thus, usually, even fewer resources. Consider Figure 1.1 showing corpora coverage by language; most languages of the world have no easily accessible resources, and most of the resources that do exist are centered around the Indo-European family–English in particular. Often, the solution to how best to deal with low-resource languages is to consider the use of unsupervised or minimally-supervised methods. The thought process behind this decision is that if there is simply not a lot of available data, trying to get the most mileage out of universal properties or the small amount of data you do have is the best course of action available.

[1]https://www.ethnologue.com/statistics/size

However, most of the current state-of-the-art methods in unsupervised parsing still rely on the existence of a large corpus, the only requirement that has been in any way relaxed is that the corpus can now be unlabeled [Naseem, 2014, Grave and Elhadad, 2015]. As already mentioned, the vast majority of the world's languages simply do not have a large enough digitized corpus of sentences to run even these unsupervised methods appropriately.

With this in mind, there is an important terminological distinction to be aware of; that of supervision vs. resource-availability. While the two are often conflated, especially in environments where resource-availability is assumed, they are very much different concepts. Many state of the art unsupervised methods are unsupervised but high-resource in the sense that they depend on large amounts of data, but small (no) amounts of annotation. In contrast, many supervised methods from the literature are able to do reasonably well in a supervised, low-resource setting with a relatively small number of annotated sentences, although they would obviously prefer (and are typically evaluated and intended for) high-resource supervised settings. The other possible combination–supervised low-resource–is more uncommon, potentially because of the idea that low-resource cases should just use unsupervised methods. However, as previously discussed, a surprisingly small amount of annotation can often outperform unsupervised methods with many times more data.

**Limited Domains** While the major use of the techniques described in this work is to enable computational resources to be developed for languages without large annotated corpora, they do have other uses and properties that make them applicable to researchers working exclusively in a high-resource context as well.

In addition to having particular languages that lack sufficient data or anno-

tators for standard tools and techniques, there are also situations in which potential users may simply not have a large amount of applicable data even if they are working in a language like English. These may occur when developing tools that target a very limited and specific domain or genre. For instance, building a parser for English-language Twitter data is not the same task as developing a parser for general-purpose English. Accordingly, it may be most appropriate to imagine the existence of a hypothetical 'Twitter English', for which all the English data in the world is only of limited use.

Instead, the situation of training a parser for this Twitter English would be more like training a parser for a low-resource language that shared some mutual intelligibility with English. You could use English as a secondary source of data, but treating Twitter English as a separate entity means that there may not be a large amount of highly relevant data available and perhaps a small annotation project would be needed to assist with the domain adaptation. Alternatively, if the limited domain is English text messages—another domain with highly-specialized language—collecting raw data may itself be very difficult.

## 1.2 Classification of Supervision

As a major part of this research concerns the relative merits of various supervision sources it is important to establish a slightly more fine-grained classification scheme than than the typical 'supervised' as opposed to 'unsupervised' distinction that most frequently appears in the literature. An honest accounting of supervision sources is also beneficial for comparative purposes; many techniques presented in the literature bill themselves as unsupervised but still make use of things like gold part of speech tags or another 'minor' form of supervision that may end up playing a substantial

role in the overall performance of a given model–having access to gold part of speech tags is no small benefit.

A truly unsupervised approach would operate exclusively over raw surface forms, and would need to generate any other meta information it required. Such techniques are few and far between, but that is not necessarily a problem in and of itself; a bigger problem is that the over-generalization of the 'unsupervised' label leads to a situation in which methods that use fundamentally different sets of supervision are being compared. The need to classify things into one of these two classes is further detrimental in that there is a pressing need for the middle ground; the rise of 'semi-supervised' techniques speaks directly to this need. Semi-supervised methods are typically only 'semi'-supervised in the sense that they provide a non-standard method of supervision.

Throughout this entire body of work, an effort is made to explicitly account for all of sources of supervision, whether they come in the form of human-provided annotations or an algorithmically-provided label. Whenever possible, the costs of this supervision are also made explicit and grounded in real world concepts of annotation time and literal dollar value costs. Too often in the literature these real costs are either glossed over or provided in terms of a non-grounded unit of measurement. For instance, dependency annotations play a major role in this body of work; measuring the cost of a dependency corpus could be done by simply counting the number of annotated arcs in the corpus. This is surely a valid cost measure, in that obtaining more arc annotations is more costly than obtaining fewer, but it ignores the human effort that goes into obtaining those annotations.

This work will make reference to a few different types of supervision, all of which are orthogonal to the coverage level of supervision (partial vs. full) and

resource availability (high vs. low).

**Instance-level** Any supervision source that directly labels instances of training data can be considered 'instance-level' supervision. This is similar in many respects to the idea of token supervision in that a single point of instance-level supervision has very little generalizing ability, it applies to the the specific case that has been labeled and that's all.

This is perhaps the most common or base form of supervision, and what much of the literature means when they refer to the idea of supervision. It is also easy to see how instance-level supervision may be further subdivided into partial and full instance-level supervision by simply not labeling all instances in the corpus.

**Task-level** If a supervision source provides information that applies to the entire task rather than a specific instance, it can be called 'task-level' supervision. Task-level supervision is similar to the concept of type supervision from part-of-speech tagging. If an annotator is provided a single word type and asked to provide all the potential part-of-speech tags that type may take on, the annotator is providing task-level supervision because they are not directly labeling specific instances of that word type in the corpus (instance-level supervision) but instead are indirectly providing limits on the task itself.

Another example of task-level supervision, this time in dependency parsing, would be providing a fixed set of universal dependency rules that the entire corpus is expected to more or less respect. The supervisor providing these rules is not directly annotating any particular sentence, but is instead specifying what constitutes a valid or good solution for the entire task.

In many cases, task-level supervision may be extracted from a collection

of instance-level supervision. This fact follows from definitions; if instance-level supervision provides direct feedback and guidance for individual instances of a task, we should be able to learn something about the overall task by examining a collection of these annotations. For instance, if we are given a part-of-speech labeled corpus, with tags provided for individual tokens, extracting a tag dictionary for each word type is trivial. An alternative method of providing task-level information is to have an annotator, typically a very experienced and knowledgeable one, directly provide the task-level information.

**Proxy** Some sources of supervision do not readily fall into the two classes above, and these will be referred to as 'proxy supervision'. An example of proxy supervision could be using parallel multilingual data to project information from one corpus to another. The defining factor of this category is the transfer of knowledge from a situation where the supervision was clearly applicable to one where it could potentially be useful but wasn't intended for. As another example, systems that allow for annotations to 'bleed' onto similar or related word forms, label-propagation setups for instance, could be considered to be providing proxy supervision by transferring that knowledge from a case where it was perfectly appropriate (and supplied) to one where it may very well be useful but no annotator has specifically validated that fact.

Proxy supervision can also be employed to provide some measure of supervision where providing a full instance-level annotation is not feasible for any reason. For instance, partial dependency annotations can be considered a proxy for the full dependency tree structure annotation for a given sentence. If an annotator is incapable of providing the full structure, the partial structure can serve as a stand-in or rough approximation of the full structure.

## 1.3 Overview

This dissertation is structured around the three goals discussed above, with the ultimate aim of showing the feasibility of providing direct, instance-level supervision for a complicated, dense annotation object–dependency trees–in a less than ideal annotation environment. While traditional methods of supervision are hindered in such environments, alternative sources of supervision can still provide the required information.

In Chapter 2 the first dissertation goal–demonstrating the benefits of working in a higher level of linguistic sophistication–is considered. More evidence is provided, through a series of smaller experiments, in support of the idea that a greater attention to linguistic sophistication and awareness should be a primary goal of computational linguistics work, and that it can potentially be very important in a low-resource scenario. This is a viewpoint expressed by others in the literature as well [Bender, 2009, Bender, 2011, Hwa et al., 2005, Manning, 2011]. In particular, I show multiple examples of tasks that can be improved by using systems capable of directly dealing with morphology, which is an aspect of language often passed over or minimally modeled in computational settings. I demonstrate that the costs associated with moving to a linguistically-richer representation are not always prohibitive, particularly when alternative supervision sources such as algorithmically-supplied supervision are used, and that the benefits of having that data can outweigh the cost of obtaining it. In relation to the main thesis of this dissertation, the work in this chapter demonstrates that complex supervision can be obtained in a variety of ways.

Section 2.1 presents work on a graph-based, morphologically-aware Part-of-Speech (POS) tagging method specifically evaluated on small corpora collected

in an artificially time-limited fashion. The major results from this work are that small amounts of noisy data are capable of producing reasonable POS taggers, type annotations are superior to tokens in this context, and that being aware of the actual morphology in a language (rather than taking a character-based n-gram approach) leads to improvements that correlate with morphological complexity. This work is an example of an expensive annotation object, namely morphological segmentations, being provided in an alternative fashion–algorithmically instead of being provided by human annotators.

Section 2.2 additionally considers the use of morphological and sub-word information in syntactic parsing. The major result from Chapter 2 is that complex supervision can be obtained efficiently and used effectively by leveraging an intelligent combination of human annotation effort and machine-provided automated labeling.

The second major goal of this dissertation–considering the possibility of working in a partial annotation environment specifically in low-resource settings–is addressed primarily in Chapters 3 and 4. Chapter 3 is an introduction to the annotation framework used throughout the rest of the work to represent partial dependencies, Graph Fragment Language (GFL). GFL has a number of properties that make it both practical and effective in the context of partial annotations and low-resource settings in particular.

While in Chapter 2 I considered the use of algorithmically supplied annotation as a alternative supervision source, the work in Chapter 4 switches to using a group of novice annotators providing partial annotations as an alternative supervision source. Chapter 4 presents work on developing parser models that can make use of a wide variety of supervision sources while also allowing for the use of incomplete

partial annotations. Specifically, there is discussion of a Gibbs sampling approach for both constituencies and dependencies, as well as a dependency parser based on minimum spanning trees. Both of these techniques are scalable from entirely unsupervised through partial supervision to fully supervised, which means they could be applicable to a wide variety of projects and scenarios. In addition to the discussions of implementation details, a number of simulation experiments are presented that are intended to validate the use of partial annotations as a supervision source in low-resource environments. By adopting various metrics of total annotation cost, it can be shown that–at least in simulation–there are cost saving benefits to be had by working with partial annotations and for a fixed cost, we might even expect better performance from systems based around partial annotations than full annotations.

The third and final goal of this work is to examine the properties of the actual annotations and annotators that make up a particular small corpus. Chapters 5 and 6 form the core of this argumentation and focus more directly on addressing the costs and practical considerations associated with developing small corpora. Chapter 5 discusses the collection of a partial dependency corpus annotated primarily by novice annotators in Spanish. This chapter is where the notion of partial annotation is first considered as a way to decrease the costs (both temporal and financial) of assembling a corpus while simultaneously increasing the precision of the annotations from a wide range of annotators who have varying personal standards for annotating different linguistic structures.

The major feature of interest for this corpus is the collected information associated with each of the annotators regarding their language background and history of annotation or computational background. This corpus is a primary data source for various experiments throughout this work. Here models are also presented

that were created from the partial dependency corpus is order to better understand the habits and tendencies of small-scale corpus annotators, as well as ways in which these models can be used during parser evaluation in order to better evaluate systems designed for use with novice annotators.

Multiple uses for partial dependency annotations are presented in Chapter 6. While interesting objects of study in their own right, partial dependencies are not directly applicable in many scenarios, especially when using standard techniques. Accordingly, here I examine a number of ways in which partial dependencies can be either converted into a more usable form or used in a limited fashion as-is. The primary result is a number of different dependency imputation schemes that are used to resolve the missing dependency arcs in existing partial annotations. These imputation schemes are quite resilient to degradation, as I demonstrate in both annotator data and simulated corpora; they can recover a substantial portion of missing arcs, reliably producing very usable results even at a missing arc rate of up to 50%. There is also evidence presented from the human-collected data that backs up the simulation-based result of partial annotations achieving better performance on a fixed cost budget.

## 1.4 Audience Impact

While much of the major methodological contributions of this research are of a technical nature, namely the production of parsers capable of working from partial or minimal training data, the potential use cases of such methods are intended to be of practical value for a wide range of researchers and audiences. In particular, there are two main sets of readers that this research is aimed towards, and this section is intended to provide some context from the different points of view of these groups.

**Linguists** Computational linguistics is ultimately a linguistic pursuit; this research is intended to, in the end, be of benefit to the linguist and not simply a computational exercise. In particular, the major practical goal of the research is to be able to produce a useful artifact (a dependency parser) in an achievable, reasonable, and efficient manner. This resulting artifact can be of use to linguists in a variety of ways. The desire for practicality above all else is a driving factor behind many of the decisions made in this research, and a major reason why it should be considered useful: no existing tools, particular theories of language, or existing data are assumed. These assumptions are often fatal in the low-resource contexts considered here, and the fact that much prior work makes these assumptions–either implicitly or explicitly–is likely one reason that computational linguists struggle to provide useful tools to the general linguistics community.

The motivation behind this research is to provide tools in many more languages than are currently considered. Linguists engaged in field work, documentation projects, or larger scale theoretical work may find that the availability of computational tools in their language of study allows for useful methods of analysis not previously practical. Embracing computational methods earlier in the life-cycle of research on languages means that corpus-based techniques–more common on high-resource languages like English–become applicable in low-resource contexts as well. For instance, the parsers developed in this research would be accurate enough to help filter field data for later analysis by facilitating some measure of syntactic search or categorization.

Additionally, apart from the base value of the parsers themselves, they can also be embedded in other tools or pipelines that provide additional value. Methods for tasks such as machine translation and semantic parsing are increasingly making

use of syntactic features that can be supplied relatively accurately by parsers like the ones built here. This means that high-level, practical tasks like translation can be accomplished or aided by the rapid construction of parsers; this would be extremely valuable in, for instance, humanitarian crisis contexts where there is suddenly a mass need to translate in and out of potentially very low-resource languages.

**Computer Scientists** On a technical level, this body of research provides a set of techniques for working in low-resource environments–be they languages or specialized domains. Although work in low-resource contexts has become more popular in recent years, the same assumptions of certain types of data, tooling, and theories discussed above means that not all of the work would transfer neatly into real-world environments; if a low-resource method requires large amounts of cleanly formatted unannotated data, it is in many important ways not a low-resource method. It may be 'low-supervision', but conflating resources for supervision is highly problematic when it comes to maximizing real-world applicability, where the two are clearly separate.

For computer scientists and computational linguistics researchers, this dissertation should be first considered as an example of truly training a parser from the ground-up in as close to a real-world low-resource context as possible. With a small group of minimally trained annotators, we can produce parsers that substantially outperform both baselines and unsupervised methods in less than a day while assuming absolutely zero prior existing tools, supervision, or data. Those parsers will almost certainly not outperform state-of-the-art systems trained on hugely expensive corpora that took years to assemble, but that doesn't mean that they aren't useful in the right contexts.

A second point of emphasis for computational readers is concerned with

evaluation. Standard evaluation metrics for parsing, such as unlabeled attachment score (UAS) are useful in that they are a good way of measuring agreement with a gold-standard annotation, but using simple agreement with a gold-standard for evaluation becomes an issue in low-resource contexts because of constraints and concerns that may be entirely external to the actual experiment. If no gold-standard exists, or if we'd rather trade some agreement for overall cost savings, it becomes harder to justify using the same evaluation metric as is typical in high-resource, standard experimental setups. In Chapter 4 for instance, I adopt an evaluation setup that factors in the cost of the training data in different ways and find different results than a straightforward UAS application. Making informed decisions about the evaluation process should be an important factor in any experiment, but particularly in such a highly contextualized environment as low-resource languages, which bring in a high degree of extrinsic considerations.

Accordingly, although the notion of 'big data' and vast training training sets may be the easiest path to high gold-standard agreement numbers and an easy comparison to prior work, this dissertation should be considered an argument that there will always be a place for smartly applied expert knowledge and for supervision in its many forms.

# Chapter 2

# Representing Linguistic Phenomena

Representation impacts performance, particularly in low-resource scenarios with languages where the linguistic facts are harder to reconcile with the standard representations used in the literature. This is an area where the limits of linguistic knowledge play a role as well; many disagreements about the nature of syntactic phenomena exist, and sometimes fieldwork has yet to come to a consensus either– for instance concerning the reality of morphemes as opposed other sub-lexical units, and whether the fact that humans process and learn morphology differently from syntax or semantics means they should be treated differently.

This chapter considers how issues of linguistic representation interact with supervision. In particular, the benefits of adopting levels of analysis that are different from the standards typically involved in computational linguistics are considered; for instance moving from a word level representation to one based on sub-word units, or abandoning the common representational unit of a sentence and moving

to a discourse-level model. While a movement towards representations that most accurately represent linguistic reality–as best as currently understood–may seem uncontroversial, there is a tension against such a move provided by a number of different facts.

The fact that the field of computational linguistics standardized around a particular set of representations—while often ignoring others—is not particularly surprising. Early technological limitations, the increased difficulty and costs of obtaining training data for more detailed representations, and the linguistic facts of commonly used languages all likely played a role in this largely tacit standardization. If most early researchers were native speakers of a highly morphologically complex language, perhaps the morpheme would have ended up playing a more prominent role in many models, for instance.

Additionally, increasing the linguistic sophistication of our models often requires providing more complex and technical forms of supervision and annotation. Moving from a model of subword classification based solely on coarse parts of speech to one based directly on the morphological forms of the words in question requires providing morphological boundaries for word forms instead of a monolithic label of 'noun' or 'verb'. This makes the task of providing supervision increasingly difficult and restricts the potential set of annotators available to the researcher.

However, just as linguists have expanded their views and theories in the light of new evidence from a multitude of languages, computational linguistics models must inevitably do the same. Models that operate over word level chunks, constrained to work on a single sentence at a time are common, yet entirely inadequate to consider a wide range of linguistic phenomena. Certainly, not all computational linguistics work suffers from this narrow view–entire sub-fields like discourse analysis

and computational morphology depend on breaking out of that view on things–but a much wider range of work could potentially benefit from insights gained in those sub-fields. For example, numerous insights into syntax have come by incorporating theories of phonology [Pullum and Zwicky, 1988], two fields which might at first glance seem entirely separable; there can be little doubt that taking a more holistic view of the representation of linguistic knowledge in computational models can be beneficial in many situations.

The rest of this chapter examines two specific tasks–part of speech tagging and dependency parsing–and considers ways in which different forms of supervision and linguistic representation impact overall performance, both at the level of linguistic knowledge requested from human annotators and the ways in which models themselves can achieve greater linguistic sophistication. Additionally, these examples of supervision enhancement make use of liberal amounts of automated annotation where possible, allowing machine learned models to supply the complex annotation objects that the human annotators may not have the time or ability to provide. Later chapters will draw parallels between these complex annotations and the provisioning of full syntactic trees–a more common yet also complex task that annotators struggle to provide.

## 2.1 Low-Resource POS Tagging[1]

Part of speech (POS) tagging is a common source of supervision when working in syntactic parsing as it provides a simple label that can hopefully provide some predictive ability for how that word form functions in the syntax of the sentences.

[1]**The pos tagging work in this section is joint work with Dan Garrette and Jason Baldridge. [Garrette et al., 2013]**

The appeal of POS tagging is that it is a relatively straightforward task of mapping word forms to a finite–and often quite small–set of labels, and that models utilizing POS tags almost universally outperform models that lack such supervision.

The task is not without its complications however; it is not, for instance, immediately clear what the set of labels should be, nor whether that set should apply to all languages or if languages are free to adopt their own specific set of labels. Additionally, while most linguists would agree that there are many more distinctions present than are commonly made (e.g., 'adverb' is a label that covers a wide variety of phenomena) there is little consensus as to what the proper number of categories should be. A minimum of two (nouns and verbs) is often assumed, but challenges to even that distinction exist as well [Hopper and Thompson, 1985].

Whether or not POS tags are capturing any deep notion of linguistic reality, the fact remains that they are extremely helpful or essential to many current NLP systems. As such, any linguistic annotation project, high- or low-resource, will likely be required to produce annotations and tools for handling these tags.

Various methods for doing automated POS tagging have been previously proposed, although many of them suffer from specific weaknesses in the low-resource case. For example, supervised learning methods can provide high accuracy, but they perform poorly when little supervision is available [Manning, 2011]. Good results in weakly-supervised tagging have been obtained by training sequence models such as hidden Markov models (HMM) using the Expectation-Maximization algorithm (EM), however most work in this area has still relied on relatively large amounts of data, both annotated and unannotated, as well as an assumption that the annotations are very clean [Kupiec, 1992, Merialdo, 1994].

Additionally, most research into weak supervision relies on simulating that

supervision with tag dictionaries extracted from existing large, expertly-annotated corpora. These resources have been developed over long periods of time by trained annotators who collaborate to produce high-quality analyses. They are also biased towards including only the most likely tag for each word type, resulting in a cleaner dictionary than one would find in a real scenario. As such, these experiments do not reflect real-world constraints and are a poor approximation of weak supervision. This work explicitly tries to track the true cost of annotation by performing all annotation in a timed fashion. By doing this, we are able both to see the actual character of data at different blocks (rather than extracting from a fully-formed version), and to accurately translate performance increases into real-world time and dollar values.

Previous work developed a different strategy based on generalizing linguistic input with a computational model: linguists annotated either types or tokens for two hours, these annotations are projected onto a corpus of unlabeled tokens using label propagation and HMMs, and a final POS-tagger is trained on this larger auto-labeled corpus [Garrette et al., 2013]. That approach uses much more realistic types and quantities of resources than previous work; nonetheless, it leaves many open questions regarding the effectiveness of incrementally more annotation, the role of unannotated data, and whether there is a good balance to be found using a *combination* of type- and token-supervision. Additionally, the possibility of increasing the linguistic sophistication of the model through the use of morphological analyzers was left unexplored.

This work addresses these questions via a series of experiments designed to quantify the effect on performance given by the amount of time spent finding or annotating training materials. We specifically look at the impact of four types of data collection:

1. Time annotating sentences (token supervision)

2. Time creating tag dictionary (type supervision)
3. Time constructing a finite state transducer (FST) to analyze word-type morphology
4. Amount of raw data available for training

We explore these strategies in the context of POS-tagging for Kinyarwanda and Malagasy. We also include experiments for English, pretending as though it is a low-resource language. The overwhelming take away from our results is that type supervision—when backed by an effective semi-supervised learning approach—is the most important source of linguistic information. Also, helping to make the case for an increased focus on proper linguistic analysis, morphological analyzers yield improvements for morphologically rich languages when there are few labeled types or tokens (and, it never hurts to use them). Finally, performance improves with more raw data, though we see diminishing returns past 400,000 tokens. With just four hours of type annotation, our system obtains good accuracy across the three languages: 89.8% on English, 81.9% on Kinyarwanda, and 81.2% on Malagasy.

Tracking the real time costs associated with the production of these numbers is an important aspect of this work. While these results are not state-of-the-art tagging numbers, they were produced with a fraction of the time and budget needed by the corpora on which state-of-the-art methods rely.

Our results compare favorably with previous work despite using considerably less supervision and a more difficult set of tags. For example, Li et al. use the entirety of English Wiktionary directly as a tag dictionary to obtain 87.1% accuracy on English, below our result.[Li et al., 2012] Täckström et al. average 88.8% across 8 major languages, but for Turkish, a morphologically rich language, they achieve only 65.2%, significantly below our 81.9% for morphologically-rich

| | KIN | | MLG | | ENG - Experienced | | ENG - Novice | |
|---|---|---|---|---|---|---|---|---|
| time | type | token | type | token | type | token | type | token |
| 1:00 | 801 | 559 (1093) | 660 | 422 (899) | 910 | 522 (1124) | 210 | 308 (599) |
| 2:00 | 1814 | 948 (2093) | 1363 | 785 (1923) | 2660 | 1036 (2375) | 631 | 646 (1429) |
| 3:00 | 2539 | 1324 (3176) | 2043 | 1082 (3064) | 4561 | 1314 (3222) | 1350 | 953 (2178) |
| 4:00 | 3682 | 1651 (4119) | 2773 | 1378 (4227) | 6598 | 1697 (4376) | 2185 | 1220 (2933) |

Table 2.1: Annotations for each language and annotator as time increases. Shows the number of tag dictionary entries from type annotation vs. token. (The count of labeled tokens is shown in parentheses). For brevity, the table only shows hourly progress.

Kinyarwanda.[Täckström et al., 2013]

### 2.1.1 Data Collection

Kinyarwanda (KIN) and Malagasy (MLG) are low-resource, KIN is morphologically rich, and English (ENG) is used for comparison. For each language, sentences were divided into four sets: training data to be labeled by annotators, raw training data, development data, and test data.

**Data sources** The KIN texts are transcripts of testimonies by survivors of the Rwandan genocide provided by the Kigali Genocide Memorial Center. The MLG texts are articles from *Lakroa*, *La Gazette*, and Malagasy Global Voices. Texts in both KIN and MLG were tokenized and labeled with POS tags by two linguistics graduate students, each of which was studying one of the languages. The KIN and MLG data have 12 and 23 distinct POS tags, respectively.

The Penn Treebank (PTB) [Marcus et al., 1993] is used as ENG data. Section 01 was used for token-supervised annotation, sections 02-14 were used as raw data, 15-18 for development of the FST, 19-21 as a dev set and 22-24 as a test set. The PTB uses 45 distinct POS tags.

| | ENG - Exp. | | ENG - Nov. | |
|---|---|---|---|---|
| time | type | tok | type | tok |
| 1:00 | 0.05 | 0.03 | 0.01 | 0.02 |
| 2:00 | 0.15 | 0.05 | 0.03 | 0.03 |
| 3:00 | 0.24 | 0.06 | 0.07 | 0.05 |
| 4:00 | 0.32 | 0.08 | 0.11 | 0.06 |

Table 2.2: Tag dictionary recall against the test set for ENG annotators on type and token annotations.

**Collecting annotations** Linguists with non-native knowledge of KIN and MLG produced annotations for four hours (in 30-minute intervals) for two tasks. In the first task, type-supervision, the annotator was given a list of the words in the target language (ranked from most to least frequent), and they annotated each word type with its potential POS tags. The word types and frequencies used for this task were taken from the raw training data and did not include the test sets. In the second task, token-supervision, full sentences were annotated with POS tags. The 30-minute intervals allow us to investigate the incremental benefit of additional annotation of each type as well as how both annotation types might be combined within a fixed annotation budget.

Table 2.1 gives statistics for all languages and annotators showing progress during the 4-hour tasks. With token-annotation, tag dictionary growth slows because high-frequency words are repeatedly annotated, producing only additional frequency and sequence information. In contrast, every type-annotation label is a new tag dictionary entry. For types, growth increases over time, reflecting the fact that high-frequency words (which are addressed first) tend to be more ambiguous and thus require more careful thought than later words.

### 2.1.2 Approach

Learning under low-resource conditions is more difficult than scenarios in most previous POS work because the vast majority of the word types in the training and test data are not covered by the annotations. When most words are unknown, learning algorithms such as EM struggle. Recall that most work on learning POS-taggers from tag dictionaries used tag dictionaries culled from *test* sets (even when considering incomplete dictionaries). We thus build on our previous approach, which exploits extremely sparse, human-generated annotations that are produced without knowledge of which words appear in the test set.

This approach generalizes a small initial tag dictionary to include unannotated word types appearing in raw data. It estimates word/tag pair and tag-transition frequency information using model-minimization, which also reduces noise introduced by automatic tag dictionary expansion. The approach exploits type annotations effectively to learn parameters for out-of-vocabulary words and infer missing frequency and sequence information. This pipeline is described in detail in the previous work, so we give only a brief overview and describe our additions.

The purpose of tag dictionary expansion is to estimate label distributions for tokens in a raw corpus, including words missing in the annotations. For this, a graph connecting annotated words to unannotated words via features is constructed and POS labels are pushed between these items using label propagation (LP). LP has been used successfully for extending POS labels from high-resource languages to low via parallel corpora, among other tasks[Das and Petrov, 2011, Täckström et al., 2013, Ding, 2011]. These works have typically used n-gram features (capturing basic syntax) and character affixes (basic morphology).

The character n-gram affix-as-morphology approach produces many features,

but only a fraction of them represent actual morphemes. Incorrect features end up pushing noise around the graph, so affixes can lead to more false labels that drown out the true labels.

While affixes may be sufficient for languages with limited morphology, their effectiveness diminishes for morphology-rich languages, which have much higher type-to-token ratios. More types means sparser word frequency statistics and more out-of-vocabulary items, and thus problems for EM. Here, we modify the LP graph by supplementing or replacing generic affix features with a focused set of morphological features produced by an FST. These targeted morphological features are effective during LP because words that share them are much more likely to actually share POS tags and be morphologically related.

FSTs produce multiple analyses, which is actually advantageous for LP. Ambiguities need not be resolved since we just take the union of all morphological features for all analyses and use them as features in the graph. Note that each FST produces its own POS-tags as features, but these do *not* correspond to the target POS tagset used by the tagger. This is important because it decouples FST development and the final POS task. Thus, any FST for the language, regardless of its provenance, can be used with any target POS tagset.

The use of FSTs in this context is an example of the use of a proxy form of automated supervision in order to provide a complex annotation object (here a morphological analysis) where annotators may not be capable of providing it themselves.

| | KIN | | | | MLG | | | |
|---|---|---|---|---|---|---|---|---|
| time | No-LP | Aff. Only | FST Only | Aff+FST | No-LP | Aff. Only | FST Only | Aff+FST |
| 1:00 | 57.84 | 76.02 | 78.78 | **79.00** | 69.25 | 76.16 | **76.58** | 75.94 |
| 2:00 | 63.66 | 80.03 | 80.84 | **80.91** | 74.10 | 79.14 | **79.72** | 79.05 |
| 3:00 | 66.04 | 81.14 | 81.21 | **81.31** | 76.24 | 79.66 | **80.55** | 79.97 |
| 4:00 | 69.16 | 81.75 | 81.47 | **81.93** | 77.47 | 80.63 | **81.00** | 80.79 |

Table 2.3: Impact of FST features

### 2.1.3 Results

The overall best tagging accuracies achieved by language are 81.9% for KIN using all types, 81.2% for MLG using half types and half tokens, and 89.8% for ENG using all types and the maximal amount of raw data. All of these best values were achieved using both FST and affix LP features. Table 2.3 contains a summary of the type-based results for KIN and MLG.

All results described in this section are averaged over five folds of raw data.

**Types versus tokens** The primary research question in this work was the relationship between supervision type and time. Annotation must be done by someone familiar with the target language, linguistics, and the target POS tagset. To make the best use of their time, we need to know which form of supervision is most useful so that efforts can be concentrated there. Additionally, it is useful to identify when returns on annotation effort diminish so that annotators do not spend time doing work that is unlikely to add much value.

The annotators produced four hours each of type and token annotations, each in 30-minute increments. To assess the effects of annotation time, we trained taggers cumulatively on each increment and determine the value of each additional half-hour of effort. Results are shown for ENG in Figure 2.1. In all scenarios, the use of LP (and model minimization) delivers huge performance gains. Additionally, the use of FST features, usually along with affixes, yielded better results than

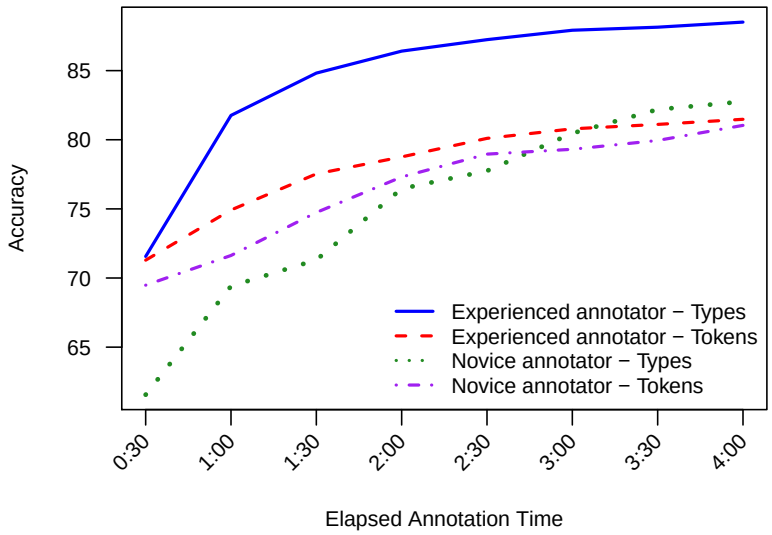



Figure 2.1: Annotation time vs. tagger accuracy for ENG type-only and token-only annotations with affix and FST LP features.

without. This indicates the LP procedure makes effective use of the morphological features produced by the FST and that the affix features are able to capture missing information without adding too much noise to the LP graph.

Performance is considerably better when type annotations are used than only tokens. Type annotations plateau much faster, so a shorter amount of time must be spent annotating types than if token annotations are used. For KIN it takes approximately 1.5 hours to reach near-maximum accuracy for types, but 2.5 hours for tokens. This difference is due to the fact that the type annotations started with the most frequent words whereas the token annotations were on random sentences. Thus, type annotations quickly cover a significant portion of the language's tokens. With annotations directly on tokens, some of the highest frequency types are covered, but annotation time is also ineffectively used on low-frequency types that happen to appear in those sentences.

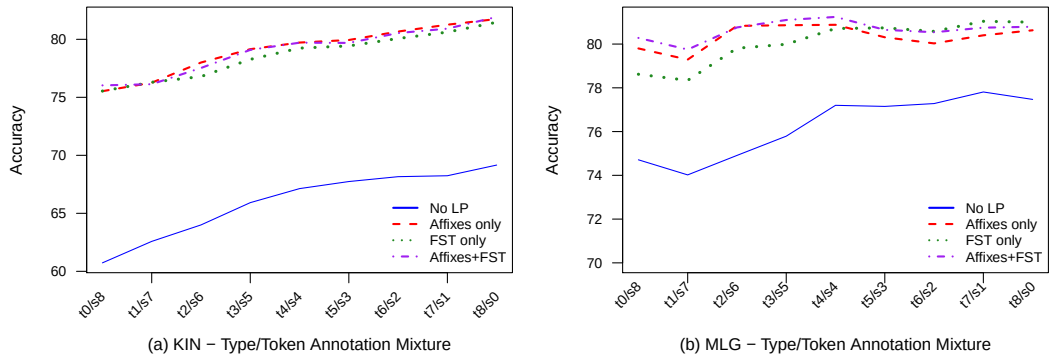



Figure 2.2: Annotation mixture vs. tagger accuracy. X-axis labels give annotation proportions, e.g. "t2/s6" indicates 2/8 of the time (1 hour) was spent annotating types and 6/8 (3 hours), full sentences.

**Type-Token Mixtures** Because type and token annotations are each better at providing different information — a tag dictionary of high-frequency words vs. sequence and frequency information — it is reasonable to expect that a combination of the two might yield higher performance by each contributing different but complementary information during training.

This matters in low-resource settings because type or token annotations will likely be produced by the same people, so there is a tradeoff between spending resources on one form of annotation over the other. Understanding the best mixture of annotations can inform us on how to maximize the benefit of a set annotation budget. To this end, we ran experiments fixing the annotation time to four hours while varying the mix of type and token annotations. Results are shown for KIN and MLG in Figure 2.2. In general, we found increasing the proportion of types led to increased performance.

Täckström et al. explore the use of mixed type and token annotations in which a tagger is learned by projecting information via parallel text. In their experiments, they—like us—found that type information is more valuable than token

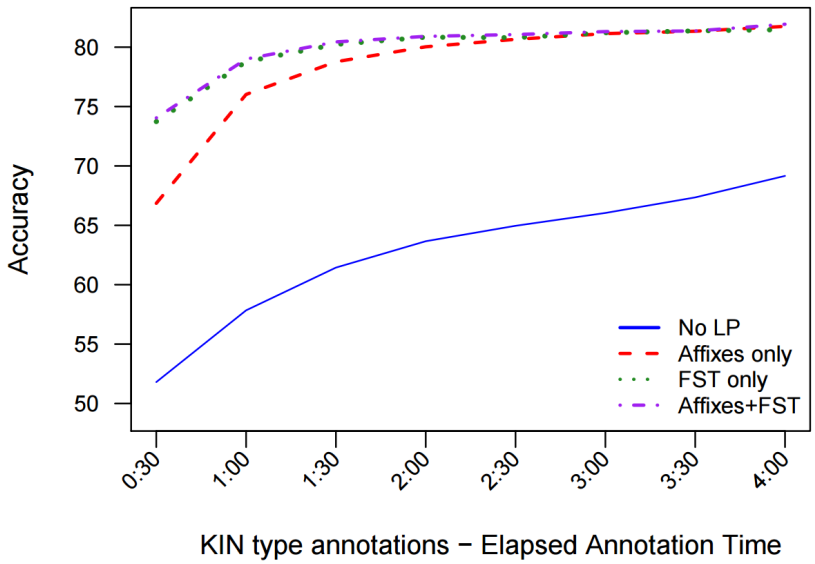



Figure 2.3: Annotation time vs. tagger accuracy for KIN type-only annotations with affix and FST LP features.

information. However, they were able to see gains through the complementary effects of mixing type and token annotations. It is likely that this difference in our results is due to the amount of annotated data used. It seems that the amount of type information collected in four hours is not sufficient to saturate the system, meaning that switching to annotating tokens tends to hurt performance.

**Transducer Features** Finally, the use of FST features yields the largest gains for KIN as shown in Figure 2.3, but particularly when small amounts of annotation are available. This makes sense: KIN is a morphologically rich language, so sparsity is greater and crude affixes capture less actual morphology. With little annotated data, LP relies heavily on morphological features to make clean links between words. But, with more annotations, the gains of the FST over affix features alone diminishes: the affix features eventually presumably capture enough of the morphology to make

up the difference.

**Conclusions** Care must be taken when drawing conclusions from small-scale annotation studies such as those presented in this section. Nonetheless, we have explored realistic annotation scenarios for POS-tagging for low-resource languages and found several consistent patterns. Most importantly, it is clear that task-level (type) annotations are the most useful input one can obtain from a linguist—provided a semi-supervised algorithm for projecting that information reliably onto raw tokens is available. In a sense, this result validates the research trajectory of efforts of the past two decades put into learning taggers from tag dictionaries: papers have successively removed layers of unrealistic assumptions, and in doing so have produced pipelines for task-level supervision that easily beat instance-level supervision prepared in comparable amounts of time.

The result of most immediate practical value is that we show it is possible to train effective POS-taggers on actual low-resource languages given only a relatively small amount of unlabeled text and a few hours of annotation by a non-native linguist. Instead of having annotators label full sentences as one might expect the natural choice would be, it is much more effective to simply extract a list of the most frequent word types in the language and concentrate efforts on annotating these types with their potential parts of speech. Furthermore, particularly for languages with rich morphology, adopting a more linguistically sophisticated model of subword types and allowing for the use of features from morphological transducer can yield significant performance gains, particularly when large amounts of other annotated resources are unavailable. This is a good example of how redefining the linguistic representation of a problem can impact performance, and how potentially negative consequences of that move–namely increased annotation effort–can be mitigated by

an efficient resource development plan.

A final result is that, as might be expected, the use of additional raw text is able to improve performance. However, these gains are small in comparison to the potential benefits of obtaining even a few hours worth of type annotations. This result helps to demonstrate that although the use of unsupervised methods may seem natural in the context of low-resource languages, the cost of obtaining those additional raw texts (which may be quite high for the majority of the world's languages) is simply not worth it relative to the benefits of enlisting the services of an annotator for a limited time.

## 2.2 Sub-Word Modeling

Supervision in the form of POS tagging, as discussed in the previous section, is about seeking and providing some generalizations for syntactic behavior. But if those generalizations are useful—and empirically they are—can we achieve even more benefit by attempting to learn and reason over the raw content that directs syntactic function? This is the goal of syntactic parsing setups that attempt to use morphological information to create more accurate syntactic parses.

The reasoning behind this idea is quite logical. Consider a sentence of three words with POS tags 'Verb', 'Noun', 'Noun'; if all we have is that information then an ambiguity exists between an analysis with just one noun phrase (using both nouns) or two. If, on the other hand, we're able to analyze the verb form and find morphology indicating that verb is anticipating two nominal arguments, the possible space of syntactic parses is further reduced.

This type of information could be encoded either as a literal representation of the morphology in the tokens by marking the morpheme transitions, or by essentially

expanding the set of available POS tags to encode relevant information. Consider the example below from Swahili [Deen, 2002].

(1) Juma alinunua vitabu

| | | | | | | |
|---|---|---|---|---|---|---|
| Juma | a- | li- | nunu | -a | vi- | tabu |
| Juma | SA3s- | past- | buy | -IND | 8- | book |

'Juma bought books.'

(2) Juma aliwanunulia watoto vitabu

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Juma | a- | li- | wa- | nunu | -li | -a | wa- | toto | vi- | tabu |
| Juma | SA3s- | past- | OA3pl- | buy | -APPL | -IND | 2- | child | 8- | book |

'Juma bought the children books.'

In these examples we could choose to represent the verb in (1) and (2) and 'Verb'/'Verb-APP', to differentiate the applicative form from the non-applicative, or we could represent them as 'a+li+nunu+a'/'a+li+wa+nunu+li+a', with each morpheme receiving a sub-token. This section concerns itself with generating supervision of the latter type, providing an actual morphological segmentation of the surface form to the parser.

As with POS tag labels, morphology is far from a settled area of linguistic theory, but rather than being based on the abstract idea of categorial labels, morphemes do in fact have a psychological reality that can be quantifiably observed in speakers. This is an attractive quality for the argument that morphological awareness (via segmentations or analyses of surface forms) could be a better form of supervision for the task of syntactic parsing than POS tags alone.

However, producing morphological segmentations of any reasonable accuracy is quite hard. There are essentially two major ways in which these segmentations

can be produced: via unsupervised morphological induction or via a finite-state transducer (FST) as was demonstrated in the section on POS tagging.

### 2.2.1 Obtaining Morphological Features

Especially for low-resource languages, where even field linguists can struggle to identify and label morphological information reliably, obtaining morphological segmentations is difficult. Standard POS tagging is dramatically easier (given a reasonable tag set size), which is one major reason that POS tagging dominates when it comes to lexical supervision.

**Baselines** The simplest possible way of producing morphological segmentations is to simply split up the surface form via some deterministic or stochastic process. For instance, given the Swahili verb form 'aliwanunulia' from Example (2), we could break it into chunks of two characters, yielding the segmentation 'al+iw+an+un+ul+ia', or randomly into chunks of 2-4 characters, possibly yielding 'al+iwa+nu+nu+lia'.

In the experiments presented in this section, baselines that split forms into uniform chunks of length 1-4 are used. Most languages of the world have an average morpheme size of less than four characters, so it is reasonable to assume any splits based on longer chunks would result in poorer segmentations overall.

Although these methods are used as 'baselines', similar simple models of subword structure have yielded excellent, interesting results in other systems. For instance, moving from a word-based model of machine translation to a character-based model has been demonstrated to not only slightly increase the performance of the system, but to increase its performance in linguistically sophisticated ways [Ling et al., 2015]. The system is capable of achieving accurate (or at least intelligent guesses) at the translation of previously unseen words by being able to identify sub-

word structures and patterns within the unknown word.

Given that the goal of these morphological feature generating models is really to provide useful features and information to the parser, machine translation model, or tagger, even 'baselines' may be successful if the downstream task is capable of sifting through the noise inherent in these less sophisticated segmentation techniques. This is a trade-off of linguistic sophistication: either the supervision, via automated process or human annotators, can try to be as accurate as possible linguistically speaking; or the machine learner can be sophisticated enough to learn the requisite patterns for the task it is performing. In the case of the POS tagger described in Section 2.1, the tagger was more successful when humans provided more accurate linguistic information via finite state transducers; alternatively, the character-based machine translation model of Ling et al. was able to sort through the many character transitions and identify useful patterns to make informed choices for the translation of unknown words.

**Unsupervised Methods** A more intelligent way to split forms, while still requiring minimal or no human effort, is to attempt to identify chunks of characters that occur together with some high frequency. If a system is given a large quantity of English data, it may determine, for instance, that the three character chunk 'ing' occurs much more frequently than we might expect by chance. Such a system could attempt to identify any number of these potential morphemes in a corpus.

This is the goal of unsupervised morphological segmentation, and the modern standard system used for unsupervised morphological segmentation is Morfessor [Virpioja et al., 2013]. Morfessor works as described above, by identifying common chunks and labeling them as morphemes throughout the training corpus.

In the experiments in this section, Morfessor is used to provide supervision

for the parser in the form of unsupervised morphological segmentations.

**Finite State Transducers** The final common way in which morphological segmentations are created is through the use of a finite state transducer (FST). An FST is a finite state automaton that maps an input sequence (in this case characters from a word form) into an output sequence (here a series of morphemes).

While FSTs have the potential to be the most accurate of all the methods described here, they also require the most human effort. All FSTs used for morphological segmentation need to be coded by hand by a linguist with either an expert understanding of the target language or a suitable reference grammar.

There is some experimental evidence to suggest that helpful FSTs may be created in a low-resource environment. The FSTs used in Section 2.1 to supply supplementary features for the POS label propagation system were created by a linguist with no knowledge of the target language (Kinyarwanda) and only a reference grammar for help. Even under these circumstances, with only 10 hours of development time, the FSTs produced were able to provide features that increased the accuracy of the final POS taggings.

Due to the difficultly in obtaining FSTs for a variety of languages, In this section only their use in the case of Finnish is described.

### 2.2.2 Morphological Embeddings for Syntactic Parsing

An increasingly popular method of parsing, continuous-state parsing, uses a method of maintaining information about the state of a parser by embedding that state in a high-dimensional vector. This high-dimensional vector can then be used to make decisions regarding the actions that the parser should take with regard to parsing the current sentence. Any number of representations could potentially be used to

encode the representation of the current state, although a popular scheme uses long short-term memory (LSTM) recurrent neural networks to do this encoding.

The LSTM-Parser of Dyer et al. crucially depends on the representation of individual words embedded in a vector space [Dyer et al., 2015], which makes it a particularly useful environment to consider the effects that different levels or methods of word representation have on syntactic parsing. The basis for this embedding space can be varied; in the original formulation, entire words are embedded based on counts of their occurrence with other words in a standard distributional representation. Ballesteros et al. made the simple change to represent a word as the concatenation of the representations for all of its characters, while the individual character representations are calculated in the same distributional manner as the words in the original formulation [Ballesteros et al., 2015]. A schematic view of the parser is shown in Figure 2.4, where the hatched blocks are the representations for the individual words that are being varied in this experiment.

This work expands further on the idea of Ballesteros et al. by allowing for the basic unit of representation to be arbitrary length sub-lexical units. Under this formulation, the character-based model of Ballesteros et al. is equivalent to using a unigram morphological segmentation, while new embedding spaces can be realized by using any number of candidate segmentation schemes. By opening up the space of representations, this modified model gains linguistic sophistication by acknowledging that words are not simply atomic units and that the sub-units they consist of contribute in some meaningful way to the behavior and function of that word in syntax.

This linguistic knowledge regarding the function of words can alternatively be provided by a separate supervision source, namely POS tags, and in the original

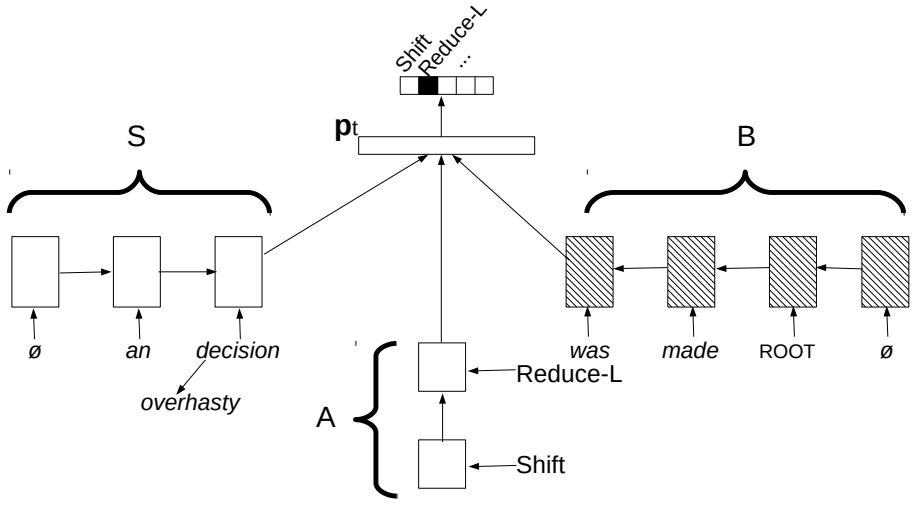



Figure 2.4: LSTM-Parser schematic from Dyer et al. (2015). In the figure, 'S' is the stack of partially completed dependency subtrees, 'A' is the stack of actions previously taken by the parser, and 'B' is the buffer of words waiting to be read. These are combined to form $P_t$, the representation of the parser state upon which a decision regarding the next action of the parser is made. This work crucially modifies the representations of the words, shown in hatching here.

formulation of the LSTM-Parser POS tags are provided–to great positive effect. In the sub-word based models (either characters or morphemes) POS tags are withheld because they are providing essentially the same information and would obscure the effects of the morphological segmentations.

I ran a set of experiments on all of the languages in the Universal Dependencies corpora. Similar patterns held for all languages, and this section will provide illustrative examples.

In all experiments, the standard training and evaluation splits were used. Morphological information was provided to the parser by either simply splitting the words into chunks of varying lengths or by passing the word through a trained Morfessor model. In most cases the Morfessor model was unsupervised, and in those cases was trained on the training data of the Universal Dependencies corpus. In the

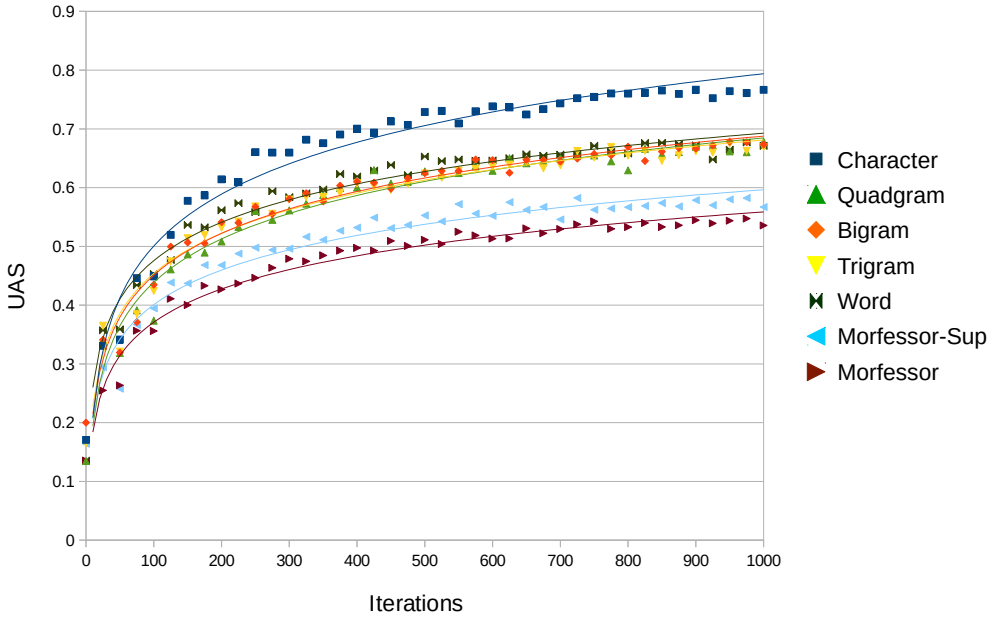



Figure 2.5: Comparison of parsing accuracy with various morphological embeddings for Finnish.

case of Finnish a small number of gold morphological segmentations (roughly 2000) was available through the Morpho Challenge 2010 task on unsupervised morphological segmentation [Kurimo et al., 2010]. This data was used to train a supervised Morfessor model, in addition to the unsupervised model.

The results of parsing the Finnish data with the LSTM model using these morphological embeddings are shown in Figure 2.5, which shows Unlabeled Attachment Scores (the percentage of tokens with correct heads) over training. The clear winner is chunking with one-character chunks, with two-, three-, and four-character chunks all essentially tying for second place. The two Morfessor-based models, despite having the best overall segmentations when scored using an FST (See Table 2.4), produced the worst results when used as supervision for parsing.

The reasons why the more sophisticated segmenting strategies were so to-

| Segmenting Method | Precision | Recall | F-Score | UAS |
|---|---|---|---|---|
| Unigrams | 0.02 | **1.00** | 0.05 | **76.2** |
| Bigrams | 0.11 | 0.89 | 0.20 | 68.8 |
| Trigrams | 0.24 | 0.88 | 0.37 | 68.5 |
| Quadgrams | 0.43 | 0.81 | 0.56 | 67.5 |
| Supervised Morfessor | 0.67 | 0.64 | **0.66** | 57.4 |
| Morfessor | **0.70** | 0.59 | 0.64 | 54.9 |

Table 2.4: Finnish Morphological Segmentation Statistics

tally dominated by the baselines is not immediately clear, but Table 2.4 provides some clues. The methods can be ranked by morphological boundary recall; that is, segmentation methods that produced greater morpheme boundary recall rates produced better parsing results when used as a source of supervision. This fact may also explain the tight grouping of the two-, three-, and four-character chunking methods as well–they all had similar recall values which is to be expected given average morpheme lengths. Figure 2.7 shows the tight correlation between boundary recall and UAS scores. This same result, with character embeddings outperforming a grouping of chunking methods followed by Morfessor models, was observed in the other languages as well, although the overall delta in UAS scores was not as substantial for languages with fewer morphemes–see Figure 2.6 for the Spanish results. Note that exact morpheme boundary recall values are unavailable for other languages due to the lack of quality FST s to provide supervision.

If it is true that morpheme boundary recall is the major determining factor when it comes to the success of parsing with morphological embeddings, it would imply that the LSTM parser is capable of learning which boundaries are useful and which aren't; as long as it is has access to enough 'true' boundaries, it can adequately filter out the excess noise. This fact may have some precedent in linguistic theory, as morphological boundaries are able to be learned at a much earlier

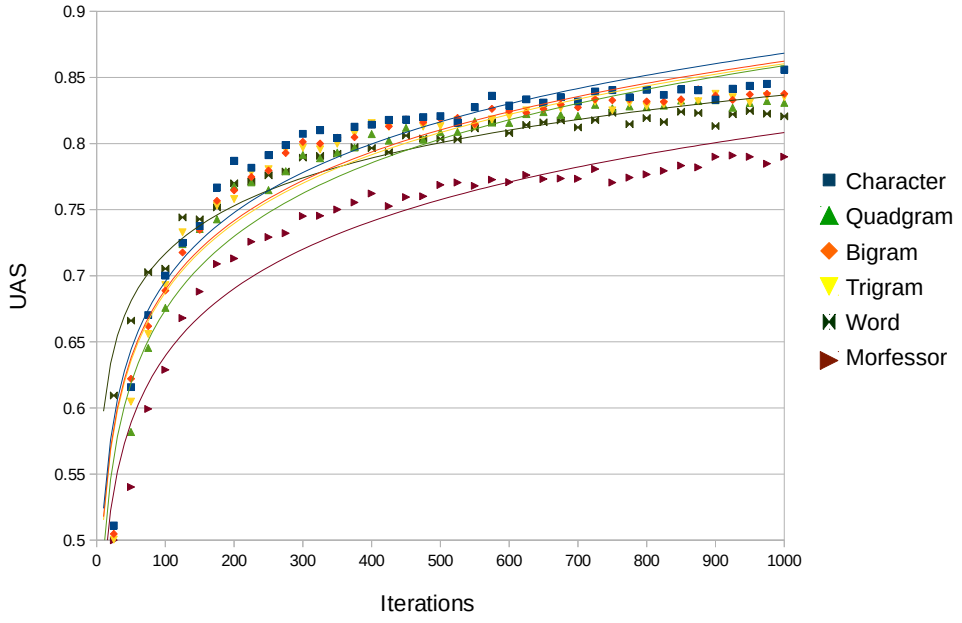



Figure 2.6: Comparison of parsing accuracy with various morphological embeddings for Spanish.

developmental stage than more complex notions like syntax and semantics; experimental evidence indicates that preverbal infants as young as 11 months can quickly and accurately identify morpheme boundaries and sub-lexical items based solely on frequency effects, without the aid of semantics [Marquis and Shi, 2012].

On the surface, the poor parsing performance of the more linguistically sophisticated representations provided by the Morfessor models is a negative result for the idea that morphological information is useful in parsing, and more generally that data with increased linguistic information is inherently preferred. However, giving the parser access to this additional level of representation did increase performance over treating words as atomic units.

In other words, the specific failing illustrated here was that none of the informed segmentation methods were able to provide useful segmentations for the

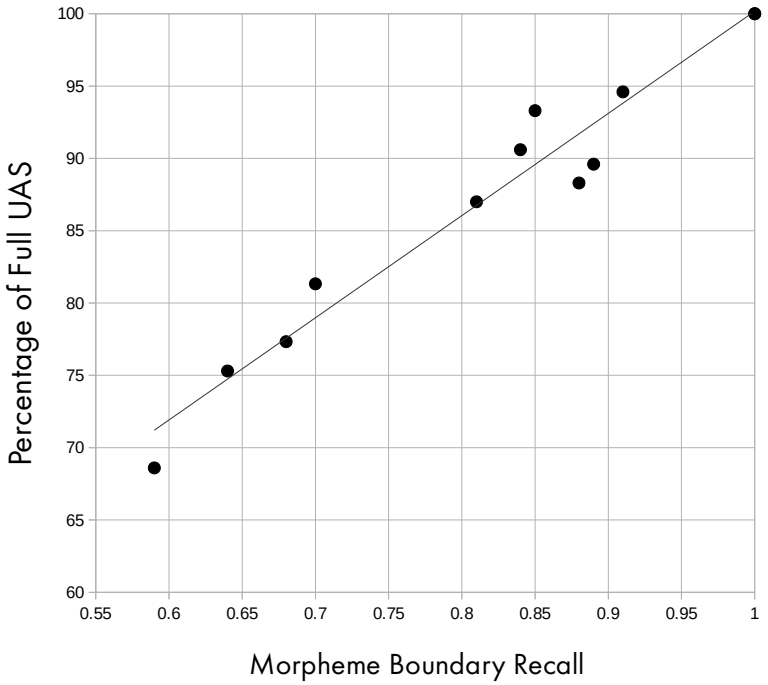



Figure 2.7: Relative UAS as a function of morpheme boundary recall in training segmentations

parser, and not that sub-word representations were of no value. It is important to note that even the segmentations produced by the supervised Morfessor model were not particularly great, with an F-Score of only 0.64. It remains to be seen whether high-quality morphological supervision could potentially improve parser performance over the single character baseline. The lack of large morphologically segmented corpora made this an impossible condition to test for, and the production of such a corpus was outside the scope of this dissertation.

# Chapter 3

# Partial Annotations with GFL

The central problem associated with the production of tree-based syntactic annotations in a small scale, small team environment is the technical knowledge and overall complexity of the annotations themselves. In the typical case, a novice annotator with only a minimal, non-specialist training in syntax will be having to learn to deal with both the tree as an object as well as the actual annotation content simultaneously. This leads to potentially long training periods where the data from these annotators may be of questionable quality with regards to standardization practices. For instance, the Penn Treebank annotation project required a multi-month training period for its annotators, and made use of a substantial set of guidelines [Marcus et al., 1993]. While the resulting data was very high quality and became a standard dataset in the field, the process was tremendously expensive and not feasible for most annotation projects.

The overall complex and sheer number of decisions that go into a tree-based annotation make them one of the more difficult annotations to obtain. In a dependency tree annotation, annotators must specify a head for each word in the sentence

in a series of perhaps 30 decisions that are not always intuitive and often interact–changing the analysis of one part of the tree may require updating other parts as well in a cascading series of changes. This is in contrast to annotation tasks like POS tagging or word sense disambiguation, where each decision is largely independent; the structured nature of the task imposes an additional burden on the annotators to adhere to that structure.

Like the difficult annotation objects from Chapter 2, notably morphological segmentations, we would like to find a way to obtain dependency trees more cheaply. The solutions in Chapter 2 were all centered around getting algorithms to provide the supervision for us, without the direct intervention of human annotators; finite state transducers and unsupervised morphological segmentation methods were considered as sources of the supervision. There is a roughly equivalent technique that is often used in the production of syntactic treebanks: an existing parser creates a best-guess parse for the sentence to be annotated and human annotators then evaluate and potentially correct the structure. This standard production method is often impossible in the area of low-resource languages, because the corpus being produced is often the first in that language–which complicates the process of finding an existing parser.

Instead of decreasing annotation cost by having a machine learner supply the annotations, the approach used in this dissertation is to allow for annotators to only partially specify the annotation for a sentence rather than requiring that they specify the full tree. This substantially reduces the annotation load on individual annotators and has a number of beneficial properties that will be considered in the remaining chapters of this dissertation. In particular, the remainder of this chapter will introduce an annotation scheme called Graph Fragment Language (GFL) that

| Annotation | Symbol | Example |
|---|---|---|
| Dependency | > X < | The > dog |
| Constituent | ( X ) | (That guy there) > ate < it |
| Bracket Head | * | (The bag* of cheese) > disappeared |

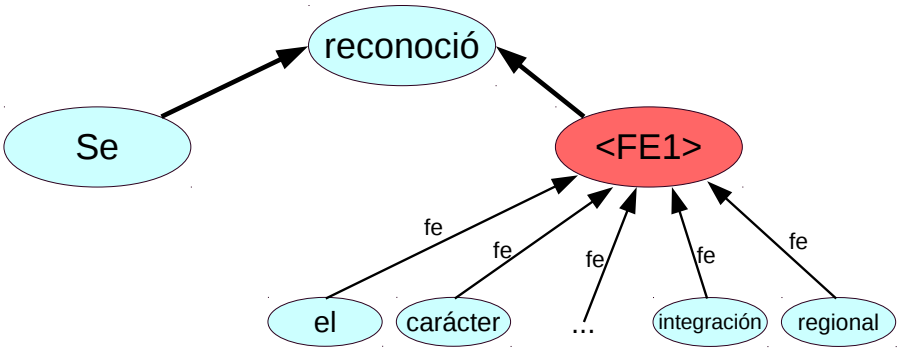
Table 3.1: GFL Elements

is specifically suited for the annotation of partial dependencies, Chapter 4 describes parsing methods that can leverage partial annotations, Chapter 5 details the creation of a corpus of partial dependencies and examines habits of the annotators, and Chapter 6 contains experimental results making use of the new corpus.

## 3.1 Graph-Fragment Language

Both in this Spanish corpus and throughout the remainder of this work, the partial syntactic annotation framework that I will be considering is the Fragmentary Unlabeled Dependency Grammar (FUDG) formalism [Schneider et al., 2013]. The primary advantage that FUDG offers over more traditional annotation schemes for producing this corpus is that it has been specifically designed to make minimal commitments to any one particular theoretical perspective. This is potentially of particular benefit to researchers working in a low-resource environment on languages for which the understanding of the syntax are still in flux. By attempting to abstract away from committing to an actual syntactic analysis, annotations from the early stages of analysis can potentially remain relevant even after the analysis of a construction has changed.

A single FUDG annotation consists underlyingly of a directed graph with nodes representing lexical units (which may be either single words, multiword expressions, or an underspecified section of the sentence known as a *fudge expression*




Se > reconoció < (el carácter dinámico y~1 creciente de~1 la~1 globalización y~2 de~2 la~2 integración regional)

*'the dynamic and growing nature of globalization and regional integration was recognized'*

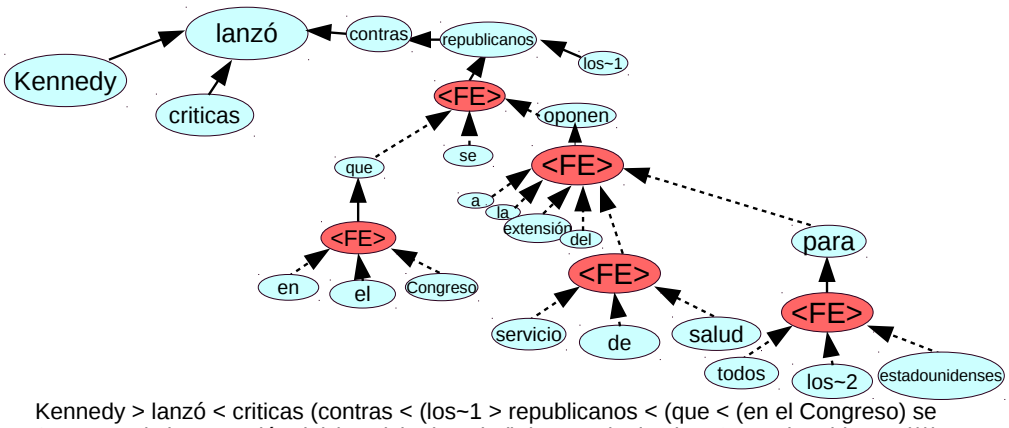
Figure 3.1: Spanish GFL Example

(FE)) and edges representing dependency links.

To specify FUDG annotations, a plain-text dependency notation format known as Graph Fragment Language (GFL) is used. GFL is used to encode the various fragments of the FUDG graph and specify the constraints that exist between them.

The elements of GFL that the annotators for this work are shown in Table 3.1 above. Of particular note is the ability to underspecify sections of a sentence; this is what allows annotators to essentially skip parts (potentially large parts) of sentences that they are unsure of the correct analysis for, for instance 'That guy there' or 'The bag of cheese' in Table 3.1. Annotators may be unsure of the analysis for different reasons, some of which I will explicitly consider in this work. For instance, the issue may be related to the annotators ability to understand the sentence—Chapter 6 presents experiments designed to consider the implications of using annotators who do not have much fluency in the target language. Alternatively, the annotators may be unsure as to the correct way to analyze a particular construction—Chapter 6 introduces experiments and models aimed at ways in which working in a FUDG/GFL setup (or any other theoretically-independent formalism) allows for much greater flexibility during parser evaluation and corpus/domain adaptation.

Some examples of GFL annotations and their corresponding graphs are shown




Kennedy > lanzó < criticas (contras < (los~1 > republicanos < (que < (en el Congreso) se Oponen < (a la extensión del (servicio de salud) (para < (todos los~2 estadounidenses))))

*'Kennedy launched criticisms against Republicans in Congress opposed to the extension of health services for all Americans"*

Figure 3.2: Spanish GFL Example—In this figure, dashed lines represent fudg-expression connections and notations such as '~1' indicate the count of the instance of a word type that occurs multiple times in the sentence.

in Figures 3.1 and 3.2. In Figure 3.1, there is a good example of an FE node, where the annotator has left the internal structure of a large noun phrase with multiple coordinations unspecified, likely saving themselves a lot of time. Additionally, in both of the examples, the annotator has chosen not to provide dependencies for all of the determiners—these are easy for parsers to get right, so not having an annotator specify them is usually a perfectly fine idea, and it saves time on each sentence as well.

These GFL examples came from an annotator with very minimal training, which is described in the next section. While there are certainly elements of the annotation that could have been done more efficiently, for instance the bracketing on 'salud para todos los 2 estadounidenses', in general the annotators reported feeling confident using the limited set of annotations.

### 3.1.1 Underspecification

Underspecification is the most practical and immediately obvious benefit of using GFL. The use of underspecification can occur in two ways: either the annotator does not make any reference to a particular token, leaving it entirely isolated from the

graph, or they include it in a FE node as discussed above and specify a section of the graph to which it may be attached, but do not explicitly name the attachment point. Underspecifications of the first type are commonly used by novice annotators you simply have no intuitions about parts of the sentence, whereas underspecifications of the second type have a wider range of uses. For instance, a novice annotator may recognize a complex noun phrase and include it in an FE node without specifying the internal structure because they are unable to, while an experienced annotator may do the same thing in order to save time and annotate more interesting parts of the sentence.

Additionally, underspecifications have a use as a sort of annotation placeholder in cases where the project has not settled on standard representation for a particular structure. Rather than forcing commitment to a particular linguistic theory early on in the annotation process, the use of underspecification allows this framework to put off the details of specification while still potentially marking the grouping of words as being of interest. Later on, once decisions have been made these sections could then be re-annotated.

Projects with multiple annotators of different ability levels gain an additional benefit from underspecification–specialization and targeted application of expert knowledge. Working in a partial annotation environment allows for expensive, expert annotators to target their annotation work to only those areas that require it.

The potential for increased value of underspecification with multiple annotators is illustrated in Figure 3.3. Depending on the types of annotators available to a given project, the use of annotations that degrade elegantly into partial annotations allows for a variety of potential annotation plans. For instance, timeline

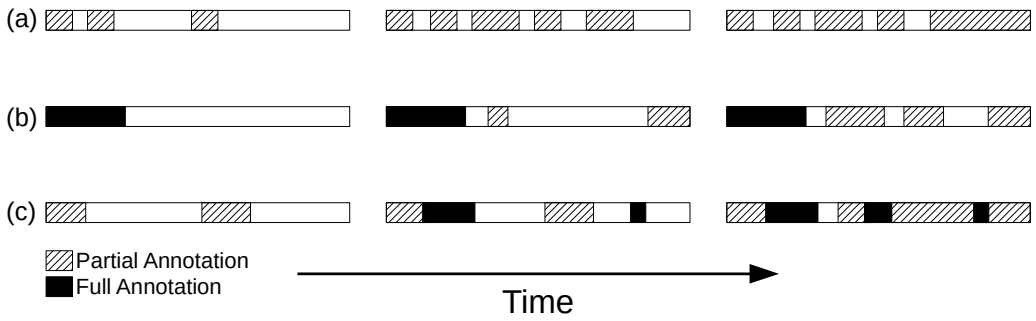



Figure 3.3: Illustration of different annotation strategies making use of partial annotations and varying annotator abilities showing potential evolutions of corpus coverage over time. Hatched areas indicate partial completion by novice annotators, while the solid areas indicate denser annotations from an expert.

(a) in Figure 3.3 shows 'worst-case' scenario where only novice annotators are available. In this case, we're forced to make use of partial annotations throughout the timeline. In contrast, timeline (b) shows a situation where a project might have the budget and availability for an expert annotator to establish a seed corpus of dense annotations that the cheaper, novice annotators can then expand on. Timeline (c) represents a realistic best-case scenario. In this situation, the hypothetical project has the ability to consult an expert throughout the project to fill in difficult sections, while making use of novice annotators and partial annotations to rapidly expand coverage of the corpus.

### 3.1.2 Simplicity

Another major benefit of using GFL is that the actual annotation symbols are simple and there are relatively few of them. The work presented in this dissertation makes use of an even further reduced set, shown previously in Table 3.1. With only a few symbols, the annotation process is easily explained to annotators and they do not require lengthy reference guides in order to look up the meaning of uncommonly used symbols. The fact that dependency arcs specified via GFL are unlabeled for syntactic relation cuts down dramatically on complexity.

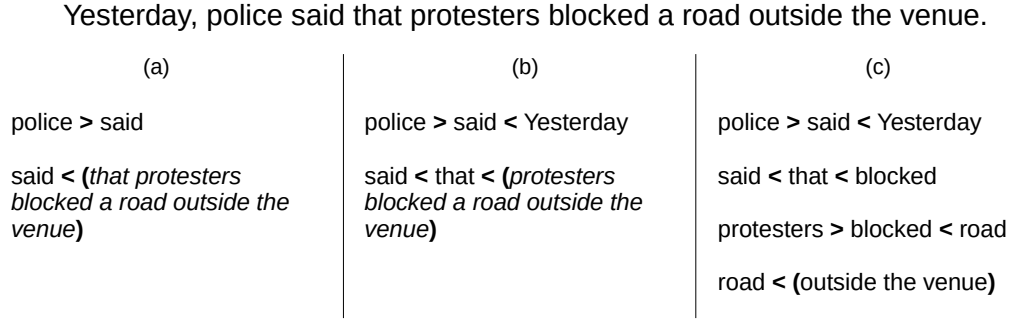



Figure 3.4: Illustration of different annotation densities for a single sentences.

The fact that GFL is relatively simple (particularly in the reduced form used here) does not mean that a lot of effort is required to produce useful annotations. When combined with the underspecification ability, GFL can rapidly produce useful annotations with minimal annotator effort or decision making. Figure 3.4 demonstrates various levels of annotation density and complexity for a single sentence. Even the lightest annotation in (a) provides valuable information concerning the structure of the sentence, namely that it contains an embedded clause, while (b) introduces additional details and (c) delves into the structure of the embedded clause itself. This property of usefulness at any complexity level is important when making use of annotators with different ability levels, otherwise either the novice annotators will be producing non-useful annotations because their simpler annotations aren't as valuable or the experience of the expert annotators will be wasted because a dense annotation is not especially more valuable than a simple annotation.

### 3.1.3 Theory-neutral

As a framework for syntactic annotation rather than a particular annotation scheme, FUDG and GFL intentionally encode very little linguistic theory. Other than the idea of dependencies and groupings of tokens–loosely constituents–there is nothing that privileges any particular theory of syntax with regards to the analysis of particular linguistic structures. This means that projects using GFL for annotation are allowed

to develop their own standards, which is of particular benefit when working with low-resource languages because often the analyses for various structures are ambiguous or changing, particularly if the language has had little descriptive linguistic work done on it.

# Chapter 4

# Low-Resource Syntactic Parsing

In this chapter, two different approaches to low-resource syntactic parsing are described: a Gibbs sampling approach used for both probabilistic context-free grammars (PCFGs) and dependencies, along with a minimum-spanning tree (MST) based approach for dependencies.

Both of these techniques are flexible in that they can optionally accept partial annotations as a form of weak instance-level supervision. The MST-based approach also uses an additional 'task-level' supervision source in the form of universal grammar rules.

This chapter primarily contains the theoretical and implementation details for these approaches, along with basic results. For a more in-depth analysis of experimental results for all of these tools, see Chapter 6.

## 4.1 Gibbs Sampling[1]

### 4.1.1 PCFGs

Despite great progress over the past two decades on parsing, relatively little work has considered the problem of creating accurate parsers for low-resource languages. Existing work in this area focuses primarily on approaches that use some form of cross-lingual bootstrapping to improve performance. For instance, Hwa et al. use a parallel Chinese/English corpus and an English dependency grammar to induce an annotated Chinese corpus in order to train a Chinese dependency grammar [Hwa et al., 2005]. Kuhn also considers the benefits of using multiple languages to induce a monolingual grammar, making use of a measure for data reliability in order to weight training data based on confidence of annotation [Kuhn, 2004b]. Bootstrapping approaches such as these achieve markedly improved results, but they are dependent on the existence of a parallel bilingual corpus. Very few such corpora are readily available, particularly for low-resource languages, and creating such corpora obviously presents a challenge for many practical applications. Kuhn shows some of the difficulty in handling low-resource languages by examining various tasks using Q'anjob'al as an example [Kuhn, 2004a]. Another approach is that of Bender et al., who take a more linguistically-motivated approach by making use of linguistic universals to seed newly developed grammars [Bender et al., 2002]. This substantially reduces the effort by making it unnecessary to learn the basic parameters of a language, but it lacks the robustness of grammars learned from data.

Recent work on Probabilistic Context-Free Grammars with latent annotations (PCFG-LA) [Matsuzaki et al., 2005, Petrov et al., 2006] have shown them to

[1] **This section is based on joint work with Liang Sun and Jason Baldridge. [Sun et al., 2014, Mielens et al., 2015]**

be effective models for syntactic parsing, especially when less training material is available [Liang et al., 2009, Shindo et al., 2012]. The coarse nonterminal symbols found in vanilla PCFGs are refined by latent variables; these latent annotations can model subtypes of grammar symbols that result in better grammars and enable better estimates of grammar productions. In this paper, we provide a Gibbs sampler for learning PCFG-LA models and show its effectiveness for parsing low-resource languages such as Malagasy and Kinyarwanda.

Previous PCFG-LA work focuses on the problem of parameter estimation, including expectation-maximization (EM) [Matsuzaki et al., 2005, Petrov et al., 2006], spectral learning [Cohen et al., 2012, Cohen et al., 2013], and variational inference [Liang et al., 2009, Wang and Blunsom, 2013]. Regardless of inference method, previous work has used the same method to parse new sentences: a Viterbi parse under a new sentence-specific PCFG obtained from an approximation of the original grammar [Matsuzaki et al., 2005]. Here, we provide an alternative approach to parsing new sentences: an extension of the Gibbs sampling algorithm of Johnson et al., which learns rule probabilities in an unsupervised PCFG. [Johnson et al., 2007]

We use a Gibbs sampler to collect sampled trees theoretically distributed from the true posterior distribution in order to parse. Priors in a Bayesian model can control the sparsity of grammars (which the inside-outside algorithm fails to do), while naturally incorporating smoothing into the model [Johnson et al., 2007, Liang et al., 2009]. We also build a Bayesian model for parsing with a treebank, and incorporate information from training data as a prior. Moreover, we extend the Gibbs sampler to learn and parse PCFGs with latent annotations. Learning the latent annotations is a compute-intensive process. We show how a small amount of training data can be used to bootstrap: after running a large number of sampling

iterations on a small set, the resulting parameters are used to seed a smaller number of iterations on the full training data. This allows us to employ more latent annotations while maintaining reasonable training times and still making full use of the available training data.

We find that our technique comes near state of the art results on large datasets, such as those for Chinese and English, and it provides excellent results on limited datasets – both artificially limited in the case of English, and naturally limited in the case of Italian, Malagasy, and Kinyarwanda. This, combined with its ability to run off-the-shelf on new languages without any supporting materials such as parallel corpora, make it a valuable technique for the parsing of low-resource languages.

**Bayesian PCFG** Our starting point is a Gibbs Sampling algorithm for vanilla PCFGs introduced by Johnson et al. for estimating rule probabilities in an unsupervised PCFG.

For a grammar $G$, each rule $r$ in the set of rules $R$ has an associated probability $\theta_r$. The probabilities for all the rules that expand the same non-terminal $A$ must sum to one: $\sum_{A \to \beta \in R} \theta_{A \to \beta} = 1$.

Given an input corpus $\boldsymbol{w}$=($\boldsymbol{w^{(1)}}, \cdots, \boldsymbol{w^{(n)}}$), we introduce a latent variable $\boldsymbol{t}$=($t^{(1)}, \cdots, t^{(n)}$) for trees generated by $G$ for each sentence. The joint posterior distribution of $\boldsymbol{t}$ and $\theta$ conditioned on $\boldsymbol{w}$ is:

$$\begin{aligned} p(\boldsymbol{t}, \theta \mid \boldsymbol{w}) &\propto p(\theta) p(\boldsymbol{w} \mid \boldsymbol{t}) p(\boldsymbol{t} \mid \theta) \\ &= p(\theta) (\prod_{i=1}^{n} p(w^{(i)} \mid t^{(i)}) p(t^{(i)} \mid \theta)) \\ &= p(\theta) (\prod_{i=1}^{n} p(w^{(i)} \mid t^{(i)}) \prod_{r \in R} \theta_r^{f_r(t^{(i)})}) \end{aligned} \tag{4.1}$$

Here $f_r(t)$ is the number of occurrences of rule $r$ in the derivation of $\boldsymbol{t}$; $p(w^{(i)} \mid t^{(i)}) = 1$ if the yield of $t^{(i)}$ is the sequence $w^{(i)}$, and 0 otherwise.

We use a Dirichlet distribution parametrized by $\alpha_A$: $Dir(\alpha_A)$ as the prior of the probability distribution for all rules expanding non-terminal $A$ ($p(\theta_A)$). The prior for all $\theta$, $p(\theta)$, is the product of all Dirichlet distributions over all non-terminals $A \in N$: $p(\theta \mid \alpha) = \prod_{A \in N} p(\theta_A \mid \alpha_A)$.

Since the Dirichlet distribution is conjugate to the Multinomial distribution, which we use to model the likelihood of trees, the conditional posterior of $\theta_A$ can be updated as follows:

$$\begin{aligned} p_G(\theta \mid \boldsymbol{t}, \alpha) &\propto p_G(\boldsymbol{t} \mid \theta) p(\theta \mid \alpha) \\ &\propto (\prod\nolimits_{r \in R} \theta_r^{f_r(\boldsymbol{t})})(\prod\nolimits_{r \in R} \theta_r^{\alpha_r - 1}) \\ &= \prod\nolimits_{r \in R} \theta_r^{f_r(\boldsymbol{t}) + \alpha_r - 1} \end{aligned} \tag{4.2}$$

which is still a Dirichlet distribution with updated parameter $f_r(\boldsymbol{t}) + \alpha_r$ for each rule $r \in R$.

**Gibbs sampler** An advantage of using Gibbs sampling for Bayesian inference, as opposed to other approximation algorithms such as Variational Bayesian inference (VB) and Collapsed Variational Bayesian inference (CVB), is that Markov Chain Monte Carlo (MCMC) algorithms are guaranteed to converge to a sample from the true posterior under appropriate conditions [Taddy, 2011]. Both VB and CVB converge to inaccurate and locally optimal solutions, like EM. In some models, CVB can achieve more accurate results due to weaker assumptions [Wang and Blunsom, 2013]. Another advantage of Gibbs sampling is that the sampler allows for parallel computation by allowing each sentence to be sampled entirely independently of the others.

After each parallel sampling stage, all model parameters are updated in a single step, and the process then repeats.

To sample the joint posterior $p(\boldsymbol{t}, \theta \mid \boldsymbol{w})$, we sample production probabilities $\theta$ and then trees $\boldsymbol{t}$ from these conditional distributions:

$$p(\boldsymbol{t} \mid \theta, \boldsymbol{w}, \alpha) = \prod_{i=1}^{n} p(t_i \mid w_i, \theta) \tag{4.3}$$

$$p(\theta \mid \boldsymbol{t}, \boldsymbol{w}, \alpha) = \prod_{A \in N} Dir(\theta_A \mid f_A(\boldsymbol{t}) + \alpha_A) \tag{4.4}$$

*Step 1: Sample Rule Probabilities.* Given trees $\boldsymbol{t}$ and prior $\alpha$, the production probabilities $\theta_A$ for each nonterminal $A \in N$ are sampled from a Dirichlet distribution with parameters $f_A(\boldsymbol{t}) + \alpha_A$. $f_A(\boldsymbol{t})$ is a vector, and each component of $f_A(\boldsymbol{t})$, is the number of occurrences of one rule expanding nonterminal $A$.

*Step 2: Sample Tree Structures.* To sample trees from $p(t_i \mid \boldsymbol{w_i}, \theta)$, we use the efficient sampling scheme used in previous work [Goodman, 1998, Finkel et al., 2006, Johnson et al., 2007]. There are two parts to this algorithm. The first constructs an inside table as in the Inside-Outside algorithm for PCFGs [Lary and Young, 1990]. 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 $\boldsymbol{\theta}$, we construct the inside table with entries $p_{A,i,k}$ for each nonterminal and each word span $w_{i,k} : 0 \leq i < k \leq l$, where $p_{A,i,k} = P_{G_A}(w_{i,k}|\theta)$ is the probability that words $i$ through $k$ were produced by the non-terminal $A$. The table is computed recursively

**Require:** $A$ is parent node of binary rule; $w_{i,k}$ is a span of words: $i+1<k$
  **function** TREESAMPLER($A, i, k$)
    **for** $i<j<k$ and pair of child nodes of $A$:$B, C$ **do**
      $P(j,B,C) = \frac{\theta_{A\to BC}\cdot p_{B,i,j}\cdot p_{C,j,k}}{\cdot} p_{A,i,k}$
    **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

by

$$p_{A,k-1,k} = \theta_{A\to w_k} \tag{4.5}$$

$$p_{A,i,k} = \sum_{A\to BC\in R} \sum_{i<j<k} \theta_{A\to BC}\cdot p_{B,i,j}\cdot p_{C,j,k} \tag{4.6}$$

for all $A, B, C \in N$ and $0 \le i < j < k \le l$.

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.

In unsupervised PCFG learning, the rule probabilities can be resampled using the sampled trees, then used to reparse the corpus, and so on.

**Experiments and Results** Our goal is to understand parsing efficacy using sampling and latent annotations for low-resource languages, so we perform experiments on five languages with varying amount of training data. We compare our results to a number of previously established baselines. First, for all languages, we use both a standard unsmoothed PCFG and the Bikel parser, trained on the training corpus. Additionally, we compare to state-of-the-art results for both English and Chinese,

which have an existing body of work in PCFGs using a Bayesian framework. For Chinese, we compare to Huang & Harper (2009), using their results that only use the Chinese Treebank (CTB). For English, we compare to Liang et al. (2009). Prior results for parsing the constituency version of the Italian data are available [Alicante et al., 2012], but as they make use of a different version of the treebank including extra sentences, and additionally use the extensive functional tags present in the corpus, we do not directly compare our results to theirs.[2]

Basic results are presented in Tables 4.1 and 4.2, which show performance when training on section 02 of the WSJ (treating English as a lower-resource language). The results show that the basic Gibbs PCFG (where K=1), with an F-score of 61.0, substantially outperforms not only an unsmoothed PCFG (the simplest baseline), but also the Bikel parser [Bikel, 2004] trained on the same amount of data. Table 4.1 also shows further large gains are obtained from using latent annotations—from 60.5 for K=1 to 78.7 for K=8.

The Gibbs PCFG also compares quite favorably to the PCFG-LA of Liang et al.—slightly better for K=1 and K=2 and slightly worse for K=4 and K=8. Table 4.2 shows that the Gibbs PCFG is able to produce results with a smaller amount of variance relative to the Berkeley Parser, even at low training sizes.

For complete results and discussion see Chapter 6.

### 4.1.2 Dependencies

While the Gibbs-PCFG parser proved itself to be a useful tool, its usefulness is limited by the nature of the data it operates on—syntactic constituencies. These annotations are expensive and time-consuming to produce, so it is unlikely that a

[2]As part of a standardized pre-processing step, we strip functional tags, which makes a direct comparison to their results inappropriate.

| System | K=1 | K=2 | K=4 | K=8 | K=16 |
|---|---|---|---|---|---|
| Unsmoothed PCFG | 40.2 | — | — | — | — |
| Bikel Parser | 57.9 | — | — | — | — |
| Liang et al. 07 | 60.5 | 71.1 | 77.2 | **79.2** | 78.2 |
| Berkeley Parser | 60.8 | **74.4** | **78.4** | 79.1 | **78.7** |
| Gibbs PCFG | **61.0** | 71.3 | 76.6 | 78.7 | 78.0 |

Table 4.1: F1 scores for small English training data experiments. 'K' is the number of latent annotations – K=1 represents a vanilla, unannotated PCFG.

low-resource project would be even using them. It is much more likely that small projects would be using dependencies due to their comparative ease of understanding and annotation. This fact is likely responsible for the proliferation of dependency corpora in recent years, while constituencies have been less influential.

Accordingly, in the interest of providing the most useful tools, we were interested in making a dependency version of the Gibbs-PCFG parser. To do this, we make use of the DMV model, a generative model for the unsupervised learning of dependency structures [Klein and Manning, 2004].

**CFG-DMV model** 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 $\omega_i$ has an corresponding part-of-speech tag $\tau_i$. We denote the set of all words as $V_\omega$ and the set of all parts-of-speech as $V_\tau$. 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 $\tau_i$ in sentence $\boldsymbol{\omega}$ is represented in a

| System | WSJ Sec. 02 | KIN | MLG |
|---|---|---|---|
| Berkeley Parser | **78.3 ± 0.93** | 60.6 ± 1.1 | 52.2 ± 2.0 |
| Gibbs PCFG | 76.7 ± 0.63 | **67.2 ± 0.92** | **57.5 ± 1.1** |

Table 4.2: F1 scores with standard deviation over ten runs of small training data, K=4.

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 $DT_L$ $DT_R$ $NN_L$ $NN_R$ $V_L$ $V_R$).

By essentially encoding dependencies into a PCFG, we are able to perform dependency parsing using the Gibbs-PCFG parser. Unfortunately, performance in this setup is rather poor. However, we found that rather than directly performing the dependency parsing, we could instead use this setup to perform 'parse imputation' on partial dependency annotations (filling in missing information to produce full annotations), and then use those completed annotations in any number of off-the-shelf dependency parsers. An in-depth examination of these experiments is found in Chapter 6.

## 4.2 Minimum Spanning Trees

Minimum Spanning Trees (MSTs) can be used to perform dependency parsing, as shown by MSTParser [McDonald et al., 2005], which achieves good cross-linguistic performance. The formulation of Grave & Elhadad (G&E) in particular has a number of nice properties that allow for both good speed and performance, along with easily modified supervision sources [Grave and Elhadad, 2015]. Although it is labeled as an unsupervised parser, the G&E-MST parser makes use of task-level supervision in the form of a set of universal dependency rules that the parser endeavors to obey (suffering a penalty for productions that do not conform to the rules). In this section, the extension of this parser to incorporate partial instance-level supervision is described, which provides greatly increased performance with minimal annotation effort. Additionally, as with the dependency formulation of the Gibbs-PCFG, the modified G&E-MST parser can be used to complete partial annotations for use in

standard dependency pipelines.

Minimally-supervised ('unsupervised') parsing solutions like the G&E-MST parser are simultaneously an attractive yet troublesome method for handling low-data scenarios. While the performance of unsupervised parsers has increased dramatically, making them a potentially viable option for users faced with limited budgets to construct labeled corpora, their performance is often outmatched by small amounts of labeled instance data. Further, recent work using linguistically informed error analysis on unsupervised combinatory categorial grammar parsing has demonstrated that entire syntactic phenomena appear to be outside the scope of existing unsupervised parsers [Bisk and Hockenmaier, 2015]. Accordingly, most recent work in this area has focused on methods of injecting various sources of annotation, whether via linguistic world-knowledge [Naseem et al., 2010, Grave and Elhadad, 2015], partial annotations [Flannery et al., 2011, Mielens et al., 2015] or cross-lingual information transfer [Naseem et al., 2012].

We present a semi-supervised parsing method that uses cheaply obtained partial annotations from non-expert annotators along with a small set of universal dependency rules to achieve performance gains in dependency parsing that would have required an infeasible amount of extra raw data to achieve using a traditional 'unsupervised' approach.

This setup exploits weak task-level supervision over our entire dataset via the universal rules of Grave & Elhadad while providing direct, instance supervision when available via the partial dependency annotations. We focus on presenting results that are as close to a realistic annotation effort as possible by not using forms of hidden supervision such as gold part-of-speech tags. We show that particularly in small data environments, these extra amounts of supervision can heavily influence

| | |
|---|---|
| Verb $\mapsto$ Verb | Noun $\mapsto$ Noun |
| Verb $\mapsto$ Noun | Noun $\mapsto$ Adj |
| Verb $\mapsto$ Pron | Noun $\mapsto$ Det |
| Verb $\mapsto$ Adv | Noun $\mapsto$ Num |
| Verb $\mapsto$ Adp | Noun $\mapsto$ Conj |
| Adj $\mapsto$ Adv | Adp $\mapsto$ Noun |

Table 4.3: Universal Dependency Rules

the outcomes of experiments.

### 4.2.1 Method

In this section we present a brief overview of the core parsing algorithm that we use in this work. For full details, see Grave & Elhadad (2015).

We begin by considering a binary vector $\mathbf{y}$ that encodes all of the dependencies in our corpus, such that $\mathbf{y}_{ijk} = 1$ if sentence $i$ has an arc with dependent $j$ and head $k$.

This representation leads to the problem formulation in Equation 4.7, where $Y$ is the convex hull of all the valid tree assignments for $\mathbf{y}$, $\mathbf{n}$ is the number of possible dependency arcs in the corpus, $\mathbf{u}$ is a penalty vector that penalizes potential dependency arcs that are not in the set of universal dependency rules (Table 4.3), and $\mathbf{w}$ is a weight vector learned during training.

$$\min_{\mathbf{y} \in Y} \min_{\mathbf{w}} \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 - \mu \mathbf{u}^T \mathbf{y} \tag{4.7}$$

This problem can be solved using the optimization algorithm described by Grave & Elhadad, reproduced here as Algorithm 1.

**Partial Dependency Features** The primary modification this work introduces is an additional penalty term into the problem formulation in Equation 4.7, where

**Algorithm 1** Optimization algorithm from Grave & Elhadad (2015)

1: **for** $r \neq 0$ **do**
  Compute the optimal $\mathbf{w}$:
  $\mathbf{w}_t = \arg\min_{\mathbf{w}} \frac{1}{2n}\|\mathbf{y}_t - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$
  Compute the gradient w.r.t. $\mathbf{y}$:
  $\mathbf{g}_t = \frac{1}{n}(\mathbf{y}_t - \mathbf{X}\mathbf{w}_t) - \mu\mathbf{u}$
  Solve the linear program:
  $\mathbf{s}_t = \min_{\mathbf{s}\in Y} \mathbf{s}^T\mathbf{g}_t$
  Take the Franke-Wolfe step:
  $\mathbf{y}_t = \gamma_t\mathbf{s}_t + (1-\gamma_t)\mathbf{y}_t$
2: **end for**

potential arcs are penalized for not agreeing with the partial annotations that the annotators have provided—for the arcs that have these annotations. This is functionally very similar to the penalty term used in the original formulation to drive adherence to the set of universal rules.

Let $\mathcal{S}$ be the set of all indicies on $\mathbf{y}$ where that head-dependent pair conforms to one of the universal rules. Then we can require that some proportion of the arcs in the corpus satisfy a rule:

$$\frac{1}{n}\sum_{i\in\mathcal{S}} y_i \geq c$$

Which is equivalent to $\mathbf{u}^T\mathbf{y} \geq c$, where:

$$u_i = \begin{cases} 1/n, & \text{if } i \in \mathcal{S}. \\ 0, & \text{otherwise.} \end{cases}$$

This is how the penalty term $\mu\mathbf{u}^T\mathbf{y}$ from Equation 4.7 is derived. Similarly, we can add another penalty term by following the same procedure, where we want a certain percentage of the arcs to conform to the trees specified by the annotators.

If we let $\mathcal{G}$ be the set of all indicies on $\mathbf{y}$ where the word pair conforms to the GFL annotations, then it is simple to construct an additional penalty term $\xi \mathbf{v}^T \mathbf{y}$.

There is a slight difference between the GFL penalty term and the universal rule penalty term. Whereas the universal rule penalty is based simply on whether the arc conforms or does not conform to the rules, the GFL annotations naturally lead to a three-way distinction: the annotation can specify that an arc *should* be present, *should not* be present, or make no commitment either way.

Accordingly, we modify $\mathcal{G}$ to be two sets, $\mathcal{G}_w$ and $\mathcal{G}_b$, where $\mathcal{G}_w$ is the set of all indicies on $\mathbf{y}$ where the word pair should have an arc, and $\mathcal{G}_b$ is the set of all indicies on $\mathbf{y}$ where the word pair should *not* have an arc. We refer to these as the whitelist and blacklist accordingly. Under this formulation, the GFL-based penalty term $\xi \mathbf{v}^T \mathbf{y}$ is now made with:

$$v_i = \begin{cases} 1/n, & \text{if } i \in \mathcal{G}_w \\ -1/n, & \text{if } i \in \mathcal{G}_b \\ 0, & \text{otherwise} \end{cases}$$

This leads to the modified objective function in Equation 4.8, which now seeks to find a solution that minimizes the number of arcs that violate both universal rules and the annotator-specified fragments.

$$\min_{\mathbf{y} \in Y} \min_{\mathbf{w}} \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 - \mu \mathbf{u}^T \mathbf{y} - \xi \mathbf{v}^T \mathbf{y} \tag{4.8}$$

Note that when there are no GFL annotations specified for the corpus (or, equivalently, for a particular sentence) the GFL penalty term goes to zero, and the objective function reverts to the original formulation.

congress has not passed < (a comprehensive plan)
passed < congress

Figure 4.1: GFL Whitelisting vs. Blacklisting

Specific arcs are added to $\mathcal{G}_w$ and $\mathcal{G}_b$ in a number of ways, based on the different types of GFL annotation.

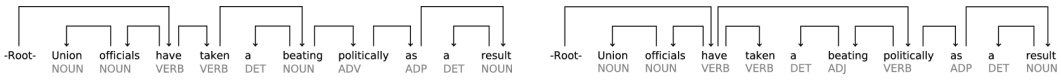
Consider the GFL annotation in Figure 4.1. Here, the annotator has specified a direct dependency with 'passed' as the head of 'congress'. This means that the arc 'passed $\leftarrow$ congress' is added to $\mathcal{G}_w$, while all other arcs of the form '$X \leftarrow$ congress' are added to $\mathcal{G}_b$ because 'congress' may only have a single head.

Brackets may also result in additions to the whitelist and blacklist. In Figure 4.1, 'a comprehensive plan' is bracketed. In this case, no arcs are able to be whitelisted, but many are able to be blacklisted. For instance, no word external to the bracket may be headed by a word in the bracket. This means arcs such as 'plan $\leftarrow$ congress' must be in $\mathcal{G}_b$.

Additionally, in this case, the entire bracket is indicated as being headed by 'passed'. While we are not able to whitelist any specific arcs with this information (because we do not know the head of the bracketed expression), we can say that no word internal to the bracket is headed by any word external to the bracket other than 'passed'. This means arcs such as 'congress $\leftarrow$ plan' must be in $\mathcal{G}_b$.

### 4.2.2 Experiments and Results

We perform a set of experiments broadly divided into two classes: Parse Imputation and Semi-Supervised Parsing. These two modes reflect the multiple potential uses for a dependency parser utilizing partial annotations, depending on where the final sentences to infer structure on come from. When the structures of sentences with ex-




(a) Gold Tags (b) Predicted Tags

Figure 4.2: Differences in parsing results due to minimal POS tagging errors.

isting partial dependencies are to be inferred, the parser is running in an imputation type setup where it must respect the human-provided dependency fragments. When the structures to be inferred are over sentences with no existing partial annotations, the parser is running in a more straight-forward semi-supervised setup.

Results using simulated data can be found in Section 4.3; the results there demonstrate the capabilities of the parser and partial annotations from a theoretical perspective. The results of running on a real-world multi-annotator corpus are presented in Chapter 6. Generally speaking, the ConvexMST Parser achieved better performance than the Gibbs Parser discussed earlier.

**POS-Tagging Impact** It was important to consider the use of imperfect POS-taggings because this entire framework is based off of the assumption that the user is working from essentially no pre-existing resources. Assuming the availability of gold-standard POS tags is antithetical to this idea, and the use of such tags is one way in which instance-level supervision can show up in otherwise unsupervised systems.

Many of the errors made by the trained taggers aren't likely to cause major problems during the task of parsing; for instance errors such as mislabeling pronouns as nouns or adverbs as adjectives are unlikely to lead to major structural issues within dependency trees. However, more unlikely errors can cause more dramatic effects as shown in Figure 4.2. Here, the phrase 'beating politically' (gold tags 'NOUN ADV') is mis-tagged as 'ADJ VERB', leading to the attachment of 'politically' to the

root word and the reorganization of a substantial chunk of the sentence.

The use of the predicted POS tags had a large impact on some of the experiments and allowed additional observations about the strengths and weaknesses of the various models, namely the robustness to noisy taggings, which is much more pressing concern when working in a low-resource setting.

## 4.3 Simulated Partial Dependencies

Although using human-annotated data is the ideal testing environment for partial annotations–and indeed for most types of experiments–for a variety of reasons including realistic rates of annotation and varying ability levels, the need for training data in a large number of languages required the use of simulation techniques due to lack of annotator availability. Additionally, using simulated data allows for better observation of the strengths and weaknesses of the different models for imputation and parsing described earlier in this chapter. By assuming that our 'annotators' are infallible and always produce annotations that are consistent with the existing gold standard for the underlying corpus, the influence of disagreements with the gold standard is eliminated and we can more accurately measure how sensitive the various methods are to missing annotations rather than conflating missing annotations and inaccurate annotations.

These simulation experiments will also provide an upper-bound on the levels of performance we can reasonably expect from experiments utilizing human data, at least for these techniques, since the impact of missing annotations is equally detrimental whether it occurs in simulated or human-sourced data.

Experiments conducted using data from human annotators are discussed in detail in Chapter 6, primarily using a large collected corpus that is described in

Chapter 5. This section will be concerned exclusively with validating the parsing models from this chapter with simulated data.

### 4.3.1 Simulation Models

Many different models for generating partial dependency information can be envisioned, although not all may be particularly well-grounded in the reality of how human annotators end up producing actual partial annotations. As such, there are certain requirements that are beneficial for the simulation models to have given that we are intending to use them as an exploratory tool to probe the limits of the parsers themselves. In particular, simulation models are evaluated by the following two criteria:

1. **Degradation Targeting**: The models must be able to produce partial annotations at various levels of degradation, in the sense of removing more or fewer head specifications, in order to provide data across a wide range of hypothetical annotation densities (head specification percentage). Additionally, the ability to specify this degradation level is highly beneficial. Because we are interested in how dense annotations should or must be in order to achieve accurate performance, this is an important factor when considering the selection of a simulation model.

2. **Adaptability to Human Models**: In addition to simply being able to reduce the number of specified heads in a gold standard sentence, it would be of obvious benefit to have the models be capable of producing degraded annotations that appear as if they could have been produced by actual human annotators in various capacities. This could include incorporating tendencies such as those that can be found in particular human annotators, such as rarely

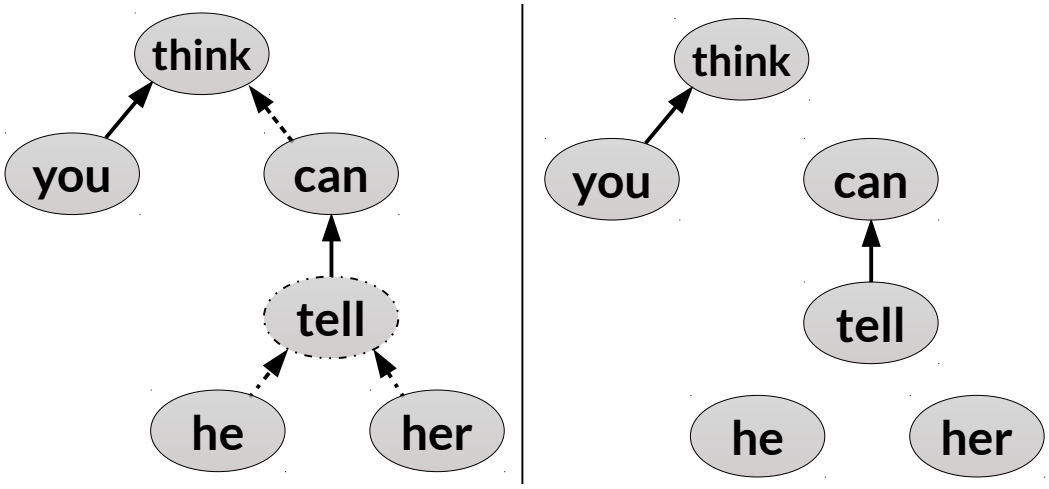



Figure 4.3: General methods for removing dependency information in simulation. 'Tell' has been selected for constituent clearing, while 'can' is being de-linked.

marking prepositional attachment or frequently specifying determiner-noun relations. Furthermore, the ideal simulation model would be able to adapt to the observed tendencies of individual annotators and essentially mimic them by modifying it's own tendencies.

All of the models described in this section are degradation models; their input is a gold standard dependency tree annotation on which they operate by removing the parent relation for some number of nodes. Partial annotation simulators that operate in a building-up fashion are also conceivable, but the benefit of working in a degradation fashion is that corpus conventions and the gold standard methods of annotating particular structures are easily maintained. This is important for this particular application, as we are using simulation primarily as a validation tool and are not interested in exploring simulated variation in structural conventions– although this an interesting area of research that could certainly be the subject of future work.

When considering ways in which the actual removal of dependency information could function, there are two obvious operations which in play a role in all of the models presented here: Node de-linkage and constituent clearing. Figure 4.3 shows an example of each of these operating on a tree. Both operations are intended to model distinct types of annotator uncertainty; node de-linkage indicates that the hypothetical annotator was unsure of where this particular node should connect, for instance a preposition that they didn't know the proper attachment for, and constituent clearing typically indicates that an annotator had specified a constituent using GFL brackets or similar but hadn't specified the internal structure.

**Randomized Removal** The most obvious method of producing partial annotations is to simply de-link a certain percentage of nodes in the gold standard tree. This has the benefit of being able to nearly perfectly match whatever desired amount of degradation we are looking for, but in practice this is not at all a good match for how real world annotators end up producing the annotations. There are a number of factors that end up making certain arcs much easier for annotators to specify a head for: distance between parent and child, the parts-of-speech involved, and overall sentence complexity all end up factoring into whether an annotator is confident enough to make an annotation for a particular head.

**Algorithm 2** Randomized Partial Annotations

```
1: procedure DegradeTreeRandom(T,r)
2:     threshold = r
3:     for each token t in T do
4:         if random(0,1) < threshold then
5:             t.parent = UNK
6:         end if
7:     end for
8:     Return T
9: end procedure
```

Although originally much of the simulation work presented in this dissertation made use of randomized knockout, the process of corpus collection discussed in Chapter 5 revealed more informative patterns that could be leveraged into more realistic simulation models. Anecdotally, different annotators of partial annotations report different overall strategies for how they approached the process of annotation. These differences are likely reflective of both differences in the background knowledge of annotators, such as their linguistic background and comfort level with the concept of constituency, as well as the relatively minimal instructions they are given with regard to how to produce annotations. The general instructions for training focused on having annotators identify the main nominal phrases and verbs, along with how they relate. However, the details of when to fill in internal structures (and how) was left in large part up to them.

**noun-seeking** One strong tendency noted among at least some–usually less experienced–annotators, is the tendency to focus on marking nominal constituents, and not much else. This motivates my first model, called NOUN-SEEKING.

NOUN-SEEKING begins by identifying all of the noun phrases in a sentence, and removes the bracketings and dependencies associated with all tokens not in a noun phrase. This results in the sentence being reduced to just fully-annotated noun phrases. From here, two parameters control both the deletion of internal noun structure and the linking of noun phrases to their noun-external parent. That is to say, the internal structure of the noun phrase is deleted at a particular rate, and the head of the noun phrases is reattached to its correct parent at another rate. The deletion of the noun structure proceeds in a top-down hierarchical fashion, which helps preserve interesting high-level structures while removing the types of frequent, low-level attachments annotators often leave out.

**Algorithm 3** NOUN-SEEKING

```
1: procedure DegradeTreeNouns(T,l,i)
2:     for each token t in T do
3:         if not inNounPhrase(t) then
4:             t.parent = UNK
5:         end if
6:     end for
7:     toProcess = {T.root}
8:     while |toProcess| > 0 do
9:         t = toProcess.head
10:        if t.pos == NOUN then
11:            if random(0,1) < i then
12:                t.deleteInternalStructure()
13:            end if
14:            if random(0,1) > l then
15:                t.parent = UNK
16:            end if
17:        end if
18:        toProcess.add(t.children)
19:    end while
20:    Return T
21: end procedure
```

Consider the dependency tree shown in Figure 4.4, consisting of a portion of the sentence, 'recently retired players from Montana organized the event'. When applying the NOUN-SEEKING pruner, this structure would be isolated from the rest of the sentence, and pruning would begin at the head of the noun phrase, 'players'. With probability $\alpha$, all of the structure under 'players' may be deleted. If the structure is not deleted, then each of the children of 'players' are considered in turn, and their sub-structures are also potentially deleted with probability $\alpha$. Finally, despite any potential deletions, 'players' is connected to 'organized' with probability $\beta$.

**Algorithm 4** CONSTITUENT-SEEKING

```
1: procedure DegradeTreeConst(T,α⃗,β)
2:     toProcess = {T.root}
3:     while |toProcess| > 0 do
4:         t = toProcess.head
5:         if random(0,1) < α_{t.POS} then
6:             t.deleteInternalStructure()
7:         end if
8:         if random(0,1) > β then
9:             t.parent = UNK
10:        end if
11:        toProcess.add(t.children)
12:    end while
13:    Return T
14: end procedure
```

**constituent-seeking** A second model, CONSTITUENT-SEEKING, generalizes the behavior of NOUN-SEEKING to all categories. Accordingly, each phrase category has a parameter controlling the rate of internal structure deletion, $\vec{\alpha} = (\alpha_{NP}\, \alpha_{VP}\, \alpha_{PP}\, \ldots)$, and there is a global parameter, $\beta$ controlling overall rate of dependency link deletion. The deletion process still happens in a top-down fashion to retain high-level information in most cases.

The primary benefit that CONSTITUENT-SEEKING offers over NOUN-SEEKING is that CONSTITUENT-SEEKING can be configured to model annotators of a much wider skill/experience level, by specifying how they perform on a wider variety of phrases, whereas NOUN-SEEKING is quite limited in this regard.

**level-pruning** The final model to consider is one motivated more by the depth of the dependencies rather than any facts about the constituency. In particular, each level of depth ($d_1$, $d_2$, ...) from the root node has a parent-deletion rate ($\alpha_1$, $\alpha_2$, ...) associated with it. The algorithm moves down the tree from the root, potentially

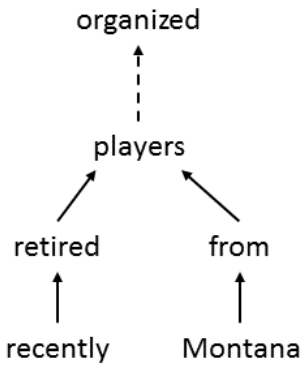



Figure 4.4: NOUN-SEEKING Pruning Example

**Algorithm 5** LEVEL-PRUNING

```
1: procedure DegradeTreeLevels(T,α⃗)
2:     toProcess = {T.root}
3:     while |toProcess| > 0 do
4:         t = toProcess.head
5:         if random(0,1) < α_{t.depth} then
6:             t.parent = UNK
7:         end if
8:         toProcess.add(t.children)
9:     end while
10:    Return T
11: end procedure
```

deleting dependency links as it goes, with the nodes deeper in the tree being more susceptible to deletion.

Currently, the parent deletion rate increases according to the simple formula $\alpha_{N+1} = \alpha_N + 0.1$, although this can be altered to produce different effects and model hypothetical annotators who may have level-based tendencies like focusing exclusively on low level attachments for instance.

This model is different than the others in the sense that it is agnostic to the category at hand, which allows the experimenter to consider a different set of effects

| | DE | EN | ES | FR | IT | SV | PT-BR | AVG |
|---|---|---|---|---|---|---|---|---|
| Total Arcs | 3390 | 6095 | 6811 | 6288 | 5327 | 3535 | 5578 | |
| 90% Arcs | 0.696 | 0.795 | 0.78 | 0.826 | 0.82 | 0.777 | 0.806 | 0.786 |
| 75% Arcs | 0.695 | 0.775 | 0.765 | 0.83 | 0.818 | 0.771 | 0.8 | 0.779 |
| 50% Arcs | 0.699 | 0.775 | 0.758 | 0.816 | 0.818 | 0.771 | 0.796 | 0.776 |
| 40% Arcs | 0.696 | 0.759 | 0.73 | 0.791 | 0.788 | 0.762 | 0.764 | 0.756 |
| 30% Arcs | 0.61 | 0.728 | 0.7 | 0.757 | 0.774 | 0.769 | 0.724 | 0.723 |

Table 4.4: Simulated partial training data results.

than seen with the other models.

### 4.3.2 Recovery of Degraded Annotations

Simulated partial training data at multiple levels of arc retention were constructed for a variety of languages in the Universal Dependencies (UD) corpora using the techniques described above. This training data was then used to train ConvexMST models (see 4) that were evaluated against the standard UD test sets. Results are shown in Figure 4.5 and Table 4.4. As can be clearly seen in the figure, the ConvexMST models are largely able to cope with the removal of arcs up to around 50% or so. However, once the retained arc percentage drops past that there is a sharp fall off in most of the languages.

It is likely that as retention rates fall, ConvexMST is benefiting greatly from the additional task-level supervision it receives in the form of universal dependency grammar rules. The occasional missing arc is easily recovered by consulting these rules, as they cover the most common types of dependencies, however once the missing sections become larger the recovery is not as straightforward anymore and errors with respect to gold are introduced.

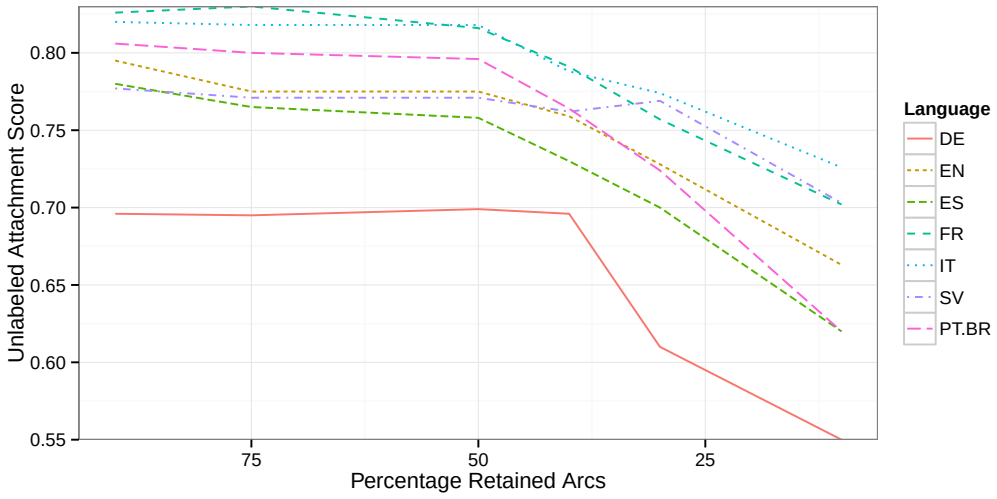



Figure 4.5: UAS with simulated partial training data for multiple languages.

### 4.3.3 Fixed Annotation Budget

In another simulation experiment, the total number of annotated arcs was held constant, while the number of sentences and their completion rates were varied. For instance, one training set could consist of 100 fully annotated sentences containing 2000 arcs, and another could consist of 200 sentences with 50% completion for a total of 2000 annotated arcs. Figure 4.6 shows the results of this experiment on Spanish data for completion rates of 100%, 70%, 50%, and 30%. In this figure, cost is calculated as simply the number of arcs specified in the training corpus.

At any given fixed annotation cost, each completion strategy yields relatively similar results within a few points, although an increased completion does appear to be associated with increased performance. This effect is most apparent at smaller training sizes. While this certainly could be considered a point in favor of using complete annotations over partial annotations when possible, Figure 4.6 should be read as being positive to the idea of partial annotations overall when the real world

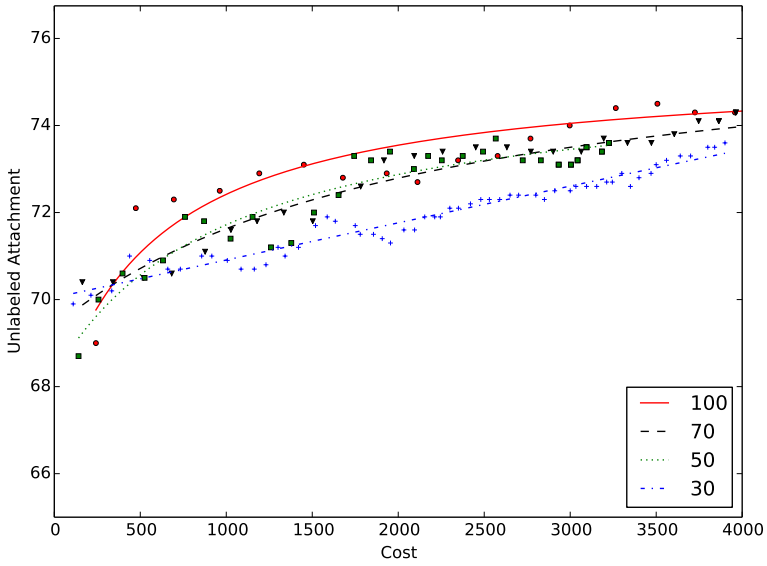



Figure 4.6: UAS at fixed annotation arc costs for varying completion rates.

costs and feasibility of producing a particular number of full vs. partial annotations are considered. The fact that a fixed budget of arc annotations leads to similar performance no matter how densely annotated the individual sentences were means that annotation projects have more options when it comes to how to obtain a desired performance level.

A more realistic measure of annotation cost would attempt to factor in the real world costs as much as possible, since the total cost of obtaining a single arc in a scenario where annotators are specifying all of the arcs in a given sentence is greater than the cost of obtaining a single arc where annotators are only specifying a few arcs per sentence. This difference in cost is made up of both a time factor, because when annotators must specify all arcs they are forced to make many more decisions that slow down annotation, and a dollar value factor, because annotators capable of producing sentences with 100% specification are typically more expen-

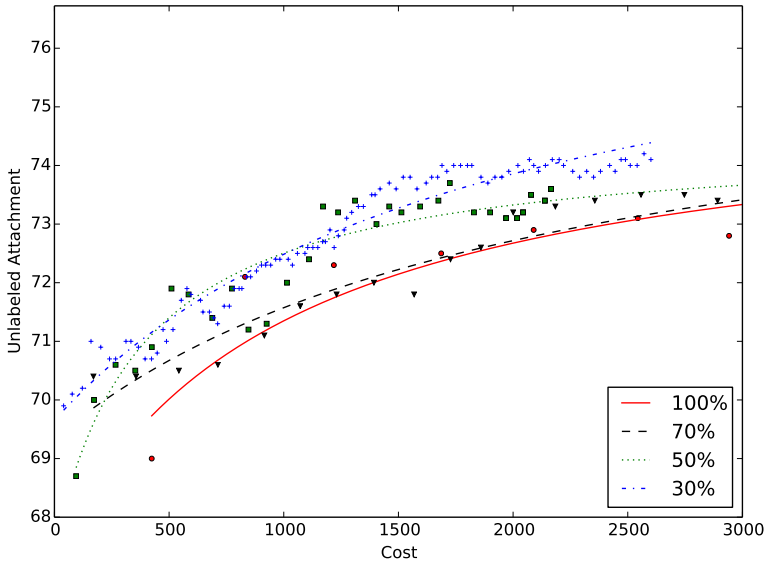



Figure 4.7: UAS at fixed annotation costs for varying completion rates, with variable costs per arc according to completion rate.

sive to hire. Previous work on measuring total annotation cost has used metrics such as the number of discriminant decisions that go into selecting a correct parse tree, and completion percentage can be seen as a similar proxy for annotator effort [Osborne and Baldridge, 2004].

For now, the total annotation cost will be not simply the number of arcs annotated but rather the total number of arcs weighted by the completion percentage, on a variable scale of cost. This modification is intended to reflect the fact that the first 10% of annotations in a sentence are cheaper to obtain than the last 10% of annotations. The cost of an arc in, e.g., the third 10% block of the annotations is $1.1^3$. This means that an arc in the final 10% costs $1.1^{10} = 2.6$, which is roughly two and a half times harder than an arc in the first 10%. This cost is likely a conservative estimate of the difficulty of complete annotations.

Under this measure of cost, Figure 4.6 is transformed into Figure 4.7, in

which the performance curves of the lower completion rates end up surpassing the fully-specified curve because the cheaper per arc cost at the low completion rates effectively squashes those curves into the cheaper total cost area of the figure. This result is potentially even more impressive given that the cost function is most likely underestimating the total cost of the 100% and 70% conditions.

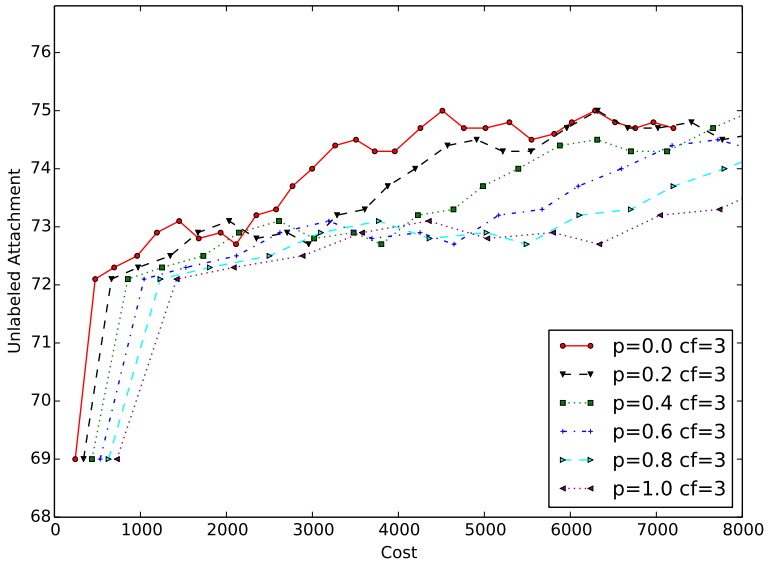
We can also envision a different model of annotation cost that focuses on the costs of obtaining full annotations rather than the costs of obtaining any given annotation density. In particular, we can assume that a certain percentage of arcs will come rather cheaply, while the remainder can be obtained at an increased cost. This is useful when thinking about partial annotations with multiple annotators, because the use of expert annotators in place of novices is nicely captured by the model, and we can then simulate use cases such as those outlined in Figure 3.3 of Chapter 3. Additionally, this cost model is similar to the previously discussed model, except with a clear cut-off between expensive and cheap annotations rather than the sliding cost previously used.

Now the calculation of the cost for a particular annotation is as follows:

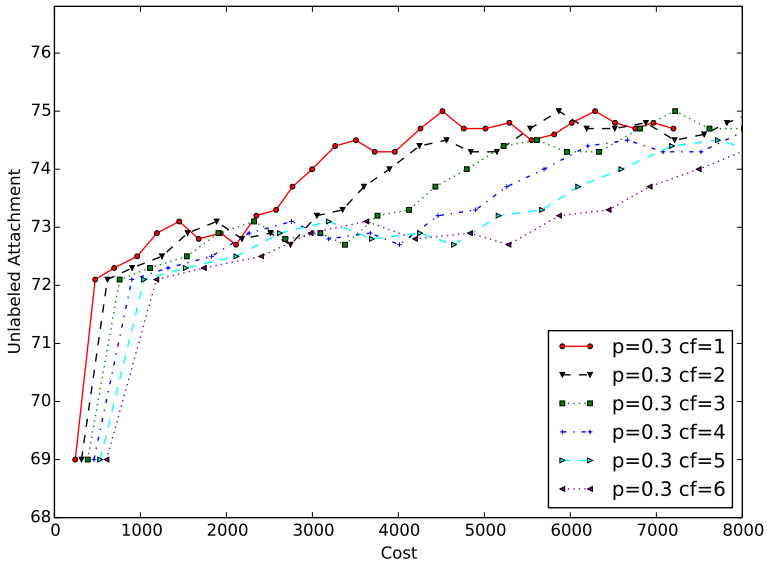
$$cost(A, \alpha, \beta) = \alpha\beta \cdot |A| + (1 - \alpha) \cdot |A|$$

Where '$A$' is the annotation, $\alpha$ is the percentage of difficult arcs, and $\beta$ is the cost factor associated with annotating difficult arcs. Simple arcs are assigned a fixed cost of 1.

The results of varying both $\alpha$ and $\beta$ individually are shown in Figure 4.8. Figure 4.8a shows cost curves over the full range of $\alpha$ (percentage of difficult arcs) values for a fixed $\beta$ (cost factor) value of 3, while Figure 4.8b shows cost curves over a reasonable range of $\beta$ values for a fixed $\alpha$ value of 0.3. In both figures, fully




(a) Variable Difficulty Percentage




(b) Variable Cost Factor

Figure 4.8: Cost curves under different cost modelling assumptions. In the figures, 'cf' is the cost factor, and 'p' is the percentage of difficult arcs.

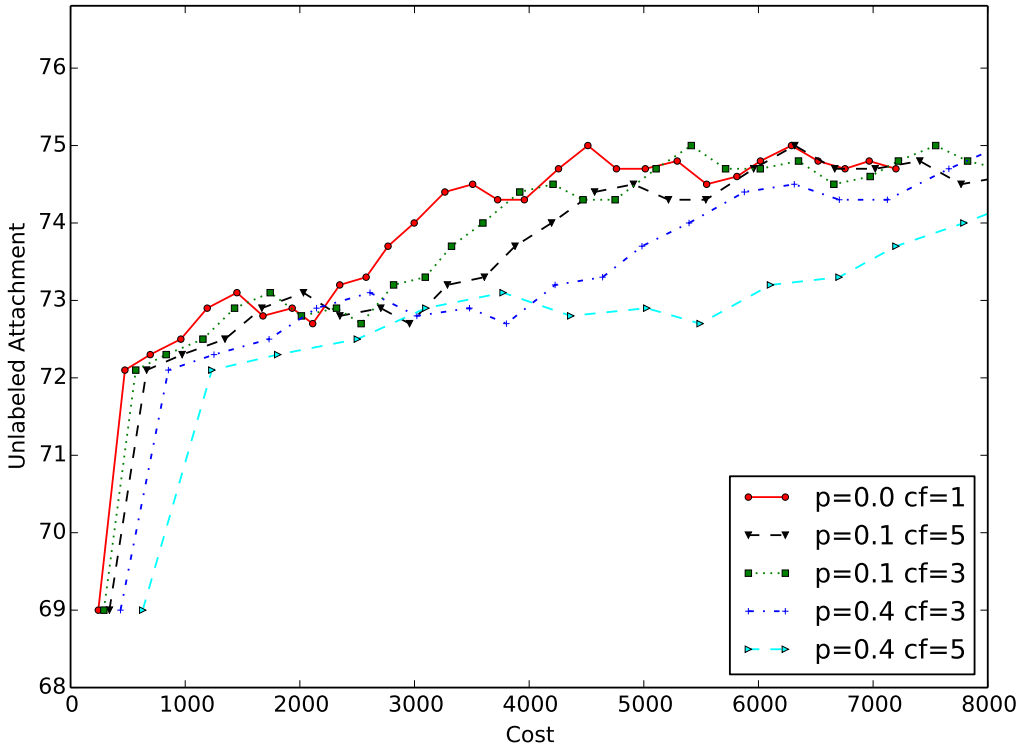



Figure 4.9: Cost curve examples for variation of both factors simultaneously.

specified sentences are being used.

These two figures demonstrate that increasing performance per unit cost under this model can be effectively achieved by either reducing the cost of the annotators or minimizing the use of expert annotators.

A few reasonable combinations of both of these factors are illustrated in Figure 4.9. Once again, the annotations being added here are fully specified rather than partial. The most salient part of this figure is the fact that over reasonable values for both the 'difficulty percentage' and the cost factor for skilled annotators, the scenarios with lower difficulty percentages win out easily. This indicates that there is much potential total cost saving to be had in opening the annotation process to as

wide a range of annotators as possible. In the extreme case of a crowdsourced annotator compared to an experienced field linguist, the cost factor may be even higher than the values presented here, further exaggerating the benefit; this is further illustration of the potential for gains from techniques that create complex annotations from a set of simpler ones obtained from cheaper annotators.

If a single parameterization of the new cost model is chosen, the behavior of partial annotations under this cost model can be considered in addition to the fully specified annotations. Setting the parameters to $\alpha = 0.4$ and $\beta = 3$ yields a very reasonable model with an 60/40 split for cheap and expensive annotations. An important note is that when using partial annotations, it is assumed that all cheap annotations are obtained first. This means that if a sentence was completed to 75%, the annotations for that sentence cover all the cheap annotations (60%), and a portion of expensive annotations (15% out of a possible 40%). While this is obviously a simplifying assumption, in our corpus development the explicit focus and instructions aimed at speed of production at the expense of completeness helps facilitate this 'cheap first' strategy and make it a more reasonable modeling assumption.

Figure 4.10 shows how Figure 4.6 is re-scaled by the use of this new cost model. For this parameterization, the 50% level ends up performing the best; the 70% and 100% level suffer too much from the added costs on expensive annotation, and the 30% level doesn't have the performance to take advantage of lower costs. In other words, a compromise between annotation coverage and costs yielded the best overall performance.

This performance is more in line with what might be expected of real world results than the original cost model described above, which rewarded partial anno-

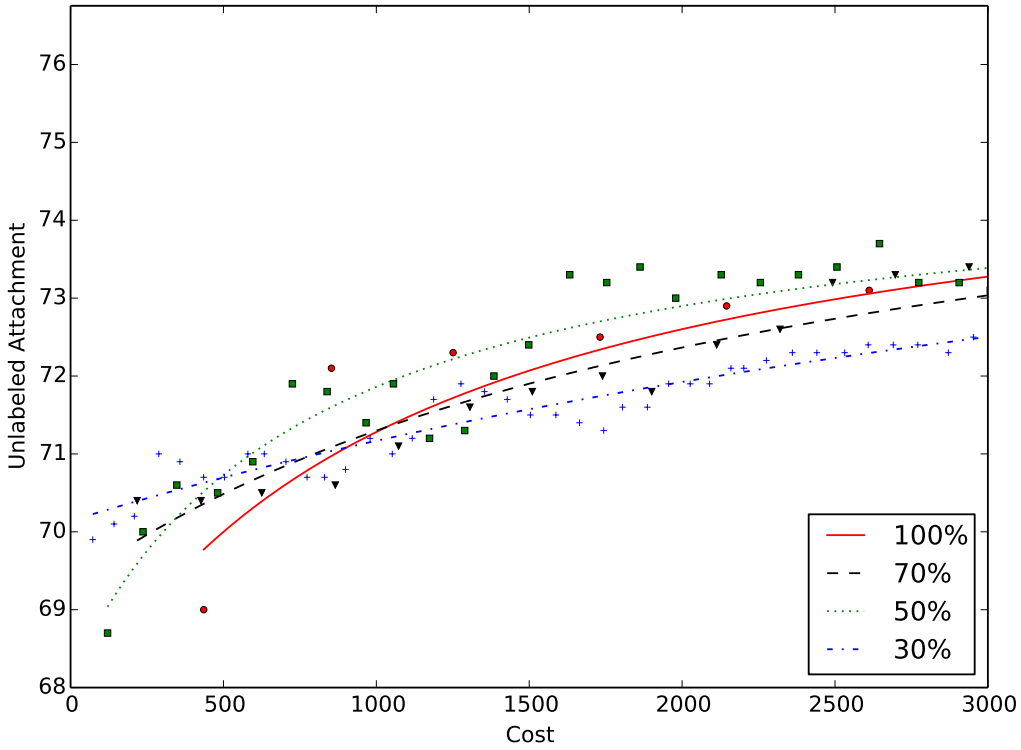



Figure 4.10: Total cost curves for partial annotations under the parameterized cost model. ($\alpha = 0.4$ and $\beta = 3$)

tations to a much greater degree, essentially making the minimization of coverage its number one priority. That previous model found the 30% level to be the best overall at a particular total cost, but empirically these is a rather low percentage of arcs, and would probably not be the most efficient under real world conditions.

Taken together, all of these simulation results are strong indicators that the ConvexMST method itself is capable of handling partial annotations at relatively low densities, at least given training data that agrees with the standards of the test data. In particular, it is effective at both recovering the missing dependencies along with parsing using the partial features as the training data. Additionally, under some

assumptions regarding the total cost of annotation, the simulation results indicate that there are economic benefits to be had from working with cheaper annotators when possible–especially if an imputation strategy can efficiently bump completion percentage for the novice annotators.

# Chapter 5

# Partial Dependency Corpus Collection

In order to effectively study small corpora, we must first create one. Although there are simple ways to achieve this, such as scaling down an existing corpus, the resulting corpora are not great proxies for a real-life small corpus that was collected under less than ideal circumstances. Instead, for the purposes of this dissertation, I will create a corpus to serve as the primary data source for my investigations into the properties of small corpora and the annotators associated with their production.

Although very small corpora of partially annotated dependencies have been collected as a data source for prior work in English, Chinese, Portuguese, and Kinyarwanda (see Chapter 4), an investigation into the properties of partial annotations must collect those annotations from a more diverse set of annotators in order to consider the differences between them, and must maintain data on the various factors about the annotators themselves.

In order to do this, I perform a series of controlled annotation experiments of

partial Spanish dependencies using annotators that have been selected for different levels of experience in both Spanish and linguistic annotation in general.

For the language level variable, participants are grouped into three categories: novice, experienced (non-native speaker), and native speakers. The distinction between novice and experienced non-native speakers is made on the basis of number of years of formal language study or residence in a primarily Spanish speaking environment, with experienced being greater than three years of study. Linguistic annotation background is represented by a binary variable simply indicating whether the participant has any prior experience in creating syntactic annotations.

As the major purpose of this corpus is to consider variation between annotators working in a minimally supervised environment, there is no attempt made to resolve dependencies to any sort of ground truth based on agreement, nor do the annotators work collaboratively at any point. Experimental sessions consist of a single participant and the experimenter, using annotation software previously developed. The annotation software is custom-built in such a way as to facilitate the annotation process specifically for this task, as described later in the chapter.

The participants provide syntactic dependency annotations in the Graph Fragment Language (GFL) framework described in Chapter 3. GFL was designed specifically to be flexible in the requirements it places on annotators, which makes it ideal for training new annotators and providing partial annotations. Another important factor in the selection of GFL is that prior work has already developed techniques that can make use of the GFL annotations.

The participant was given a large set of sentences from a standardized corpus to annotate, and worked through them at their own pace for two hours, with brief breaks every half hour. Three of these sessions are conducted per participant in

order to collect the desired six hours of annotations.

## 5.1 Corpus Development Background

The corpus discussed here is substantially different from the majority of the available corpora, but to contextualize its creation I will briefly discuss a few important corpora from the literature that contributed either novel construction techniques or took more radical approaches from a linguistic perspective.

The first electronic corpora began appearing in the 1960's, with the Brown Corpus being the first to gain much popularity. The earliest corpora did not contain syntactic structure annotations, but were instead simply collections of texts, some with part-of-speech tags. Syntactically annotated corpora began appearing in the mid 80's, when Sampson and others started updating the existing text corpora with a layer of constituency information. The Lancaster-Oslo-Bergen Corpus (LOB) was the first to be updated in this way. The development of large-scale annotated corpora in the 80's allowed for relatively simple statistical analysis techniques to be used for the first time - leading to a huge paradigm shift in the way that corpora were used.

The Penn Treebank (PTB) in particular represented a significant leap forward in corpus construction techniques, especially in terms of overall size. The PTB was larger than any other syntactically annotated corpus at the time of its release in 1993, due primarily to the nature of the annotation process. The PTB was the first large corpus to use the strategy of allowing an existing parser to parse raw sentences prior to annotation by humans; this initial parse was then corrected by the annotator - saving massive amounts of time over having the same annotator specify the entire parse tree. This corpus development process very quickly became a standard, despite the fact that it required an existing parser in order to function.

A great comparison of corpus construction techniques with the PTB is the SUSANNE Corpus. Developed nearly simultaneously (very early 90's), the two corpora took nearly opposite approaches to the construction method. As opposed to the parse correction method of the Penn Treebank, the SUSANNE corpus developers created a very specific set of guidelines, and then had annotators specify the entire sentence according to this scheme. The stated goal of the SUSANNE scheme was to:

> [...] provide a method of representing all aspects of English grammar which are sufficiently definite to be susceptible of formal annotation, with the categories and the boundaries between categories specified in sufficient detail that, ideally, two analysts independently annotating the same text and referring to the same scheme must produce the same structural analysis.

In other words, the goal of the SUSANNE scheme was to encode a particular theoretical perspective for annotators to follow, whereas the PTB aimed to be comparatively theory neutral. The corpus being constructed here takes a more extreme approach to theory neutrality by providing only minimal training to annotators and allowing them to essentially make up their own standards.

The explosion of interest in annotated corpora following the popularity and success of the PTB resulted in a wide variety of corpora being developed in the late 1990's and early 2000's. These quickly began expanding the number of languages and formalisms represented in corpus form. The TUT Treebank (2000) was both an early Italian corpus and an early native dependency corpus, for instance. Most of the well known, standardized corpora have their roots in this time period; The TIGER Treebank (German, 2002), the Alpino Dependency Corpus (Dutch, 2002),

Projecto Floresta Sint(c)tica (Portuguese, 2002), and the Prague Dependency Treebank (Czech, 2003), among many others—in my survey of 87 corpora, 31 were released in the first half of the 2000's. However, the vast majority of these 87 corpora (75%) are of Indo-European languages, there simply isn't much diversity with respect to language families.

Within a few years, some amount of work had been put into treebanks for smaller, minority languages like Catalan (Cat3LB; 2004), with additional work being done on the machinery and techniques surrounding the construction of these corpora. For instance, LREC 2004 ran a workshop on 'first steps in language documentation for minority languages', where many of the papers focused on either the use of linguistic universals (Bender et al., 2004, among others) or on porting existing tools to new languages or somehow leveraging parallel corpora in other languages. As a result, most of these corpus construction techniques ended up minimizing the amount of work to be to done in the actual target language (a reasonable idea given the difficulty of finding speakers for particular languages). From the mid 2000's on, the focus on multilingual comparable and parallel corpora continued, likely driven by the CoNLL-X shared task on multilingual dependency parsing (2006) popularizing the task and creating a ready set of data. However, the low-resource corpora in our research come from a slightly different scenario; for the purposes of this research I am assuming the availability of at least one speaker/annotator, which opens up the possibilities for direct supervision and saves us from having to lean too heavily on external resources.

## 5.2 Purpose

The purpose of constructing a corpus in this manner is clearly not to provide a perfect, 'gold standard' analysis for the complete set of the raw sentences available to the annotators. Given the duration of the project, the number of annotators, and their proficiency in both linguistics and Spanish this would be an entirely unrealistic goal.

In fact, even for the sentences that do obtain complete annotations, the resultant trees output from the corpus are likely to be full of inconsistencies, misapplied conventions, and other types of annotator error. The trees are not cross-checked by other annotators, and no automated cleanup is done.

The existing literature from prior annotation projects has provided well-developed pipelines for producing gold standard data that work well with the time, funds, and linguistic resources available to them. However, obtaining the messy corpus described above is in fact the goal of this project. While there are innumerable benefits to having large collections of clean, well-annotated natural language data around, much less attention has been paid to studying the earliest stages of these collections. Some messy collections do end up becoming standardized, cleaned, corpora – but the vast majority do not, at least not anytime soon.

The reasons that a small, messy corpus may exist in that state for an extended period of time are varied. For instance, perhaps the corpus was being specifically constructed to help facilitate domain adaptation of a larger model, and only needed to be a few hundred sentences. Or, the small collection could have been part of a language documentation effort that has temporarily lost the ability to consult with speakers or pay for annotators capable of correcting the corpus. In these scenarios, small, messy corpora may end up being at least a temporary end-product

that is pressed into use.

A major impediment to studying these small corpora is that they are often not publicly available because the authors never intended them to be polished, publishable artifacts or they think they have no value due to the fact that many of the annotations are not particularly accurate.

By constructing a corpus that is intentionally (and realistically) messy, I am providing a resource that will intentionally remain frozen in this early stage of development and provide a test-bed for research on small corpora that does not rely on artificial means to simulate its properties. This corpus could be used to answer questions related to the required language background of annotators; for instance, are native speakers more reliable on particular facets of annotation? Or it may provide insight into the partial annotation process; for instance, how large must a developing corpus be before annotators can begin to hand-wave over particularly common constructions? These are some of the questions I hope to address using the collected data.

## 5.3 Annotators

There are 12 total annotators that produced annotations for this corpus. In order to get an adequate representation for various levels of Spanish-speaking experience, annotators were recruited based primarily on their background in Spanish. The breakdown of experience levels for the annotators is shown in Table 5.1

As the table shows, there is an equal breakdown by background in Spanish, with four annotators in each category, but the breakdown by annotation experience level is slightly unbalanced towards the unexperienced. However, even (some) annotators who fell into the 'no previous annotation experience' category reported and

| | Prior Experience | No Prior Experience |
|---|---|---|
| Beginner | 2 | 2 |
| Experienced | 1 | 3 |
| Native | 1 | 3 |

Table 5.1: Number of annotators by experience level in Spanish (rows) and prior annotation projects (columns).

exhibited confidence with the relatively simple annotation scheme after just a small amount of training.

The annotators were recruited from undergraduate and graduate students in the Linguistics and Spanish departments at the University of Texas at Austin. Accordingly, all the annotators were already familiar with the concept of parts-of-speech; this was the only prerequisite for understanding the instructions of the task, which did not assume any formal knowledge of dependencies, phrase-structure, or constituency.

### 5.3.1 Instructions to Annotators

Annotators underwent a roughly thirty minute training period at the beginning of their first annotation session. The purpose of this training session was to guide the annotators to produce the types of annotations that were of the highest value possible. In particular, the primary goal presented to the annotators was to try to annotate as many sentences as possible rather than focus on completely perfecting a small number of annotations. This instruction was intended to prevent annotators from focusing on a single sentence, and to convey that the goal of the task was not to produce perfect, complete annotations, but rather to give the general, 'big-picture' information for a larger number of sentences.

In this setup, the 'big-picture' information consisted of the identification of

constituents (primarily noun phrases) and the specification of their dependency relations. Prior work has identified this basic strategy as an effective one as it allows for rapid identification and specification of high-level structures in the sentence. Accurate annotation of high-level sentence structures has been shown to be particularly beneficial for the task of syntactic parsing (Hwa, 1999), so the basic annotation procedure taught to the annotators attempted to highlight this information.

The annotation procedure, presented during the training period, is listed below. An example of an annotation progressing over the four main stages of the procedure is shown in Figure 5.1.

1. **'Main Word' Identification**: For most sentences in the corpus, the identification of the root was relatively straight-forward. Annotators were told that they should try to identify the most important, 'main word' in the sentences, and that this word would very likely be a verb. The existence of multiple 'main-words' was allowed, but annotators were told that finding one was the goal.

2. **High-Level Constituents**: Annotators were given an informal definition of a constituent–'a series of words that make a bigger unit, or block'– and provided with several examples in their native language, either Spanish or English. Annotators were told that this step was important in order to create bigger blocks to work with, rather than using individual words. The instructions were to focus on bigger blocks first, and smaller blocks later. The possibility for nested constituencies was also explained with examples.

   Examples focused mostly on high-level noun phrases and clauses, but the possibility of grouping preposition and adjective phrases was also demonstrated.

3. **Main Dependencies**: Once the annotators had created a set of blocks, they were told it was additionally helpful to know how those blocks related to each other as well as to the important words (roots and main verbs) they had already identified. The idea of dependencies was explained by saying that certain words or blocks needed extra information to be completed. For

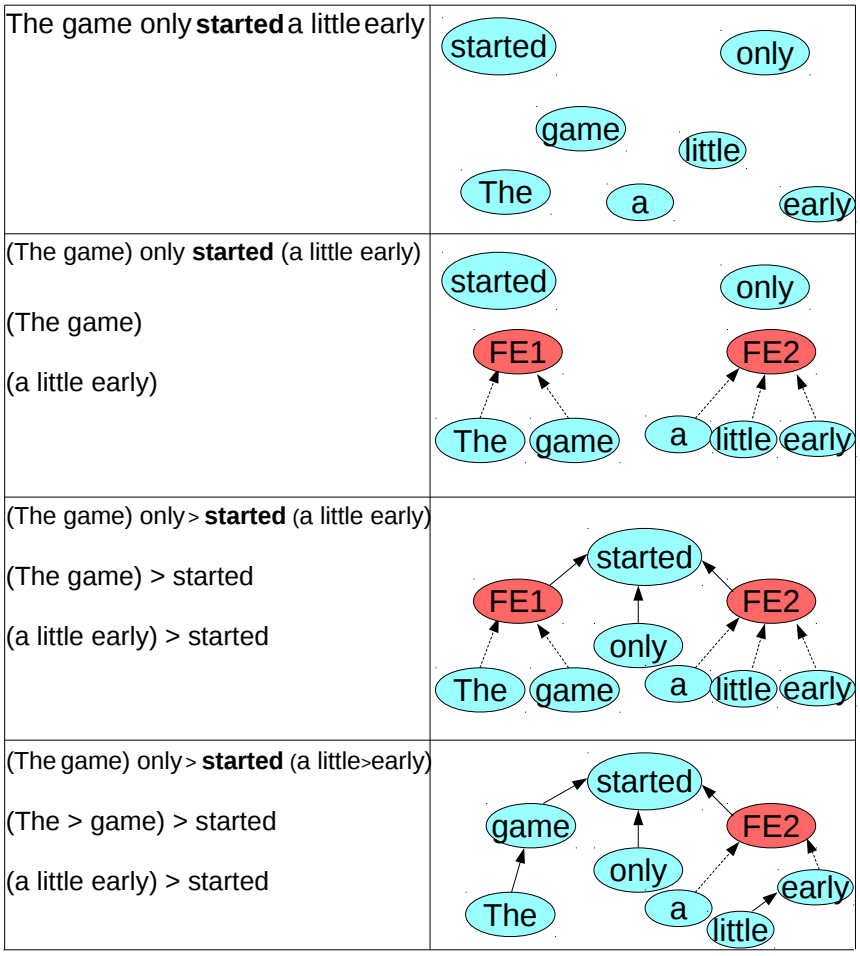



Figure 5.1: Evolution of a GFL annotation over the four main stages of the annotation procedure.

instance, if they had labeled 'jumped' as the main word in a sentence, they probably want a block that describes who or what jumped.

The head vs. tail distinction was explained as one of 'importance', with the head being more important to the overall meaning of the sentence than the tail.

4. **Extra Information**: Once the annotator had completed the above steps, they were told that if they felt confident labeling any smaller units they could do so, but that they shouldn't take too much time to do so.

This procedure was presented in a way that could be followed roughly in stages,

| Section | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | com. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Avg. Length | 29.1 | 30.2 | 28.5 | 29.2 | 30.1 | 30.6 | 28.6 | 29.8 | 30.4 | 29.7 | 30.8 | 28.9 | 29.0 |
| % <10 | 8.4 | 9.2 | 8.1 | 7.8 | 8.0 | 7.3 | 8.5 | 8.8 | 7.6 | 7.7 | 8.1 | 7.8 | 7.5 |
| % >40 | 20.6 | 23.5 | 22.6 | 25.4 | 24.3 | 25.1 | 24.5 | 23.9 | 26.0 | 24.7 | 25.6 | 24.4 | 27.1 |

Table 5.2: AnCora Annotator Split Statistics

although no formal constraint was made on this–annotators were free to label constituents after beginning to specify dependencies, for instance. Additionally, the annotation software aided adherence to the procedure to some degree. In particular, when annotators specified a constituent, it was pulled out from the main text block and was easily available for reference during the dependency annotation phase.

In practice, most of the annotators reported that this procedure was relatively simple to understand and implement. The most common questions and problems that arose were concerning the direction of dependency relations using the angle bracket notation of GFL. To help minimize confusion annotators were provided with a minimal set of guidelines that stated common rules such as 'Nouns tend to point at the verb they are associated with', 'Determiners almost always point at Nouns', and so on.

## 5.4 Data

The sentences to be annotated are drawn from the AnCora-ES Spanish corpus, which is a collection of roughly 500K tokens of primarily newswire texts[Taulé et al., 2008].

The data is divided into a 13-way partition of the AnCora corpus, one partition for each individual annotator and a single common partition that was annotated by all annotators. Each annotator worked on their individual section for two of the three two-hour sessions, and the common section for the third session. The decision to have a single common section was made with the desire to have as many

| | |
|---|---|
| Sentences Annotated | 2162 |
| Tokens Annotated | 55492 |
| Avg. Sentences per Hour | 36.03 |

Table 5.3: Basic Corpus Statistics

annotators annotate a particular set of sentences as possible, to consider the effect of annotator knowledge (both language-specific and computational) on a common set of sentences, as well as to allow for inter-annotator agreement measures to be calculated.

Additionally, each section has similar basic statistics, shown in Table 5.2, to provide a similar experience for every annotator and to prevent the annotations from specific annotators being biased towards particular sentence lengths—as has been a problem for us in the past, and has previously been shown to impact parser performance.

The total number of annotations obtained can be seen in Table 5.3. Overall, the results of the annotation met expectations in terms of average number of sentences per hour, and the rate of approximately 750 tokens per hour significantly exceeds previous annotation projects using different construction techniques. For instance, the original GFL paper conducted experiments that resulted in a rate of 430 tokens/hr for English data by native English speakers [Schneider et al., 2013]. The experiments presented here are different in that not all the speakers were native Spanish speakers–a factor which should slow annotators down–but we also have an explicit focus on speed, with annotation instructions designed to facilitate that, and are utilizing a subset of the GFL markup symbols. In other work using GFL on English, Chinese, Kinyarwanda, and Portuguese, similar rates of 400-500 tokens/hr were found, this time with non-native speakers although each annotator was relatively experienced in the GFL standard [Mielens et al., 2015].

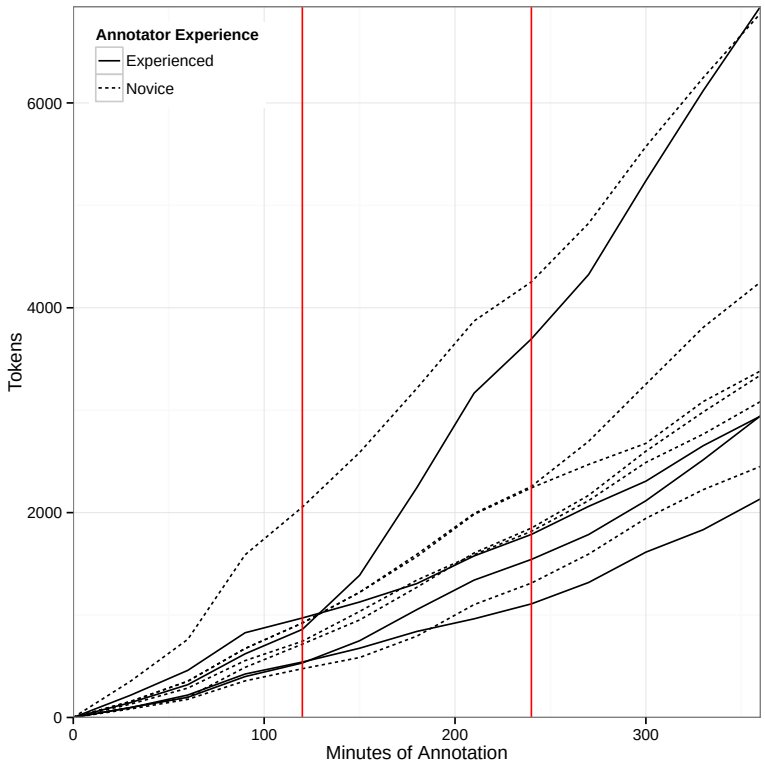



Figure 5.2: Annotated tokens over time by annotator. Vertical lines indicate session breaks.

Non GFL-based annotation rates varied significantly based on the technique used to construct them. The Penn Treebank and Chinese Treebank both made use of correction-based annotation in which an existing parser produced an analysis for a sentence and annotators were then asked to verify and correct it if necessary. The Penn Treebank reported rates of 750-1000 tokens/hr using this method (after a four month training period), and the Chinese Treebank reported rates of 300-400 tokens/hr. The large difference in these results may be related to the accuracy of the underlying parser and the amount of corrections each team needed to make on average.

The Ancient Greek Dependency Treebank (AGDT) is perhaps the most relevant example from the literature because they did not employ an existing dependency parser for initial guesses since there was no existing parser, and for obvious reasons no annotator was a native speaker [Bamman and Crane, 2011]. They report rates of 97-211 tokens/hr for individual annotators.

Figure 5.2 shows the progression of token annotation over time for all annotators. Note that annotator speed generally increased over time, with the first session being the slowest, although there was an effect of annotator fatigue where the final 30 minutes of each of session tended to be slower than the preceding blocks of that session.

## 5.5 Annotator Modeling

### 5.5.1 Annotator Agreement

Inter-annotator agreement has a number of interesting interpretations and consequences in the context of partial annotations, and accordingly I have conducted multiple analyses of this agreement in the corpus. One basic question that must be resolved is whether to focus attention on the annotations that have been provided, or focus on the potential agreement of the hypothetical structures that we might infer from the provided annotations. This is a seemingly small distinction that has important consequences for the measured agreement numbers.

Another point to consider is the utility of inter-annotator agreement in a partial annotation environment, and this is directly impacted by our choice of how we measure that agreement. In a standard annotation project, inter-annotator agreement is primarily used to provide some context for the difficulty of the annotation task, along with a measure of how 'clean' we might expect the data to be. In a

| Annotator | Commitment | Promiscuity | Gold Prec. | Gold Recall | Gold F1 | Spanish | Experience |
|---|---|---|---|---|---|---|---|
| 1 | .774 | 737 | .851 | .254 | .392 | B | Yes |
| 10 | .317 | 1296 | .161 | .030 | .051 | B | No |
| 11 | .778 | 758 | .416 | .613 | .425 | B | No |
| 12 | .422 | 798 | .648 | .300 | .410 | B | Yes |
| Avg-B | .572 | 897 | .513 | .299 | .319 | | |
| 2 | .541 | 1304 | .673 | .312 | .426 | E | No |
| 3 | .622 | 27 | .520 | .619 | .565 | E | Yes |
| 5 | .457 | 3283 | .462 | .152 | .229 | E | No |
| 8 | .740 | 1004 | .511 | .695 | .588 | E | No |
| Avg-E | .59 | 1404 | .542 | .442 | .451 | | |
| 4 | .597 | 1361 | .750 | .040 | .077 | N | No |
| 6 | .686 | 40 | .723 | .400 | .515 | N | Yes |
| 7 | .783 | 1 | .823 | .172 | .284 | N | No |
| 9 | .341 | 31 | .730 | .034 | .066 | N | No |
| Avg-N | .602 | 358 | .757 | .162 | .236 | | |

Table 5.4: Coverage statistics for partial dependency corpus collection. 'Spanish' indicates an annotator's background in Spanish: B=beginner, E=experienced, N=native speaker. 'Experience' indicates whether the annotator had prior experience doing any sort of syntactic annotation work.

partial annotation setup, the interpretation may be slightly different; under some metrics, annotators that have annotated entirely disjoint sections of a particular sentence may still impact inter-annotator agreement, even if their specific annotations don't themselves disagree at all. In this work I provide a variety of different metrics aimed at addressing multiple views on agreement in a partial annotation context.

Also considered was 'agreement with gold', or accuracy on some gold-standard in other words. Table 5.4 contains statistics for annotator agreement with the gold standard supervision from the original AnCora corpus. With a few exceptions, precision was quite good with a median value of 0.723 although there were several outliers on the negative end, most notably annotator 10 (precision=0.161). Recall values were low (median=0.172), which is to be expected when the explicit instructions of the task were to focus more on getting through as many sentences as possible rather than providing heads for every single token.

One interesting finding from Table 5.4 is the correlation of Spanish ability with Gold Precision. While this is not entirely unexpected, since more accomplished speakers of a language are less likely to make mistakes when locating things like syntactic structures or parts-of-speech in sentences, it might have been expected that prior annotation experience would have been a better predictor of agreement with gold due to a familiarity with how annotations typically look. Given the nature of GFL annotation, I would hypothesize that the impact of annotator experience is diminished; when annotators are free to skip confusing structures then it doesn't matter as much whether they have seen similar structures annotated in the past. GFL annotation does not diminish the effect of language experience however, because there is still great value in being able to accurately identify things like part-of-speech in the texts.

While we might expect that annotators would trade off precision for recall, by either quickly annotating many heads leading to decreases in precision or more carefully annotating a smaller proportion of heads for higher precision, Figure 5.3 shows that precision and recall tended to increase together. Although some part of this effect may be explained by factoring in the total number of sentences annotated, with high precision/high recall annotators tending to have fewer sentences overall, this is not always the case–annotator 1 had a large volume for instance.

Within Figure 5.3 there are a few rough clusters: one containing annotators 1, 4, 7, and 9, another containing annotators 2, 6, 5, 10, and 12, and a third with annotators 3, 8, and 11. The first cluster consists of native speakers (4,7,9) and the most experienced annotator (1), while the second, looser cluster, contains a mix of mostly experienced and beginner speakers (2, 5, 10, 12) with a single native speaker (6). This is a clear point in support of the use of native speakers as annotators

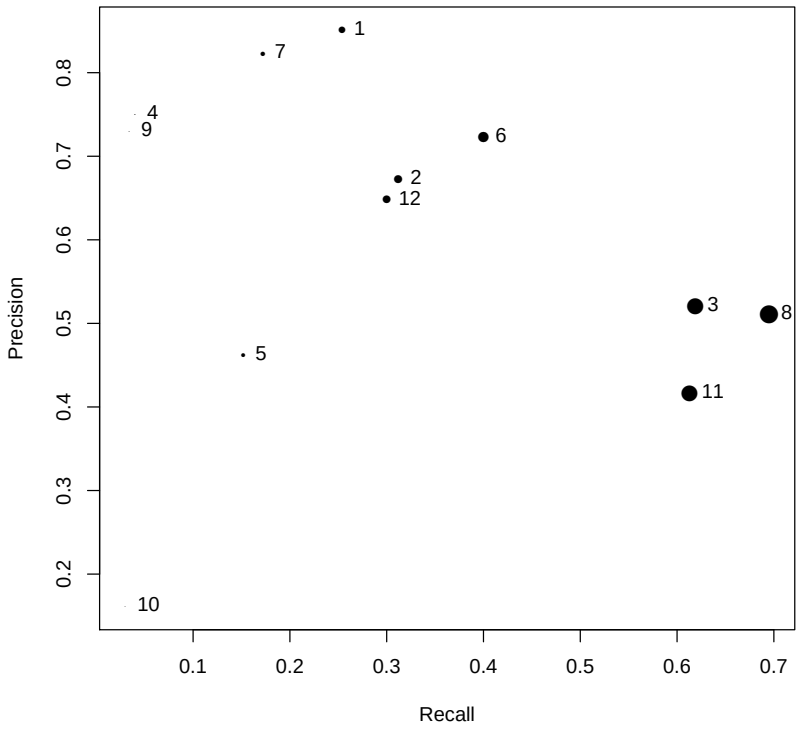



Figure 5.3: Precision vs. Recall for the individual annotation sets. Larger markers indicate faster rates of annotation.

whenever possible, which speaks to the importance of developing annotation schemes that can be used by anyone with minimal training given the difficulty in finding native speakers of various languages with a background in annotation.

The third cluster contains relatively productive annotators with high recall and moderate precision; these are all non-native speakers.

Inter-annotator agreement was also explored by Schneider et al. (2013) in their development of GFL. Their measurement of agreement involves computing the 'promiscuity' of a particular annotation, which is the cardinality of the full set of

dependency trees supported by the partial annotations. The agreement between two annotators is then computable from these promiscuity sets–pairs of annotators who produce annotations that yield similar sets of possible trees are judged to have a high agreement rate.

In addition to the promiscuity measure of how many potential trees are supported by a given annotation, Schneider et al. also define a commitment quotient (Equation 5.1) that varies from 0 to 1, with 1 indicating the annotation supports a single tree, while 0 indicates the annotation does not limit the space of trees at all. Here, $prom(\mathcal{A})$ is the promiscuity measure of the annotation $\mathcal{A}$ and $n$ is the number of lexical items (including the root symbol).

$$com(\mathcal{A}) = 1 - \frac{\log prom(\mathcal{A})}{\log n^{n-2}} \tag{5.1}$$

Single annotator coverage statistics are available in Table 5.4. There was a wide range of variability in both commitment and promiscuity, as might be expected for a collection of naïve annotators covering a wide range of ability levels.

As a first method of inter-annotator agreement, Schneider et al. define *comPrec* shown in Equation 5.2 which measures agreement between two annotations, $\mathcal{A}_1$ and $\mathcal{A}_2$ on a single sentence $s$, with $supp(\mathcal{A})$ being the set of trees supported by annotation $\mathcal{A}$.

$$comPrec(\mathcal{A}_1|\mathcal{A}_2) = com(\mathcal{A}_1)\frac{|supp(\mathcal{A}_1) \bigcap supp(\mathcal{A}_2)|}{|supp(\mathcal{A}_1)|} \tag{5.2}$$

However, as this measure involves taking the intersection of the two sets of supported trees, if the two annotations are incompatible in some way (for instance, if one annotator has marked 'the paper' as 'the ⟨ paper' and the other 'the ⟩ paper')

the intersection will be empty, and the reported agreement will be zero. This makes *comPrec* highly sensitive to true annotator disagreements of this type.

Given that the directionality of dependencies was the most common source of confusion and questions from the inexperienced annotators working during this study, the use of *softComPrec* shown in Equation 5.3 is much more appropriate, where $\ell$ stands for an individual lexical item in a sentence.

$$softComPrec(\mathcal{A}_1|\mathcal{A}_2) = com(\mathcal{A}_1)\frac{\sum_{\ell \in s} \bigcap_{i \in \{1,2\}} suppParents_{\mathcal{A}_i}(\ell)}{\sum_{\ell \in s} suppParents_{\mathcal{A}_1}(\ell)} \tag{5.3}$$

This variant decomposes the intersection from an intersection of trees to an intersection at the level of individual lexical items. In situations where annotators are likely to simple errors, this measure minimizes the impact of those errors on overall agreement scores. It is also easily extended to the multiple annotator case, by simply changing the $\bigcap_{i \in \{1,2\}}$ term to $\bigcap_{i \in \{1 \ldots k\}}$ where $k$ is the number of annotators.

Each of the inter-annotator agreement metrics proposed by Schneider et al. are asymmetric in that they are weighted by the commitment of a particular annotator. This means that $comPrec(\mathcal{A}_1|\mathcal{A}_2) \neq comPrec(\mathcal{A}_2|\mathcal{A}_1)$, provided $\mathcal{A}_1 \neq \mathcal{A}_2$. This makes these measures more of a score, implicitly penalizing annotators for not having a high commitment for the sentence.

In this work, it will also be useful to refer to a measure of inter-annotator agreement that more simply measures disagreements between annotators rather than scoring based on the supported trees. This disagreement is measured over the definite heads that are unambiguously specified by the annotator, either directly or through exhaustion of options by various methods. Given a set of annotators $\mathcal{A}_{1 \ldots k}$, we can calculate a measure of agreement for a sentence $s$ as shown in Equation 5.4,

| Annotator | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.73 | 0.9 | 0.88 | 0.55 | 0.77 | 1 | 0.94 | 0.9 | 0.28 | 0.95 | 0.67 | .80 |
| 2 | 0.73 | 1 | 0.78 | 0.83 | 0.95 | 0.62 | 0.77 | 0.75 | 1 | 0.27 | 0.8 | 0.85 | .78 |
| 3 | 0.9 | 0.78 | 1 | 0.85 | 0.64 | 0.8 | 0.9 | 0.82 | 0.96 | 0.3 | 0.85 | 0.72 | .79 |
| 4 | 0.88 | 0.83 | 0.85 | 1 | 0.6 | 0.88 | 0.83 | 0.92 | 1 | 0.33 | 0.91 | 0.68 | .81 |
| 5 | 0.55 | 0.95 | 0.64 | 0.6 | 1 | 0.46 | 0.64 | 0.55 | 0.83 | 0.23 | 0.59 | 0.85 | .66 |
| 6 | 0.77 | 0.62 | 0.8 | 0.88 | 0.46 | 1 | 0.88 | 0.74 | 0.88 | 0.16 | 0.75 | 0.6 | .71 |
| 7 | 1 | 0.77 | 0.9 | 0.83 | 0.64 | 0.88 | 1 | 0.94 | 1 | 0.2 | 1 | 0.67 | .82 |
| 8 | 0.94 | 0.75 | 0.82 | 0.92 | 0.55 | 0.74 | 0.94 | 1 | 1 | 0.36 | 0.94 | 0.7 | .81 |
| 9 | 0.9 | 1 | 0.96 | 1 | 0.83 | 0.88 | 1 | 1 | 1 | 0 | 1 | 0.81 | .87 |
| 10 | 0.28 | 0.27 | 0.3 | 0.33 | 0.23 | 0.16 | 0.2 | 0.36 | 0 | 1 | 0.11 | 0.12 | .28 |
| 11 | 0.95 | 0.8 | 0.85 | 0.91 | 0.59 | 0.75 | 1 | 0.94 | 1 | 0.11 | 1 | 0.68 | .80 |
| 12 | 0.67 | 0.85 | 0.72 | 0.68 | 0.85 | 0.6 | 0.67 | 0.7 | 0.81 | 0.12 | 0.68 | 1 | .70 |
| Total | .85 | .86 | .86 | .9 | .82 | .77 | .98 | .92 | .96 | .61 | .94 | .82 | |

Table 5.5: Pair-wise and total agreement by annotator. The 'Total' row shows agreement with the set of all other annotators under the gflAgree metric, the 'Avg.' column is the average pairwise agreement.

where $\mathcal{A}_1(\ell)$ is the head specified by annotator 1 for lexical item $\ell$, and $\hat{\ell}$ is the most common head specified for $\ell$ amongst the set of annotators. In other words, the numerator of the summation in Equation 5.4 is the number of annotators who agreed with the most commonly selected head for that lexical item.

$$gflAgree(s) = \frac{1}{|s|} \sum_{\ell \in s} \frac{\sum_{i=1}^{k} [\mathcal{A}_i(\ell) = \hat{\ell}\,]}{k} \tag{5.4}$$

This agreement measure has the benefit of not being impacted by lack of commitment by any one particular annotator. While measuring commitment is valuable in some instances, utilizing agreement measures that respect the fact that annotators may skip certain heads for any number of reasons is important as well.

Table 5.5 shows agreement values both for all pairwise comparisons and agreement with the set of all annotators as well. Overall, agreement within this dataset is high; most annotators have average pairwise agreement in the 70-80's, and agreement for individual annotators with the set of all annotators is even higher–

mostly in the 80's. A few outliers in the agreement deserve some consideration, in particular annotators 9 and 10. Annotator 10 has an average pairwise agreement of .28, which is the worst by a large margin–annotator 5 is second with .66. This is additional evidence that annotator 10 struggled with the task, particularly when combined with the precision/recall information from Figure 5.3. Annotator 9 had very high pairwise agreement, including 100% agreement with several annotators. This is likely the result of the low volume of annotations actually produced by annotator 9, combined with their status as a native speaker. Because they weren't making that many commitments and the ones they did make were relatively accurate, it's not surprising that they ended up producing a subset of some other annotators data.

Interestingly, there was no correlation between the UAS performance of dependency parsing models trained on an annotator's data and either the average pairwise agreement (r=0.01) or the agreement between that annotator and the full set of annotations (r=-0.07). There are likely to be a number of factors influencing this point, including annotation volumes and densities, but in general there doesn't seem to be a great benefit from producing widely accepted or generic annotations.

### 5.5.2 Construction Accuracy

Because the experience levels of the different annotators are recorded, there is an ideal opportunity to consider the variation in the accuracy different groups of annotators achieve on specific linguistic constructions, with consideration for which types of annotation are best left to certain types of annotator. For instance, are relative clauses more readily identified as a constituent by more experienced annotators, or should non-native speakers stick to just bracketing major constituents?

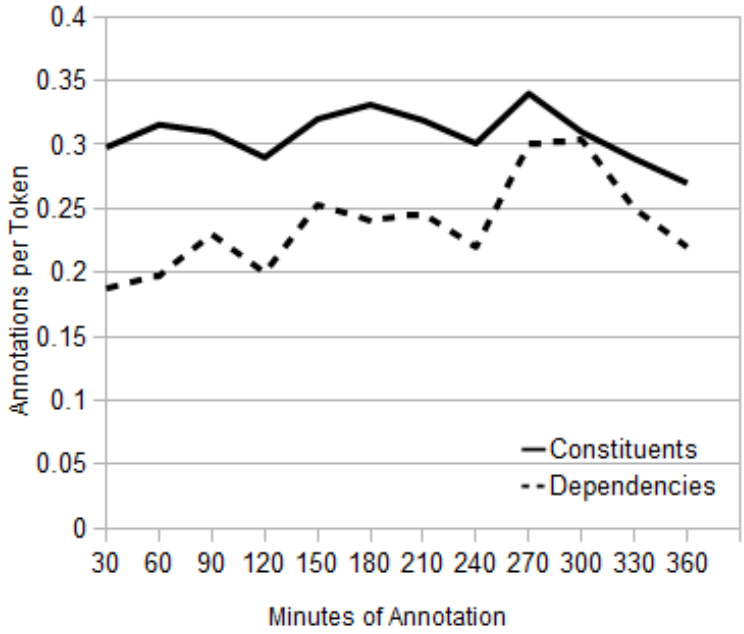



Figure 5.4: Relative densities of annotation types for a single annotator over time.

As a part of these statistics I am collecting figures on the accuracy of the different annotators on specific GFL markups (for instance, parentheses versus carets), the density of different annotations (looking for differences both between annotators and over time), the percentage of different constituent types annotated (nouns versus verbs, etc.), and others. An example of these statistics is shown in Figure 5.4, which shows the relative densities of annotation types over time for one annotator.

In this particular case the annotator seems to be experiencing a few different things. First, there is a clear fatigue effect that is lowering the overall density over time within the two hour blocks, as indicated by the jumps in density at 150 and 270 minutes followed by falling off. This fatigue is also seen within single half-hour time blocks, though not in this figure, as sentences annotated later in the blocks tend to have fewer annotation symbols. Secondly, the annotator is increasing the

number of dependency links relative to the number of brackets over time. The participant reported that while marking constituents was very straightforward, the extra variation associated with dependency marking (direction in particular) made them less confident about providing them initially.

**Determiner / Noun Phrases** A common type of construction that is subject to a clear theoretical choice is the analysis of noun phrases containing determiners. In computational linguistics, the noun is almost always taken to be the head of these phrases, with the determiner being a dependency of the head noun.

The linguistics literature, on the other hand, is split on the correct analysis for nominal phrases containing determiners. The traditional view mirrors the one used in most computational settings, with the noun being taken as the head, while the DP-hypothesis holds that the determiner is the correct head of such phrases [Abney, 1987]. The DP-hypothesis is commonly viewed as true in most areas of generative grammar today (Minimalist Program, etc.), while other theories of syntax still hold the traditional analysis to be true.

The gold standard data for the AnCora corpus has nouns as the heads of nominal phrases with determiners, which is quite common amongst dependency grammars. Figure 5.5 shows that the annotators mostly all agreed with this decision, producing annotations that in most cases respected the notion that determiners should be dependents of their nouns. However, annotators with less experience in Spanish (e.g., 5, 9, and 10) were much more mixed in their provided analyses. Annotators with prior annotation experience (e.g., 1, 6, and 12) were more likely to use the computational standard of nouns over determiners as well.

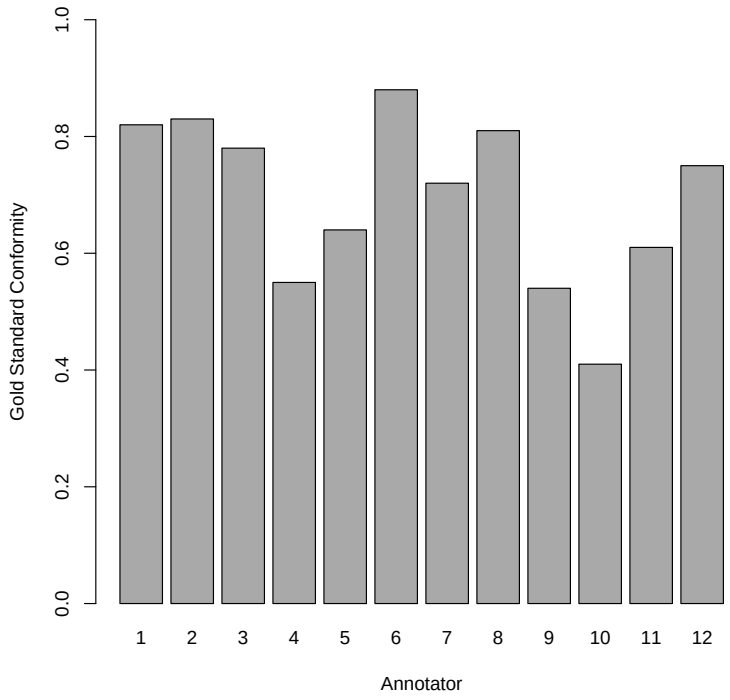



Figure 5.5: Percentage agreement with gold standard on determiner / noun phrase construction

**Modal Verb Phrases** Modal verb phrases are, like nominal phrases with determiners, a case where varying the scope taken by the object in question leads to two functionally similar but theoretically separate hypotheses. We may hypothesize that the modal verb takes a wide scope over the infinitive verb that it appears in complex with; this is a common analysis under different forms of generative grammar, and has been specifically argued for in the case of Spanish [Picallo, 1990, Zagona, 1988]. This analysis has been used in a number of dependency corpora.

An alternative analysis, which is used in the gold data of the AnCora corpus, is to say that the modal verb is a dependent of the verb, rather than the head. On this point, unlike with the nominal phrases just discussed, our annotators had substantial and clear disagreement with the gold standard data. Figure 5.6 shows

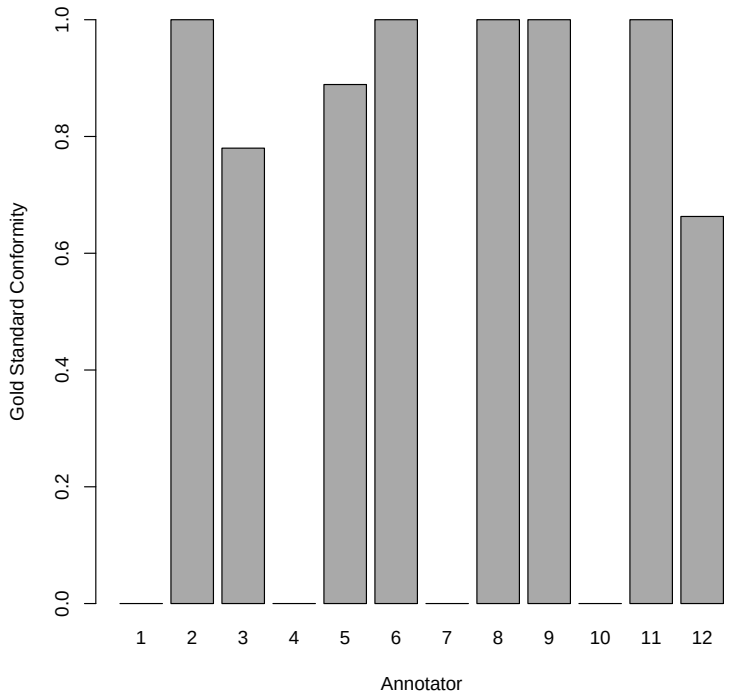



Figure 5.6: Percentage agreement with gold standard on modal verb constructions

this disagreement; in this figure, the y-axis shows the percentage of modal verbs that were annotated as the dependent of the neighboring verb, thereby agreeing with the gold standard. Roughly half the annotators annotated according to the convention, while the other half chose the opposite direction of dependency–making the modal the head.

Compared to the nominal phrase accuracy in Figure 5.5, annotations of modal verb attachment were very polarized; here, annotators were very consistent with the convention they chose, whether that was the gold convention or not, as opposed to the nominal phrase head which saw a lot more variation at the level of a single annotator. This fact may be due in part to the smaller number of modal verbs than determiner-noun complexes, both in absolute number and the fact that

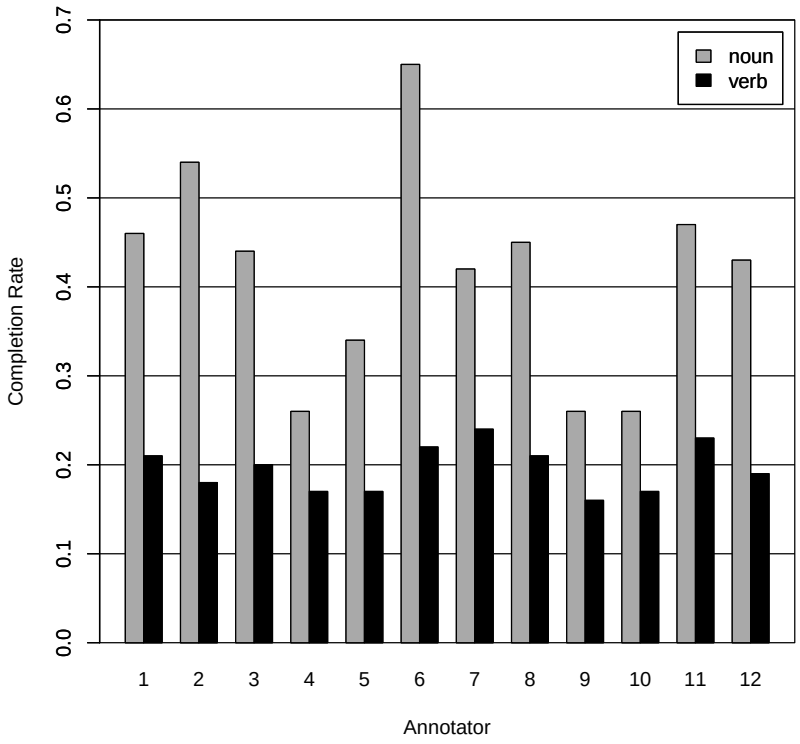



Figure 5.7: Rate of phrase completion by annotator and tag

annotators tended to skip the more theoretically complex modal verbs as compared to the relatively straight-forward determiners. This would imply that the annotators who ended up annotating more modal verbs were more confident, and this turns out to be visible in the data; native speakers and experienced annotators annotating many more modal verbs than non-native and novice annotators.

**Collected Statistics** In Section 4.3.1, a number of methods for constructing simulated partial annotations were provided. A key parameter for most of those methods was completion rates for various categories. In order to provide realistic rate parameters for the methods described in that section, I have collected statistics on category completion rates from the human annotators. Specifically, Figure 5.7

shows the rates of completion for verb and noun phrases by annotator. Rates were significantly higher for noun phrases across all annotators, which is to be expected given that most noun phrases are embedded in verb phrases–making noun phrase completion a prerequisite for verb phrase completion in many instances.

Overall, rates of completion for nouns and verbs were strongly correlated (r=0.65), but some annotators displayed biases towards annotating noun phrases relative to other categories–for instance annotator 2–which was a motivation for building a simulation model that focused in large part on the accuracy of simulation for noun phrases. The fact that annotators tended to lean towards nouns rather than verbs is not altogether surprising given both the annotation process provided, which specifically targets the production of noun phrases as an explicit step, along with the simpler structure they offer relative to verbs and prepositions for instance.

# Chapter 6

# Partial Annotations for Parsing

The use of partial dependency annotations, as discussed in detail in Chapter 5, can open the annotation process to a wider variety of annotators in terms of experience both in the target language and in the process of annotation. In this chapter, results from across this body of research that pertain to the use of partial annotations are discussed. In particular, a number of specialized issues that systems using partial annotations face which may be less relevant in fully-supervised systems will be considered. Through simulation experiments, the tolerance of the ConvexMST parser for partial data is established and an example of performance on real world data is illustrated using the corpus developed in Chapter 5.

Chapter 5 also established that, as might be expected, the use of inexperienced annotators with a wide variety of backgrounds combined with minimal training and guidelines leads to a variety of choices when it comes to standardization of structures and choices regarding representation. Despite this variability, the annotations from all these annotators must be combined and managed in such a way as to provide the maximum utility to the parsers and other tools learning from them.

Despite these occasionally substantial differences, combining annotations from multiple annotators does indeed lead to a system that performs better against a gold standard than when trained on any one particular set of annotations, as will be shown in this chapter.

## 6.1 Agreement with Gold Standards

The major benefit that partial annotations offer is the ability to skip portions of sentences for various reasons. When working with relatively inexperienced annotators, the most common reason for skipping portions of sentences is probably the simple fact that the annotator isn't sure how to label that section. Experienced annotators also skip sections, but their reasons for doing so are more often time-related; they can skip sections that are similar to previously annotated data, resulting in increased speeds.

Considering the inexperienced annotators, this ability to skip difficult sections can lead to a more accurate, clean annotation set overall by not forcing annotators to make a decision one way or another on something.

To attempt to quantify this effect, single annotators annotated the same set of data, with the most experienced annotators fully specifying all dependents in a sentence and allowing others to skip words/sentences at their own discretion when they weren't sure how to proceed. The accuracies of the specified dependency heads for this experiment are shown in Table 6.1.

Considering the results in Table 6.1, the fully specified annotations achieved a similar although slightly lower accuracy than the partial annotations. Particularly in the Spanish case, it is important to remember that the partial specification value is a median of the many annotators that provided partial annotations; several anno-

| Language | Full Specification | Partial Specification |
|---|---|---|
| English | .689 | .715 ± .104 |
| Spanish | .707 | .723 ± .168 |

Table 6.1: Accuracy of Full vs. Partial Annotations. The partial specification value is the median of all annotators.

tators produced accuracies in the 85-95% range–much higher than our experienced annotator providing full annotations. Additionally, the fully-specified annotations were not associated with any increase in performance when used as training data for dependency parsing; the partial annotations led to a UAS score of 70.6, while the fully specified annotations led to a score of 66.6.

Another factor to consider is the time spent annotating and the costs associated with that number. In the Spanish case, the experienced annotator took approximately 13 hours to fully annotate the sentences provided. The inexperienced annotators were each able to partially annotate the same set in roughly two hours, although some were unable to reach the end of the set. If we assume a model of annotation cost based solely on hours of annotation, this result is similar to the result obtained via simulation in Chapter 4: on a fixed budget (here, total hours of annotation), partial annotations may provide increased performance. This distribution of annotations over multiple annotators has potentially valuable cost savings; paying an experienced annotator for 13 hours of work can potentially be much more expensive than paying six annotators for two hours work each, especially when we consider that the inexperienced annotators on this project had a range of abilities in Spanish that included some with just a year or two of experience, while the experienced annotator was a fluent speaker. Finding fluent speakers with annotation experience for many languages is difficult or impossible, and potentially prohibitively expensive if it is indeed possible.

The types of disagreements that annotators had with the gold standard varied from annotator to annotator. In some cases, as with those outlined in Chapter 5 such as the headedness of noun phrases with determiners or verb phrases with modal, the disagreements are readily apparent and relatively principled. In other cases the reasoning for the disagreement can likely be attributed to little other than annotator inexperience, either in the language or annotation in general. For instance, in the case of the annotator with the lowest gold agreement numbers (annotator 10) there was little generalization to be made–the errors were essentially random.

## 6.2 Comparison with Task-Level Supervision

While partial annotations are one way of cheaply adding supervision to a corpus, there are other options as well. Most commonly, some type of task-level supervision is employed that serves to limit the potential space of solutions.

In my work, the ConvexMST parser is a perfect example of this. In the original formulation[Grave and Elhadad, 2015], it operates exclusively on task-level supervision in the form of universal grammar (UG) rules that the dependency trees should obey. However, the modified version presented in Chapter 4 allows for instance-level partial dependency annotations in addition to these UG rules. To consider the relative benefits of these feature sets, we ran a set of experiments broadly divided into two classes: Parse Imputation and Semi-Supervised Parsing. These two modes reflect the multiple potential uses for a dependency parser utilizing partial annotations, depending on where the final sentences to infer structure on come from. When the structures of sentences with existing partial dependencies are to be inferred, the parser is running in an imputation type setup where it must respect the human-provided dependency fragments. When the structures to be in-

| Parser | Features | Gold Tags | | Predicted Tags | |
|---|---|---|---|---|---|
| | | EN | ES | EN | ES |
| RB | *N/A* | 16.7 | 30.1 | 16.7 | 30.1 |
| Gibbs Parser | GFL | 50.1 | 60.4 | 48.7 | 52.5 |
| ConvexMST | UG | 48.8 | 57.1 | 42.5 | 38.3 |
| | GFL | 60.2 | 68.0 | 59.3 | 68.3 |
| | GFL+UG | 63.0 | 70.6 | 60.3 | 69.6 |

Table 6.2: Imputation accuracy on English and Spanish GFL-annotated texts, 10 or fewer words.

ferred are over sentences with no existing partial annotations, the parser is running in a more straight-forward semi-supervised setup.

The results from Tables 6.2 and 6.3 demonstrate that the partial dependency features (GFL in the tables) strictly outperform the task-level UG rules, although a combination of the two yielded the best results overall.

From Table 6.2, we can see that UG-related imputation methods benefit more from gold POS tags than GFL-related imputation methods. Or, alternatively, feature sets that include GFL-based information are more tolerant of noisy POS taggings. This may indicate that GFL annotations are more independent of POS tags, while universal grammar rules depend on having correct POS tags and are brittle in the face of noise. Table 6.2 also shows that adding GFL annotations leads to better imputation results whether we start from a blank baseline or from UG rules. This pattern holds even when we greatly expand the size of the data with raw texts. This may indicate that in imputation, universal grammar rules may introduce too much extra noise to be particularly useful on their own. This result is not entirely unexpected, given the relative performances of the partial dependency and UG constraints on their own, but it provides an additional piece of evidence that the utility of instance-level supervision, even in small amounts, can trump generalized task-level supervision.

| Parser | Features | Gold Tags | | Predicted Tags | |
|---|---|---|---|---|---|
| | | EN | ES | EN | ES |
| RB | *N/A* | 17.1 | 28.0 | 17.1 | 28.0 |
| Gibbs Parser | GFL | 60.2 | 65.3 | 55.8 | 52.7 |
| ConvexMST | GFL+Raw Text | 62.7 | 67.8 | 57.6 | 51.8 |
| | UG | 63.1 | 63.5 | 56.9 | 50.0 |
| | GFL | 65.9 | 70.5 | 61.2 | 67.1 |
| | GFL+UG | 68.2 | 71.3 | 63.2 | 67.3 |

Table 6.3: Directed dependency accuracy on English and Spanish universal treebanks using annotator provided GFL annotations, 10 or fewer words.

Additionally, when using both partial dependencies and UG-based constraints, there is optionally a weighting factor that can bias the parser towards being more likely to respect either the partial dependencies or the UG constraints. This weighting was varied during development, and it was found that for basically all the datasets, the best results were obtained when violations of partial dependency constraints were treated as worse than violations of UG constraints.

Table 6.3 shows the semi-supervised parsing results on the English and Spanish universal treebanks for sentences with 10 or fewer words. Methods *GFL*, *GFL+UG* and *UG* are trained with GFL-annotated texts, and guided by GFL annotations, weighted GFL annotations and universal grammar rules, and universal grammar rules. Method *Raw Texts* is trained with raw texts from universal treebanks, and guided by universal grammar rules. This condition is equivalent to the method used by Grave & Elhadad.

It can be seen in Table 6.3 that the *GFL+UG* method achieves the best performance in our parsing task. These results use an equal weighting for all annotators, although smarter weighting schemes based on annotator experience or ability could produce improved results.

The Spanish results from Tables 6.2 and 6.3 use the combined annotation sets

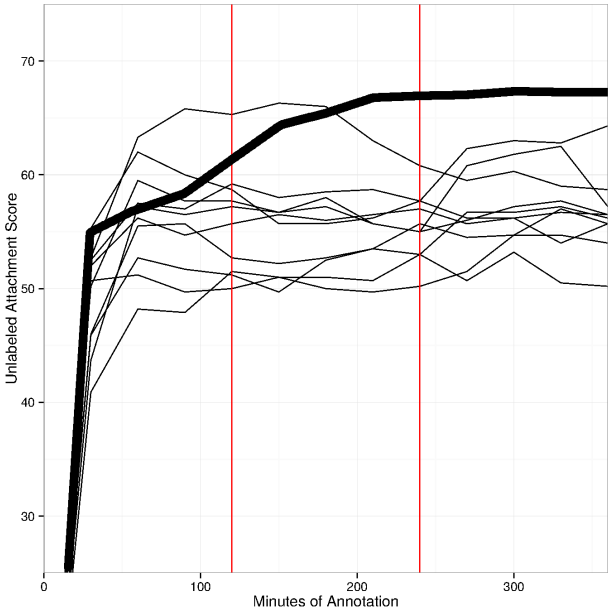



Figure 6.1: UAS for individual annotators over time. The thick line represents the combination of all annotators.

from all of the annotators that provided partial annotations. Together, they achieve results that are greater than any one individual annotator was able to provide. Figure 6.1 shows the UAS scores over time for each individual annotator–note that all are lower than the 67.3 achieved through their combination. The combination of all annotators is represented by the thicker line, which is trained using all of the annotators' data for a given annotation time frame.

When compared to the simulation results from Section 4.3, these parsing results are poorer than we might have expected. A likely source of this extra error is the individual annotator's choices with respect to linguistic structure standards. As shown in Chapter 5, individual annotators made very different decisions with respect to how to annotate particular structures. While each annotator was more or less consistent with respect to their own choices, those choices were not always

| | EN | ES |
|---|---|---|
| UG | 39.2 | 31.2 |
| GFL | 49.6 | 42.1 |
| GFL+UG | 52.2 | 44.8 |

Table 6.4: Directed dependency accuracy on English and Spanish universal treebanks RBG-Parser trained on imputed texts.

in agreement with either other annotators or the gold standard around which the test set is built.

## 6.3 Completing Partial Annotations

When using partial annotations, two basic pipelines can be envisioned. Either the partial supervision features are incorporated into a model that directly parses new sentences, or the partial sentences can be completed and then used to train a standard dependency parsing model.

### 6.3.1 Fill-then-Parse vs. Fill+Parse

The results from Table 6.3 are illustrative of the Fill+Parse technique, where the partial features are used to build a model that directly parses the test sentences. Table 6.4 contains the results of completing the partial annotations and then subsequently training RBG-Parser on the filled in annotations (Fill-then-Parse).

As can be seen, the Fill+Parse pipeline, illustrated in Figure 6.2, gives better results overall by up to ten points. There are quite a few factors that go into the relative performance of these two pipelines, most importantly the strength of imputation and the tolerance of the final parser to noisy data, but these results seem to indicate that using a single model is more effective in this case.

The reasons for the lower performance of RBG-Parser when using the im-

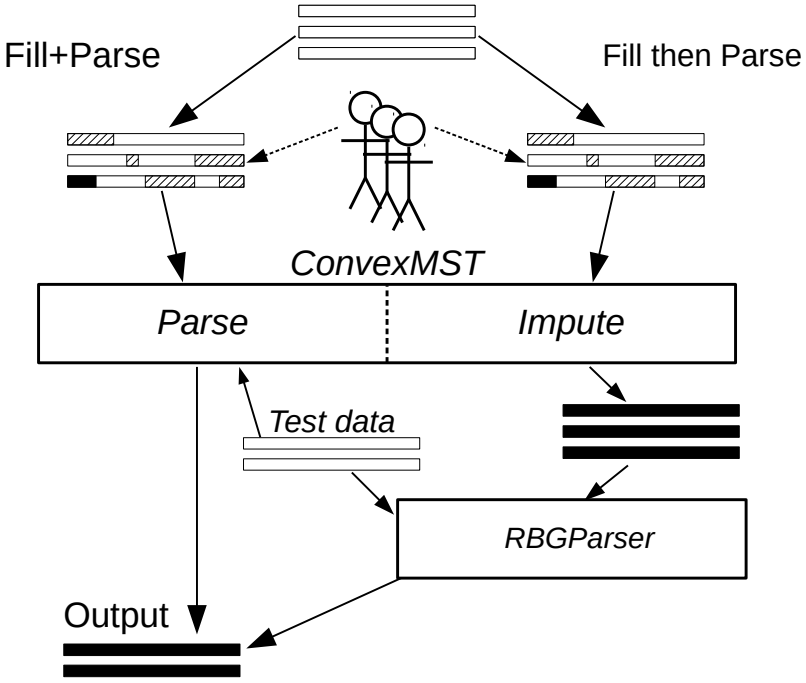



Figure 6.2: Fill+Parse vs. Fill then Parse data paths.

puted data are likely similar to the reasons the UG features struggle on noisy POS tags–with a noisy input, these tools are dependent on relatively clean input data in order to generate high-performing output. The use of higher order features in RBG-Parser also means that parsing errors in the data end up being propagated onto many more input features, potentially spoiling the model. It may be the case that the model is learning what the annotator has 'taught' it perfectly well, but because that data was so noisy, the resulting model is similarly noisy.

This suggests that if we wish to build a parsing model using partial annotations as the training data, the most effective method of doing so is to use a parser that is able to understand partial annotations as a native supervision type. Imputing missing dependencies for training with standard models is possible, but much less effective. Imputation schemes may still have uses however; for instance during the annotation process, where a partial dependency may be completed in a hypothetical fashion and presented to a human annotator for validation and revision, similarly

to how many corpora are produced using an existing parser to provide candidate parses for human consideration.

## 6.4 Selective Learning

Partial annotations provide a straightforward way to break down the rather complex task of syntactic annotation by decomposing the monolithic tree into usable, smaller units down to the level of individual arcs. As such, partial annotations are a natural match for active learning setups potentially up to the scale of crowdsourcing by making individual annotation decisions cheap yet effective.

True active learning experiments are left for future work, but in this section I present the results of a pilot 'selective learning' experiment that was conducted on the partial Spanish dependency corpus. Selective learning and active learning are similar in that they both operate in an iterative fashion, where new data is annotated at each time step based on some set of criteria. Differentiating selective learning from active learning is done on the basis of how the selection of new data is made. In selective learning, we assume an existing corpus which is being reanalyzed and reannotated based on the actual annotations themselves, whereas in active learning some external model (for instance a parser) is used to weight potential corpus additions–typically by some measure of confidence from the model.

Selective learning experiments, similarly to active learning, are obviously best done using human annotators, but some initial evidence is presented here from simulations that shows partial annotations can be useful in such experiments.

As an initial experiment, the Spanish partial corpus was taken as a starting point upon which the simulations would expand. Over multiple rounds, a selection function chose individual dependency arcs for annotation or re-annotation of their

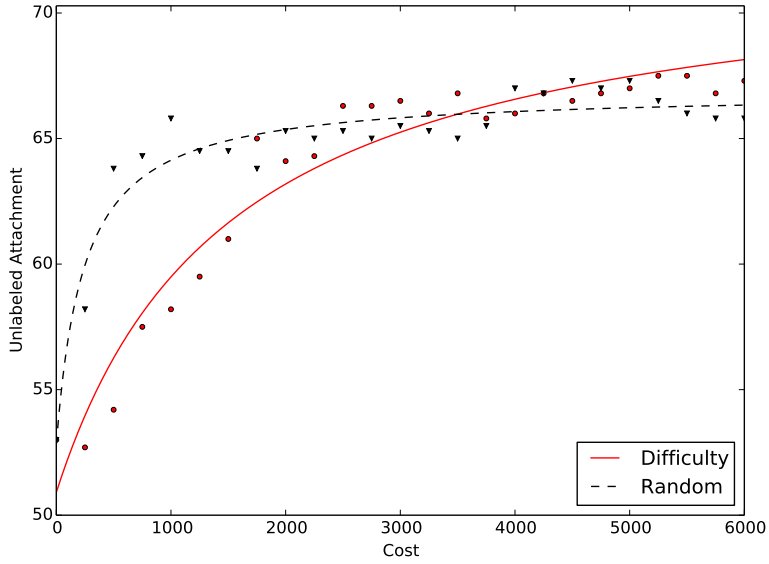



Figure 6.3: Selective Learning Simulation

parents. In each round, 250 arcs (roughly 10 sentences worth) were chosen. In the ideal scenario these arcs would have been sent to a human annotator, but in this simulation they were annotated by an oracle using the gold standard dependencies.

The selection function was either 'random', which simply selected random arcs for annotation, or a 'difficulty' function. The difficulty function assigns a weight to each arc that is equal to the number of different heads it has been given across the dataset (disagreement) plus the percentage of arcs in the sentence that are unspecified. This biases the difficulty function towards sentences containing large numbers of unknown arcs as well as arcs with disagreements among the human annotators.

Figure 6.3 contains the results of this simple selective learning experiment. Although the random selection function is better initially, the difficulty function eventually catches up and surpasses it. It should be noted that defining the diffi-

culty/uncertainty function in terms of the annotations themselves rather than on the output on the parser is unusual in active learning experiments. The ConvexMST parser used here currently has no ability to reason about confidence, and as such it is impossible to define uncertainty in terms of parser output. Future work in this area should certainly include parser output as at least a factor in the determination of uncertainty.

## 6.5 Future Directions

This work on the properties of partial dependencies, in concert with the results from Chapter 5 concerning the behavior of individual annotators, suggests a number of possible interesting directions that could be explored.

### 6.5.1 Probabilistic Models of Annotation

The use of partial annotations and multiple annotators naturally leads to a number of different situations where different annotators either agree or disagree at the level of either the full parse tree or the level of individual arcs. The natural question that arises from these circumstances is how to determine the correct label (head) for any given arc given the full set of annotations on that particular arc. In the work presented in this dissertation there was no explicit attempts to determine the ground truth for an arc; if annotators disagreed, the conflicting annotations were all added to the training set and simply allowed to conflict.

There are a variety of smarter techniques that could be potentially employed to resolve annotator disagreements at the arc level. One simple method would be establish some measure of annotator accuracy on a known tuning set, and then weight the contributions of annotators more highly when that annotator has been

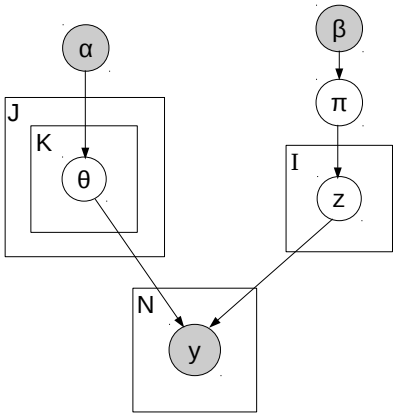



Figure 6.4: Graphical model sketch of Dawid and Skene's probabilistic model of annotation. 'J' is the number of annotators, 'K' is the number of categories, 'I' is the number of items, 'N' is the number of annotations. $\theta$ is the annotator accuracy, $\pi$ is the category prevalence, $z$ is the true category. 'y' are observed labels, $\alpha$ and $\beta$ are priors on the accuracies and prevalences.

shown to agree with the provided standard frequently. An generalization of this procedure over multiple categories is provided by Dawid and Skene (1979); originally used to make inferences concerning consensus among patient histories taken by multiple doctors, it was recently applied to a large scale NLP annotation task– namely word sense labeling [Passonneau and Carpenter, 2014].

The graphical model sketch of the Dawid and Skene model, as presented by Passonneau and Carpenter, is shown in Figure 6.4. The model includes the following parameters:

- $z_i \in 1 : K$ is the true category of item $i$.
- $\pi_k \in [0, 1]$ is the probability that an item is in category $k$.
- $\theta_{j,k,k'} \in [0, 1]$ is the probability that annotator $j$ assigns label $k'$ to an item whose true category is $k$

Under this model, given a set of observed annotations $y$, we can estimate the true label for an item $i$ using Bayes's rule:

$$\begin{aligned} p(z_i|y,\theta,\pi) &\propto p(z_i|\pi)p(y|z_i,\theta) \\ &= \pi_{z[i]} \prod_{ii[n]=i} \theta_{jj[n],z[i],y[n]} \end{aligned}$$

This model works particularly well for tasks such as word sense disambiguation, because each item (word) has well-defined categories (the senses) and each annotation is independent and unstructured. The task of syntactic annotation, on the other hand, is structured and it is not immediately clear what the categories and labels should be, since the actual annotation (namely the parent token index) is not particularly informative across different sentences.

However, if we allow the items $i$ from the Dawid and Skene model to be not the actual dependency arc ('the $\rightarrow$ dog') or even the indicies ($3 \rightarrow 4$), but rather a delexicalized version of the arc based on POS tags (D $\rightarrow$ N), then we obtain a reasonable number of categories that do have cross-sentential meaning. Using the actual words in the arc leads to an intractable number of categories, and using the indicies means that the categories are only relevant for the sentence we are currently considering.

Furthermore, by modeling the categories as a function of the POS tags involved, we can reason about the prevalence of different categories and annotator biases in ways that capture interesting linguistic information. For instance, Chapter 5 showed that the annotators on our project had strong and consistent tendencies for annotating the heads of nominal phrases as either the determiner or noun. Under the POS-based category formulation we would be able to model these biases as

a difference in the tendency to annotate 'D' as the head of 'N' or vice-versa. This could provide an effective way of allowing annotators some freedom to work the way that makes the most sense to them, and then 'correcting' their biases after the fact towards a standard that we choose to impose by setting the prior on category prevalences according to a gold standard we provide, either by extracting the information from an existing corpus or by hand specifying the desired prevalences for the different categories.

### 6.5.2 Crowdsourcing

The discussions in this chapter related to active learning, combined with the potential for splitting up the annotation workload of a sentence into arbitrarily small chunks using partial annotations, leads to the question of how far this partiality can be taken. This dissertation has demonstrated as one of its main goals that novice annotators with a minimal amount of training and feedback are capable of producing annotations that are ultimately useful in constructing parsing models. Would it be possible and effective to potentially break things up even more, and deconstruct the task of syntactic annotation to asking many annotators questions about a single arc, for instance?

With the complicated process of syntactic annotation broken up into units of individual arcs, the possibility would also exist for this annotation to be done via crowdsourcing. Typically, tasks that can be crowdsourced involve annotators making many, simple decisions. This allows for the annotation process to essentially be parallelized across many annotators who are not even required to be able to do the full task; there would be no requirement that annotators be able to specify a full parse tree in order for them to function effectively as workers in a crowdsourced

dependency annotation project, for instance. This desire for simplicity of annotation decisions means that syntactic tree annotations have not typically been considered as viable targets for crowdsourcing efforts.

It's also easy to see how the model of annotation described in the previous section would be quite valuable in a crowdsourced environment. As the number of annotators increases, less principled methods of correcting for annotator biases such as simply weighting annotators differently quickly become impossible.

# Chapter 7

# Conclusion

Building computational linguistics tools and resources is a challenging task that only gets harder when the language or domain of interest has few existing resources from which to build. In such an environment, everything becomes more expensive to obtain: annotators become scarce and are more frequently inexperienced, datasets don't come pre-built, and standards for how to analyze the linguistic facts at hand may not even exist. Working in low-resource environments demands that attention by paid to all aspects of the data collection, analysis, and evaluation process, both because the data itself is expensive to produce and because each single bit of data has the potential to dramatically alter models and performance if left unchecked.

This dissertation has presented work from two main lines of research: In the first, techniques for working with small amounts of data were discussed, and in the second, looking directly at the nature of small corpora as distinct entities from their larger, more established relatives was the goal. These investigations found support for the central thesis of this work, that supervised techniques can provide a superior level of performance relative to either unsupervised or indirectly supervised

methods–even when faced with minimal, noisy data.

**Representation of Supervision** This dissertation has argued that in low data scenarios, there are big potential benefits to be had by adopting a more linguistically sophisticated approach to supervision. This was explored in Chapter 2, and whether linguistic sophistication means asking for more involved annotations from annotators or simply allowing models access to additional levels of linguistic analysis such as discourse or sublexical units, it is clear that an informed linguistic point-of-view can contribute to performance gains. This was supported by results from both POS-tagging and dependency parsing, where allowing the models to consider sublexical segmentation features resulted in performance gains. Additionally, for the POS-tagging results, there was an observable gradation in performance from the worst performance using no sublexical features, through 'pseudo-morphemes' obtained as n-gram chunks, to finite state transducer features that led to the best performance. The results from dependency parsing indicated that while there is potential benefit to be had from gaining access to morphological information by using morphemes as the base unit of analysis, the unsupervised methods used to generate the segmentation features were simply not effective enough to realize those gains. Should better methods of obtaining morphological segmentations become available, it would not be surprising to find better parsing results as well.

Figure 7.1 illustrates the various paths to complex representations and supervision sources considered in different areas of this dissertation. The use of a lone expert annotator was looked at as a direct comparison to the 'many annotators' scenario in the Spanish partial dependency corpus, while pre-trained models were used to produce morphological features for the POS-tagging and morphologically-aware dependency parsing experiments. While the possibilities for the partial annotation

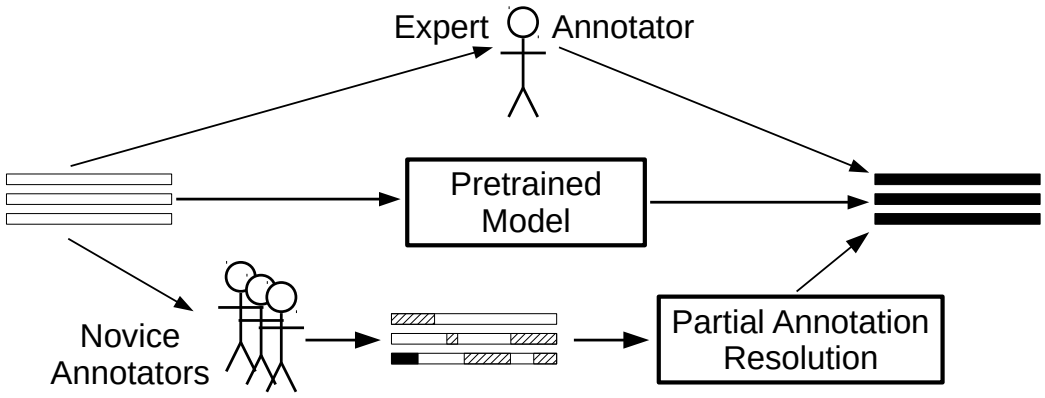



Figure 7.1: Diagram of annotation paths for complex annotation representations.

resolution component of the pipeline in Figure 7.1 were left largely unexplored in this work, the potential exists for techniques like the probabilistic models of annotation of Passonneau and Carpenter to increase performance in this area, making it even more practical to build complex representations from simpler, partial annotations.

**Benefits of Partial Supervision** On techniques for working with low-resource languages, this dissertation has argued that the use of supervised or semi-supervised techniques is perhaps more appropriate in many instances than using comparatively unsupervised approaches despite the general trend in the literature towards using minimally supervised methods to allow for easy cross-lingual application. While this conflicts with the desire for complex, informed annotation objects from Chapter 2 like morphological segmentations and full parse trees, partial annotations provide a proxy supervision source capable of being provided by humans and able to be combined and used similarly to full trees. Using time-controlled experiments, it has been demonstrated that relatively accurate models can be built in a very reasonable amount of time; and when even 'unsupervised' methods require the time to put together a corpus, the time costs associated with semi-supervision are often worth it from a performance perspective.

Unsupervised methods are certainly better, almost by definition, at out-of-

the-box cross-lingual applicability, but this can come at the expense of playing to a common denominator from the point of view of syntax. As demonstrated by my results on feature selection for the ConvexMST parsing model, universal properties, rules, and tendencies can do a reasonable job at parsing any given language, but ultimately suffer from an upper-bound due to the desire for maintaining generality– language specific features are not limited like this. A small amount of direction supervision, even in a partially specified context, can begin to shape the grammar for that individual language in ways that outperform the generalized unsupervised model. This fact was demonstrated by the success of the ConvexMST model in any condition involving the use of instance level partial annotations, as opposed to the conditions that rely solely on the universal grammar rules.

While partial annotations as a supervision source are clearly valuable, the question of how best to integrate them into a parsing pipeline was also considered. In particular, this dissertation considered the relative benefits of filling in the partial annotations to produce full annotations usable by any parser versus using a parser with the partial features directly. A clear performance boost was found for using the parser that directly utilizes the partial annotations. The disagreements with a gold standard that the use of an imputation scheme introduces seem to be magnified by feeding them back into a general dependency parser, whereas a parser that is aware of partial annotations can maintain a separation between human-provided annotations and the hypothetical completions for the missing values; this allows the partial parser access to implicit confidence values that the general parser is blind to due to the imputation process.

**Annotator Behavior** To consider the nature of small corpora and their associated annotation processes, a rapidly constructed partial dependency corpus for

Spanish was developed in a controlled environment using mostly inexperienced annotators with minimal training time. This corpus has provided the raw data needed for an in-depth examination of the habits of individual annotators along with exploring inter-annotator agreement in a partial annotation environment, as well as training individual parsing models and examining how differences in annotation habits can lead to biases in trained parsing models and impact performance, especially relative to an established gold standard dataset.

This corpus provided evidence that, using various measures of total cost, performance levels for a partially annotated corpus can meet or even exceed that of a fully specified corpus. In particular, a grouping of six annotators each providing two hours of partial annotations was able to provide training data that outperformed that of a single experienced annotator fully specifying annotations for twelve hours. This was a win for partial annotations in terms of time; the process was effectively parallelized between annotators leading to 'wall clock' annotation time that was much shorter. This was also an economic win for partial annotations; the experienced annotator receives a higher rate of pay, which makes the distributed partial corpus both cheaper, quicker, and better-performing.

Examining the nature of the Spanish corpus itself leads to the result that while individual annotators may differ both from one another and from an established gold standard, even most novice annotators are relatively consistent with respect to how they tend to label individual constructions and phenomena–at least when allowed to defer and skip those they are unsure of. It might be an interesting extension of this finding to consider annotator consistency on complicated syntactic structures by requiring full annotations from novice annotators on a specially constructed dataset. As the annotators are often capable of identifying the difficult

constructions, they may be capable of producing a consistent analysis even if they are basically making it up the first time they encounter the structure. In any case, the observed consistency is an encouraging finding for the use of novice annotators in annotation projects, and points to the potential feasibility of adopting a useful probabilistic model of annotation to help resolve inter-annotator disagreements or opening the annotation process even further through the use of crowdsourcing techniques. Current methods for combining differing analyses from throughout this work essentially threw all the various features together, but in a crowdsourced scenario with many more annotators techniques will need to be in place for doing more intelligent resolution of differences.

**Summary** To conclude, this dissertation has provided evidence that there are benefits to working in a direct, instance level supervised format even when considering low-resource languages or limited domains; the benefits of direct supervision can provide superior performance when compared with unsupervised or task-level supervised models, even when the total annotated volume is very low. In order to rapidly produce the required annotations, partial annotations provide a helpful representation that allows annotators from a wide variety of backgrounds to assist in production. By lowering the barriers to entry when it comes to recruiting annotators, projects needing annotation can accomplish their goals at lower costs while maintaining good performance. While there will always be a place for extremely high-quality, guideline heavy, gold standard annotations, this dissertation has shown that annotation projects of all sizes are capable of producing the data they need to achieve competitive performance while also allowing for linguistic exploration of the annotation objects and annotators themselves.

# Bibliography

[Abney, 1987] Abney, S. P. (1987). *The English noun phrase in its sentential aspect*. PhD thesis, Massachusetts Institute of Technology.

[Alicante et al., 2012] Alicante, A., Bosco, C., Corazza, A., and Lavelli, A. (2012). A treebank-based study on the influence of Italian word order on parsing performance. In Chair), N. C. C., Choukri, K., Declerck, T., Doan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of LREC'12*, Istanbul, Turkey. European Language Resources Association (ELRA).

[Ballesteros et al., 2015] Ballesteros, M., Dyer, C., and Smith, A. N. (2015). Improved transition-based parsing by modeling characters instead of words with lstms. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

[Bamman and Crane, 2011] Bamman, D. and Crane, G. (2011). The ancient Greek and Latin dependency treebanks. In *Language Technology for Cultural Heritage*, pages 79–98. Springer.

[Bender, 2009] Bender, E. M. (2009). Linguistically naïve!= language independent: Why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32. Association for Computational Linguistics.

[Bender, 2011] Bender, E. M. (2011). On achieving and evaluating language-independence in NLP. *Linguistic Issues in Language Technology*, 6(3):1–26.

[Bender et al., 2002] Bender, E. M., Flickinger, D., and Oepen, S. (2002). The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of

Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In Carroll, J., Oostdijk, N., and Sutcliffe, R., editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

[Bikel, 2004] Bikel, D. M. (2004). Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

[Bird, 2009] Bird, S. (2009). Natural language processing and linguistic fieldwork. *Comput. Linguist.*, 35(3):469–474.

[Bisk and Hockenmaier, 2015] Bisk, Y. and Hockenmaier, J. (2015). Probing the linguistic strengths and limitations of unsupervised grammar induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[Cohen et al., 2012] Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2012). Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 223–231. Association for Computational Linguistics.

[Cohen et al., 2013] Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2013). Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL-HLT*, pages 148–157.

[Das and Petrov, 2011] Das, D. and Petrov, S. (2011). Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL-HLT*, Portland, Oregon, USA.

[Deen, 2002] Deen, K. (2002). *The acquisition of Nairobi Swahili: The morphosyntax of inflectional prefixes and subjects*. PhD thesis, UNIVERSITY OF CALIFORNIA Los Angeles.

[Ding, 2011] Ding, W. (2011). Weakly supervised part-of-speech tagging for Chinese using label propagation. Master's thesis, University of Texas at Austin.

[Dyer et al., 2015] Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, A. N. (2015). Transition-based dependency parsing with stack long short-term memory. 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 334–343. Association for Computational Linguistics.

[Eisner and Satta, 1999] Eisner, J. and Satta, G. (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.

[Finkel et al., 2006] Finkel, J. R., Manning, C. D., and Ng, A. Y. (2006). Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

[Flannery et al., 2011] Flannery, D., Miayo, Y., Neubig, G., and Mori, S. (2011). Training Dependency Parsers from Partially Annotated Corpora. In *IJCNLP*, pages 776–784.

[Garrette et al., 2013] Garrette, D., Mielens, J., and Baldridge, J. (2013). Real-World Semi-Supervised Learning of POS-Taggers for Low-Resource Languages. In *Proceedings of the 51th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

[Goodman, 1998] Goodman, J. T. (1998). *Parsing Inside-Out*. PhD thesis, Harvard University Cambridge, Massachusetts.

[Grave and Elhadad, 2015] Grave, E. and Elhadad, N. (2015). A convex and feature-rich discriminative approach to dependency grammar induction.

[Hopper and Thompson, 1985] Hopper, P. J. and Thompson, S. A. (1985). The iconicity of the universal categories nounand verb. *Iconicity in syntax*, pages 151–183.

[Hwa et al., 2005] Hwa, R., Resnik, P., and Weinberg, A. (2005). Breaking the Resource Bottleneck for Multilingual Parsing. In *The Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*. Conference on Language Resources and Evaluation.

[Johnson et al., 2007] Johnson, M., Griffiths, T., and Goldwater, S. (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; Proceedings of the Main Conference*, pages 139–146.

[Klein and Manning, 2004] Klein, D. and Manning, C. D. (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.

[Kuhn, 2004a] Kuhn, J. (2004a). Applying computational linguistic techniques in a documentary project for Qanjobal (Mayan, Guatemala). In *In Proceedings of LREC 2004*. Citeseer.

[Kuhn, 2004b] Kuhn, J. (2004b). Experiments in parallel-text based grammar induction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 470. Association for Computational Linguistics.

[Kupiec, 1992] Kupiec, J. (1992). Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language*, 6(3).

[Kurimo et al., 2010] Kurimo, M., Virpioja, S., Turunen, V. T., et al. (2010). Proceedings of the morpho challenge 2010 workshop. In *Morpho Challenge Workshop; 2010; Espoo*. Aalto University School of Science and Technology.

[Lary and Young, 1990] Lary, K. and Young, S. (1990). The estimation of stochastic context-free grammars using the inside-outside algrithm. *Computer, Speech and Language*, 4:35–56.

[Li et al., 2012] Li, S., Graça, J., and Taskar, B. (2012). Wiki-ly Supervised Part-of-Speech Tagging. In *Proceedings of EMNLP*, Jeju Island, Korea.

[Liang et al., 2009] Liang, P., Jordan, M. I., and Klein, D. (2009). Probabilistic grammars and hierarchical Dirichlet processes. *The handbook of applied Bayesian analysis*.

[Ling et al., 2015] Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015). Character-based Neural Machine Translation. *arXiv preprint arXiv:1511.04586*.

[Manning, 2011] Manning, C. D. (2011). Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In *Proceedings of CICLing*.

[Marcus et al., 1993] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

[Marquis and Shi, 2012] Marquis, A. and Shi, R. (2012). Initial morphological learning in preverbal infants. *Cognition*, 122(1):61–66.

[Matsuzaki et al., 2005] Matsuzaki, T., Miyao, Y., and Tsujii, J. (2005). Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 75–82. Association for Computational Linguistics.

[McDonald et al., 2005] McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (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*, pages 523–530. Association for Computational Linguistics.

[Merialdo, 1994] Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2).

[Mielens et al., 2015] Mielens, J., Sun, L., and Baldridge, J. (2015). Parse imputation for dependency annotations. In *Proc. of ACL*.

[Naseem, 2014] Naseem, T. (2014). *Linguistically motivated models for lightly-supervised dependency parsing*. PhD thesis, Massachusetts Institute of Technology.

[Naseem et al., 2012] Naseem, T., Barzilay, R., and Globerson, A. (2012). Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

[Naseem et al., 2010] Naseem, T., Chen, H., Barzilay, R., and Johnson, M. (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.

[Osborne and Baldridge, 2004] Osborne, M. and Baldridge, J. (2004). Ensemble-based Active Learning for Parse Selection. In *HLT-NAACL*, pages 89–96. Citeseer.

[Passonneau and Carpenter, 2014] Passonneau, R. J. and Carpenter, B. (2014). The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

[Petrov et al., 2006] Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

[Picallo, 1990] Picallo, M. C. (1990). Modal Verbs in Catalan. *Natural Language & Linguistic Theory*, 8(2):285–312.

[Pullum and Zwicky, 1988] Pullum, G. K. and Zwicky, A. M. (1988). The syntax-phonology interface. *Linguistics: the Cambridge survey*, 1:255–280.

[Schneider et al., 2013] Schneider, N., O'Connor, B., Saphra, N., Bamman, D., Faruqui, M., Smith, N. A., Dyer, C., and Baldridge, J. (2013). A framework for (under)specifying dependency syntax without overloading annotators. *CoRR*, abs/1306.2091.

[Shindo et al., 2012] Shindo, H., Miyao, Y., Fujino, A., and Nagata, M. (2012). Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 440–448. Association for Computational Linguistics.

[Sun et al., 2014] Sun, L., Mielens, J., and Baldridge, J. (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.

[Täckström et al., 2013] Täckström, O., Das, D., Petrov, S., McDonald, R., and Nivre, J. (2013). Token and Type Constraints for Cross-Lingual Part-of-Speech Tagging. In *Transactions of the ACL*. Association for Computational Linguistics.

[Taddy, 2011] Taddy, M. A. (2011). On estimation and selection for topic models. *arXiv preprint arXiv:1109.4518*.

[Taulé et al., 2008] Taulé, M., Martí, M. A., and Recasens, M. (2008). AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *LREC*.

[Virpioja et al., 2013] Virpioja, S., Smit, P., Grönroos, S.-A., Kurimo, M., et al. (2013). Morfessor 2.0: Python implementation and extensions for morfessor baseline.

[Wang and Blunsom, 2013] Wang, P. and Blunsom, P. (2013). Collapsed Variational Bayesian Inference for PCFGs. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 173–182, Sofia, Bulgaria. Association for Computational Linguistics.

[Zagona, 1988] Zagona, K. (1988). Proper Government of Antecedentless VP in English and Spanish. *Natural Language & Linguistic Theory*, 6(1):95–128.