From Aari to Zulu: Massively Multilingual Creation of Language Tools using Interlinear Glossed Text

Ryan Georgi

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Fei Xia, Chair

William D. Lewis

Emily M. Bender

Program Authorized to Offer Degree:
UW Department of Linguistics

University of Washington

Abstract

From Aari to Zulu: Massively Multilingual Creation of Language Tools using Interlinear Glossed Text

Ryan Georgi

Chair of the Supervisory Committee:
Professor Fei Xia
Linguistics

This dissertation examines the suitability of Interlinear Glossed Text (IGT) as a computational, semi-structured resource for creating NLP tools for resource-poor languages, with a focus on the tasks of word alignment, part-of-speech (POS) tagging, and dependency parsing. The creation of a massively multilingual database of IGT instances called the **O**nline **D**atabase of **IN**terlinear text (ODIN) made possible the potential for creating tools to harness this particular data format on a large scale. Xia and Lewis (2007); Lewis and Xia (2008) demonstrated the potential of using IGT instances from ODIN to answer some typological questions such as basic word order for a large number of languages by means of utilizing the language–gloss–translation line structure of IGT instances to bootstrap word alignment, and consequentially syntactic projection. This dissertation seeks to perform a thorough investigation as to the potential for creating these NLP tools for endangered or otherwise resource-poor languages with nothing more than the IGT instances found in ODIN.

After introducing the IGT data type and the particulars of the resources that will be used (Sections 3.1 to 4.4), this thesis presents each task in detail. Word alignment will be discussed in Chapter 5, POS tagging in Chapter 6, and dependency parsing in Chapter 7. In Chapter 8, **IN**terlinear **T**ext **EN**richment **T**oolkit (INTENT), the software created to enrich the IGT data and extract NLP tools from it will be introduced, and a brief summary of

where to find and use the software will be included. Lastly, Chapter 9 will discuss aims of the experiments, and the overall viability of IGT for the tasks attempted.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Fei Xia, for all of her guidance, expertise, and patience with me throughout my graduate career. Her attention to detail instilled a methodological rigor and skepticism in me and my work that contributed to my overall maturity as a scientist. I would also like to thank my co-advisor, Will Lewis, for his input and enthusiasm for my chosen topic of study, which helped me maintain confidence that my work was meaningful, even when the results were disappointing. My final committee member, Emily Bender, was not only instrumental in helping me greatly strengthen my dissertation, and providing guidance through the last several years of the program, but is also largely responsible for there being a program for me to have started in the first place. Thanks also to my Graduate School Representative, Gail Stygall, for giving her time to participate in my defense and celebrate at my commencement.

Thanks also to the Linguistics Department Administrator, Mike Furr, and our Academic Counselor, Joyce Parvi, who helped me navigate the complicated world of funding, travel, and academic bureaucracy. And to our Systems Administrator, David Brodbeck, for running a tight ship, despite the many attempts of us graduate students to run the computing cluster into the ground.

I would also extend my thanks to the National Science Foundation, which provided a significant portion of the funding for my research through Grant No. BCS-0748919, and for recognizing the need for the type of exploratory work on resource-poor languages that this research consisted of.

I would also like to thank my family, who supported me in uncountable ways throughout my graduate career, and enabled me to tackle the greatest challenges by knowing that they

had my back should I fall, and without whom this dissertation surely would not exist. Last but not least, my friends and partners, who supported me when I was down, and made sure I was never without joy and hope when things were at their most difficult.

DEDICATION

To Deanna and Shawn

Who not only gave of themselves to support me as the person I was, but supported my growth throughout the years into the person I was to become.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF IGT INSTANCES

LIST OF CODE SAMPLES

Chapter 1
INTRODUCTION

1.1 Motivation

With the remarkable achievements accomplished by data-driven Natural Language Processing (NLP), much recent work in the field has been done on expanding available sources of data. The approaches have varied from adapting comparable corpora (Hana et al., 2004; Feldman et al., 2006) to tasks typically requiring parallel corpora (Yarowsky and Ngai, 2001; Hwa et al., 2004), to using partial or noisy automatic annotation to train models (Spreyer and Kuhn, 2009), to crowdsourcing using Amazon's Mechanical Turk (Callison-Burch and Dredze, 2010). Expanding the available data can be used for improved performance and domain adaptation on existing tasks, or implementing NLP tasks in new languages entirely.

Curated resources have become available for more languages in recent years, enabling data-driven methods for a variety of new tasks in these languages. Between the LRE Language Map[1] (Calzolari et al., 2012) and the catalog of the Linguistic Data Consortium (LDC)[2], there are resources available for approximately[3] 205 languages. Despite this increase, this is still far short of the 7,000+ living languages on Earth (Lewis, 2009). Furthermore, as Fig. 1.1 shows, a large plurality of these 205 languages have only a single resource available, with the top twelve languages accounting for 73% of the total available resources, but only 29% of the world's population.

---

[1]Language Resources and Evaluation, `http://www.resourcebook.eu/`

[2]https://www.ldc.upenn.edu/

[3]I use "approximately," because in putting together a list of the language resources available in each catalog, the language identifiers are not normalized, and may say "Multiple Languages" or misspellings, so this number is an estimate.

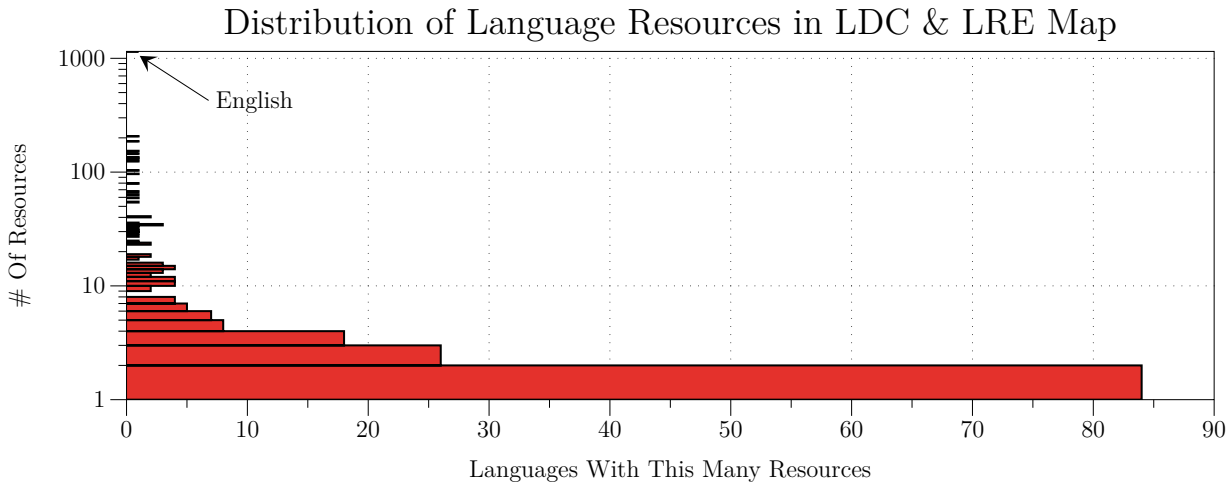


Figure 1.1: Graph showing how many languages have n resources available for them among the LDC and LRE collections, out of 205 total languages.

As a result, solutions are needed for a researcher to turn computational methods toward a language that is not among those 205 with curated resources. If the language in question has raw, unannotated language data available in large quantities, that may make unsupervised machine learning methods possible. If the researcher has access to enough resources to spend the time or money funding a language expert to create a small amount of annotation, they may attempt to constrain an unsupervised induction method with this expert knowledge. As discussed in Bender (2011), there is often a scarcity of funding, resources, or interest in the form of shared tasks to languages not among the 20 most studied languages. This makes it very difficult for researchers to find the time or money to fund an extensive annotation task aimed at creating resources for data-driven tools. While unsupervised and semi-supervised methods can be successful, they often require large amounts of unannotated data which is difficult to find for infrequently studied languages.

It is this problem that this thesis seeks to address, through the use of a data source in common usage, but with few computational applications. That data source is Interlinear

Glossed Text, or IGT, an example of which is shown in Instance 1.1. IGT instances have been in widespread use in the field of linguistics for years, and as such are found in a wide selection of publications covering phenomena for thousands of languages. In Xia and Lewis (2007), the authors demonstrated that a unique benefit of this data type was the ability to leverage the gloss to align the source language with resource-rich English. Using available tools for English, the English could be processed and this resulting annotation "projected" to the resource-poor language. The work of Lewis and Xia (2010) in developing ODIN, the **O**nline **D**atabase of **IN**terlinear text, has made IGT instances readily available for over a thousand languages. Both Xia and Lewis (2007) and Bender et al. (2013) demonstrate the potential for IGT to be used in answering broad typological questions, such as basic word order.

IGT as a resource has limitations, but its availability for a wide selection of languages, and word-level, or sometimes morpheme-level information make it an enticing resource. While this previous work has shown the potential for IGT as a resource in typology, I seek in this work to create an end-to-end system that can produce bootstrapped tools for any of the languages in the ODIN database.

This work outlines the research I have performed on the utility of IGT as a resource in real-world scenarios, focusing on base-level component tasks that can be harnessed individually or in combination to develop more complex NLP systems for resource-poor languages.

```
(21) Mary-ga   hon-o    watas-are-ta.
     Mary-nom  book-acc pass-pass-pst
     'Mary was passed the book.'
```

Instance 1.1: An example of Japanese (`jpn`) IGT found in Baker (1996).

1.2 Research Question

While building high-performance NLP tools for resource-poor languages remains a difficult challenge, IGT is an intriguing resource with which to create a foothold for such languages. Ultimately, this work seeks to answer the following question:

> *Can the existing language knowledge contained in Interlinear Glossed Text be harnessed to perform basic NLP tasks on resource-poor languages in a repeatable and broadly applicable manner?*

These tasks would include part-of-speech (POS) tagging, dependency parsing, and performing word alignment on IGT instances for the purposes of projection and building translation dictionaries.

I will look at the ways these various tasks may be attempted given a variety of data sources that may be available for a given language pair and how access to IGT may improve or make possible the task at hand.

1.3 Outline

In the next chapter (Chapter 2), I will discuss the relevant work that has been done previously on word alignment, part-of-speech (POS) tagging, and dependency parsing as related to the particular domain of resource-poor languages and how they relate to or influence the work of this thesis. Section 3.1 will provide a more detailed description of the IGT data type, and the particular challenges in dealing with it. Section 3.2 will give a more detailed discussion of what the IGT data format that will be used throughout this thesis looks like, and how it will be used. Additionally, I will give an overview of the various components that make up the overall IGT enrichment system of this thesis, and how the different components connect into other parts of the system.

I will next introduce the various data sources that will be used for the experiments that I have carried out during the process of this work in Chapter 4, followed by a discussion of dealing with POS tags, one of the main difficulties in working across a wide variety of languages, and the particular decisions that were made for the languages and corpora presented here. Also pertinent to the specific issues that IGT presents is the cleaning of the ODIN data, which is discussed in Section 4.4.

Next, I will detail the experiments that were performed for each of the main tasks mentioned above: word alignment in Chapter 5, POS tagging in Chapter 6, and dependency parsing in Chapter 7.

After the description of the experiments performed, I will introduce the **IN**terlinear **T**ext **EN**richment **T**oolkit (INTENT), the software package that I have built to perform the IGT enrichment tasks presented here, which is being used at the time of writing to provide enriched data for the ODIN database, starting with the release of version 2.1, and will be freely available for users desiring to perform their own IGT-related research. Chapter 8 will describe the package's main features and usage, as well as where it may be obtained and how to find documentation.

Finally, I will provide a summary of the work that has been done and the work that remains for this topic in Chapter 9.

Chapter 2
LITERATURE REVIEW

The main motivation behind this work was to find a method by which to create NLP tools for resource-poor languages that have little to no data available. Given this aim, I will discuss several broad categories in the literature that have in some way contributed toward this goal. In Section 2.1, I will look at how languages with little or no annotated data but large amounts of unannotated data have been approached in the past. In Sections 2.2 and 2.3, I will discuss ways in which data available for one or more language can be leveraged to be used on another language, either by virtue of being typologically similar in some way, or by other methods. Finally, in Section 2.4, I will discuss attempts to fill in the gaps in resource coverage for resource-poor languages.

2.1 Unsupervised and Semi-supervised Induction Methods

One of the first paths to consider in approaching the problem of resource-poor languages are methods for which little or no annotated data is required. This class of methods relies on analyzing unannotated data for the target language and inducing patterns from the text.

2.1.1 Fully Unsupervised Methods

POS Tag Induction The principle hypothesis in unsupervised POS tag induction is that words in a language can be grouped into natural classes by virtue of the contexts within which they co-occur with other words. In English, for instance, words that follow determiners are typically either a noun or some sort of noun modifier.

One of the earliest studies to propose such a method was Brown et al. (1992), who grouped

Cluster 1	immediate urgent ongoing absolute extraordinary exceptional ideological unprecedented appalling overwhelming alleged automatic [...]
Cluster 2	worried concerned skeptical unhappy uneasy reticent unsure perplexed excited apprehensive legion unconcerned [...]
Cluster 3	cover include involve exclude confuse encompass designate preclude transcend duplicate defy precede [...]
Cluster 4	encourage promote protect defend safeguard restore assist preserve coordinate convince destroy integrate [...]
Cluster 5	china russia iran israel turkey ukraine india japan pakistan georgia serbia europol [...]
Cluster 6	waste water drugs land fish material meat profit alcohol forest blood chemicals [...]

Table 2.1: Examples of clusters discussed in Rishøj and Søgaard (2011), extracted using the Brown clustering algorithm (Brown et al., 1992) over the English section of the Europarl corpus.

words into n-gram based clusters based upon this principle. In the study, the authors use agglomerative hierarchical clustering, with a subsequent maximization step to move words between clusters if it improves the expected probability of the observed corpus. Clark (2003) extended this concept of clustering by incorporating morphological information, a feature useful for languages morphologically richer than English.

Table 2.1 shows six example clusters of the 1,000 constructed in Rishøj and Søgaard (2011) using the Brown algorithm on the Europarl corpus. As the authors point out, these clusters seem to actually be subsets of traditional part-of-speech tags, with clusters 1 & 2 being adjectives, 3 & 4 being transitive verbs, and 5 & 6 being nouns. Furthermore, the nouns in 5 are countries or governmental actors, while the nouns in 6 are tangible objects. This would seem to suggest that while the clusters might not have one-to-one mappings between cluster and POS tag, the hypothesis that word clusters have a relationship to part of speech tags appears supported.

Dependency and Phrase Structure Induction In addition to POS tagging, work has also been done on fully unsupervised grammar induction for both constituent structures (Clark, 2001; Klein and Manning, 2002) and dependency structures (Klein and Manning, 2004; Smith and Eisner, 2005b; Daumé III, 2009). While inducing word classes that resemble POS tags is a task with a large search space, introducing latent structure adds a new range of complexity. These unsupervised grammar induction methods tend to use variations on the inside-outside algorithm (Baker, 1979), a type of Expectation Maximization (EM) algorithm (Dempster et al., 1977) for use with tree structures. The inside-outside algorithm initializes a Probabilistic Context-Free Grammar (PCFG) that can be used to parse the corpus and produce an *expectation* that represents the current grammar's view of how probable the current corpus is. The result of this step is a large space of *partial counts* that represent the probability of each portion of the grammar being observed in some version of the underlying structure. In the *maximization* step, the parameters of the model—for a PCFG, the probabilities for the rules—are recalculated using a Maximum Likelihood Estimation (MLE) of the partial counts.

The result is a hill-climbing algorithm that attempts to find the grammar that maximizes the likelihood of the observed data. While this is an extremely clever solution to learning something as complex as grammar rules from unlabeled corpora, the search space this algorithm needs to traverse is massive, and thus the algorithm usually gets stuck in a local optimum. Daumé III (2009) cites Unlabeled Attachment Score (UAS) results in the 33–45% range for unsupervised dependency parsing, compared to a supervised system at 79–82%.

While unsupervised methods are aimed at finding the best methods to search an complex search space, another path of research is to find ways to constrain the search; this notion of producing "semi-supervised" or "weakly" supervised approaches will be discussed in the following section.

2.1.2 Semi-Supervised Methods

Using Linguistic Constraints While the fully unsupervised methods in Section 2.1.1 show a degree of promise in their ability to work on any language, regardless of knowledge about that language, finding the best global optimum in a search space filled with latent variables is extremely difficult. The semi-supervised class of methods seeks to use some of the same techniques, but constrain the search space.

Semi-Supervised POS Tagging Methods There has been a great deal of work on semi-supervised POS tagging (Smith and Eisner, 2005a; Haghighi and Klein, 2006b; Toutanova and Johnson, 2007; Mann and McCallum, 2008; Graça et al., 2011). Two of these techniques include determining what a "prototypical" word for a POS tag is, so as to assign probability mass to that POS tag for contexts that the prototype appears in; and incorporating partial dictionaries and utilizing the intuition that though a word may be ambiguous, its distribution over possible tags is usually sparse (Toutanova and Johnson, 2007). All of these approaches share the same goal of reducing or otherwise constraining the search space for POS induction.

In Haghighi and Klein (2006b), the authors use a list of words that are prototypical of a given POS tag as a way to constrain the forward-backward sequence labeling algorithm (Rabiner, 1989) to only consider model parameters for which the prototypes are given their specified POS tag. In the study, the unsupervised forward-backward system achieves 42.2% POS tagging accuracy using the Penn Treebank tagset and sentences of length 10 or shorter from the first 48K tokens of the Wall Street Journal (WSJ) section of the treebank (Marcus et al., 1993). Using only this constraint, the authors report that a list of prototypes automatically extracted from the WSJ corpus and designed to be unambiguous achieves a 61.9% tagging accuracy. In contrast to the fully unsupervised approach, where the forward-backward algorithm is initialized randomly and unconstrained, the prototype constraints serve to guide the model to a more global optimum.

In another experiment, Haghighi and Klein expand the set of prototypes using distributional similarity features, and rerun the experiment with a much larger group of prototypes. This expansion increases POS tag accuracy from the 61.9% using the original prototypes to 79.1% using the distributional similarity features. While this methodology seems promising, it should be noted that the choice of prototypes is extremely important, as the authors note that "tuning [of the prototype list] could greatly reduce the error rate," and that the automatic extraction method produces best-case results.

An alternate take on constraining the POS tagging search space is the approach taken by Toutanova and Johnson (2007). In this work, the authors use a dictionary that lists words and their POS tags; however, rather than condition upon the POS tags directly, the system models the *ambiguity class*, or the set of possible tags for a word. For instance, activity-type words like 'run,' 'walk,' or 'feed' that may have both a verbal meaning and a nominal meaning might form the ambiguity class of {VERB, NOUN}. This allows for the model to allocate probability mass for a particular ambiguous word without affecting the best overall sequence for the sentence.

With a dictionary created from words and their ambiguity classes that appear 3 or more times in the corpus, this approach achieves 89.7% accuracy. Although the authors acknowledge that building a dictionary from an annotated source is "arguably unrealistic" (Toutanova et al., 2002, p. 6), modeling these ambiguity classes would appear to allow for somewhat more flexibility in constraining the tagging problem than the prototype solution.

Semi-Supervised Grammar Induction Methods In addition to work done in POS tagging, work on semi-supervised grammar induction is fairly extensive as well (Haghighi and Klein, 2006a; Koo et al., 2008; Naseem et al., 2010; Mirroshandel et al., 2012). Following up on the prototype method for constraining an EM algorithm's search space, Haghighi and Klein (2006a) describe a method for doing a similar prototype-based system for constituent parsing. In this work, prototypes are given as Context Free Grammar (CFG) rules such as

NP → DT NN or VP → VBD DT NN. The prototypes are then used during the inside-outside algorithm such that when the production on the right is encountered, partial counts are discarded for rules that do not match the prototype. Using this approach, the authors report a labeled F_1-score of **0.57** on the WSJ-10 corpus.[1] The authors also perform an experiment intersecting the deep-structure PCFG induction algorithm of the inside-outside algorithm with the shallow, bracketing-inducing Constituent-Context Model (CCM) of Klein and Manning (2002). Using the constraints for the inside-outside algorithm given above, Haghighi and Klein use the matrix of possible bracketings and their likelihood scores and multiply those scores with the partial count matrix of the inside-outside algorithm, to reinforce the likelihood that the structure being induced in the PCFG is also a likely constituent as determined by the CCM algorithm. Combining these two models and using the constraints and distributional features of the prototypes to expand the prototype list ultimately produces a labeled F_1-score of **0.65**.

Haghighi and Klein (2006a) inspired the work from my master's thesis (Georgi, 2009), where I extracted prototype CFG rules from IGT instances to constrain an inside-outside algorithm in a similar manner. The results of this research on the German-language NEGRA corpus (Brants et al., 2003) showed only 0.27 labeled F_1-score when the terminals and nonterminals were mapped to a common reduced tagset. Compared to the uninformed EM algorithm, which produces labels that are essentially random, this 0.27 is still better than the uninformed system's F_1-score of 0.007. When the labels were remapped many-to-one to the label that greedily optimized the score, this brought the scores to 0.45 and 0.41 respectively. Despite this improvement, requiring such remapping is precisely what semi-supervised methods seek to avoid. Despite the promise of this prototype-driven method, it seems that the prototypes must be chosen to be extremely unambiguous, and the noise that projection with IGT introduces is likely too great for this approach to work well.

[1]The Wall Street Journal section of the Penn Treebank, consisting only of sentences of length ten words or fewer.

Semi-Supervised Word Alignment Unsupervised word alignment using parallel corpora has been common practice in machine translation for decades, whether using the IBM Models (Brown et al., 1993), or an HMM model for word alignment (Vogel et al., 1996). When looking to perform word alignment for resource-poor languages, which do not have substantial amounts of parallel text, some modifications to the typical approach are needed.

One such approach to adapt such statistical approaches to resource-poor languages is proposed in Fraser and Marcu (2006), where word alignment present in a supervised set of alignments is used as "gold" data. In the approach taken by Fraser and Marcu (2006, 2007), the IBM Model 4 parameters of translation probability, fertility, and distortion are cast as sub-models h_m with weights λ_m of a larger log-linear model. In a modified EM algorithm, the sub-model parameters θ_m are estimated and used to produce a vector of sub-model weights λ. To augment the M step, a discriminative re-ranker is used to choose the sub-model weights with the minimum error according to the training data, and the process is repeated until convergence.

While I will discuss how IGT may be used to obtain high-precision alignments in Section 3.1.3 and Section 5.2, the difficulty in using this approach is that it requires that a small gold-standard corpus be available for each language desired, something that would require specific language knowledge to produce. Although such resources may require substantial human effort to produce, the experiments on word alignment I present here could be used as a preliminary step to generate a small, relatively high-quality corpus that could then be corrected by annotators, thus greatly reducing the effort required to produce such a resource.

2.2 Adapting Resources by Leveraging Typological Similarities

Utilizing Morphosyntactic Similarities One approach to dealing with a resource-poor language is to find a typologically or genetically related language with comparatively more resources, and leverage the similarity between the languages to use the resource-rich language

of the pair as a guide for the related, but resource-poor language. This is the type of approach taken by Hana et al. (2004) and Feldman et al. (2006) for doing morphological analysis of resource-poor Russian using resource-rich Czech and Polish. In these studies, the morphosyntactic features of the lexical items and their word order are the basis upon which Russian, Czech, and Polish all share some degree of similarity. Despite the large amount of monolingual text in each of these languages, the authors note that parallel text is difficult to come by for many Slavic languages, and so the work focused on exploiting the typological similarities rather than projection-based approaches such as those in Section 2.3.

These studies adapt the morphological information between these languages by using the TnT tagger (Brants, 2000) and using surrogate data for estimating the transition and emission probabilities. The morpheme-level POS tags from the resource-rich language are used to approximate the transition probabilities of the related target language, while a statistical morphological analyzer on the resource-poor target language is used to approximate the emission probabilities by bootstrapping the emissions from cognates in the other language. While the transition properties can be transferred between the related languages, the morphological analyzer is needed in order to adapt the tagger to the different lexical items in the target. Between Czech and Russian, Hana et al. (2004) show that their tagger achieves an accuracy of 88%, compared to the commercial Xerox tagger[2] at 82%, showing a great deal of promise for this approach. Another important finding of Feldman et al. (2006) is that combining Polish and Czech training data for the Russian tagger performs better than either language alone. The authors hypothesize that the two languages may have complementary similarities, such that combining the training data ultimately produces a more Russian-like pseudo-language.

Another instance of adapting resources from related languages is described in Snyder and Barzilay (2008), where Hebrew, Arabic, Aramaic, and English are analyzed in par-

[2] `http://www.xrce.xerox.com/competencies/content-analysis/demos/russian`

		Source Language								
		Danish	German	Greek	English	Spanish	Italian	Dutch	Portuguese	Swedish
Target Language	Danish	**79.2**	45.2	44.0	45.9	45.0	48.6	46.1	48.1	47.8
	German	34.3	**83.9**	53.2	47.2	45.8	53.4	55.8	55.5	46.2
	Greek	33.3	52.5	**77.5**	63.9	41.6	59.3	57.3	58.6	47.5
	English	34.4	37.9	45.7	**82.5**	28.5	38.6	43.7	42.3	43.7
	Spanish	38.1	49.4	57.3	53.3	**79.7**	68.4	51.2	66.7	41.4
	Italian	44.8	57.7	66.8	57.7	64.7	**79.3**	57.6	69.1	50.9
	Dutch	38.7	43.7	62.1	60.8	40.9	50.4	**73.6**	58.5	44.2
	Portuguese	42.5	52.0	66.6	69.2	68.5	74.7	67.1	**84.6**	52.1
	Swedish	44.5	57.0	57.8	58.3	46.3	53.4	54.5	66.8	**84.8**

Table 2.2: Unlabeled Attachment Score (UAS) results of transfer parsing from McDonald et al. (2011). The bold numbers are results for the parser being trained and parsed on itself. The underlined numbers are the best results for a source language other than the target itself.

allel to induce morphological segmentation. Using a corpus of short parallel phrases, the authors model potential segmentations and alignments between morphemes among the parallel phrases, assuming morphemes may be aligned to zero or one other morpheme. Arabic, Hebrew, and Aramaic are Semitic languages, known for highly productive morphology (Bravmann, 1977). While English is not a related language, it is an isolating one (Sapir, 1921). Snyder and Barzilay (2008) notes that, although English's lack of morphological ambiguity helps in this parallel induction, when character-to-phonetic correspondences are added from a related language, the related languages perform significantly better. Furthermore, in all but the Aramaic→Hebrew pair, the inclusion of the other languages in the analysis outperforms the state-of-the-art monolingual morpheme induction system Morfessor Creutz and Lagus (2007).

Utilizing Delexicalized Syntactic Information Moving on from the morphological analysis case, another interesting approach to resource adaptation is the "transfer parsing" approaches of Zeman (2008) and McDonald et al. (2011, 2013). In these approaches, the

authors use language pairs for which words have been replaced with part-of-speech tags either from a gold standard or by a tagger to hide the lexical features specific to a given source or target language from the parser, thus "delexicalizing" it. In order to ensure compatibility between tags, these papers use some form of mapped POS tags, either an "Interlingua-like" tagset (Zeman, 2008) or the Universal POS tagset (Petrov et al., 2012). Täckström et al. (2013) take the "delexicalization" a step farther and replace the mapped POS tags entirely using induced word clusters.

Whatever the method for grouping words into classes, the now-delexicalized parser is agnostic as to which language it is actually looking at, since the only tokens presented to it are universal POS tags that can be mapped between languages. This allows for the parser to be trained on dependency structures in one language, and parse POS-tag sequences into dependency structures in another language.

Given that these transfer-parser approaches are agnostic with regards to specific lexical items of the language, the languages that the parsers "transfer" between need not be related. There is, however, an expectation that syntactic information expressed in the form of basic word order is still encoded in the stream of POS tags. While unrelated languages may still see good results in such transfer parsing methods, the transfer between the source and target language may be unpredictable when the languages are syntactically divergent.

Putting this intuition to the test, the cross-linguistic results mapping source and target languages from McDonald et al. (2011) are shown in Table 2.2, where it can be seen that the Romance languages do tend to strongly prefer other Romance languages. Interestingly, however, Portuguese performs better as a source language parsing Swedish than either German, Dutch, or English, other Germanic languages. Similarly, Greek performs better as a source language for Dutch than the other Germanic languages do as sources.

McDonald et al. (2011) make two particular adjustments to the direct transfer method of Zeman (2008); namely, adding an alignment constraint similar to that use for projection in

Hwa et al. (2005) to guide parallel sentences, but also a system the authors call "Multi-Source Transfer" where training data from all available languages is concatenated and then used as training data. Despite the naïve nature of this approach, the authors find little detriment to some languages and modest improvements to others. This finding makes the case for using potentially unrelated languages to transfer linguistic information from, which I will discuss in the following section.

2.3 Adapting Resources by Other Methods

Bilingual Parsing One approach to leveraging information from a language pair, rather than a single language, can be to use methods that ingest parallel data instead of monolingual data. Recent work on this concept includes Inverse Transduction Grammars (ITGs), as introduced in Wu (1997), as well as Multitext Grammars (Melamed, 2003). Both of these approaches are extensions of a far older concept of parallel parsing algorithms often called *synchronous* grammars or *syntax directed translation schemata* (Lewis and Stearns, 1968; Aho and Ullman, 1969).

While such approaches are largely focused on machine translation and many studies have been geared towards optimizing machine translation results using such parallel grammars (Och and Ney, 2003a; Galley et al., 2006; Chiang, 2007), Wu states that a trained ITG that is provided with fully parsed trees for one language can then be adapted to transfer "grammatical expertise in one language toward bootstrapping grammar acquisition in another" (399). That is, if a transduction grammar is trained for the language pair and a high-quality parse is available for one language, it is easy to guide the parse of the second language in the pair by using the transduction grammar model to constrain the parse.

While synchronous grammar approaches such as these may indeed be helpful in leveraging language resources in the manner described above, these grammars have the requirement of needing at least some labeled training data for both languages in order to build the initial

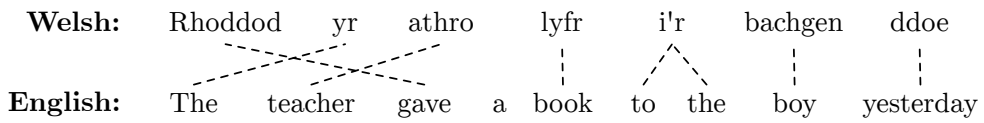


Figure 2.1: A parallel Welsh–English sentence, with word alignments.

model.

Projection-Based Transfer A particularly active area of research in transferring annotation between languages is what I will collectively call "projection" methods, which function by obtaining word alignment between two or more languages and "projecting" information from one language from word to word along these alignments. The concept as I will discuss here was introduced in Yarowsky and Ngai (2001), although syntactic transfer had been explored previously, such as in Twisted Pair Grammar (Jones and Havrilla, 1998).

The basic notion behind projection with parallel corpora is that given a parallel corpus consisting of a resource-rich language and a resource-poor language, the resource-rich language either has annotations available, or can have them generated with high-quality NLP tools. This annotation can then be mapped between languages by means of word alignment. This approach relies on the fundamental assumption that basic syntactic concepts, such as POS tags, can be mapped between languages. This assumption is something that Hwa et al. (2002) describe as the Direct Correspondence Assumption (DCA), and as will be discussed later, this assumption does not always hold up. Proceeding with this assumption, however, one might take a parallel sentence such as that in Fig. 2.1, and start by finding the word alignment. Once the word alignment has been obtained, annotation is generated on the resource-rich language, as shown in Fig. 2.2. Having obtained the POS tags for the English, it is now a simple matter of following the word alignments and "copying" the POS tags from the English to the aligned Welsh sentence, as shown in Fig. 2.3. Yarowsky and Ngai (2001) use projection to project POS tags in this way as well as NP bracketers, while Hwa et al.

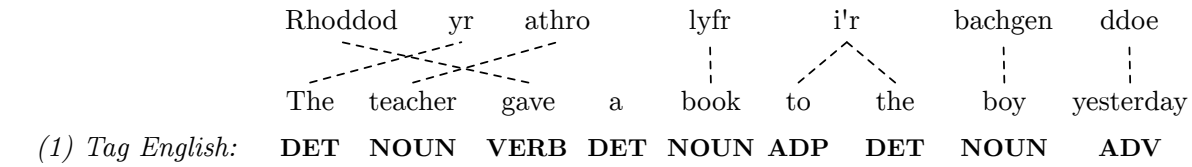


Figure 2.2: Sentence from Fig. 2.1, with English Sentence POS Tagged.

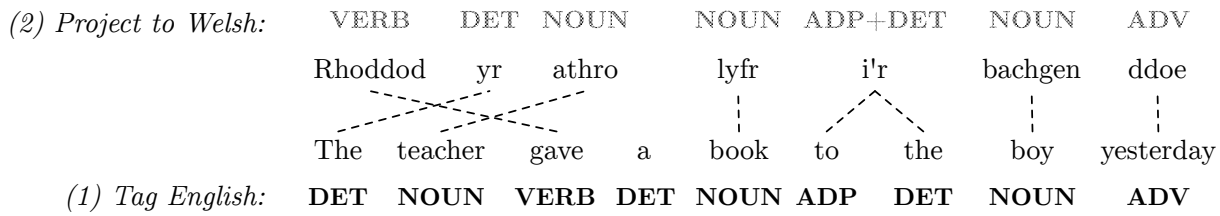


Figure 2.3: Sentence from Fig. 2.1, showing how POS tags would project using provided alignment.

(2005) uses projection to bootstrap parsers.

Already, Fig. 2.3 shows some of the potential issues that the DCA can raise, as the multiple alignment of the English words *to* and *the* with the single Welsh word *i'r* means that there is a potential conflict between how English and Welsh represent this dative construction. While English uses the preposition *to* to mark the indirect object and the determiner *the* to mark the definiteness, it appears that Welsh is doing both in the single word *i'r*. This results in a conflict in assigning the POS tag that I will address methods of handling later, in Chapter 6.

While part-of-speech tags are one shallow form of annotation that may be projected, dependency structures are another, deeper form that will be discussed in this thesis.

2.4 Creating Resources for Resource-Poor Languages

Finally, and tautologically, perhaps the most straightforward path to tackling resource-poor languages is to create or discover resources for them. In this area of research, there are

a few interesting projects that have a particular focus in broad coverage of resource-poor languages.

Indigenous Tweets Indigenous Tweets[3] (Scannell, 2014) is a directory of Twitter users that tend to tweet in various minority languages of the world. Due to the nature of social media content in general, and microblog content in particular, this data tends to be noisy and difficult to work with (Derczynski et al., 2013). Furthermore, the tweets are not filtered by language, so this resource is not a homogeneous data source. That said, as discussed in Nilsson (2015), the very multilinguality and code-switching that make it a poor corpus for monolingual NLP methods makes it an exceptional source for sociological and sociolinguistic studies of minority language speakers.

The Crúbadán Project The Crúbadán Project[4] (Scannell, 2007) is another project from the same researcher behind Indigenous Tweets. At the time of writing, the database contained entries for 2,124 languages, and provides ISO-639-3 code (SIL International, 2006), countries in which the language is spoken, and script type for each of the languages. Included for each language is a list of URLs for pages that contain the given language, a word list, character trigrams, and word bigrams.

Such resources allow for the building of tools for resource-poor languages, such as grammar checkers (Scannell, 2016), language identifiers (Lui et al., 2014), and others. This resource appears to be very well-curated, and, as far as I can tell, underutilized.

The Online Database of INterlinear text (ODIN) The principal works of relevance in this thesis are those of Lewis (2006) and Lewis and Xia (2010), which describe the development of the **O**nline **D**atabase of **IN**terlinear text (ODIN). ODIN was created in a multi-step

[3] `http://indigenoustweets.com/`

[4] `http://crubadan.org/`

process. First, a meta-crawling approach was used, creating queries based upon phenomena often discussed in IGT instances (such as *ERG* or *ABS*) and directing these queries to existing search engines. After scraping the results of each search, a classifier for detecting IGT instances within a document was then trained, and the IGT instances isolated. Once isolated, language identifiers were created for the instances, at first by leveraging the fact that the language name for the instance was typically given by the author of the document in which it was embedded. With a sufficient number of examples for each language thus identified, language models could then be built using the previously identified instances for the identification of future instances.

The resulting ODIN-2.1[5] dataset used for the experiments in this thesis consists of 151,633 instances covering 1,487 languages. More statistics on the breakdown of the ODIN data can be found in Section 4.1.1.

With ODIN created, Xia and Lewis (2007) describes some preliminary ways in which this resource can be leveraged to create NLP tools for the languages contained in the database via projection similar to Hwa et al. (2005), but with the added benefit of the high-quality alignment provided by the gloss line contained in IGT. The methods I used in this work for alignment and projection largely follow this study, and as such are explained further in Chapter 5 and Section 6.2.

Moving the scope of research outward from generating resources for particular languages, Lewis and Xia (2008) further investigates the possibility of using ODIN as a resource to answer broad typological questions. Lewis and Xia (2008) sought to predict basic word order parameters (Greenberg, 1963) using the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013) as a gold standard, and were able to achieve 99% accuracy with 40 or more IGT instances, and 79% accuracy for between 10–39 instances. This shows that IGT holds promise as a computational resource not only for language-specific queries,

[5] http://depts.washington.edu/uwcl/odin/

but also for answering broader typological questions.

2.5 Summary

I have discussed a great number of methods which may be used to approach languages for which annotated data is scarce, but many of these approaches have limitations. When it comes to unsupervised approaches, the amount of data required to achieve the published results is typically on the order of millions of sentences. For instance, while the projection-based approach of Das and Petrov (2011) sounds extremely promising, in that POS tagging accuracy figures between 79.5% and 87.9% are reported over 8 languages, with only linguistic knowledge of English required. These languages are all European, however, and the data requirements are in the millions of sentences, something that will not be achievable for many resource-poor languages.

Bilingual parsing methods using Inverse Transduction Grammars as described in (Wu, 1997, p. 399) also sound promising, but require gold-standard parse trees for training before parsing on unseen sentences can be performed, and also require large parallel corpora.

The semi-supervised approaches too, hold promise, but in Haghighi and Klein (2006b) and Toutanova and Johnson (2007), the authors noted that the selection of the supervision can greatly affect the results. In these two studies, the authors used corpus-driven approaches to select the prototypes and tagging dictionaries, respectively. While this produced good results on these corpora, how will such approaches function when the available supervision is far sparser or noisier, as it is with IGT?

In this thesis, I will make the case for working with data that does exist for resource-poor languages, namely Interlinear Glossed Text. The ODIN database provides broad, semi-structured data for over a thousand languages, and if this data can be properly harnessed, it may open the door to hundreds of previously intractable languages. In the following chapters, I will describe the ways in which the unique data source that is IGT can be used to form a

foundation upon which massively multilingual tools might be built.

Chapter 3
METHODOLOGY OVERVIEW

Having discussed previous attempts to address the creation of language tools for resource-poor languages, I will now outline the methods that I will be pursuing in this thesis. Following on the promise of Interlinear Glossed Text (IGT) demonstrated by Xia and Lewis (2007) and Lewis and Xia (2009) as a computational resource with broad coverage, this work concentrates on expanding upon different methods of exploiting IGT to bootstrap NLP tools.

Though such flexibility exists, and the format is not formally defined, the Leipzig Glossing Rules (Bickel et al., 2008) provide a set of guidelines with which

I will begin by giving a more detailed introduction to IGT as a data type (Section 3.1), before giving an overview of the IGT enrichment system (Section 3.2) and its individual components: the word alignment module (Section 3.2.1), the POS tagging module (Section 3.2.2) and the dependency parsing module (Section 3.2.3).

3.1 Interlinear Glossed Text

Interlinear Glossed Text, or IGT, is a simple way of presenting linguistic examples. IGTs are used to present information about a language, such as morphology, syntax, or lexical distinctions to readers who may not have specific knowledge about the language at hand. Because the examples are used to illustrate particular purposes, the author will often present a varying amount of information, depending on what he or she feels is needed to illustrate the particular linguistic phenomenon or phenomena at hand. Though there is no formal specification for the IGT format, the Leipzig Glossing Rules (Bickel et al., 2008) is a set of guidelines that aim to standardize the format as much as possible.

The typical instance of IGT includes a **language** line, a **gloss** line, and a **translation** line, and represents a single sentence, as in Instance 3.1, which shows a simple IGT instance for the German sentence *Peter erzählt den Kindern eine Geschichte* ("Peter tells the story to the children").

Although examples like 3.1 are the norm, IGT instances also occur in other forms, such as that in Instance 3.2, where a single IGT instance is used to represent two or more similar sentences in the same example. Other variation includes instances with additional lines of metadata, such as alternative translations, citations, or description of linguistic phenomena. While these other forms are not uncommon, I will focus on the canonical form shown in Instance 3.1. I will discuss these alternate forms in more depth in Section 4.4.

3.1.1 Using IGT for Alignment

One of the most immediately noticeable aspects of the IGT data format is the gloss line, and in particular, the English translations provided by the gloss. Given that there is often an unambiguous repetition of the same English word on both gloss line and translation line, this can be used to assume alignment between the gloss and translation line tokens. Furthermore, since the gloss tokens and the language-line tokens are aligned in a one-to-one, monotonic manner, the gloss line can typically be used as a pivot to align the translation line with the language line.

Instance 3.3 illustrates this alignment for the sentence from Instance 3.1. Just as we can

`Peter`	`erzählt`	`den`	`Kindern`	`eine`	`Geschichte`	**Language**
`Peter`	`tells`	`the:DAT`	`children:DAT`	`an:ACC`	`story:ACC`	**Gloss**
`‘‘Peter tells a story to the children.’’`						**Translation**

Instance 3.1: IGT with German (`deu`) on the language line and English as the translation. (Nakamura, 1997).

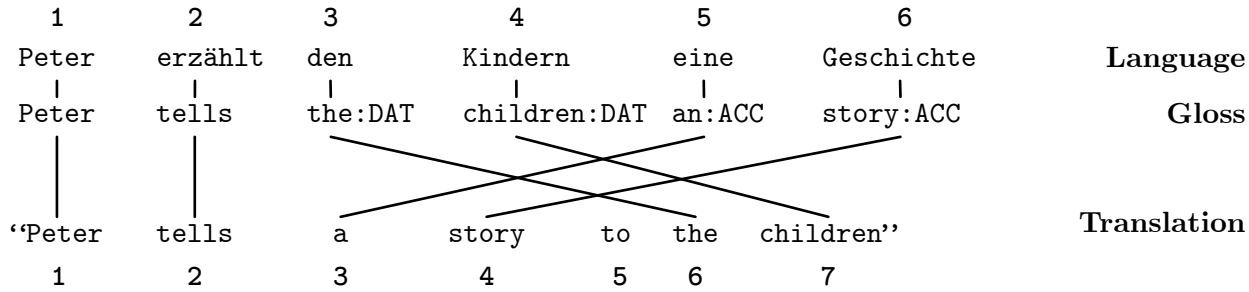
```
János-nak/Neki    győz-ni/győz-ni-e    kell.
John-dat/he.dat   win-INF/win-INF-1PL  must
‘It is necessary for John/him to win.’
```

Instance 3.2: A Hungarian (hun) IGT instance from Laczkó (2002) that combines two examples into one.

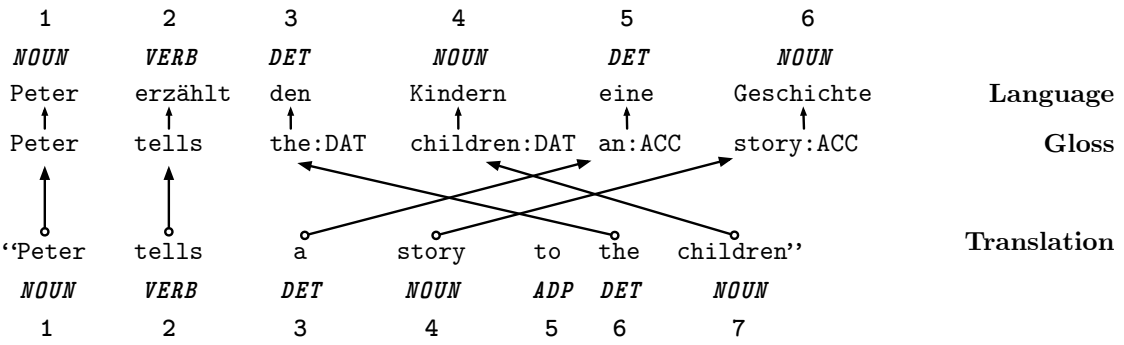
align the translation and gloss tokens children[7] ⟷ children:DAT[4], we can assume that the fourth gloss token aligns with the fourth language token and thus transitively align children[7] ⟷ Kindern[4].

3.1.2 Using IGT for Projection

In the alignment example in Instance 3.3, it happens that the alignment is felicitous, and can be used to make assumptions about the German based on what is known about the English, namely that *children* is a noun, and in this case is the indirect object of the sentence. Using word-to-word alignment to make this kind of correspondence is the basis of **projection** methods, such as those used by Yarowsky and Ngai (2001); Hwa et al. (2004); McDonald et al. (2005); Xia and Lewis (2007) and Das and Petrov (2011). Though I will discuss how I use projection in more detail in Section 6.2 and Section 7.2, Instance 3.4 shows a simple



Instance 3.3: IGT Instance from Instance 3.1, but this time showing the alignment information provided by the gloss.



Instance 3.4: IGT instance demonstrating how POS tags can be projected from the translation line to the language line by way of the the gloss line.

illustration of using the alignments to project part-of-speech tags from the translation line to the language line.

3.1.3 The Gloss Line As a “Pseudo-Language”

Since English translations are typically found both on the gloss line and translation line of IGT instances, these words can be a useful aid in aligning the translation line with the words in the language line with which the gloss words match. Besides the English words, however, the gloss line also contains additional information in the form of **grams**[1]. One such example would be the gram `3SG`, indicating that the word contains some form of inflection for third person and singular number. The following Instances 3.5 to 3.8 contain instances of Oriya, a language spoken in southern and eastern India; Turkish; Yukaghir, an endangered language of Eastern Russia with a population of 70–80 speakers (Hammarström et al., 2016); and |Gwi, a threatened language of Botswana from the Khoisan family consisting of roughly 2–4,000 speakers (Lewis, 2009).

Despite such wide variation, all three instances contain English words, and familiar gram-

[1]My usage follows that of Bybee and Dahl (1989), where a “gram” refers to a token of gloss-line annotation used to mark a grammatical feature of the token, such as inflection or case.

```
0   da    zo-ro      gɛ-rɛ     wuo-ro         la    haane.
3sg PAST  run-IMPERF go-IMPERF collect-IMPERF FACT  berries
He/she was always running there collecting berries.
```

Instance 3.5: Instance of Oriya (ori) from Beermann and Hellan (2002).

```
Para-yi    kim      čal-di-ø?
money-ACC  who-NOM  steal-PAST-3SG
‘Who stole the money?’
```

Instance 3.6: Instance of Turkish (tur) from Zwart (2002).

```
taːt joːnro-lu-ge     tudel   pon’oː-l’el.
so   forget-1PL-CONV  he      remain-3SG
‘Since we forgot him, he remained alone.’
```

Instance 3.7: Instance of Yukaghir (ykg) from Nedjalkov (1998).

```
Cire       !koõ          Da         !koõ
1.SG.NOM   go            1.SG.IMP   go
‘I go.’                  ‘Let me go.’
```

Instance 3.8: Instances of |Gwi (gwj) from Nordlinger and Sadler (2000) showing how an imperative mood for a clause is encoded on a pronoun.

matical indicators. As a result, the gloss line makes an appealing target for building a general system that can analyze any IGT instance, based on previously seen gloss lines. This is possible, with a few caveats. For one, as the IGT instances in ODIN cover over a thousand languages, and the gloss line tokens mirror the word order of the language they annotate, word order will vary from language to language. As such, when building such a general system, context between tokens should not be relied upon. Second, though grams may indicate concepts such as person or number as verbal inflection in one language, in other languages other word classes may exhibit these grammatical features. For instance, in Instance 3.8, the authors illustrate what appears to be a case of the imperative mood being encoded on a pronoun.

In Instances 3.5 to 3.7, the variation of the grams `3SG`, `NOM`, and `IMPERF`, could be extremely useful in answering interesting typological questions such as: how different types of agreement are expressed, what case system (Bender et al., 2013, 2014) or gender systems might be used on nouns, or how tense and mood are expressed on verbs. Because of this variation, however, training a system to analyze glosses based on knowledge from one language may not always map correctly to other languages.

While my work does not focus on these more complex associations between the grams and the words they modify, it does take advantage of these grammatical markers as clues to the part-of-speech tag of the language-line word the gloss token is intended to represent. For instance, the `IMPERF` gram in Instance 3.5 or the `PAST` grams in Instance 3.5 or Instance 3.6 are likely to be strong indicators of a `VERB` POS tag. However, despite this strong positive correlation, grams like `3SG` can indicate the inflection of either a `VERB` (as in Instances 3.6 and 3.7), a `NOUN` (Instance 3.5) or a `PRON` (Instance 3.8).

Through all of this linguistic analysis, in a format meant to be relatively standardized, the IGT gloss line can be thought of a "pseudo-language" in that it uses a common vocabulary, mixing English words and grammatical markers, though with a word order that changes

according to the language it annotates. This unique aspect of IGT is something that I will explore further in Sections 5.3.2 and 6.3.

3.2 System Overview

Having described what IGT is, and some of its unique features, I will now provide an overview of the system that I have created for the purpose of using IGT's unique features to automatically enrich these examples. I call this system **IN**terlinear **T**ext **EN**richment **T**oolkit, or INTENT. The flowchart in Fig. 3.1 gives a very high-level overview of the modules in the system.

The overview in Fig. 3.1 first draws a distinction between "Raw" IGT and "Enriched" IGT. "Raw" IGT in this setting is the plain text IGT instances, whether from ODIN or another source. The "enriched" IGT are IGT instances with added POS tags, translation-

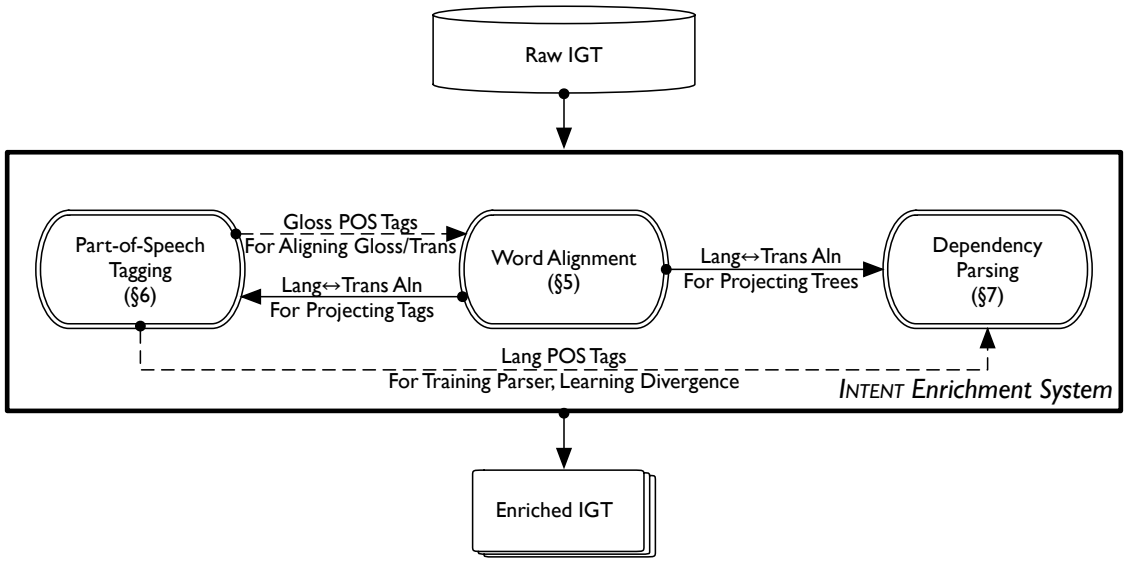


Figure 3.1: High-level overview of the end-to-end INTENT system which will be discussed in Chapter 8. INTENT enriches raw IGT with POS tags, word alignments, and Dependency Structures. Required inputs are shown with solid lines, while optional inputs are shown with dotted lines.

to-gloss alignments, or dependency structures.

Figure 3.1 illustrates the three main modules of the system: the POS tagging module (Chapter 6), the word alignment module (Chapter 5) and the dependency parsing module (Chapter 7). Each module may have interactions with other modules depending upon the settings of the module. For instance, the solid arrow between word alignment and dependency parsing indicates that the IGT-based word alignment module is required for the dependency parsing module, as the dependency parsing experiments require some form of word alignment to project the structures between the translation line and the target language. In the next several sections, I will give a brief overview of the system and the settings for each of the component modules.

3.2.1 Word Alignment

Figure 3.2 gives an overview of the different methods for aligning the IGT data as will be presented in Chapter 5. I describe a method for aligning the language and gloss lines of

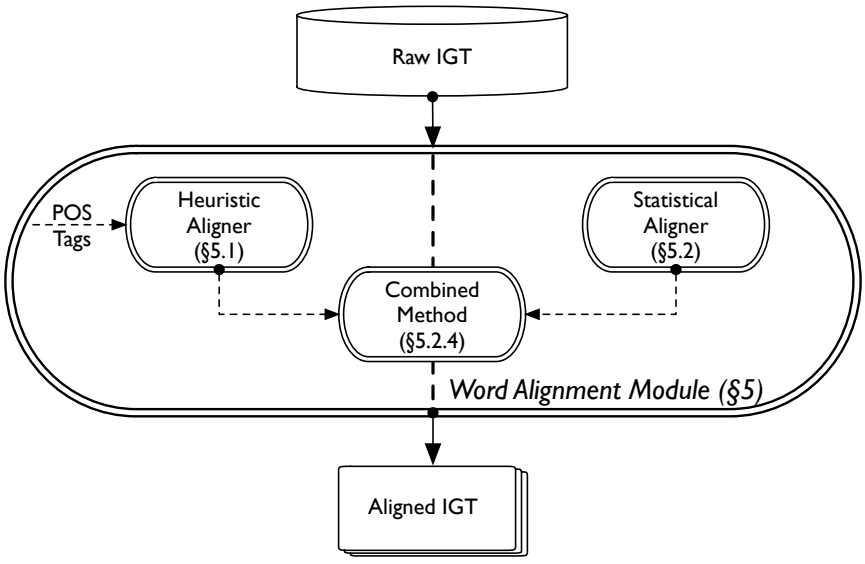


Figure 3.2: Overview of the word alignment module and its different settings.

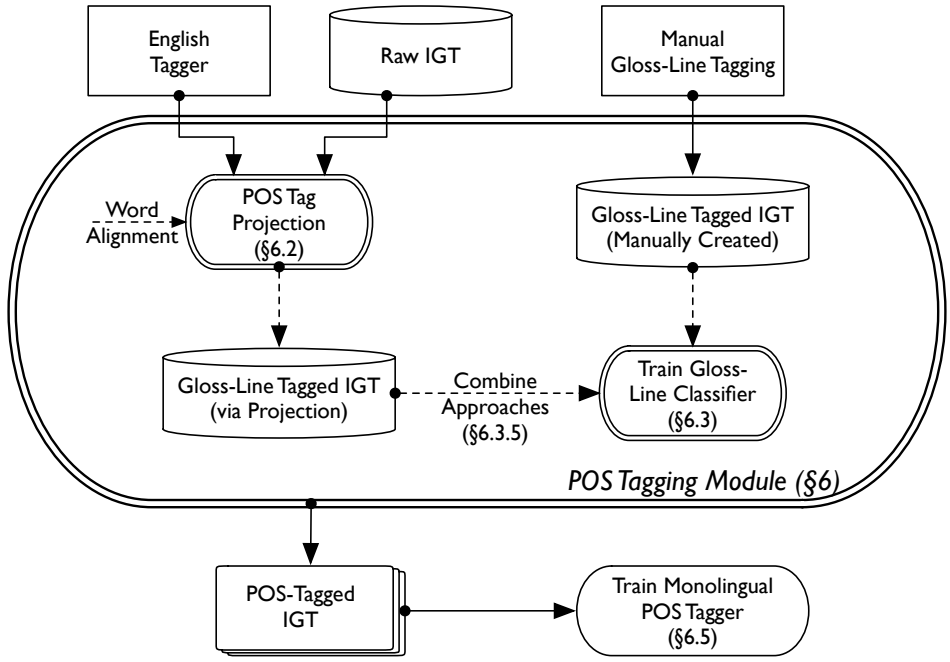


Figure 3.3: Overview of the POS tagging module and its different settings.

IGT instances using various different matching heuristics (Section 5.2), and compare the performance of this method against statistical approaches (Section 5.3). In addition to these approaches, I discuss how elements that have been heuristically matched can be combined with a statistical approach on the IGT instances to create a combined alignment approach (Section 5.4). The end result of this module is a set of IGT instances that have alignment either between the language and gloss lines, with a 1-to-1 alignment assumed between gloss and language lines, or instances with language and translation lines aligned directly.

3.2.2 POS Tagging

The POS tagging module (Fig. 3.3) also consists of three potential approaches for training POS taggers from IGT sources. The first is POS tag projection, which is described in Section 6.2. This requires Raw IGT, an English POS tagger, and IGT word alignment

(from the word alignment module). The second method is a classification-based approach (Section 6.3), which requires a training corpus of manually POS-tagged gloss line data. The third approach combines both projection and classification by using the projection approach to project POS tags to the gloss line, and using those projected labels to train the classifier (Section 6.4). Whether projected or labeled directly by means of a classifier, the POS tags for these latter two approaches may be only applied to a subset of IGT instances and languages, as this approach leverages the "pseudo-language" qualities of the gloss line to apply gloss line POS tags to IGT instances regardless of the target language. These gloss line POS tags can then use the one-to-one alignment with the language line to label the language line tokens with the appropriate POS tags.

Once the language line has POS tags, those target language sentences can be used as a source of supervision to train a POS tagger that can be used directly on the target language. Those experiments are described in more depth in Section 6.6.

3.2.3 Dependency Parsing

Finally, the flowchart in Fig. 3.4 illustrates an overview of the experimental settings for dealing with the topic of dependency parsing. Like POS tagging, the basic setting for obtaining DS parses for IGT data involves parsing the translation line, aligning the instance, and projecting the structure to the language line, which is covered in Section 7.2.

Unlike the POS tagging module, the dependency parsing module includes a series of processes aimed at comparing the DS trees used in projection with another set of trees to learn how the two sets of annotation diverge, and improve the results. There are two approaches for this divergence-learning task: the first system uses manually corrected trees alongside the projected DS trees and, in combination with a statistical DS parser, uses the projected trees as features in the parser training process. This system may then be run on new IGT data of the same language, and the projection is weighted against other parser

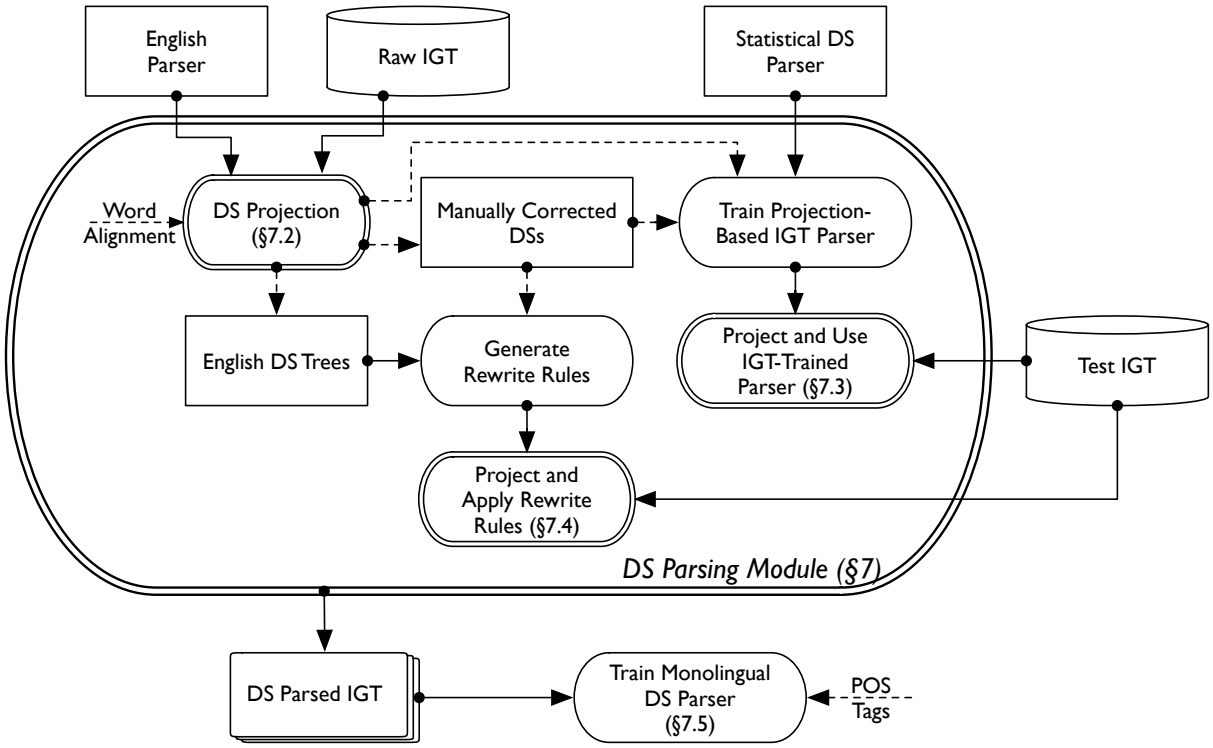


Figure 3.4: Overview of the dependency parsing module and its different settings.

features to produce better parses than projection alone (Section 7.3). The second system compares the English DS with the manually corrected IGT DSs to find divergence patterns, and then applies these rewrite rules to subsequent projections (Section 7.4).

While any of the dependency parsing methods above ultimately produce IGT instances with target language DSs, only the projection-based approach does not require additional manual intervention, and so it is the most broadly applicable. For that setting, Section 7.5 tests training monolingual DS parsers using different combinations of alignment methods and POS tag sources.

3.3 Summary

In this section, I have laid out a number of methods to use the unique data source that is raw IGT instances, and utilize the particular features of IGT to add word alignment, POS tags, and dependency structures. In the next chapter, I will discuss the particular data sources in more detail, as well as some of the peculiar difficulties in dealing with different data sources, such as mapping between different POS tag sets (Section 4.3) and dealing with the very important issue of noise in many of the IGT instances (Section 4.4).

Chapter 4

THE DATA

As the goal of my research has been to develop a system capable of working on any language that is represented within ODIN, a complete evaluation of the system would be run against many languages, belonging to many different language families. In addition, I have three tasks that I will evaluate: POS tagging, word alignment, and dependency parsing. While corpora with manually-created gold standards are often available for one or more of these tasks, finding all three in the same place is difficult. Therefore, I have used a number of different corpora to try and answer all of these questions.

In the following sections I will give an overview of the specific corpora used for this research (Section 4.1) and a number of details about how the data was used, including tokenization and transliteration (Section 4.2), dealing with multiple tagsets (Section 4.3),

Resource Type	ODIN	**XL-IGT**	**RG-IGT**	**UD-2.0**	**HUTP**	**CTN**
IGT	✓	✓	✓		✓	✓
POS Tags (Tagset)			✓ (Universal)	✓ (Universal)	✓ (Hindi)	✓ (CTN)
Dependency Structures		✓		✓	✓	
Word Alignment		✓	✓			
# Of Sentences	151,633	796	82	85,625	147	8,695
# Of Languages	1,487	7	5	8	1	1

Table 4.1: High-level overview of the corpora used in this thesis, and the type of annotation that they contain.

and the issues of data cleanliness involved with using IGT from ODIN (Section 4.4).

4.1 Corpora Overview

Table 4.1 shows a very high-level overview of the six corpora used in this thesis, and the gold-standard annotations that each contain: IGT instances, POS tags, Dependency Structures, or word alignment. Table 4.2 shows a different view of the corpora, breaking down the annotation by language, covering the sixteen different languages that will be tested on in this thesis. The key in Table 4.2 may be used as reference for the resources provided by each corpus.

Language Family	**Language**	**ISO**	**ODIN**	**XL-IGT**	**RG-IGT**	**UD-2.0**	**HUTP**	**CTN**
Afroasiatic	Hausa	hau	☲	⁘ ☲ 🌳				
Austronesian	Indonesian	ind	☲			🏷 🌳		
	Malagasy	mlg	☲	⁘ ☲ 🌳				
Indo-European	Bulgarian	bul	☲		⁘ ☲ 🏷			
	French	fra	☲		⁘ ☲ 🏷	🏷 🌳		
	Gaelic	gla	☲	⁘ ☲ 🌳				
	German	deu	☲	⁘ ☲ 🌳	⁘ ☲ 🏷	🏷 🌳		
	Hindi	hin	☲				☲ 🏷 🌳	
	Italian	ita	☲		⁘ ☲ 🏷	🏷 🌳		
	Spanish	spa	☲		⁘ ☲ 🏷	🏷 🌳		
	Swedish	swe	☲			🏷 🌳		
	Welsh	cym	☲	⁘ ☲ 🌳				
Koreanic	Korean	kor	☲	⁘ ☲ 🌳		🏷 🌳		
Sino-Tibetan	Chintang	ctn						☲ 🏷
Uto-Aztecan	Yaqui	yaq	☲	⁘ ☲ 🌳				

†Orthography is not transliterated.
* Only languages used in this thesis are listed; these corpora contain more.

🏷 = POS tags ⁘ = Word Alignment 🌳 = Dependency Trees ☲ = IGT

Table 4.2: Matrix describing the different resources used (across the top row) and the information provided within each resource, represented by the icons defined at the bottom.

# Langs	# Instances	# Tokens	Avg Tokens/Instance
1,487	151,633	231,602	4.78

Table 4.3: Data overview for ODIN v2.1 data.

4.1.1 ODIN *v2.1* *(*Provides: ☲ *)*

ODIN, the **O**nline **D**atabase of **IN**terlinear text (Xia et al., 2016), is the main repository for raw IGT instances that I will draw upon. The IGT instances within ODIN are raw text, automatically extracted from the PDF documents which contained them. As a result of the PDF-to-text conversion, the IGT instances often exhibit corruption, and require cleaning to fix some of the more common errors (see Section 4.4). The version of ODIN used for the experiments in this thesis was version 2.1, available at `http://depts.washington.edu/uwcl/odin/`.

Table 4.3 shows an overview of the amount of data contained in the version of the ODIN data used for this thesis. Due to the large amount of coverage provided, the ODIN data provides IGT instances for nearly every language that I will discuss in this paper, and great deal more. With over 150,000 instances representing nearly 1,500 languages, the ODIN database holds the potential to bootstrap language tools for an extremely wide array of languages. Despite having the unique IGT data format to leverage, the ODIN database does not have gold-standard labels on the language data itself, and so other resources are used for evaluation.

4.1.2 XL-IGT *(*Provides: ⁘ ☲ 🕸 *)*

A small subset of instances drawn from the ODIN database used for the experiments in Xia and Lewis (2007) consisting of data for German, Gaelic (Scottish), Hausa, Korean, Malagasy, Welsh, and Yaqui. This dataset was of particular interest because the IGT had been cleaned

	Instances	# Lang Tokens	Avg Tokens/Instance
Gaelic	67	394	5.88
German	174	1,290	7.41
Hausa	130	780	6.00
Korean	138	718	5.20
Malagasy	128	728	5.69
Welsh	75	460	6.13
Yaqui	84	506	6.02
Total	796	4,876	6.13

Table 4.4: Overview of XL-IGT data

and so was of optimal quality, but also because these languages span a range of language families. Hausa is a great example of a language that has a large speaker population, but few resources. Hausa is an Afro-Asiatic language spoken primarily in Nigeria, with a speaker population of approximately 41 million (Lewis, 2009) but not very many online resources.

Yaqui exemplifies another side of resource-poor languages, an indigenous language of the Uto-Aztecan family with 12,230 speakers in the Sonoran region of Mexico and Arizona in the United States (Lewis, 2009). While Welsh and Gaelic have more speakers than Yaqui, they might similarly be considered threatened minority languages. In this context, NLP tools might be less valuable in traditional roles, but still of interest for tasks such as language documentation and preservation. For instance, minority-language detection on microblogging platforms could be used to help minority-language speakers find others who share the language, or spark interest in those who might like to learn.

The instances in the XL-IGT corpus have been annotated for word alignment and dependency structures. Table 4.4 gives the breakdown of this data set by language. No gold-standard POS tags are available.

Language	Instances	# Lang Tokens	Avg Tokens/Instance
Bulgarian	9	43	4.8
French	40	282	7.1
German	70	419	6.0
Italian	7	28	4.0
Spanish	15	83	5.5
Total	141	855	6.1

Table 4.5: Overview of RG-IGT Data

*4.1.3 RG-IGT (*Provides: *)*

This is another subset of the ODIN database, this time with the language line, gloss line, and translation line manually annotated with POS tags, as well as translation-gloss and gloss-language line alignment.

The gloss line being directly annotated with POS tags was done for the purpose of training the gloss-line POS classifier described in Section 6.3, but also provides the ability to evaluate projected POS tags independent of any language-gloss misalignment that may occur. The languages here were chosen primarily for the author's ability to understand the orthography and ensure that the data was clean and well-formatted. This unfortunately limited the set of languages to only Indo-European languages, but since the primary target for this corpus was the gloss line, this annotation should be portable to other languages and language families.

This data set also provides annotation for word alignment, but no tree structures. The same data were developed and used in Georgi et al. (2013). Table 4.5 provides an overview of the data.

*4.1.4 UD-2.0 (*Provides: *)*

This corpus, the Universal Dependency Treebank, v2.0 (McDonald et al., 2013), provides dependency trees in CoNLL-X format (Buchholz and Marsi, 2006), which provides labeled

dependency trees, as well as POS tags. Table 4.6 provides a breakdown by language of this resource.

This is the primary data source that will be used for evaluating POS tags and DSs on monolingual data. The POS tags used are the Universal POS tags defined by Petrov et al. (2012), and are the same used for the RG-IGT data. Additionally, the Universal Dependency project has sought to create treebanks that are syntactically consistent in their analyses across languages, and thus should reduce errors that might arise from independently constructed resources, such as in the Hindi-Urdu Treebank Project (Section 4.1.5).

Language	Sentences	Tokens	Types	Avg. Tags Per Type
French	16,422	396,511	41,452	1.08
German	15,918	293,460	50,461	1.05
Indonesian	5,593	121,923	19,915	1.08
Italian	7,189	167,873	20,655	1.06
Korean	6,339	69,690	36,323	1.02
Portuguese (Brazilian)	11,998	298,323	30,756	1.08
Spanish	16,007	424,425	46,045	1.08
Swedish	6,159	96,319	15,037	1.03
Total	85,625	1,868,524	260,644	1.03

Table 4.6: Overview of the UD-2.0 data.

4.1.5 HUTP *(*Provides: ☲ 🏷 🕸 *)*

This corpus consists of a set of IGT instances that were used as guidelines sentences for the construction of the Hindi-Urdu Treebank Project (Bhatt et al., 2009). The IGT instances are very clean, and represent the languages in a romanized transliteration. In addition to the IGT text, this resource also provide POS tags and labeled dependency parses. The dependency structures created for this project were not created with the same goal of cross-lingual representation as those in the UD-2.0 corpus, however, so some choices in the format

of the dependency trees differ, and as a result, evaluation on the two data sets may differ significantly for reasons other than the particular methodology.

This corpus serves to complement the other IGT-containing corpora in order to evaluate the POS tagging and dependency parsing methods, and increase the breadth of linguistic families for which these evaluations are performed. The HUTP uses its own POS tagset, which can be found at `http://verbs.colorado.edu/hindiurdu/guidelines.html`.

Language	# Instances	# Lang Tokens	Avg Tokens/Instance
Hindi/Urdu	147	963	6.55

Table 4.7: Overview of the HUTP data.

4.1.6 CTN (Provides: ☲ 🏷)

This corpus contains a set of IGT examples from the Chintang language of Nepal (Bickel et al., 2009). The IGT instances themselves are very clean, but as will be discussed in Section 4.3, the POS tags used in the database are not the same as the Universal Tagset, and so the use of this resource for evaluation is not as straightforward as the other annotated IGT corpora shown above.[1]

Language	Instances	# Lang Tokens	Avg Tokens/Instance
Chintang	8,695	38,896	4.47

Table 4.8: Overview of the CTN data.

[1]See also Georgi et al. (2015).

4.2 Tokenization and Transliteration in IGT

While examples like the one in Instance 3.3 show an idealized case of IGT, not all instances are as amenable to being worked with as German and English. Besides increasing divergence between languages, there are languages that do not share orthography with English. One such issue arising from such orthographic differences is how meaning-bearing tokens are delimited. Since IGT is ultimately a format for presentation, and typically for a predominately English-speaking audience, this results in IGT instances that are not given in the original orthography, or if they are, a romanization that includes whitespace tokenization, and often morphological analysis using hyphenation. Without the original tokenization, however, an IGT-bootstrapped tool will be unable to correctly process untokenized input.

Besides the whitespace issue, an orthographic difference between the typical language representation and the representation in the IGT presents a different problem. While many languages use orthographic representations that are now available in Unicode formats, transliteration of the orthography is a difficult issue. Standards do exist [e.g. ISO 15919 (Stone, 2004)], but the linguists who create the IGTs may or may not use one. Reasons for not using the native orthography or phonetic transcription may be to prefer another approach, such as the International Phonetic Alphabet (IPA), or to simplify the presentation based on the paper's intended audience. This means that when it comes to utilizing the information extracted from these languages with non-Latin orthographies, matching them with other resources that may be available for the language may prove difficult. As a result, for the purposes of this work, experiments that evaluate on monolingual data will be limited to languages which use Latin orthographies, or perfectly regular transliteration.[2]

[2]Japanese and Korean, for instance, occur in both ODIN and the UD-2.0 treebank, but are found entirely in transliteration in the former, and in native orthography in the latter.

Tag	Description
ADJ	Adjectives
ADP	Adpositions (Prepositions/Postpositions)
ADV	Adverbs
CONJ	Conjunctions
DET	Determiners and Articles
NOUN	Nouns
NUM	Numerals
PRON	Pronouns
PRT	Particles
VERB	Verbs
X	Catchall (Abbreviations/Foreign Words)
.	Punctuation

Table 4.9: Universal POS Tagset following Petrov et al. (2012)

4.3 POS Tagsets

One of the biggest difficulties in working with massively multilingual data is that the tools and annotations used in different languages are geared toward a language-specific analysis for the target language. Using an English POS tagger will typically mean using the Penn Treebank tagset, which tends to be focused on English constructions. In Yarowsky and Ngai (2001), the authors focused on projecting English to French, and made the concession to not distinguish between the mood, person, or number distinctions that are realized on verbs in French but not English, yet they still worked to distinguish between the `VB/VBN/VBG/VBD`[3] forms. While projecting tags may be feasible between these two Indo-European languages, using these tags may make less sense when projecting between English and a highly isolating language where tense is formed by the bare verb and a tense-bearing particle.

A number of works in recent years (Zeman, 2008; de Marneffe and Manning, 2008; Petrov et al., 2012) have posited different approaches to selecting an appropriate set of part-of-speech

[3]Present/Past Participle/Gerund/Past

(POS) tags that can be applied to a wide variety of languages, ideally a tagset that is universal among languages. This is a difficult undertaking, as Zeman (2008) pointed out, given that the primary goal of POS tags is to encode a particular bundle of morphosyntactic information carried by a given word, specifically, the way in which that information is combined varies from language to language. Despite this difficulty, Xia and Lewis (2007) used projected POS tags to reliably determine basic word order for hundreds of languages, highlighting the utility of a coarse tagset for answering typological questions.

Mapping Between Tagsets With the UD-2.0 corpus using the Petrov et al. universal tagset, shown in Table 4.9, this tagset seemed a good set to use in my work with other languages. As a consequence, I used the universal tagset when creating labels for the RG-IGT corpus, and also had to create mappings for the HUTP and Chintang corpora. The mappings for these tagsets are shown in Appendix C in Tables C.1 and C.2, respectively.

A number of tagset mappings to the Petrov et al. (2012) set for other languages is also provided by the authors at `https://github.com/slavpetrov/universal-pos-tags`, including mappings from the Penn Treebank tagset. Both the mappings performed for this thesis and those done by Petrov et al. are made many-to-one, with multiple target tags being mapped to potentially the same universal tag.

4.4 IGT and Data Cleaning

While much of the data contained in ODIN is very straightforward, such as the example in Instance 3.1, there are ways in which the data in ODIN requires modification for use with the automated methods I will be using here. The data may be non-ideal for one of two reasons. Either it exhibits corruption resulting from being extracted from PDF documents, or the author has included metadata in the example that is not linguistic in nature.

In both cases, the INTENT system preprocesses the raw ODIN data in an attempt to make it more readily usable. In the former case, fixing the unintended errors arising from

```
(6)  a. Carapana          (E. Tucanoan; Metzger 1981:34)
        pa-w
        work-3.SG.FEM.PAST
       'She worked.'  (no evidential reading)
```

Instance 4.1: An instance of Carapana (`cbc`), demonstrating some of the non-linguistic information that is found in IGT instances, including the language name, example numbering, author citation, and explanation of linguistic phenomena

the format conversion is what I call **cleaning**, and is the focus of Section 4.4.1. In the latter case, the removal of inline metadata from the example is what I refer to as **normalization**, and address in Section 4.4.2.

Instance 4.1 illustrates several such cases, with example numbering, the language name (Carapana), a citation for the source of the instance, and an explanation of the intended interpretation. Instance 4.2 contains an example with both inline metadata, as well as irregular whitespace.

While these issues are different in nature, both can present problems in extracting proper inferences from the IGT instances. I will describe some of the most common issues below, and what preprocessing steps are taken by the INTENT system to clean and normalize the data.

```
doc_id=430 513 520 L M B B B B G T+AC+LN
language: arabic (arb)
line=513 tag=L:   8)    daxal{ *-na / -at}              n-nisaa -u              makaatib-a-hunna
line=514 tag=M:
line=515 tag=B:
line=518 tag=B:
line=519 tag=G:         entered{ *-3.   PL.F.   / -3.SG.F.}   the-women(3.  PL.F.)-NOM   office(PL.)-ACC-their
line=520 tag=T+AC+LN:          'The women have entered their offices.'            (Fassi Fehri, 1993: 32)
```

Instance 4.2: A raw IGT instance of Arabic (`ara`) in ODIN, extracted from Nasu (2001). This instance shows corruption on the gloss line, numbering on the language line, and other errors. Dealing with these sources of noise is key to improving performance.

4.4.1 Cleaning: Fixing PDF-to-Text Corruption

Spurious or Missing Whitespace In Instance 4.2, both lines 513 and 519 show tokens that have had spurious whitespace inserted, likely because of the pdf-to-text extraction process mistaking kerning for whitespace. The token `‘n-nisaa -u’` on line 513 should not have whitespace between the final `-u`, nor should the `‘3. PL.F.’` on line 519 have space between the `‘3’` and `‘PL’`. These may seem like simple errors, and they can sometimes be easily fixed by reattaching sentence-internal characters separated by morphological boundaries such as `‘.’` and `‘-’`, but similar problems sometimes occur between non-punctuation symbols making such reattachment heuristics trickier.

Though the example in Instance 4.2 does not contain any cases where whitespace has been erroneously omitted, such cases do exist in the data. These instances are more difficult to detect, particularly if the whitespace is missing from the language line, where less is known about language in question. Unlike the semi-structured gloss line, which contains a known vocabulary of English and grammatical terms, it is more difficult to reconstruct the intended structure of words and morphemes for a previously unseen language.

In cases of either spurious or missing whitespace, the instance is discarded if the language line and gloss line do not contain the same number of whitespace-delineated tokens. Whether this mismatch is due to a conversion error or was the result of the author's formatting, since I assume that the gloss line refers one-to-one to language-line elements, a mismatch means this assumption can no longer be made. Conversely, instances where the number of whitespace-delineated tokens do match, but corruption has caused a shift, can lead to cases where one-to-one gloss alignment produces incorrect data.

If the language line of the instance being cleaned were from a language with abundant resources, language models could be built that might help, such as detecting out-of-vocabulary words that are likely metadata, or at the character level, where spaces are likely erroneous. However, with the IGT data, although such methods may be possible on the gloss line, the

language line is likely not from a language where there is not enough data for this, and so this language line corruption is ultimately left in.

Line Corruption Instances can also end up being corrupted in ways such as that in Instance 4.3, compared to the original instance in Instance 4.4. In this case, a single line that was intended to contain a word with several subscripts and diacritics had the line split into two lines, losing the diacritic from the character it was meant to modify, and either splitting the subscripts or combining them with the neighboring word. In ODIN, at least 36,716 out of the total 151,633 instances were flagged by the original IGT detection algorithm as exhibiting line corruption.

In the cleaning process, two approaches to fix these sorts of corruption were attempted. The first involved finding whitespace between the two lines, or cases where a combining character, such as the `‘^’` on line 27, is present in isolation, and attempt to combine the characters. In this case, this would yield the desired result of `hâld`, but sometimes lines

```
line=26 tag=M+LN:      (3)        Frisian
line=27 tag=L+CR:                 Maxi h^ d
line=28 tag=L+CR:                       al       himi /*himsels        i,j
line=29 tag=G:                  Max behave.3SG him/himself
line=30 tag=T:                  ‘Max behaves (himself ).’
```

Instance 4.3: Example of a Frisian (`fry`) IGT instance exhibiting line corruption, character corruption, and language names and parentheticals, extracted from de Swart (2003).

(3) *Frisian*
Max$_i$ hâld him$_i$/*himsels$_{i,j}$
Max behave.3SG him/himself
‘Max behaves (himself).’

Instance 4.4: Original text of Instance 4.3 as intended by the author.

wrap instead of combining, and in these instances the appropriate behavior would be to concatenate the lines rather than merge. Due to the difficulties in attempting to merge, when such multi-line corruption is encountered, I have chosen to concatenate the lines.

4.4.2 Normalization: Removing Non-Linguistic Data

Judgments and Judgment Alternation Instance 4.2 also has another interesting phenomenon which is helpful for illustrative purposes, but is difficult to parse programmatically. This phenomenon is the alternation in judgment between the `-na/-at` suffixes on line 513, which the gloss line (519) clarifies are the `3.PL.F/3.SG.F` suffixes, respectively. Judgment markings in themselves are a particularly interesting feature of IGT, and potentially a very useful one for providing negative examples to parsers or language models. However, as the data is often intended for display and not automated parsing, whether the ungrammaticality refers to the utterance being unacceptable, or a particular interpretation being unavailable, is often unclear without the explanatory prose. I have chosen to save that topic for future study.

Instance Numbering Many IGT instances occur with a label of some form, such as `‘a)’`, `‘ix.’`, or `‘[3]’` on the language line, or elsewhere, for subsequent reference in the containing publication. While a simple regex can catch most of these instances, they can also contribute to language–gloss-line misalignment.

Citations Instance 4.2 contains a reference to the publication from which the IGT instance itself was cited. When such citations occur on the translation line, as in this example, they may contribute to the English-language parser or POS-tagger producing an incorrect result. These cases are not particularly problematic, however. That said, if they occur on the language or gloss line, they can again contribute to misalignment.

	Languages with ≥1 Instance	Instances
Unfiltered	1,497	157,974
Filtered	1,223	61,711

Table 4.10: Result of INTENT filtering on instances in ODIN.

4.4.3 INTENT *Filtering*

After the cleaning and normalization processes have been run, INTENT checks to see whether the instances are valid and useful for further processing. After the cleaning and normalization processes have run, a check is done on whether the number of whitespace-delineated tokens between language and gloss lines match. If they do not match, the instance is discarded. An instance is similarly discarded if it is lacking a language, gloss, or translation line, or if any of those lines are empty after the cleaning and normalization processes have run. Table 4.10 shows the results of this filtering procedure on the instances in ODIN, filtering out 61% of the total instances.

Cleaning and Normalization Failures Neither properly cleaned instances nor properly filtered instances contribute erroneous data. The cleaning and normalization processes can, however, make errors that do not get caught and filtered out. For instance, Instance 4.5 shows an instance of an IGT instance where the language-line cleaning has correctly removed the author citation, but erroneously removed the token `*(dee)` using the regular expression `/\*.*[a-z]\)/` intended to remove instance numberings. Meanwhile, the language name `Pashto` was not removed. As a result, the number of whitespace-delimited language line tokens and gloss line tokens match, but are incorrectly glossed.

```
Plaar mee *(dee) léeg-i Pashto
father 1.SG 2.SG send-PRES.3.SG
‘Your father is sending me.’
```

(a) Original IGT instance, before cleaning.

```
Plaar   mee   léeg-i   Pashto
father  1.SG  2.SG    send-Pres.3.SG
‘Your father is sending me’
```

(b) Cleaned IGT instance with faulty lang–gloss alignment

Instance 4.5: A Pashto (`pst`) instance extracted from Neeleman and Szendrői (2007) showing faulty cleaning that results in an incorrectly glossed language line.

4.4.4 Future Work in Cleaning

These are just a few of the issues with noise that are present in the IGT instances found in ODIN database. At the time of writing, there are two ongoing projects to address some of the noise issues identified here. The first is a continuation of the work done by the *i*nformation *e*ngineering and *s*ynthesis for **R**esource-**P**oor **L**anguages (R*i*PL*es*) project (Xia et al., 2016), using PDFLib's Text and image Extraction Toolkit (TET) (PDFLib GmbH, 2015), which performs substantially better than the original `pdftotext` program (Foo Labs, 2014) used for the initial conversion. The second approach uses human annotators to manually clean the extracted instances, using the interface developed as a part of this work, and discussed further in Section 8.8.

While data from these projects is not available at the time of writing, I expect that many of the results presented here would benefit modestly by use of the cleaner data. Any amount of noise is likely to affect the performance of these IGT-based systems, particularly those languages for which the number of available instances is lower. Minimally, and crucially, data cleaning will increase the number of IGT instances available for downstream processing. Even a modest increase in the number of available instances could improve the performance of the tools developed over the data, and possibly, the number of languages that can be directly affected.

4.4.5 Consistency Issues

Beyond issues with cleanliness and normalization, IGT data also has another issue, and that is consistency between authors. While there is a set of guidelines for how IGT data are to be represented in linguistics paper, in the form of the Leipzig Glossing Rules Bickel et al. (2008) (LGR), these guidelines are not always strictly followed. In some instances, authors delineate all morphemes in the language line with hyphens, in others they are not separated. Many rules are also optional, such as using a colon when morphology is present, but the author does not wish to show the segmentation. As a result, between instances, not all rules are used consistently, so a system looking to deal with these glosses must be robust to such inconsistencies.

4.5 Data Formats

The corpora listed above are made available in various data formats. The UD-2.0 corpus is distributed in the CoNLL-2009 (Hajič et al., 2009) format, and the HUTP, XL-IGT, and CTN corpora were provided in various text formats. The 2.1 version of ODIN and the RG-IGT corpora are currently available in *Xigt-XML*, a format first presented by Goodman et al. (2014) that can preserve the original text of an IGT instance while allowing for terse standoff annotation layers. The *Xigt-XML* format is also used internally by the INTENT package that will be discussed in Chapter 8. Sample 4.1 shows a snippet of a Hindi IGT instance rendered in the *Xigt-XML* format, showing the raw text in the top annotation "tier," as well as word segmentation and POS tags. For a full discussion of this data format, refer to Goodman et al. (2014).

```
<igt id="igt1">
    <tier id="n" type="odin" state="normalized">
      <item id="n1" tag="L">rAma ne mohana ko nIlI kiwAba xI</item>
      <item id="n2" tag="G">ram erg Mohan acc blue book gave</item>
      <item id="n3" tag="T">Ram gave a blue book to Mohan</item>
    </tier>
    <tier id="p" type="phrases" content="n">
      <item id="p1" content="n1" />
    </tier>
    <tier id="w" type="words" segmentation="p">
      <item id="w1" segmentation="p1[0:4]" />
      <item id="w2" segmentation="p1[5:7]" />
      <item id="w3" segmentation="p1[8:14]" />
      <item id="w4" segmentation="p1[15:17]" />
      <item id="w5" segmentation="p1[18:22]" />
      <item id="w6" segmentation="p1[23:29]" />
      <item id="w7" segmentation="p1[30:32]" />
    </tier>
    <tier id="w-pos" type="pos" alignment="w">
      <item id="w-pos1" alignment="w1">NNP</item>
      <item id="w-pos2" alignment="w2">PSP</item>
      ...
    </tier>
    ...
</igt>
```

Code Sample 4.1: Example of the *Xigt-XML* data format.

Chapter 5
WORD ALIGNMENT

One of the features of IGT that makes it extremely useful for resource-poor languages is the gloss line. As discussed in Section 3.1, the gloss line can be used to align the language line with the translation line using matching tokens, leading to the ability to transfer language annotation by these alignments. This chapter will discuss how IGT as a resource can be used to obtain high-quality alignments between the language lines and translation lines using a variety of methods. Due to its unique format, these alignments are of a much higher quality than would otherwise be achievable for the same amount of data using traditional statistical alignment methods. As intra-IGT word alignments of this type are required for projection-based transfer methods, obtaining alignments that are precise and with as broad a coverage as possible is extremely important for any task requiring projected data.

Given the importance of obtaining high-quality word alignment among language and translation lines, I will discuss and compare two different approaches for aligning IGT data, using the manually aligned instances in the RG-IGT and XL-IGT corpora as evaluation targets. A heuristic approach based upon the repeating tokens with several variations is described in Section 5.2, and several different types of traditional statistical approaches are discussed in Section 5.3. Finally, I also discuss some methods by which the two approaches may be combined in Section 5.5.

5.1 Evaluating Word Alignment within IGT Instances

In order to evaluate and compare the different methods and settings used for the experiments in this section, I will be using the manual alignments in the RG-IGT and XL-IGT corpora.

When evaluating, every alignment between one word and another is considered. Each of these word-to-word pairs is counted in the alignment to be evaluated, and compared to the word-to-word pairs in the gold standard to calculate precision, recall, and F_1-Score. For the purposes of this work, I will not differentiate between sure and possible alignments, and will consider every alignment in the gold standard as a sure alignment.

5.2 Heuristic Alignment

The first approach to alignment leverages the basic format of IGT, discussed in Section 3.1; namely, the repetition of tokens from the gloss line in the translation line. As the gloss line is intended to be an intermediary step between the foreign language and the language of the reader, the gloss line is typically populated with many of the same words (or synonyms) found in the translation. Producing an alignment, then, can be done programmatically much in the same way that a human reader might do so; by finding the words in translation and gloss that share meanings, and then associating the . One major complication, as I will discuss, is precisely what "sharing a meaning" means for certain gloss and translation tokens, as this is not always entirely straightforward.

I examined the use of six different manipulations of the gloss line in order to achieve the best possible alignment: (1) Basic String Matching, (2) Tokenizing for Morphology, (3) Allowing Multiple Matches, (4) Stemming Words, (5) Matching Grammatical Markers, (6) Matching POS tags. As each heuristic is added, the alignment algorithm was run and evaluated using the alignments RG-IGT and XL-IGT corpora.

The rest of this section discusses the implementation of these approaches, and the evaluation is shown in Fig. 5.1 and Table 5.2.

1	2	3	4	5	6			
inepo	mache'eta-m	into	kuchi'i-m	kecha-k	.			
1SG	machete-PL	and	knife-PL	put.up.SG.OBJ-PST	.			
I	put	up	the	machete	and	the	knife	.
1	2	3	4	5	6	7	8	9

Instance 5.1: IGT Instance in Yaqui (yaq) showing how morphology is given on the gloss line. Gloss token #5, 'put.up.SG.OBJ-PST', shows both the morpheme marker '-', which typically marks morpheme boundaries, and the combining marker '.' indicating that multiple aspects are contained within a single morpheme.

String Matching

The most simple baseline approach to aligning the translation and gloss lines is simply tokenizing the strings on whitespace and searching for exact string matches. This is the baseline approach used—acting as a strong baseline, given the F_1-scores shown—yet there are many reasons why a token on the gloss line doesn't match a token on the translation line exactly. Differences in capitalization, inflection, or use of non-language tokens and symbols are among the issues that other approaches address. As an extremely simple first pass beyond the baseline system, I perform case-insensitive string matching – this improves the alignment performance somewhat, but not drastically.

Tokenizing for Morphology

As is shown in Instance 5.1, gloss tokens do not merely consist of the words used in the translation, but also contain morphological information and inflection. According to the Leipzig Glossing Rules (Bickel et al., 2008), morphemes should be delineated in the gloss by hyphens ('-'), clitics by equals signs ('='), and inflection or other information that is not morphological in nature with periods ('.'). While this is not always the case in the ODIN data, I would still like to be able to recover alignments between strings that are separated by

these characters. For instance, in Instance 5.1, I would like the system to create the alignment `5:{2,3}`, mapping the gloss token `'put.up.SG.OBJ-PST'` to both `'put'` and `'up'`. For the system to find matches like these, it tokenizes each portion of the gloss line, splitting the strings on the morpheme separators (`-, =` and the compound gram marker, period `'.'`)t. There were also a few cases where parentheses and colons were used, so I included these as well. Tokenizing and performing case-insensitive matching boosted the F_1-score from 0.71 to 0.83, a 41% reduction in error.

Finding Multiple Matches

I make one further simple modification to finding matches when there are repetitions of a token on one or both lines of the gloss or translation. Instance 5.2 shows two instances where this occurs. When a token is found that has multiple occurrences on one or more

1	2	3	4	5	6		
i	mwuncey-nun	ku	mwuncey-wa	kath-ta	.		
this	***problem***-Top	that	***problem***-as	same	.		
This	***problem***	is	the	same	as	that	***problem***
1	2	3	4	5	6	7	8

(a) IGT Instance from Korean (`kor`) demonstrating repetition of words. `'problem'` is repeated twice in both gloss lines and translation lines, so alignment is done from left to right in order of the pairs.

1	2	3	4	5	6	7	8
nanomboka	niteny	ity	tonon-kira	ity	Rabe	indroa	.
began	knock	***this***	door	***this***	Rabe	twice	.
Rabe	twice	began	to	knock	on	***this***	door
1	2	3	4	5	6	7	8

(b) In this example from Malagasy (`mlg`), `'this'` is repeated twice in the gloss, but only appears once in the translation. While in this case, it is possible that the occurrence in `'this Rabe'` is not meant to align with the instance in `'this door'`, but the algorithmic output is in keeping with that in the gold standard.

Instance 5.2: Examples of multiple occurrences of tokens between gloss and translation lines.

lines the words are aligned in order of occurrence left-to-right until one side runs out of occurrences, and then the remaining tokens are aligned to the last token on the other side. In Instance 5.2a, the alignment is one-to-one and done left to right. In Instance 5.2b, there are multiple occurrences of `‘this’` on the gloss line, but only one on the translation line. This modification aligns the single gloss occurrence of `‘this’` with the multiple occurrences on the translation line. While in this case, it may be the case that the occurrence in `‘this Rabe’` is meant to be a determiner which is unaligned in the translation, such an ambiguity is hard to discern with certainty, and the gold standard alignment agrees with the output of the algorithm for this case.

Finding Root Forms

While splitting the gloss line by morpheme improves performance, the words on the gloss line are transliterations and not always inflected as they would be in the translation. Figure 5.3 gives an example of this issue, showing an instance of Malagasy where the token on the gloss line `‘dry-IV-VN’` does not match the inflected form found in the translation, `‘drying’`.

To solve this mismatch, I next attempted a chain of stemming approaches, starting with the implementation of the snowball stemmer (Porter, 2001) found in NLTK (Bird et al., 2009). If a stem could not be found this way, NLTK's implementation of the `morphy` lemmatizer from WordNet (Princeton University, 2010) is used to attempt to lemmatize verbs, nouns, or adjectives.[1] After this modification, the performance is boosted to an F_1-score of 0.93.

[1] I did not use the `morphy` package directly, as the software for this thesis was written in Python. NLTK is easily installable for Python, and this allowed for a simpler code implementation.

1	2	3	4	5
ìigaa	tanàa	gàa	tsanèewaa	.
gown	3fs-CONT	at	dry-IV-VN	.
The	gown	is	drying	.
1	2	3	4	5

Instance 5.3: IGT Instance from Malagasy (`mlg`) demonstrating the need for finding the root forms of morphs in the gloss line. Gloss token #4, `'dry-IV-VN'` shows the word `'dry'` whereas the translation uses the English inflected form `'drying'`.

Finding Grammatical Markers (Grams)

Finally, the presence of non-English subwords such as `'1SG'` or `'MASC'` in the gloss line is part of the power of the IGT format and is key to allowing a heuristic aligner to pick up

1	2	3	4	5	6	7	8		
Cheisiodd	Gwyn	ddim	beidio	ag	ateb	y	cwestiwn		
try-PAST-3SG	Gwyn	*NEG*	*NEG*	with	answer	the	question		
Gwyn	did	*n't*	try	to	*not*	answer	the	question	.
1	2	3	4	5	6	7	8	9	10

(a) IGT Instance from Welsh (`cym`) demonstrating how the two `'NEG'` grams representing negation are aligned with `'n't'` and `'not'` when gram matching is enabled.

1	2	3	4	5	6	7	8	
inepo	Diana-ta	bicha-k	,	apoik	achai	into	ketchia	.
1SG	Diana-NNOM.SG	see-PST	,	*3SG.POSS*	father	and	too	.
I	saw	Diana	and	*her*	father	.		
1	2	3	4	5	6	7	8	9

(b) IGT Instance from Yaqui (`yaq`) demonstrating how the `'1SG'` and `'3SG.POSS'` tokens from the gloss line are aligned correctly to `'I'` and `'her'` on the translation line when gram matching is enabled.

Instance 5.4: IGT instances demonstrating the importance of looking for grammatical markers, or "grams," on the gloss line.

Gram	Possible Meanings
1sg	i, me
2sg	you
2pl	you
3sg	he, she, him, her
3sgf	she, her
3sgm	he, him
3pl	they, their
poss	his, her, my, their
neg	n't, not
det	the

Table 5.1: Gram-to-word lookup table used to match grams with possible translations.

a few more matches. Instance 5.4 shows two IGT instances in which matching the grams leads to improvements in alignment. After previous matching has been performed, a second pass looks for gloss tokens in a user-defined list and looks for possible matches from a simple table lookup, matching either full words or sub-word level tokens. My list containing the mappings can be found in Table 5.1. This list is not exhaustive, but is targeted at the fact that IGT instances frequently provide only the person/number gloss for pronouns, while using the lexical item on the translation line.

Matching POS Tags

The final alignment heuristic I experimented with was using the POS tags of the remaining unaligned gloss and translation tokens to attempt to recover additional alignments. POS tags are generated on the gloss tokens with the classifier in Section 6.3, and POS tags on the translation line are generated using the Stanford Tagger (Toutanova et al., 2003), both trained using the universal POS tagset (Petrov et al., 2012) mentioned in Section 6.1. Next, the tokens that remain unaligned after the previous methods are aligned the same way as described in the multiple alignment section; left-to-right and one-to-one, and any remaining

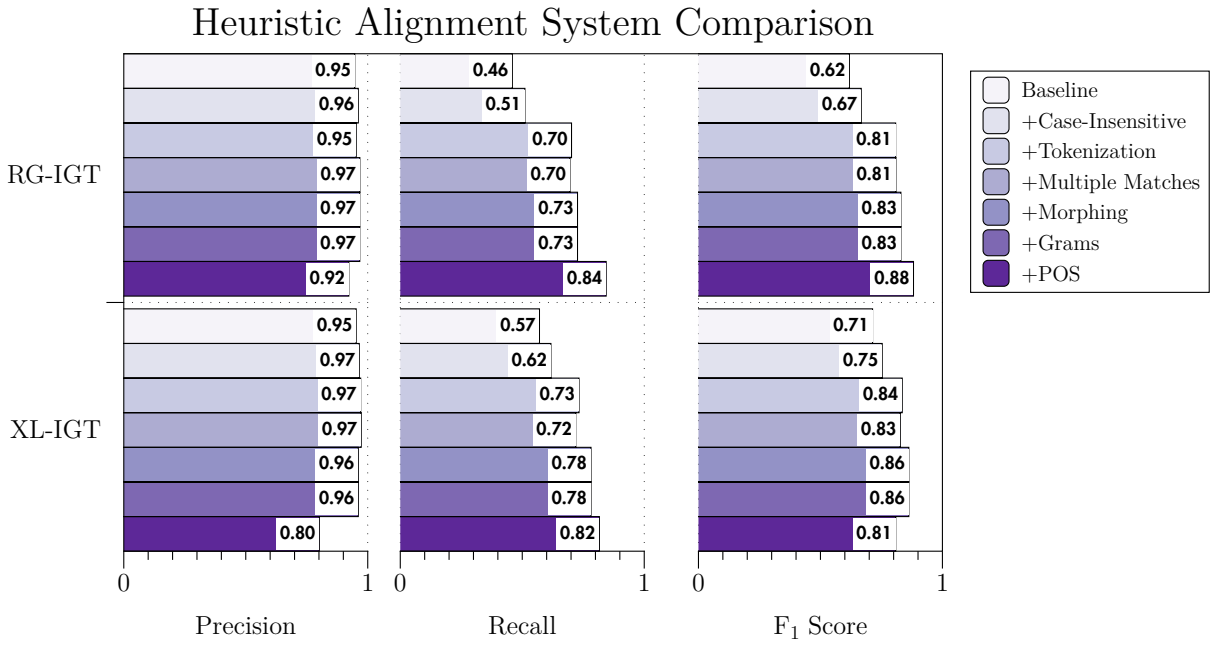


Figure 5.1: Graph showing the performance of the heuristic matching approach between gloss and translation as heuristics are added going from top to bottom.

tokens on either side are aligned with the rightmost token of the other.

Summary

Fig. 5.1 shows the results of incrementally adding heuristic approaches to the alignment algorithm on both RG-IGT and XL-IGT corpora. The baseline approach, matching only full strings between translation and gloss lines, gives an F_1-score of 0.62 on the RG-IGT data, and 0.71 on the XL-IGT data. While the precision of this baseline approach is extremely high at 0.95, the recall is very low, at 0.46 and 0.57 for the two corpora. Each additional heuristic improves recall, culminating in 0.84 on the RG-IGT data and 0.82 on the XL-IGT data. The precision remains nearly the same for all heuristics, save the POS matching heuristic, where despite the increase in recall, the precision drops significantly. Despite the drop in precision, the RG-IGT data still shows an increase in F_1-score from 0.83 to 0.88, while the XL-IGT

data drops from 0.86 to 0.81.

The addition of heuristics up until the POS tag matching improved overall performance, and so the heuristic system to use in other experiments will be the system containing the set of heuristics leading up to, but excluding the POS tag matching, and referred to as "Heur." Very high-precision alignments might be more useful for some tasks, while higher recall, or more balanced approaches might be better overall, so given the large increase in recall that the POS tag matching system offers, this system will also be present in subsequent comparisons, labeled as "Heur +POS."

Table 5.2 shows the precision, recall, and F_1-scores for the "Heur" system on both corpora, broken down by language. Given the small size of the corpora, I am wary of making conclusions about differences in the individual languages, but the variation between languages also corresponds with variation between documents and authors, and so is a useful reminder that conventions and formatting of IGT instances may differ greatly among the data in ODIN.

In summary, this heuristic approach to word alignment within IGT is simple, straightforward, and viable for languages with as few as a handful of IGT instances. Despite the multiple strengths of this approach, it is a rule-based approach, and performs the same whether given ten IGT instances or ten thousand. Given the domain of resource-poor languages, such a rule-based approach is a good place to start, but an important path of inquiry is whether performance be improved when additional data is available. The next section explores the possible ways in which the large and expanding set of instances in ODIN might be harnessed *en masse* to improve IGT alignment performance.

5.3 Statistical Alignment

The heuristic approach above takes advantage of IGT's unique format to align the language line and translation line by using the gloss line as a pivot. The typical approach to such

XL-IGT			
Language	**Precision**	**Recall**	**F_1-score**
Gaelic	0.97	0.90	0.94
German	0.93	0.81	0.87
Hausa	0.98	0.66	0.79
Korean	0.98	0.73	0.83
Malagasy	0.97	0.89	0.93
Welsh	0.96	0.79	0.87
Yaqui	0.99	0.72	0.83

RG-IGT			
Language	**Precision**	**Recall**	**F_1-score**
Bulgarian	0.93	0.66	0.77
French	0.97	0.73	0.83
German	0.96	0.76	0.85
Italian	1.00	0.61	0.76
Spanish	1.00	0.64	0.78

Table 5.2: Results of the heuristic alignment by language, without using POS tags.

bilingual alignment would be to use a statistical alignment system for phrasal alignment, such as Giza++ (Och and Ney, 2003b), to align language and translation lines directly. I will examine this approach in Section 5.3.1, while laying out an alternate approach to statistical alignment in Section 5.3.2 that aligns gloss lines with translation lines in a manner similar to the heuristic approach. Evaluation was performed following the description in Section 5.1, and results are shown in Fig. 5.2.

5.3.1 Baseline Statistical Alignment Approach

For a baseline approach of aligning two languages using parallel sentences, I build a set of parallel sentences between the language lines and translation lines of all the instances for a given language in ODIN, and use `mgiza` (Gao, 2013) to produce word alignments. This approach is labeled as "L-T" in Fig. 5.2, as this approach aligns the language line ('L') directly with the translation line ('T'). As noted in Chapter 4, these corpora are relatively small, sometimes only a few hundred sentences, which is typically not large enough to achieve good word alignment results. This is reflected clearly in the results, as the L-T systems perform well below any other alignment system shown here, with F_1-scores of 0.47 and 0.51 for the RG-IGT and XL-IGT data, respectively.

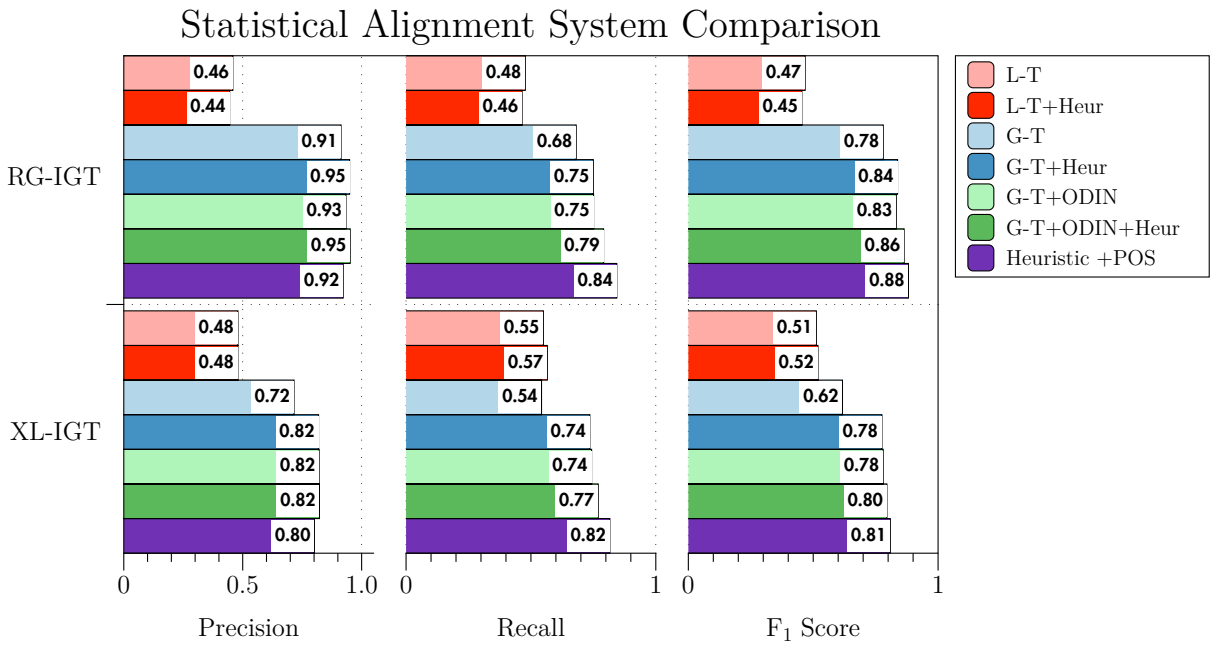


Figure 5.2: Graph showing performance of the different statistical approaches on the different corpora, with the Heuristic +POS system for comparison.

5.3.2 Gloss/Translation-Based Approach

While using the language and translation lines directly is possible, and has the benefit of being able to use the IGT instances in the ODIN database that do not include a gloss line, it is also possible to align the IGT instances based upon their translation and gloss lines, as in the heuristic approach. Rather than creating a parallel corpus of language and translation lines for this approach, the parallel corpus consists of gloss lines and translation lines for each language. This simple change improves alignment F_1-scores from the L-T system's 0.47 and 0.51 to 0.78 and 0.62 on the RG-IGT and XL-IGT corpora.

There are even more substantial gains to be made over using only the gloss and translation lines for a single language, however. Using the gloss line affords a sort of a "pseudo-language," as described in Section 3.1.3, that is found in every IGT instance with an English translation

throughout ODIN. Therefore, instead of attempting to create a statistical alignment based upon only the several hundred sentences for a selected language, the entire set of 151,633 instances across all languages can be used. This does in fact increase performance, as seen in Figs. 5.2 and 5.6, improving F_1-scores in the RG-IGT data from 0.78 to 0.84, and in the XL-IGT data from 0.62 to 0.78.

While this improvement is notable, the gloss-line "pseudo-language" presents a particular challenge to a statistical word alignment system in that the word order is variable. Its order reflects that of the glossed language, and therefore is not fixed over the whole set of ODIN instances. As a consequence, it is likely that further experiments on tweaking the distortion parameter for the alignment model might boost performance even higher.

5.3.3 Other Settings

Two other settings that I looked at with the statistical alignment systems were the choice of symmetrization heuristic and software package. Symmetrization heuristics are the methods by which bidirectional alignments are combined. Fig. 5.3 shows an hypothetical set of unidirectional alignments between English→French and French→English. Also shown are two potential solutions for combining these alignments; either by taking the *Intersection* of the alignments and selecting only the word pairs that are aligned in both instances, or the *Union* of all word pairs aligned in either direction.

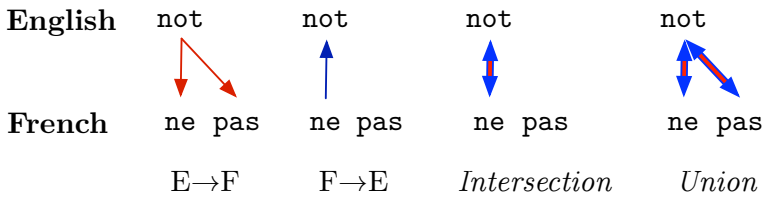


Figure 5.3: Illustration of resolving a potential difference in unidirectional alignments between French and English using *Intersection* and *Union* Symmetrization heuristics.

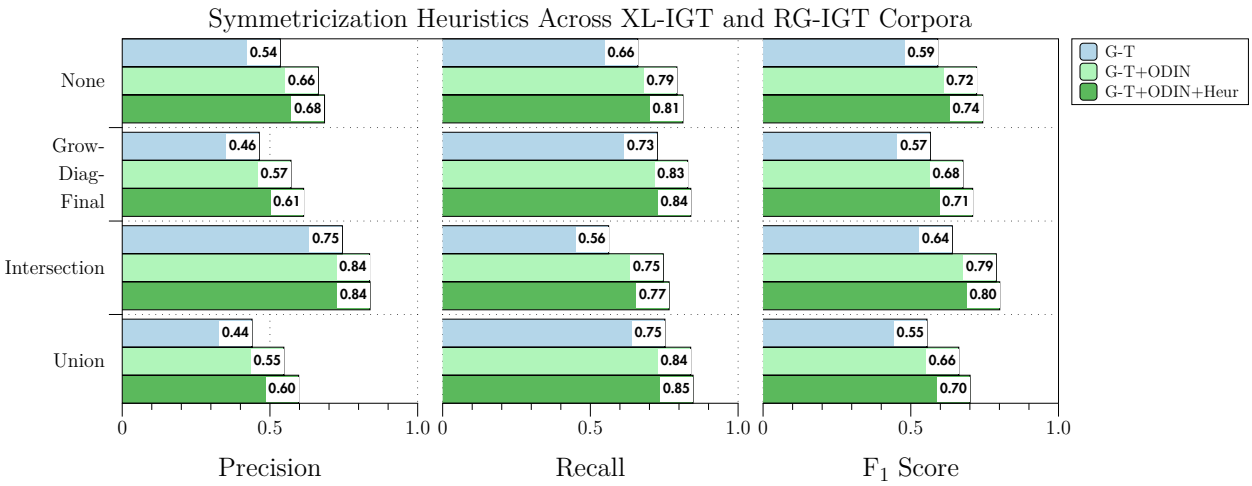


Figure 5.4: Precision, Recall, and F_1-score for the different symmetrization heuristics on the statistical approaches on the XL-IGT and RG-IGT corpora.

To compare symmetrization heuristics, I ran `union`, `intersection`, and `grow-diag-final`, and compared each of these against the single translation→gloss alignment. Figure 5.4 shows the results of these experiments. While the `grow-diag-final` and `union` methods did boost recall over `intersection`, the `intersection` approach was higher than any other method in precision by 0.2 (absolute), and thus still comes out ahead for all methods in terms of F_1-score. As a result, I used the `intersection` symmetrization heuristic as the default statistical alignment symmetrization heuristic in all other experiments.

Finally, to compare software packages, I also ran the `fastalign` software package (Dyer et al., 2013), an optimized reparameterization of IBM model 2 on the G-T and G-T+Heur systems. The graphs in Fig. 5.5 show the comparison. While performing roughly the same in recall, `mgiza` does outperform `fastalign` with substantially better precision evaluating on the XL-IGT data, and modestly better on the RG-IGT data, for an overall higher F_1-score. The `mgiza` package also has the further advantage of being able to precompute some

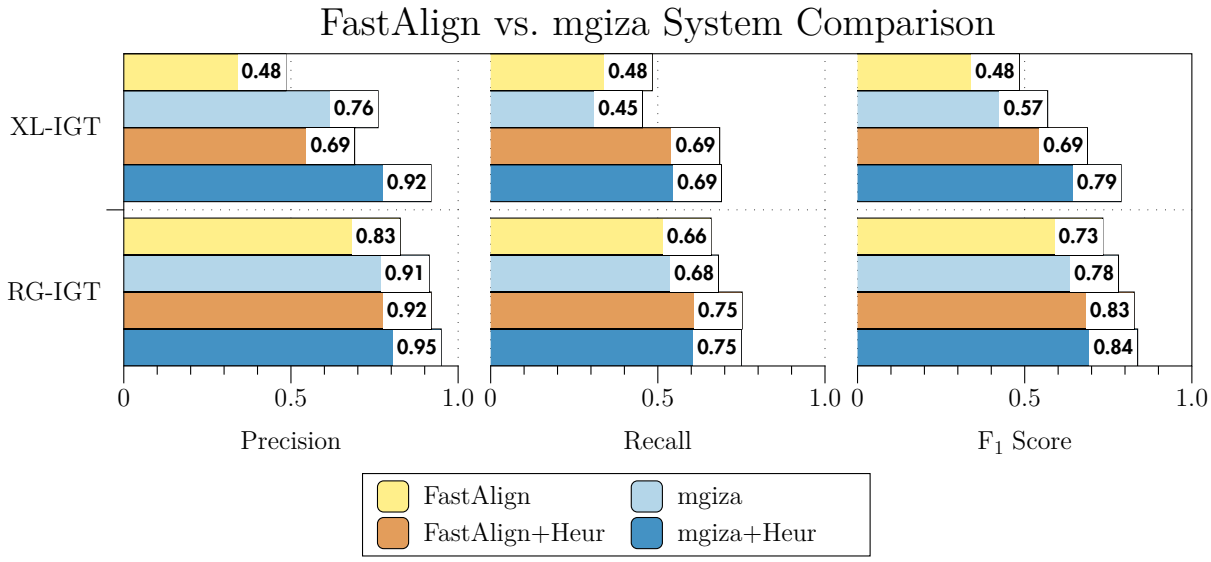


Figure 5.5: Comparisons of the `fastalign` (Dyer et al., 2013) software package with `mgiza` (Gao, 2013).

of the alignments and "force" further alignments using a transductive approach,[2] resulting in much faster program execution time. For these reasons, I also use this software package for statistical alignments in this thesis.

5.4 Combining Statistical and Heuristic Alignment

In an attempt to capture both the precision of the heuristic approach with the possible breadth of the statistical approach, I also performed heuristic alignment across the Odin corpus, and added the matching gloss/translation token pairs identified by the heuristic aligner as parallel sentences to the input to the statistical aligner. As the graph in Fig. 5.2 shows, this improves performance greatly over the baseline, and when used in conjunction with the gloss/translation lines from the Odin database, results in the best performance

[2] `https://web.archive.org/web/20160322220830/http://kyloo.net/software/doku.php/mgiza:forcealignment`

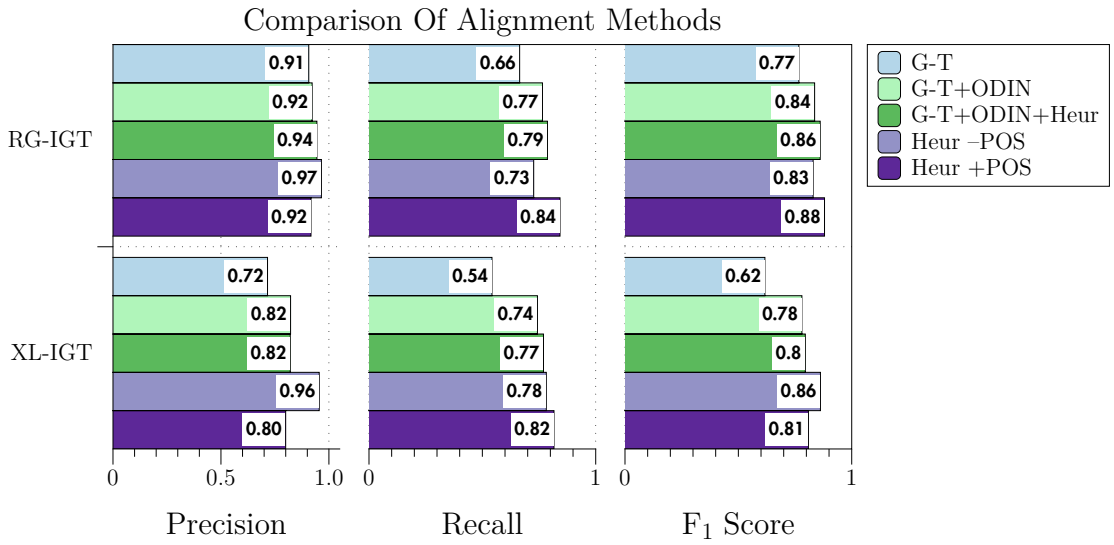


Figure 5.6: Precision, Recall, and F_1-score comparing alignment methods.

among the statistical systems.

5.5 Future Work

While the experiments here cover some of the ways in which IGT instances can be aligned, there are still other methods that I have not explored, and will briefly mention a few that may be attempted in future work.

5.5.1 "Clue-based" Alignment

Tiedemann (2003) described an interesting approach to word alignment, wherein heuristics are run between words in a language pair to build up a "clue matrix" that is used to determine alignment. Some of the heuristics include a dice coefficient, for identifying co-occurrence throughout the corpus; and the longest common sub-sequence ratio (LCSR), for identifying cognates and matching POS tags. Each heuristic is combined to produce a score for each word pair in the clue matrix, and then a Viterbi-like algorithm is used to select the best

combination of alignments from the matrix.

This seems to be an approach that would be extremely well-suited to combining a statistical analysis of the translation/gloss line with the strong string-similarity benefits, and the ability for the INTENT system to produce gloss-line POS tags.

5.5.2 Constrained Giza++ Search

A second approach outlined by Gao et al. (2010) plays to the strengths of being able to obtain high-precision alignments through IGT, while allowing for the weakness of incomplete coverage in the provided alignment examples. The essence of this approach is to use a source of high-precision alignments to constrain the EM hill-climbing of the word alignment. In brief, this constraint means that for an alignment $a = \{a_1, a_2, \ldots, a_j\}$ and partial alignments $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_j\}$ on the sentence pair f, e, the translation probability $t(f, a|e, \alpha)$ will be zero if the alignment a does not fit the constraints of the partial alignment α. This approach is extremely promising for being able to maintain the high precision of the alignments obtained by the heuristic approach, while recovering some of the unaligned tokens that such a statistical approach may recover.

5.5.3 Bootstrapping Alignment for Other Bitexts

With IGT containing language and translation lines, as well as being able to identify extremely high-precision word pairs through the use of the heuristic alignment methods, the resource provides both full parallel sentences, as well as likely word pairings between the English translation line and the target language. Although I have focused primarily on work aimed at very resource-poor languages in this thesis, there may be languages for which parallel corpora with English are available. For these languages, it would be possible to use the word pairs produced by the heuristic approach to seed the word alignment of a statistical alignment approach. Using the aligned word pairs from the IGT data in this way

might not improve performance for languages where a large amount of parallel sentences are available, but for languages where only minimal parallel corpora are available, this kind of bootstrapping might produce significant gains in word alignment performance.

5.6 Summary

In this chapter, I laid out two main approaches to finding word alignment within IGT instances; heuristic and statistical. I also looked into combining the approaches by feeding the results of the heuristic matches into the statistical data in order to combine the higher-recall statistical approach with the higher-precision heuristic method. In the end, the Heur +POS system achieved an F_1-score of 0.81 on the XL-IGT corpus data, and 0.88 on the RG-IGT data, to the best statistical system's 0.80 and 0.86. Given that both of these evaluation corpora are relatively small, these differences may not be particularly significant, but it seems that optimizing for high precision with the fairly noisy ODIN data achieves the most favorable word alignment results. While the heuristic systems with and without POS tags were largely similar in F_1-score, due to the Heur −POS system's superior precision, I will use this system as the default heuristic system elsewhere in this thesis.

Similarly, the `mgiza` software outperforms `FastAlign`, intersection works as the best symmetrization method, and the G-T+ODIN+Heur system outperforms the other statistical methods. I will use this combination of settings as the default for the statistical alignment method, referring to this combination elsewhere in the thesis as simply **Stat** for the G-T+ODIN setting, or **Stat+Heur** for the G-T+ODIN+Heur setting.

While aligning IGT instances may be helpful in developing word lists on its own, the primary reason for its inclusion here is as an upstream task for projection of linguistic information such as part of speech tags and dependency parsing. In the following sections, Chapter 6 and Chapter 7, I will use the results of these aligned IGT instances to perform further enrichment.

Chapter 6
PART-OF-SPEECH TAGGING

The next task I will describe is how IGT data may be used to perform part-of-speech (POS) tagging. POS tagging is a subset of many other annotation tasks, such as dependency parsing. Automatically created annotators that could apply POS tag labels to IGT data could be generally useful, such as in linguistic fieldwork, or large-scale typological studies.

In this section, I will describe two primary goals, the first being using IGT to produce POS tags for the target language line of IGT. The second goal is using this POS-tagged IGT to train POS taggers that can be used to tag novel, monolingual data. Within the first goal of obtaining POS tags for IGT instances, I will look at three approaches. The first of these approaches will be the standard projection-based IGT Tagger (Section 6.2), while the second will be an IGT Tagger based upon classification of the gloss line (Section 6.3). A third approach will look at how both of these first two approaches may be combined (Section 6.4). I will discuss research done as a case-study in using both of these approaches on an endangered languages in Section 6.5, and finally, how both of the approaches for obtaining language-line POS tags could be used to train monolingual POS taggers (Section 6.6).

6.1 Task Overview

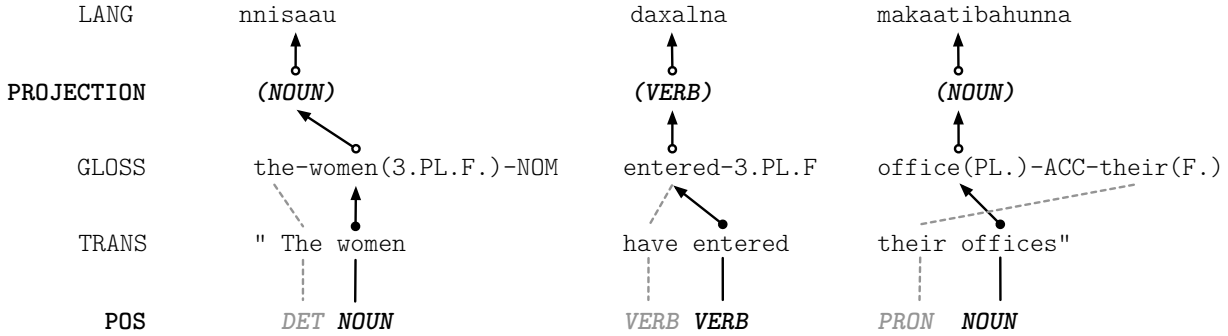
The tasks that I will be describing in this section will be the POS tagging of both the language lines of IGT instances from the RG-IGT corpora, as well as monolingual language data from the UD-2.0 corpus. Gold-standard IGT tags from the Petrov et al. universal tagset are available on both the RG-IGT and UD-2.0 corpora, and both tasks will be evaluated for their tagging accuracy on the gold-standard tags provided by the respective corpus.

LANG	nnisaau	daxalna	makaatibahunna
GLOSS	the-women(3.PL.F.)-NOM	entered-3.PL.F	office(PL.)-ACC-their(F.)
TRANS	"The women	have entered	their offices."

Instance 6.1: An Arabic (ara) IGT instance of the sentence "*The women have entered their offices*".

To give a concrete example of how IGT may be used to accomplish POS tagging in these settings, Instance 6.1 shows a typical IGT instance in Arabic. In this example, there are several English words that are shared between the gloss and the translation, and the gloss is extensively marked-up. A linguist reading this example will immediately notice that there is a substantial amount of information provided about each word in the Arabic.

Section 3.1.2 gave a high-level overview of using a projection algorithm to use English-language tools on translation line then transfer this annotation to the language line. Instance 6.2 illustrates just such a case, where the POS line at the bottom shows the POS tags assigned by an English-language POS tagger. These tags are associated with the English words on the TRANS line, which in turn are aligned with portions of the tokens on the GLOSS



Instance 6.2: An illustration of the IGT instance from Instance 6.1 where the translation has been POS tagged, and then projected via the gloss line.

LANG	Di Gianni	rimpiango	di non	essermi	ricordata
GLOSS	of Gianna	regret	of not	be-SE.1SG	remembered.F.SG
TRANS	"Of Gianni	I regret	not	having	**thought of**."

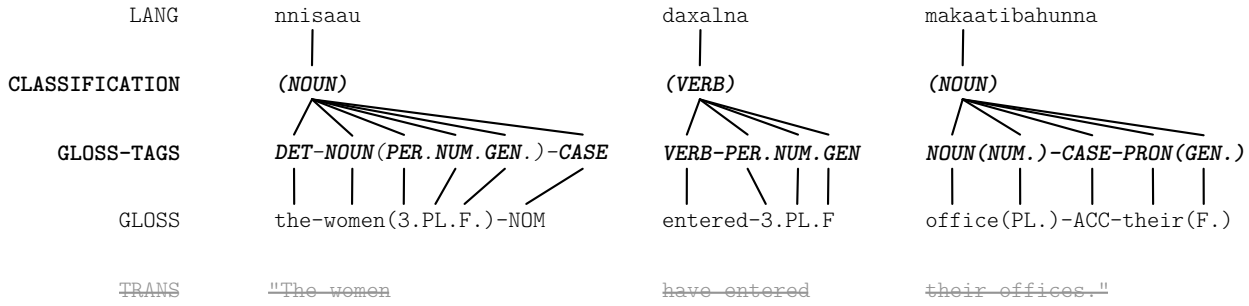
Instance 6.3: An example of IGT from Italian (`ita`) where gloss and translation words do not match precisely.

line. The `PROJECTION` line serves to indicate that the overall POS tag for the gloss token has been assigned to the given POS tag associated with one of its elements, and this tag is ultimately transferred up to the `LANG` line.

Projection via this method is not unique to IGT, and has been used in other studies on projection that focus predominantly on parallel texts, such as Hwa et al. (2004) and Yarowsky and Ngai (2001). What is unique to using IGT as a data source is how the gloss line is able to aid in aligning the translation and language lines with a far lower data requirement than methods that require statistical alignment.

This method does have downsides, though. In Instance 6.2, only those parts of the gloss that match the translation line are utilized in the projection process, while there is a far greater amount of text available. Second, as shown in Instance 6.3, sometimes the translation between the gloss and translation lines are not exact, and so words that might otherwise be used to project might be skipped.

It is for reasons such as these that I worked to extract more information from the gloss line in the POS tagging task. Instance 6.4 gives an alternative approach to finding information for part-of-speech tagging by looking directly at the gloss line itself, shown by the layer labeled `GLOSS-TAGS`. The intuition here is that the content on the gloss line is a combination of English words and **grams**—elements which specify grammatical features of the word on the language line. The first word in the Arabic in Instance 6.1, *nnisaau*, is marked on the gloss line as *the-women(3.PL.F.)-NOM*. Aside from the root, *women*, each of the tokens *3*,



Instance 6.4: The Arabic (`ara`) IGT from Instances 6.1 and 6.2, but using a `GLOSS-TAGS` layer to label the elements of the `GLOSS` line directly, and use that information to make a label choice.

PL, and *F* indicate inflection for person, number, and gender, while the initial *the-* appears to indicate a form of definite marker, with *-NOM* marking the nominative case. Collectively, I will refer to both grams and words within a single whitespace delimited token as subwords.

Looking at these subwords, I sought to exploit my intuition that even when the translation does not align perfectly with the gloss, the gloss can be used as a resource on its own to gain information on the words in the language line, both independently from, and in conjunction with, the translation line. I will discuss how I implemented this approach further

Corpus	**#Tokens**	**#Types**	**Average** $\frac{Tags}{Type}$
German Instances in XL-IGT	739	321	1.13
NEGRA Corpus	33,133	8,890	1.04
TIGER Corpus	34,093	8,726	1.04
UD-2.0	30,460	7,015	1.05
European Corpus Initiative Multilingual Corpus	333,882	15,755	N/A

Table 6.1: A comparison of the count of words and unique words in the German data of the XL-IGT corpus and other German-language corpora, illustrating the sparsity of IGT.

in Section 6.3.

One other concern in the POS tagging task when IGT is used is that of data sparsity, and particularly, the out-of-vocabulary (OOV) rate. As Table 6.1 shows, the number of German words covered by IGT data is extremely sparse, with only 321 unique words compared to the thousands in other German-language corpora. This means that, if the IGT data were to be used directly to train a POS tagger, it would have to contend with an extremely high OOV rate. Thus, maximizing the number of words that are tagged in the IGT data is very important, particularly by expanding the number of POS tags that can be recovered when alignment cannot be obtained. This concern is also one that the alternative approach presented in section Section 6.3 seeks to address.

6.2 Projection-Based Tagging

To begin the discussion of my implementation of POS tagging, I will first focus the method that has been traditionally been used for transferring annotation with bilingual corpora:

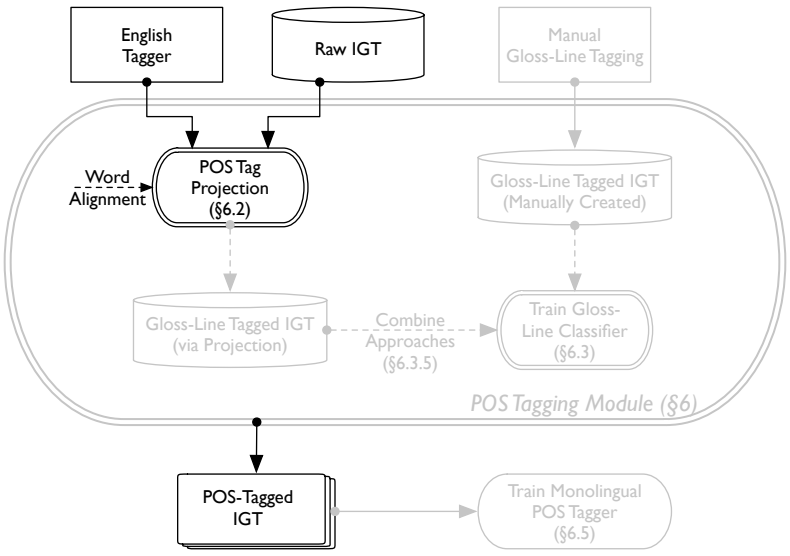


Figure 6.1: Flowchart of the high-level overview of the POS tagging module, with the projection pathway highlighted.

projection-based POS tagging. As shown by Fig. 6.1, this method requires only a set of raw IGT instances, translation line POS tags, and gloss-to-translation-line word alignment. Using the conventions from Appendix A, I use F for the language line, E for the translation line, and P_E and P_F for the part-of-speech tags for each.

The raw IGT instances may be obtained through ODIN, or any of the other IGT-containing corpora in Chapter 4. The part-of-speech tags to project from the translation line may be obtained either through manual labelings, such as are available in the RG-IGT corpus; or from an English-language part-of-speech tagger, such as the Stanford Part-of-Speech Tagger (Toutanova et al., 2003). The third element, the gloss-to-translation line alignment A, can be any of the methods described in Chapter 5.

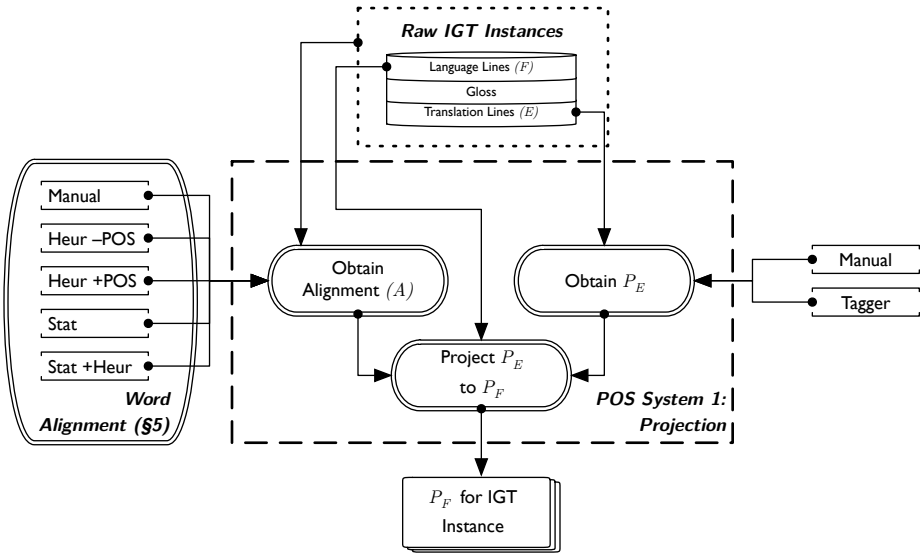


Figure 6.2: Flowchart illustrating the pathway from raw IGT to tagged language line (P_F) using IGT projection.

6.2.1 Projection-Based Tagging Algorithm

The basic mechanism for projecting part-of-speech tags using IGT is illustrated in Fig. 6.2, and the pseudocode is given in Algorithm 6.2.1. The English POS tagger and gloss/trans aligner are represented in the pseudocode by the functions `engTag(`*Sentence*`)` and `getAlignedTrans(`*GlossWord*`)`. As IGT instances are encountered, they are separated into translation and gloss lines (Lines 7 and 9), and the translation line is tagged (Line 8). Iterating over the gloss words, they are examined for alignment with the translation line (Line 11) and if an aligned translation word exists, it is assigned the translation word's tag (Lines 12 to 14), otherwise it is assigned the unknown `'UNK'` tag (Lines 15 to 16). Since gloss and language lines have a word-to-word alignment, the language word is obtained this way (Line 17). With the language word available, and the gloss word's projected or `'UNK'` tag, this (*Word*, *Tag*) pair is collected into the tagged language sentence (Line 19) and this sentence is added to the corpus (Line 20).

6.2.2 Projection-Based Tagging Experiments

Table 6.2 shows the accuracy of the projection algorithm in a number of different settings. The **Alignment Type** columns distinguish whether the alignment between translation and gloss used for the projection was from the heuristic system or manual alignments. The **POS Tag Source** columns refer to whether the tags being projected from the translation line are the manually labeled **Gold** tags, or the automatically produced **Tagger** tags.

It is worth noting that, even when gold alignment data is used with gold POS tags, the projection algorithm achieves an accuracy of 90% overall, while the usage case that more readily simulates real-world performance on an unknown language—heuristic alignment and automatic tags—achieves only 66.8% overall accuracy. While these results would still be promising for a language with no other resources, these figures highlight some of the weaknesses of the projection algorithm for POS tagging.

Algorithm 6.2.1: Projection-Based Tagging Algorithm

```
   input  : IgtCorpus ;                                  // This corpus is pre-aligned
            1 engTag(Sentence)
            2 getAlignedTrans(GlossWord)
   output : PosTaggedCorpus

 3 Function projectTags:
 4     Let PosTaggedCorpus be new Corpus;
 5     foreach Instance in IgtCorpus do
 6         Let TaggedLanguageSentence be new Sentence;
           /* Get the translation line from the instance and tag it */
 7         TransLine ⟵ getTrans(Instance) ;                    // Get the trans line
 8         TransTags ⟵ engTag(TransLine) ;                          // ...and tag it
 9         GlossLine ⟵ getGloss(Instance) ;                    // Get the gloss line

10         foreach GlossWord in GlossLine do
11             TransWord ⟵ getAlignedTrans(GlossWord) ;
12             if GlossWord has an aligned TransWord then
                   /* If the gloss is aligned, assign the trans tag to it */
13                 TransTag ⟵ getTagFor(TransWord, TransTags) ;
14                 GlossTag ⟵ TransTag;
15             else
                   /* Else if unaligned, assign the 'unknown' tag */
16                 GlossTag ⟵ "UNK" ;
17             LangWord ⟵ wordToWordAlignment(GlossWord) ;
18             LangTag ⟵ GlossTag;
19             addToSentence( TaggedLanguageSentence, (LangWord, LangTag) ) ;
20         addToCorpus(PosTaggedCorpus, TaggedLanguageSentence);
21     return PosTaggedCorpus;
```

Projection Algorithm Accuracies

Alignment Type	**Gold**		**Heuristic**	
POS Tag Source	**Gold**	**Tagger**	**Gold**	**Tagger**
Bulgarian	90.5	95.2	71.4	76.2
French	90.6	80.2	71.9	63.0
German	90.9	83.9	74.8	69.4
Italian	94.4	88.9	66.7	61.1
Spanish	80.0	69.1	65.5	60.0
Overall	90.0	82.1	72.8	66.8

Table 6.2: POS tagging accuracies for the projection algorithm on the five languages in the RG-IGT corpus. The table shows results for projection using either gold word alignments or heuristic alignments, and whether the translation-line POS tags were the gold tags or generated by the Stanford POS tagger (Toutanova et al., 2003).

6.2.3 Problems with Projection-Based Tagging

As mentioned in Section 6.1, while this method does produce a POS-tagged language line, it has several major limitations. First, as illustrated in Instance 6.1, when an English translation word does not share a root with the word or words on the gloss line it is translating, it may not align, thus leaving the projected tag as `'UNK'`.

A second source for problems is when the English translation shares a root with the gloss, but the part-of-speech associated with the translation differs from that of the glossed word,

```
borda kA   gaTana    kiyA    gayA
board of   formation do-perf go-perf
The board was formed
```

Instance 6.5: Example Hindi (`hin`) IGT where the Hindi *gaTana* ("formation") is a `NOUN` whereas the English *formed* is a `VERB`. The words share the same root, so the `VERB` tag will be incorrectly projected to *gaTana* from the translation line using the heuristic alignment approach.

such as in Instance 6.5. In this case, in an attempt to find the best match for the word, the heuristic approach will project a tag from the translation line that does not match the intent of the word on the gloss line, despite the shared root.

Finally, projecting POS tags has the problem of deciding which of multiple aligned words decides the tag for a gloss word. As Instance 6.1 shows, *the-women(3.PL.F.)-NOM* aligns with both *The/DET* and *women/NOUN*, while *office(PL.)-ACC-their(F.)* aligns with *their/PRON* and *offices/NOUN*. Deciding how to assign which aligned word is problematic. The algorithm solves this problem by using a list of precedence to favor what are likely content words over potential function words. The list used is:
VERB > NOUN > ADV > ADJ > PRON > DET > ADP > CONJ > PRT > NUM > PUNC > X.
While solution does address the concern, it is rule-based and prone to error.

For reasons such as these, it is attractive to consider a classification-based approach, where more features can be considered for a given word, and every word is assigned a tag.

6.3 Classification-Based Tagging

An alternative to using a projection-based approach is to use one in which the gloss line is annotated without requiring alignment with the translation line, and the word-to-word alignment between the gloss and language lines is used to assign the gloss line decisions to the language line. In order to accomplish this, I use a classifier that works on each of the gloss line words. While it would be possible to consider a Hidden Markov Model (HMM) or Conditional Random Field (CRF) model that is context-sensitive for this approach, if we treat the gloss lines of IGT as a pseudo-language, this pseudo-language will vary widely in its ordering between IGT language samples, and so predicating decisions based upon context that is trained upon this multilingual source may not work well for all languages.

Figure 6.3 shows the high-level overview for this approach. Unlike the projection step, which can be performed on IGT instances without anything other than off-the-shelf tools,

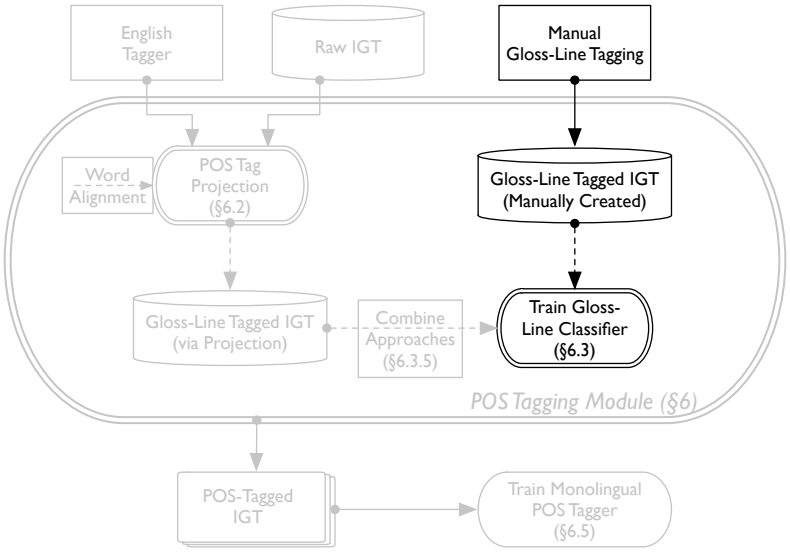


Figure 6.3: High-level overview of the classifier-based POS tagging approach that functions directly on the gloss line of an IGT instance.

this classification approach requires a training set of POS tags on the gloss line. While plenty of annotated data exists to train a POS classifier on English, IGT glosses are instead an English-like pseudo-language.

6.3.1 IGT Gloss-Line Annotation

Both to serve as a method by which to train the classifier, as well as evaluate it, I created the RG-IGT corpus mentioned in Section 4.1.3, consisting of 141 instances over 102 separate documents with manual POS labels on translation, gloss, and language lines. I made an attempt to sample IGT instances from multiple different ODIN source documents, so that the bias of a focus to only a few specific constructions was avoided. In addition, I spread the annotation over a number of languages such that annotation pertaining to constructions unique to different languages might be obtained.

Table 6.3 shows how many tokens and types were annotated for each language on gloss, language, and translation lines, as well as the average tags per type for each language, also

Language	Lang Tokens	Lang Types	Avg $\frac{Tags}{Type}$	Gloss Tokens	Gloss Types	Avg $\frac{Tags}{Type}$	Trans Tokens	Trans Types	Avg $\frac{Tags}{Type}$
Bulgarian	20	20	1.00	44	44	1.00	32	25	1.33
French	208	114	1.89	224	132	1.71	84	56	1.53
German	346	221	1.62	365	256	1.46	270	159	1.74
Italian	22	19	1.16	28	27	1.08	18	16	1.13
Spanish	83	75	1.15	69	64	1.08	66	58	1.16
Overall	679	449	1.56	730	523	1.42	470	314	1.53

Table 6.3: Breakdown of all IGT instances in the RG-IGT corpus by token and type.

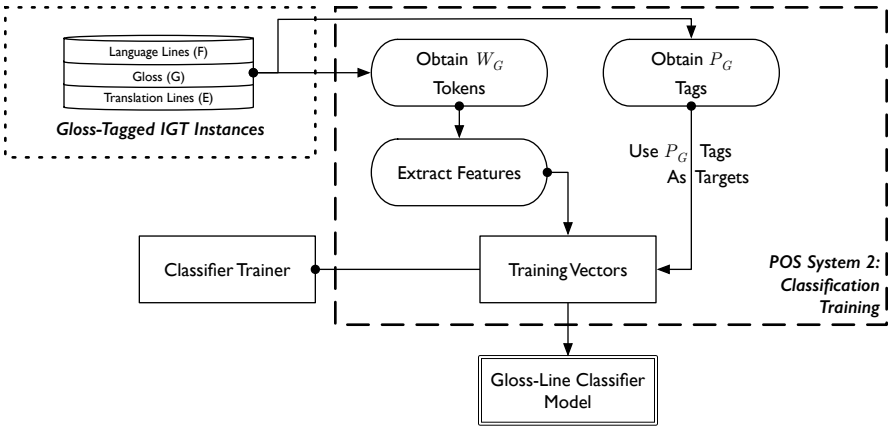


Figure 6.4: Flowchart illustrating how the classification-based approach works, both in training the gloss-line classifier and applying it to test IGT instances.

broken down by IGT line. With labels now obtained for 730 gloss tokens, the classifier can be trained on this data. Figure 6.4 shows a flowchart outlining the training and testing steps of this approach, and Algorithms 6.3.1 to 6.3.3 give the pseudocode.

6.3.2 Training

After the IGT instances have been annotated, the next steps as shown in Fig. 6.4 are to extract the features from the annotation and train a classifier for future gloss instances. Section 6.3.3 will explain the features and feature extraction step in full, but aside from this, as shown in Algorithm 6.3.1, the training step is rather simple. Line 2 instantiates a new

Algorithm 6.3.1: Classification-Based Tagging Algorithm: Training Phase

```
input  : GlossAnnotatedIgtCorpus
         ClassifierTrainer
         runTrainer(Trainer, FeatureVectorCollection)
         getFeatures(Word)
         addTrainingInstance(FeatureVector, Target, FeatureVectorCollection)
output : ClassifierModel

1 Function trainGlossClassifier:
2     Let GlossTrainingVectors be new FeatureVectorCollection;
3     foreach Instance in GlossAnnotatedIgtCorpus do
4         AnnotatedGlossLine ⟵ getGloss(Instance);
5         foreach GlossWord, GoldGlossTag in AnnotatedGlossLine do
6             InstanceVector ⟵ getFeatures(GlossWord, Instance, EnglishDictionary);
7             addTrainingInstance(InstanceVector, GoldGlossTag, GlossTrainingVectors);
8     Let GlossClassifierModel ⟵ runTrainer(ClassifierTrainer,GlossTrainingVectors);
9     return GlossClassifierModel;
```

Algorithm 6.3.2: Classification-Based Tagging Algorithm: Feature Extraction

```
input  : GlossWord
         IgtInstance
         EnglishDictionary
output : FeatureVector
1 Function getFeatures:
2     Let FeatureVector ⟵ be new FeatureVector;
3     Alignment ⟵ getAlignment(IgtInstance);
4     foreach featureFunction() in ActiveFeatures do
          /* Args are optional arguments, e.g.  Alignment and EnglishDictionary  */
5         featureValue ⟵ featureFunction(GlossWord, ... Args) addToVector(FeatureVector,
          featureValue)
6     return FeatureVector
```

collection of training vectors to feed to the classifier trainer. In my experiments I used a MaxEnt classifier, but any classifier may be used. Lines 3 to 4 iterate over all the instances and extract the gloss line, while Lines 5 to 7 iterate over the gloss tokens on the gloss line and add feature vector extracted from each token to the training vectors. Finally, Lines 8 to 9 use the classifier trainer to train a model from the input vectors, and return the trained model.

6.3.3 Features

In order to train a classifier, features must be extracted for the training instances. Table 6.4 contains a full list of features used in the classifier. Most features take only the current gloss $Word$ as an argument, but `alignedTag` takes an *Alignment* between the gloss line and translation line, and a set of $TranslationTags$ for the POS tags on the translation line. Algorithm 6.3.2 shows the pseudocode for how features are extracted at both training time and testing time.

While some features can be extremely reliable, others are not particularly helpful. In order to understand what features help the classifier the most, I conducted a test by enabling each of the features in Table 6.4 individually and in combinations, the results of which can be seen in Table 6.5.

In this test, each feature was used in isolation to train the classifier, except where combinations of features are indicated. For comparison, the performance of the projection method across the entire dataset is provided, as is the result of assigning the most frequent tag (`NOUN`).

The `subWords` feature alone was quite successful, achieving a 79.5% accuracy by itself. Interestingly, the `prefix` feature achieved a higher performance, at 83.6%. The projection-based `alignedTag` feature also performed rather well, though as gold tags are not available for all translation lines, the feature fired far less frequently when using gold tags, and thus

`subWords(`*Word*`)`
Simply return, as features, each subword contained in the word.
Example: `'Woman.3sg.FEM'` → {`'Woman'`, `'3sg'`, `'FEM'`}

`alignedTag(`*Word*, *Alignment*, *TranslationTags*`)`
Given the current word, and it's alignment with the translation line, return the POS tag of the word(s) with which it is aligned. This feature also must be provided with an alignment to the translation line and POS tags for the translation.
Example:

```
GLOSS     the-women(3.PL.F.)-NOM
            |       \
TRANS     "The/DET women/NOUN
```

`'the-women(3.PL.F.)-NOM'` → { `'DET'`, `'NOUN'` }

`wordHasNumber(`*Word*`)`
Return `1` if there is a number somewhere in the word, `0` otherwise.
Example: `'3sg.FEM'` → `1`; `'dog.NOM'` → `0`

`suffix(`*Word*`)`
Return up to the last three characters of the word.
Example: `'leading'` → {`'ing'`, `'ng'`, `'g'`}.

`prefix(`*Word*`)`
Similar to suffix, return up to the first three characters of the word.
Example: `'disavow'` → {`'d'`, `'di'`, `'dis'`}.

`numSubwords(`*Word*`)`
Return an integer value of the number of subwords contained in the word.
Example: `'Women.3sg.FEM.PL'` → `4`

`prevSubwords(`*Word*`)`
Return, the subwords contained in the preceding word. The same as calling `subWords(`*PrevWord*`)`
Example: { `PREV: '3sg.FEM' --- CURRENT: 'dog.NOM'` } → {`'3sg'`, `'FEM'`}

`nextSubwords(`*Word*`)`
Return the subwords contained in the following word. The same as calling `subWords(`*NextWord*`)`
Example: { `CURRENT: '3sg.FEM' --- NEXT: 'dog.NOM'` } → {`'dog'`, `'NOM'`}

`dictTag(`*Word*`, `*Dict*`)`
Using a dictionary generated by scanning the words in the Penn Treebank, return the most common POS tag for each subword found in the dictionary, if it is found. (In this example, no tag is found for `PL`).
Example: `'the.women.PL'` → {`'DET'`, `'NOM'`}

`prevDictTag(`*Word*`, `*Dict*`)`
Same as dictTag, but for the previous word.
Example: { `PREV: 'woman.3sg.FEM' --- CURRENT: 'went.PRES'` } → {`'NOUN'`}

`nextDictTag(`*Word*`, `*Dict*`)`
Same as dictTag, but for the following word.
Example: { `CURRENT: 'woman.3sg.FEM' --- NEXT: 'went.PRES'` } → {`'VERB'`}

Table 6.4: List of features used in the classifier. While most features simply take a *Word* as an argument, the `alignedTag` feature takes an additional *Alignment* and *TranslationTags*, and the `dictTag` takes an additional *Dict* argument.

	Feature or Ensemble	Train	Test
Baselines	Most Common Tag (NOUN)	27.9%	26.9%
	Projection(H,A)	75.1%	
Basic Features	subWords	86.3%	79.5%
	numSubwords	31.2%	31.5%
	wordHasNumber	31.5%	38.4%
Alignment	alignedTag(H,A)	78.5%	82.2%
	alignedTag(H,G)	60.3%	63.0%
	alignedTag(G,A)	84.5%	89.0%
	alignedTag(G,G)	64.4%	67.1%
Affixes	suffix	84.0%	65.8%
	prefix	88.7%	83.6%
	suffix + prefix	96.7%	80.8%
Context	prevSubWords	60.9%	47.9%
	nextSubwords	66.1%	37.0%
	prevSubwords + nextSubwords	84.2%	60.3%
Dictionary	prevDictTag	42.8%	47.9%
	nextDictTag	40.5%	46.6%
	dictTag	85.8%	89.0%
	prevDictTag + nextDictTag	53.4%	56.2%
	prevDictTag + nextDictTag + dictTag	88.6%	89.0%
Best System	subWords + alignedTag(H,A) + suffix + prefix + prevSubwords + nextSubwords + prevDictTag + nextDictTag + dictTag	**99.8**%	**94.5**%

Table 6.5: Feature tests for features listed in Section 6.3.3 on RG-IGT corpus. The performance on a training set and test set are given using a 90/10% split. For the `alignedTag` feature, the arguments provided show the $(AlignmentType, TagType)$ provided to the function, where `'H'` and `'G'` represent **H**euristic alignment and **G**old alignment respectively, while `'A'` and `'G'` likewise represent **A**utomatically labeled tags, and **G**old tags.

suffered a performance hit.

The `dictTag` feature also performed extremely well, showing the helpfulness of the English-language words on the gloss line, similar to the `alignedTag` feature when using gold alignments and automatic tags.

Finally, the Best system, chosen by testing all feature combinations and choosing the highest performing, consists of a large ensemble of features to achieve an extremely high accuracy of 94.5%, a 78% error reduction over the projection method, and a 73% reduction over using the basic features alone.

6.3.4 Running the Classifier on New Instances

Now, having trained a classifier capable of labeling gloss-line instances with their POS labels, the classifier can be run upon previously unannotated IGT instances as shown in Fig. 6.5, and implemented in Algorithm 6.3.3. In the algorithm, Line 7 uses the same `getFeatures` function that was described previously to extract features, and Line 8 uses the classifier produced in the training phase to produce the POS label for the current gloss word. Finally, the word-to-word alignment is again utilized in Line 10 to assign this label to the language word.

6.4 Combining Projection and Classification

While the creation of the RG-IGT corpus was originally intended to serve as a small set of labeled instances for both training and evaluation of the gloss-line classifier, there is an alternative method of obtaining the gloss-line POS tags for training, and that is by using projection over the entire Odin corpus. Figure 6.6 shows this revised training step.

Although the projection-based approach leaves many gaps when words are not aligned, and thus has low overall accuracy for POS tagging, the classifier approach does not require complete sequences, but can instead use only the tokens for which projection is successful.

Algorithm 6.3.3: Classification-Based Tagging Algorithm: Testing Phase

```
input  : GlossClassifierModel
         IgtCorpus
         EnglishDictionary
         classifyWord(FeatureVector, Model)
output : PosTaggedCorpus

1  Function testPosClassifier:
2      Let PosTaggedCorpus be new Corpus;
3      foreach Instance in IgtCorpus do
4          Let TaggedLanguageSentence be new Sentence;
5          GlossLine ⟵ getGloss(Instance);
6          foreach GlossWord in GlossLine do
7              Features ⟵ getFeatures(GlossWord);
8              GlossTag ⟵ classifyWord(Features, ClassifierModel);
9              LangTag ⟵ GlossTag
10             LangWord ⟵ wordToWordAlignment(GlossWord, Instance, EnglishDictionary);
11             addToSentence(TaggedLanguageSentence, (LangWord, LangTag));
12         addToCorpus(PosTaggedCorpus, TaggedLanguageSentence);
13     return PosTaggedCorpus;
```

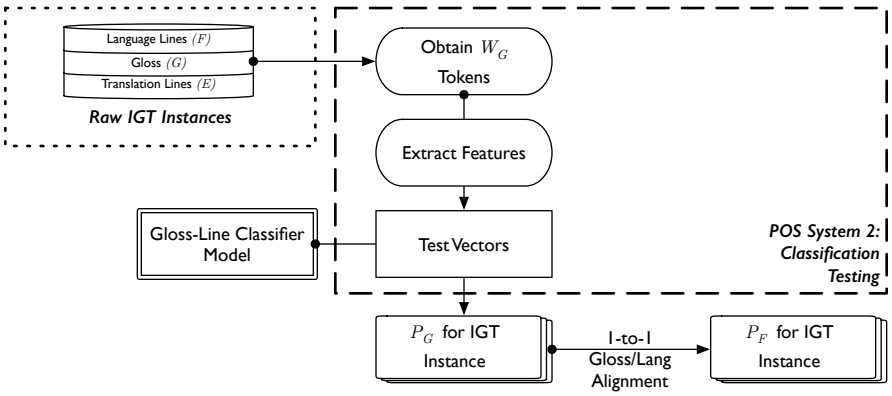


Figure 6.5: Flowchart illustrating the testing step of the classification-based tagging approach.

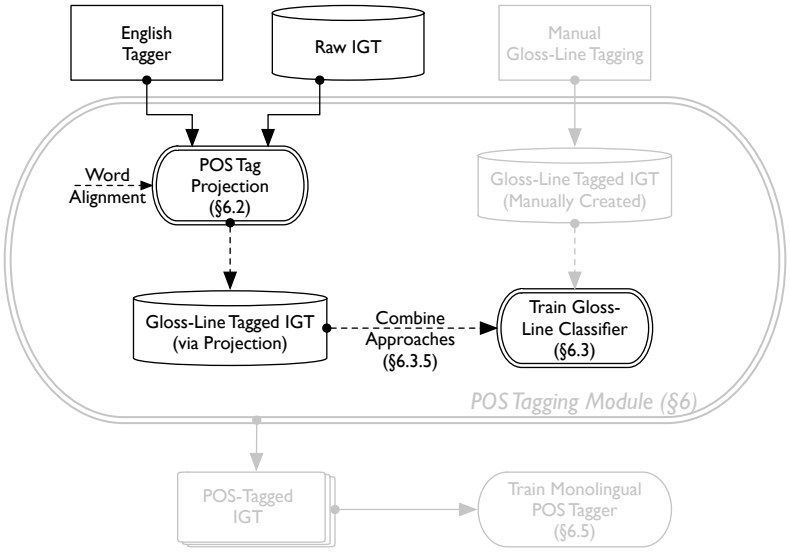


Figure 6.6: Flowchart of classification-based POS tagging training step using projected labels in place of gold-standard labels (Fig. 6.4).

Using this approach would thus potentially expand the number of training instances from the 432 tokens in the RG-IGT corpus to the 231,602 in the ODIN corpus, though with the potential of introducing noise of various types mentioned in Section 4.4.

6.4.1 Results on IGT Data

Figure 6.7 shows the results of the classification approach either using training data from 90/10% splits and 10-fold validation using the RG-IGT corpus (labeled "Manual"), or using projected POS tags from the full ODIN corpus (labeled "ODIN").

As the figure shows, the "Manual" system, despite having many fewer training instances, performs better on the test data than does the "ODIN" system. Given that the set of training data is very close to the test data, covering the same languages and being likely biased by use of the similar set of subwords, this result is not particularly surprising, and should not be given too much weight. While both classifier-based approaches show substantial improvement over the projection method, the amount of evaluation data available in these

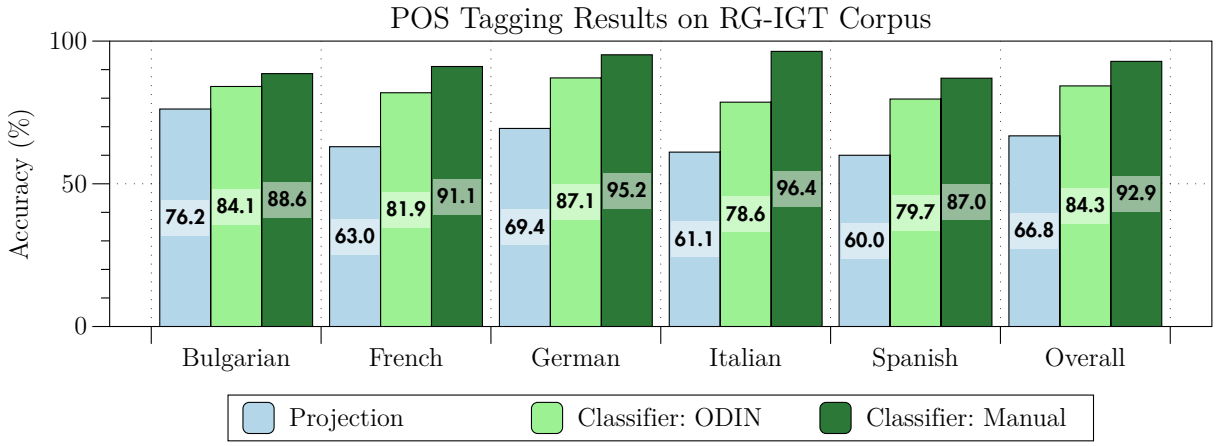


Figure 6.7: POS tag accuracies of the projection-based and classifier-based approaches, using the RG-IGT corpus for evaluation. The classifier built using the manually-created RG-IGT corpus is labeled "Manual" and the classifier built using automatically projected POS tags over the entire ODIN corpus is labeled "ODIN."

experiments is limited. The following sections Section 6.5 and Section 6.6 will discuss POS tagging experiments on a more varied set of evaluation data.

6.5 A Case study in Projection Methods: Part-of-Speech Tagging Chintang

While testing on the languages in the RG-IGT corpus provided some amount of diversity in the languages being tested, all the languages still belonged to the Indo-European language family, and so I also sought to perform part-of-speech tagging on a non-Indo-European language for which a very clean collection of IGT data was available, namely, the Chintang (`ctn`) corpus (Bickel et al., 2009)[1]. Chintang was a good candidate to study, because in addition to being an endangered language without traditional computational resources, no language-specific data had been created for this language for any of the POS tagging approaches thus far. For these reasons, running the POS tagging experiments on Chintang

[1]My work on this corpus was previously published as Georgi et al. (2015)

Method	Accuracy	% Unaligned
Basic Mapping	12.6	83.8
Extended Mapping	39.6	57.1

Table 6.6: POS tag accuracies for the projection-only approach on Chintang data.

would represent a proof-of-concept for other resource-poor languages.

As shown in Section 4.1.6, the Chintang IGT data from Bickel et al. (2009) consisted of 8,695 IGT instances, converted into the *Xigt-XML* format (Goodman et al., 2014), for use in Bender et al. (2014). Experiments included running POS tagging, both via projection and classification approaches. Gold-standard POS tags come from the Chintang corpus itself, though with mappings as will be discussed in Section 6.5.2.

6.5.1 Projection-Based Tagging

POS tagging experiments were first attempted using the projection-based methods from Section 6.2, but as shown in Table 6.6 shows, an initial 12.6% POS tagging accuracy quickly suggested an issue with the data.

Looking more closely at the Chintang data, as Table 6.6 shows, a great deal more gloss-line tokens were unaligned than was typical for IGT data. The reason for this unusual amount of unaligned instances was due to the particular conventions of the Chintang data where the overlap between words used in the gloss line and the words used in the translation

```
hun-ko-i      tis-u-m           pache
DEM-NMLZ-LOC  put.into-3P-1/2nsA SEQ
(pro)         (vt)              (gm)
after putting dal or arum
```

Instance 6.6: An instance from the Chintang (`ctn`) development set, showing the lack of alignment between gloss line and translation line, as well as the gold standard POS tags.

line were far fewer, and some gloss tokens consisted only of grammatical features. This lack of overlap is illustrated in Instance 6.6. In this particular instance, only the gloss `'put'` has a matching segment on the translation line, `'putting'`. Most of the rest of the gloss-line tokens provide grammatical information, such as `'DEM'` (demonstrative).

This lack of overlap is crucial, since when words are unaligned, projection cannot happen. Thus, 83.8% unaligned tokens means an upper bound on accuracy of 16.2%, using the alignment heuristics developed in Section 5.2.

6.5.2 Subword Mapping

The tag `gm` in the Chintang data set refers to *grammatical markers*. It seems that while some words tagged `gm` do not have counterparts in English (e.g., a TOPIC marker), others are listed as the English words *and*, *or*, or *but*.

In order to see if projection accuracy could be improved, I attempted to map the top twelve frequently seen subwords in the gloss line that had been labeled with the `'gm'` POS tag. Table 6.7 gives the list and frequencies of these twelve gloss tokens, which show that *BUT* and *and* were labeled as `'gm'` in the gold standard—words that would be labeled as `'CONJ'` in projection, if they were seen on the translation line. As shown in Table 6.6, this mapping dropped the number of tokens that were unable to receive tags from 83.8% to

Gloss Word	**# Tokens**	**Tag**	**# Tokens**
FOC	1049	CIT	360
TOP	1027	REP	243
SEQ	855	and	237
ADD	621	SURP	236
EMPH	504	SPEC.TOP	223
BUT	365	COND	207

Table 6.7: Top twelve gloss tokens labeled *gm* in the CTN corpus sorted by decreasing frequency.

57.1%, and raised the tagging accuracy from 12.6% to 39.6%.

For classification, these `'gm'` tags are added to the existing dictionary of Penn Treebank word-tag pairs used for the `dictTag` feature described in Table 6.4. This expanded dictionary would now contain CTN-specific (gloss-token, tag) pairs in addition to the high-frequency tags for other English words contained by the gloss-line token. This dictionary is then used to provide the best-guess tag feature to the classifier at training and test time.

6.5.3 Classification-Based Tagging

In the next set of experiments, the classifier was run using four different settings regarding the training data for the classifier. The classifier first used was the classifier produced on the full ODIN corpus, using projected tags, as described in Section 6.4. No instances from the Chintang corpus were used for training the classifier in this approach.

For the second classification based approach, I trained the classifier with the training portion of the Chintang corpus, ignoring the gold-standard POS labels and instead using projected POS tags, as was the case in the first scenario. The intention of this experiment was to see what improvement using instances specific to Chintang might have, since many of the gloss line tokens in the Chintang corpus were rarely, if ever, seen in much of the original ODIN database. Additionally, since all the instances used to train the classifier were coming from the same language and being sensitive to word order might be beneficial, I used this experimental setting to test whether adding context features, such as the `prevSubwords` and `nextSubwords`, would improve performance.

The third approach combined the ODIN and Chintang training data. This setting was designed to test how the addition of the Chintang instances would affect a model trained primarily on instances from other languages. The fourth approach used the training portion of the CTN data, this time utilizing gold-standard POS tags rather than projected tags.

Two additional parameters were included for the training procedure; the adding of the

Training Data	Expanded Dictionary	Context Features	Accuracy
ODIN			43.1
	✓		53.0
CTN			75.0
	✓		74.9
		✓	74.8
	✓	✓	74.9
CTN+ODIN			61.6
	✓		70.7
	✓	✓	72.6
Supervised (with Labeled CTN)			89.6
	✓		90.6
	✓	✓	90.1

Table 6.8: POS tag accuracies for classification-based method on CTN test data comparing different sets of training data and classifier features. The "Expanded Dictionary" refers to adding the `gm` tokens from Table 6.7 to the `dictTag` lookup dictionary. "Context Features" are the features labeled "Context" in Section 6.3.3.

twelve most frequent `'gm'` labeled tokens from Table 6.7 to the dictionary used by the `dictTag` feature described in Table 6.4, labeled in the results in Table 6.8 as "Expanded Dictionary," and the inclusion of contextual features, due to the target of this particular training being focused upon a single language, rather than the large variety of languages the classifier is typically intended to cover.

The results of the classification experiments are shown in Table 6.8, where the ODIN-trained classifier shows a poor performance of 43.1% with the standard settings–better than the 39.6% displayed by the projection-based approach, even with remapping, but still far short of the 94.5% demonstrated on the RG-IGT data by the feature ablation test in Table 6.5. Adding the twelve `'gm'` tagged tokens to the expanded dictionary improved performance an absolute ten percent, but this result was still far lower hoped for.

Training instead on the CTN data with projected POS tags improved the classification

drastically, reaching up to 75%. Adding the expanded dictionary and contextual features appeared to have no significant effect, possibly because the amount of training data available was so high already.

The combination of Chintang data and ODIN training data resulted in a tagging accuracy of 61.6%. This decrease in performance is expected given that the classifier model that results is less targeted toward this specific data. Adding the expanded dictionary again has a substantive effect, improving accuracy from 61.6% to 70.7%, and the contextual features generated from using the Chintang training data do help compensate further, bringing the accuracy for this set of data up to 72.6%. Finally, using the gold-standard POS tags from the Chintang data show an 89.6%–90.6% range, with the additional features again showing little effect.

6.5.4 Varying the Amount of Data

Finally, I also wanted to see what performance one might expect to see with various amounts of available IGT training data. Figure 6.8 shows the result of this experiment, comparing the classifier over a range of training data sizes, from only 50 tokens to nearly 32,000 tokens.

In the graph, we see that the classifier using the expanded dictionary achieves almost 75% accuracy with approximately 400 tokens, coming very close to the supervised approach at around 77%. With an increased amount of data, the unsupervised methods converge at 75%, and the supervised method approaches 90%. While the supervised method clearly outperforms the projection-based methods, the fact that the classifier-based method, without context features, can achieve 75% with fewer than 500 tokens is more representative of the typical amount of IGT data available for a language, and so is a very promising result.

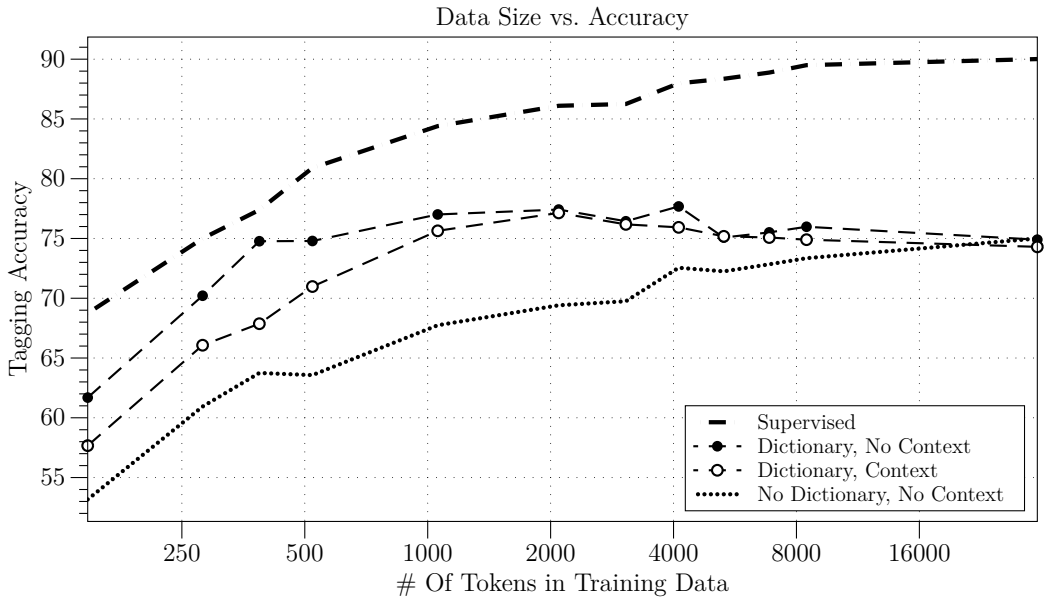


Figure 6.8: Graph showing the result of using the classifier-based tagging approach on CTN test data using varying amounts of training data. The legend refers to the different feature combinations used to train the classifier.

6.5.5 Analysis

While using the Chintang data was crucial for examining how the classification-based POS tagging approach performed on an endangered language, evaluation was difficult due to a combination of gloss-line conventions and the tags used in the gold standard. Without more knowledge of the specifics of Chintang, it is hard to determine whether the issues encountered mapping the gold standard tagset to the universal POS tags are merely an issue of design choices, or a more fundamental typological difference that the universal tagset is unable to account for.

6.6 Extending to Monolingual Corpora

While the results shown in Section 6.4.1 and Sections 6.4 and 6.5 are interesting, they are also systems that rely on IGT input data. Instead of requiring input to be IGT data, the POS-tagged IGT data could be used as training data for a monolingual POS tagger, which would result in a system able to be more widely used for the target language. The last set of POS tagging experiments I describe were done in order to achieve this goal.

6.6.1 Training Monolingual POS Taggers

In order to create monolingual POS taggers, I experimented with both methods of transferring tags to the language line described in the preceding section; either projection from translation to gloss, or classification of the gloss line, then subsequent 1-to-1 projection from the gloss line to language line.

Once a POS-tagged language line has been acquired, the resulting sentence can be used as a training instance for a standard sequence-labeling POS tagger, since we might assume that the contextual information assumed by traditional sequence labelers is more helpful when dealing with a single language, than the gloss line which can represent multiple languages. For my experiments, I used the Stanford Tagger (Toutanova et al., 2003).

The flowchart in Fig. 6.9 gives a high-level overview of how the POS tagger is trained, using either of these approaches.

6.6.2 Evaluating The Monolingual Taggers

In order to evaluate the new, monolingual POS taggers, I use the test sections of the Universal Dependency Treebank, v2.0 corpus (UD-2.0) (Agić et al., 2015), as it provides gold-standard POS in a variety of different languages, all using the universal tagset from Petrov et al. (2012). The English language POS tagger used for tagging the translation lines was also trained on the WSJ corpus using POS tags mapped to the universal set. Finally, as Section 4.2

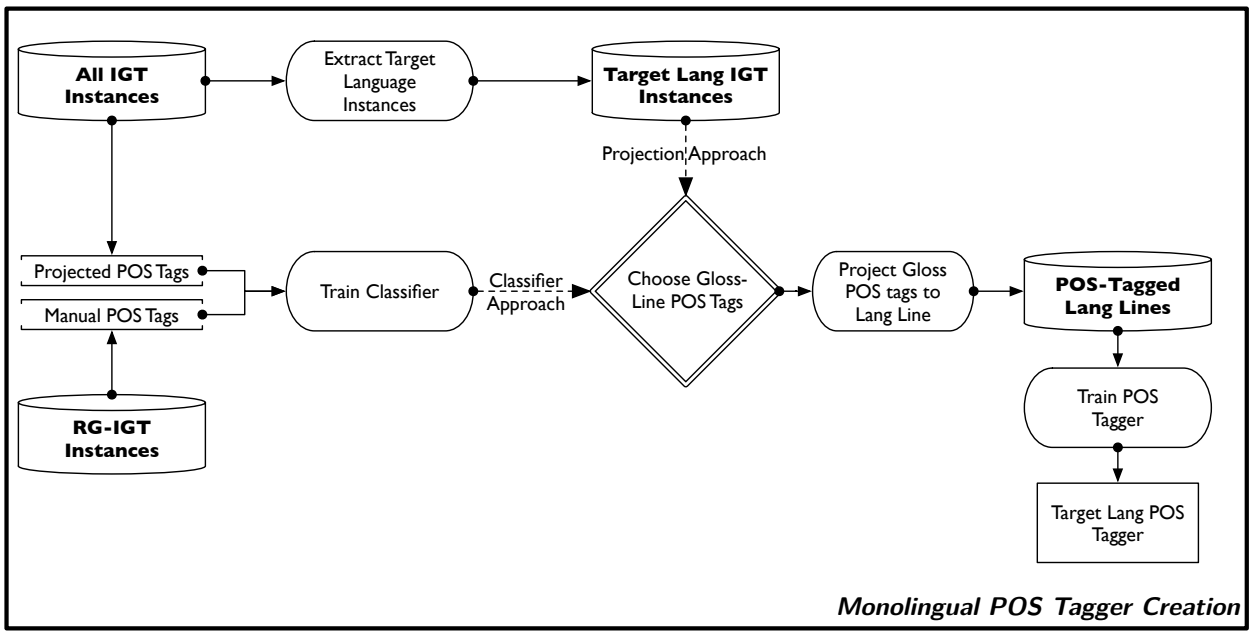


Figure 6.9: Flowchart depicting the process for training the monolingual POS tagger, using either the projection based approach from Section 6.2 or the classifier-based approach from Section 6.3.

mentions, one of the constraints chosen to limit the scope of this work was attempting to avoid complications due to transliteration of orthography, so although the UD-2.0 corpus provides data for Japanese and Korean, these languages were not used for these POS tagging experiments.

The languages chosen for evaluation were: French, German, Italian, Spanish, Indonesian, and Swedish. It is worth noting that although no language-specific information was used in creating the classifier for POS tagging the gloss line, IGT instances from the first four languages were found among those which had their gloss lines annotated in the RG-IGT corpus. The last two languages, Indonesian and Swedish, did not.

6.6.3 Results

Figure 6.10 and Table 6.9 show the results for these monolingual results, comparing between monolingual POS taggers created with IGT language lines created via projection or classifier methods, and POS taggers created with data from the training sections of the UD-2.0.

As the results from the RG-IGT (Section 6.4.1) and Chintang experiments (Section 6.5.5) suggested, the classification-based approach outperforms projection in almost all of the tested languages. Two settings for projection were attempted in the data, one labeled **Projection: Partial**, where IGT instances were used regardless of whether all words in the instance were aligned or not. The other setting, labeled **Projection: Full** only used IGT instances for which all tokens in the instance were aligned with the translation line and received a label. This **Full** setting drastically improves monolingual tagging performance for all languages except Indonesian.

Among the classification-based approaches, the classifier trained on a small amount of manually labeled gloss-line tokens (**Classifier: Manual**) performed better in most cases than the classifier trained on a much larger set of projected labels (**Classifier: ODIN**). German and Swedish were the exception, with the ODIN classifier outperforming the manual

Language	Projection: Partial	Projection: Full	Classifier: ODIN	Classifier: Manual	Supervised: 1K Tokens	Supervised: All Tokens
French	50.8	61.9	67.4	70.6	79.3	95.8
German	67.8	70.1	73.7	72.6	70.5	92.6
Italian	55.0	59.1	63.4	66.6	73.5	96.2
Spanish	61.6	68.8	70.4	73.2	80.2	95.9
Indonesian	61.5	59.7	66.2	70.9	80.7	93.9
Portuguese	52.5	65.5	60.8	65.8	78.5	96.0
Swedish	47.0	50.1	55.1	53.5	67.3	93.8
Overall	56.3	63.4	66.3	69.2	77.7	95.4

(a) Tagging accuracies for all sentences in the UD-2.0 test data.

Language	Projection: Partial	Projection: Full	Classifier: ODIN	Classifier: Manual	Supervised: 1K Tokens	Supervised: All Tokens
French	62.3	69.9	74.1	75.8	76.8	96.4
German	68.5	66.8	75.0	72.8	74.0	92.6
Italian	65.1	65.6	69.6	72.3	69.6	95.3
Spanish	64.7	65.3	70.2	73.4	78.2	95.8
Indonesian	63.8	63.3	66.7	70.1	79.4	90.9
Portuguese	56.7	66.7	64.1	65.7	75.9	94.1
Swedish	55.6	57.5	61.5	60.3	69.1	93.1
Overall	62.7	65.5	69.5	70.4	74.9	94.1

(b) Tagging accuracies for sentences in the UD-2.0 test data with length of ten words or fewer.

Table 6.9: POS tagging accuracies for monolingual taggers trained on IGT language lines and tested on on UD-2.0 test data (McDonald et al., 2013). While no tagger involved annotation that was specific to a particular language, the first four languages were languages for which some IGT instances had been annotated on the gloss line for the classifier labeled 'Manual.' Neither Indonesian, Portuguese, nor Swedish were among the languages with gloss-line annotations, and no manual labeling was done for the 'ODIN' classifier.

classifier by a percentage point. Overall, however, the Manual classifier achieved 69% tagging accuracy, to the ODIN classifier's 66%.

Shown in the results are two supervised systems, using a varying amount of training data. In the first system, labeled "1K Tokens," only the first 1,000 tokens of the training data were used. In the second system, all available training sentences were used. Given that 1,000 tokens is a far more realistic amount of data to be available for a resource-poor language IGT, these scores are given to provide more of a real-world comparison than the "All Tokens" systems, which are fully supervised with tens or hundreds of thousands of available tokens. While this 1K token system generally outperforms the IGT-based methods by a substantial

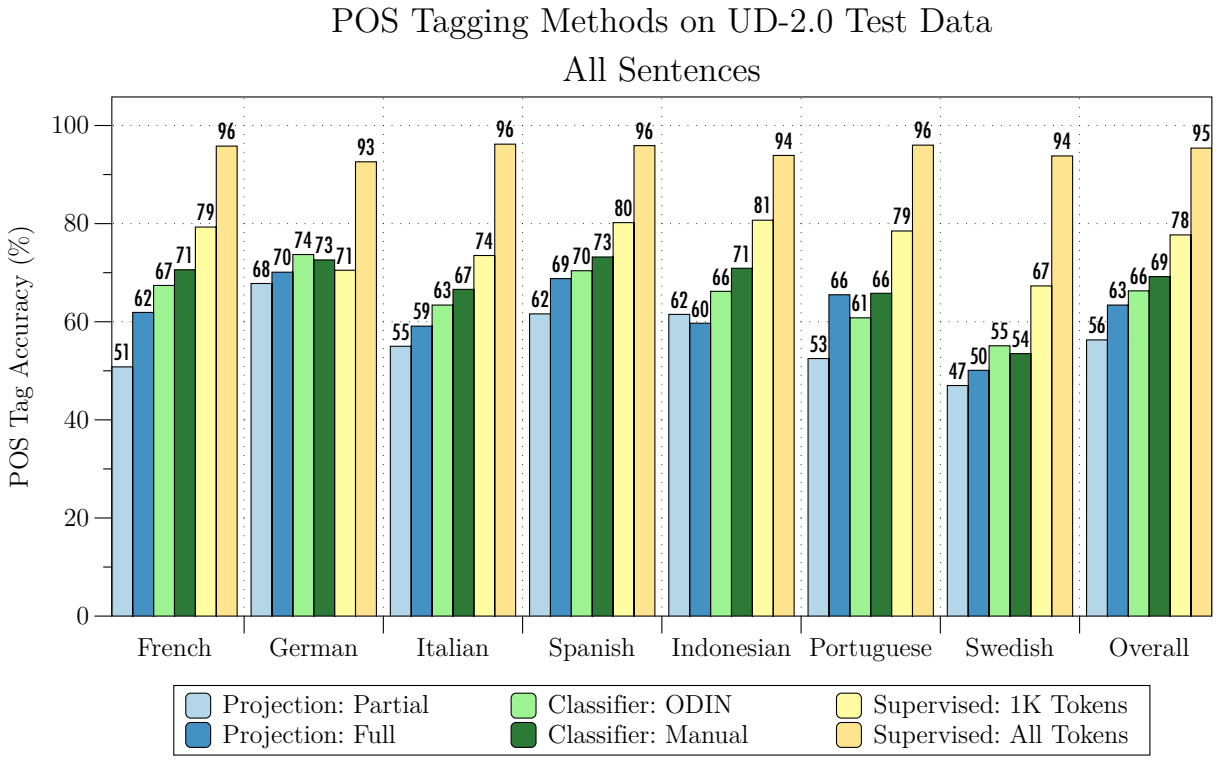


Figure 6.10: Bar graph showing the POS tag accuracies for the POS tagging on monolingual UD-2.0 test data.

margin, in German, the classifier-based approaches manage to beat the German 1K approach by 2-3 percentage points.

Another setting seeks to address the shift in domain that the UD-2.0 data set represents. The average length of a sentence in the UD-2.0 corpus is 21.8 words, while the average length of IGT in ODIN is 4.78 words. Furthermore, the domain of the UD corpus is more along the lines of newswire articles than the extremely simple instances found in IGT. In order to compare differences in sentence length, I ran a second set of experiments shown in Table 6.9b where the systems were evaluated only on sentences in the UD-2.0 test data that consisted of ten words or fewer. On this set of test data, all IGT-based methods perform better, while the supervised approaches perform worse, but the trend shown for evaluation on all sentence lengths (Table 6.9a) stays the same.

We see in these results that, although the supervised systems do still regularly outperform the IGT-based systems, given that such supervised data will not be available for the typical resource-poor language, the IGT-based systems do appear to perform decently at the task.

6.7 Future Work

While the experiments performed here give a foundation for the ways in which IGT data may be used as a source for POS tagging resource-poor languages, there some paths that would be interesting to pursue further.

6.7.1 Incorporating Classification Probabilities

Given that the classification approach I outline uses a MaxEnt classifier, which can output a top-n list of the potential POS tags and their probabilities, this is information that could be exposed to a downstream tagger, which could incorporate this ambiguity with the contextual information from the target language to produce higher-quality tagging on the monolingual task.

6.7.2 POS Induction

In the case where the amount of supervised information available for a language is limited, but a good deal of unsupervised information is available, it is possible that the POS tagging task might instead be cast as a semi-supervised POS induction task.

The prototyping approach of Haghighi and Klein (2006b) is one which would ideally hold promise for this setting. In this work, the authors take the approach of taking a typically unsupervised approach on monolingual data, and use "prototypes" as constraints on a Markov random field model that allows for the observed monolingual data to guide the induction process, while guiding the algorithm toward a solution that includes the pre-defined prototypes. While I performed some preliminary experiments along these lines, I was unable to replicate the published results, and it seemed that the choosing of the prototypes may require a deal more work.

Chapter 7

DEPENDENCY STRUCTURES

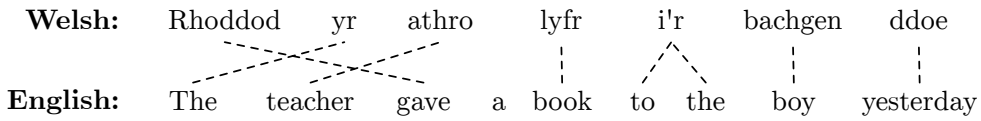
The previous chapters have discussed word alignment and POS tagging, and the problems presented when using IGT as a data source. In particular, the previous tasks face the issues of (1) noisy data and (2) sparse data. For the task of dependency parsing, the target to be learned moves from the shallower task of word alignment and POS tagging to more complex linguistic analysis. While POS tags may differ between languages, if we treat this correspondence as a statistical likelihood, the relationship between two aligned words is even less likely to hold. Thus, dependency structures that contain many such relationships between words are an even harder target for bootstrapping than previously discussed tasks.

To get a full picture of the dependency parsing task, I will present it in five parts. First, Section 7.1 will give a brief introduction to dependency structures. Second, Section 7.2 will discuss the fundamental task of projecting dependency structures using the projection algorithm alone, using different alignment methods. Next, Section 7.3 will discuss how projected parses were used in combination with a modified dependency parser to see if improvements could be made over projection alone.[1] Section 7.4 will discuss measuring the divergence between languages by means of the dependency structures, and how these patterns might be learned and applied to the output of a parser.[2] Finally, Section 7.5 will talk about training a monolingual dependency parser for use in the monolingual parsing task.

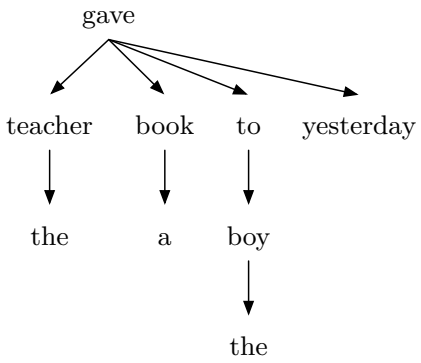
---

[1]Previously published as Georgi et al. (2012)

[2]Previously published as Georgi et al. (2014)



(a) Parallel Welsh-English sentence pair.



(b) Dependency structure representation of the English sentence.

Figure 7.1: Example of a Welsh–English sentence pair, and English dependency structure, represented as a DAG in tree form.

7.1 Introduction

Dependency structures (DSs) are used to describe individual sentences not as constituents, but rather a set of head–dependent pairs, where the pairings describe a minimal relationship between words, such as an adjective and a noun. This concept was introduced to the field of modern linguistics in Tesnière (1959), and further supported by Hays (1964), who introduced the possibility of using such a formalism in machine translation. An example sentence and its DS representation is shown in Fig. 7.1.

The primary advantage of DSs over constituent-based models is that, as mentioned by Hays (1964), in a multilingual setting, the more basic relations between head/dependent as represented in a dependency structure are more likely to be retained between languages than the complex interior and language-specific structures present in a constituent-based

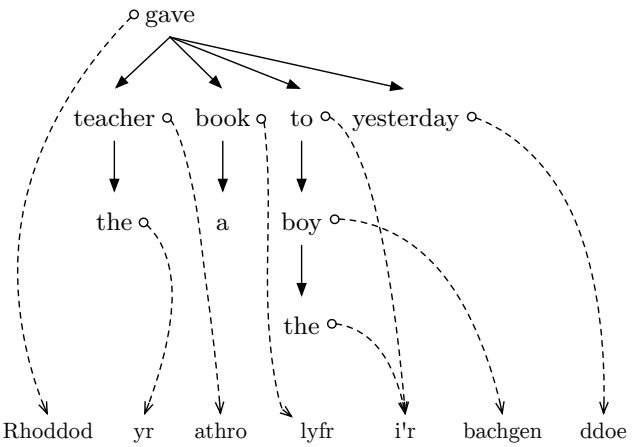
tree. It is for this reason that I choose to focus the syntactic elements of this work upon DS representations, for which a literature base exists to compare against, particularly with regard to other projection methods.

In this chapter, I will present five different DS parsing systems. Section 7.2 will describe System 1, a projection-only system which functions on IGT data. Section 7.3 will discuss Systems 2 and 3, which are ways by which a modified dependency parser can use projected edges as a feature, potentially correcting for unreliable projections. Section 7.4 will describe System 4, in which by comparing manually corrected DSs between the target language and the English projection source DSs, divergence patterns may be detected and corrected for. Finally, Section 7.5 describes System 5, which uses IGT to extract training data in the form of POS tags and projected DS structures, then train a dependency parser directly on the IGT-extracted data for running on monolingual data in the target language.

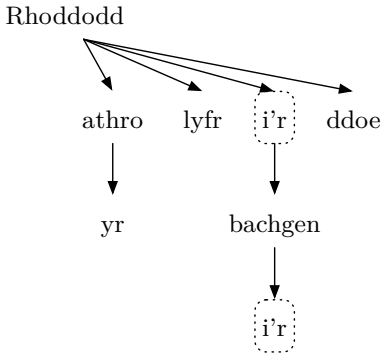
7.2 Projecting Dependency Structures

I start with the most straightforward method of obtaining dependency structures for a given language for which IGT data is available: use the IGT itself to obtain the structures. This method is essentially the same as standard bitext DS projection methods such as that implemented by Hwa et al. (2004), with the exception that IGT data allows for different alignment methods, as discussed in Chapter 5.

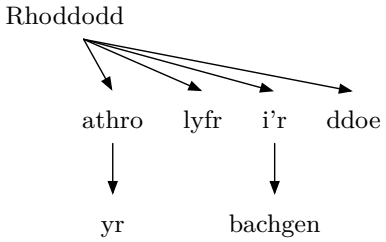
Figure 7.2a shows the English DS from Fig. 7.1, along with the word alignments to the Welsh sentence from Fig. 2.1. The projection method used follows Quirk et al. (2005) and Xia and Lewis (2007). To start, the system parses the translation line of the IGT, E, into a dependency tree, T_E, and aligns the nodes with the words in the language line, F. For each translation-line node e_i that aligns with a language-line word f_i, that node is replaced with the language-line word. If a single translation node e_i aligns with multiple language-line words (f_i, f_j), I make multiple copies of e_i as siblings in the tree for each language-line



(a) Using the English DS and the word alignment from Fig. 7.1, the next step is to project the structure via the alignments.



(b) Replacing the words in the English DS with those from the Welsh in Fig. 7.2a results in the tree above. Note that due to the Welsh word *i'r* being aligned with the English words *to* and *the*, it occurs twice.



(c) For target words that were aligned to multiple source words, all but the shallowest are removed.

Figure 7.2: Illustration of the projection of dependency structures using the aligned English–Welsh sentence and the English DS from Fig. 7.1, following Xia and Lewis (2007).

word, then replace the translation-line words with those from the language line. If multiple translation nodes align to a single language-line word, the node highest up in the tree is kept, and all others removed. Finally, remaining unaligned words are reattached using the following heuristic: for the indices $i < j < k$, where j is the unaligned word's order in the sentence, and i and k are the closest aligned words to the left and right, respectively, j is attached to the lower of i or k if one is descended from the other. That is, if k is a dependent child of i, then j will attach to k. If there are no aligned words i to the left of j, j is attached to the leftmost word that is aligned, and vice versa for k. The full pseudocode for this algorithm as implemented here can be found in Appendix B.1 as Algorithm B.1.1.

This method has the obvious advantage of being available for any language that is available in Odin, as it produces the dependency structures from the IGT instances themselves. The downside of this method is that it does not directly produce a model which can process non-IGT data; I will discuss how that may be done in Section 7.5.

Figure 7.3 illustrates the way in which the DSs are produced. In creating the dependency structures, I looked at five ways to obtain the alignment: two variants of the heuristic method (Section 5.2), two variants of the statistical G-T alignment methods (Section 5.3), and the gold-standard alignments themselves. Additionally, I look at two ways of obtaining trees for the translation-line: the manually-produced gold standard trees, and those produced by using a parser.

The first four systems, including the projection-only system described here will be evaluated against the gold-standard dependency structures in the XL-IGT and HUTP corpora described in Chapter 4. The evaluation metric will be Unlabeled Attachment Score (UAS) which was chosen over Labeled Attachment Score (LAS) to focus the content of the experiments on the projected structure, rather than the complex semantics involved in labeled dependencies, which may map poorly between languages.

Figure 7.4 shows an overview of the UAS results for the projection-only methods across

Using Manual English DSs

Language	Stat	Stat +Heur	Heur −POS	Heur +POS	Manual
Gaelic	74.6	74.6	75.0	79.8	81.8
German	74.0	74.0	66.7	74.6	88.2
Hindi	59.9	60.0	49.0	57.3	62.8
Hausa	68.1	68.1	51.3	59.1	79.5
Korean	80.9	80.9	67.3	77.8	88.1
Malagasy	81.2	81.2	72.8	77.7	88.8
Welsh	79.2	79.2	73.1	78.2	88.1
Yaqui	76.1	76.6	56.9	82.0	85.8
Overall	72.6	72.6	61.9	71.2	81.0

(a) UAS for Projected DSs, using the gold standard DSs as the source.

Using Parser-Produced English DSs

Language	Stat	Stat +Heur	Heur −POS	Heur +POS	Manual
Gaelic	61.5	61.5	66.3	69.1	65.5
German	66.9	66.9	59.5	66.6	75.4
Hindi	54.5	54.4	46.8	55.3	57.9
Hausa	57.5	57.5	47.6	53.3	61.2
Korean	75.8	75.8	63.7	73.7	83.2
Malagasy	69.3	69.3	63.0	67.5	73.2
Welsh	70.5	70.5	65.4	70.2	73.4
Yaqui	59.9	59.9	42.6	67.3	67.1
Overall	63.8	63.7	55.5	64.1	69.1

(b) UAS for projected DSs, using the Stanford Parser (de Marneffe and MacCartney, 2006) to produce the source DSs.

Table 7.1: Unlabeled Attachment Score (UAS) for projected dependency structures in the XL-IGT and Hindi datasets, showing various alignment methods and using either gold standard trees or parser-generated English trees as the projection sources.

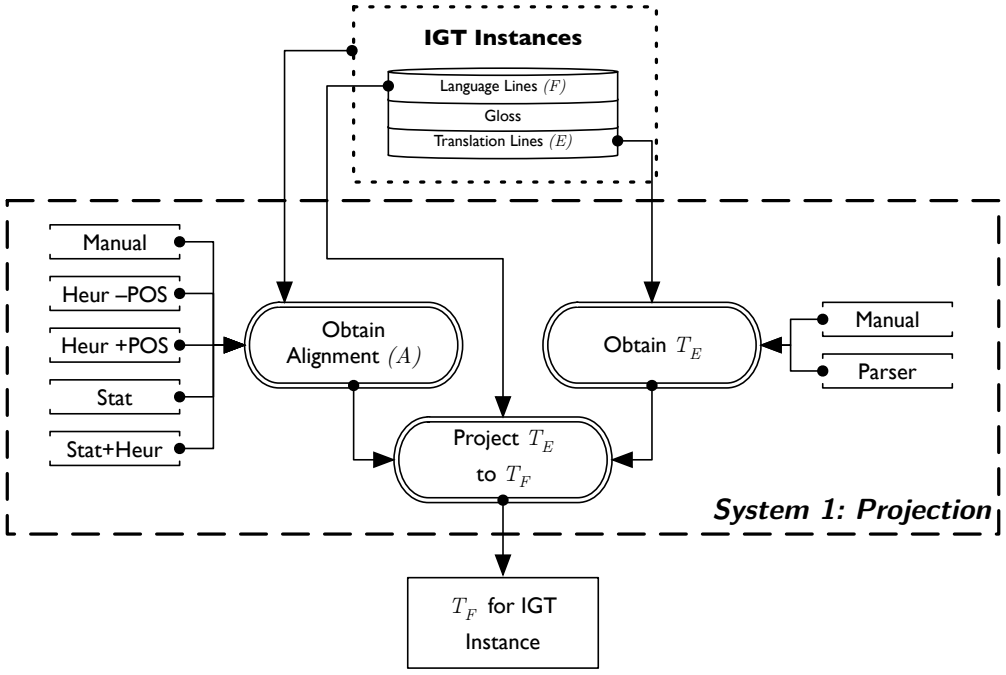


Figure 7.3: Flowchart illustrating the different options that can be used to create DSs via projection. There are five alignment methods, more detailed descriptions of which may be found in Chapter 5, and two options for the translation-line trees. Results for all ten pairwise combinations can be found in Table 7.1.

all languages in the XL-IGT and Hindi datasets, while Tables 7.1b and 7.1b show the results for the ten pairwise combinations of settings across all the languages in the XL-IGT and Hindi datasets (see Chapter 4 for further descriptions of these datasets).

Looking at the overview in Fig. 7.4, we see that the projected DSs that use the gold standard as the source perform reliably better than the automatically-generated source ones, and that using manual alignment similarly boosts the scores. While some languages achieve scores as high as 88% when using gold standard sources DSs and alignment, an important takeaway from these results is that the overall UAS achieves only 81.0%. This 19% error is important, as it represents an upper bound on what we may reasonably expect from systems that rely upon projection. This demonstrates that even with perfect word alignment and

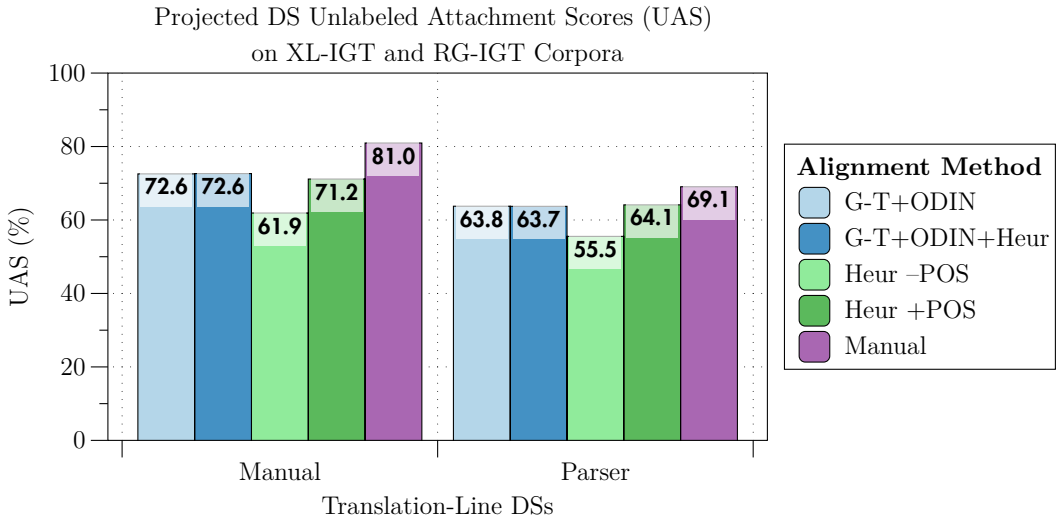


Figure 7.4: An overview of unlabeled parse accuracy for the dependency structures over the languages in the XL-IGT corpus when the projection algorithm is used with different alignment methods. The "Translation-Line DS" groups the two methods for creating the dependency structure on the translation line, manually, or generated by a parser.

gold-standard translation-line DSs, the assumption that a projected DS will be correct for the language line does not always hold.

Keeping this in mind, the 70+% UAS achieved for some languages, and 63–4% overall using non-gold sources, while not particularly impressive, is a comparatively better result than it might otherwise be for languages in which the only data available may be the IGT instances in Odin.

Hwa et al. (2004) reports figures of 67–72% for Spanish, and 54–64% for Chinese; languages for which the authors have available 100,000–240,000 parallel sentences, respectively. More recent work on projection by Ganchev et al. (2009) reports similar ranges without using language-specific rules, 63.8% for projection between English and Bulgarian, and 67.6% for English to Spanish. These methods use different data with longer sentences, so the numbers are not strictly comparable. For comparison, the languages here have between 67–174 IGT

instances, with an average length of between 6–7 words.

Beyond the potential suspects of noise and difficulty with projection, Section 7.4 will discuss how part of this low overall score may be attributed either to structural dissimilarities between languages or particular annotation decisions made in creating the evaluation data.

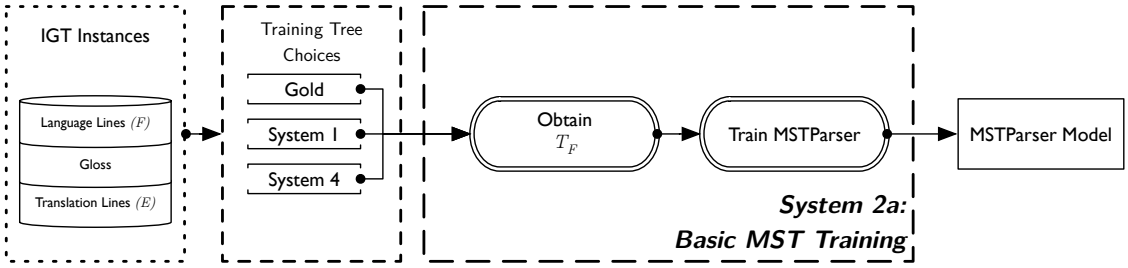
7.3 Combining Dependency Parsers with Projection

While the projection-based approach works for languages with very few resources, it has two major limitations. First, this deterministic approach is unable to take word/tag ambiguities into account unless they have been explicitly coded for. Second, systematic differences in how dependency structures are represented between languages, either intrinsic to the language or due to a design decision made in the treebank present a challenge to the algorithm, which operates on the assumption that the dependency structures are English-like in nature.

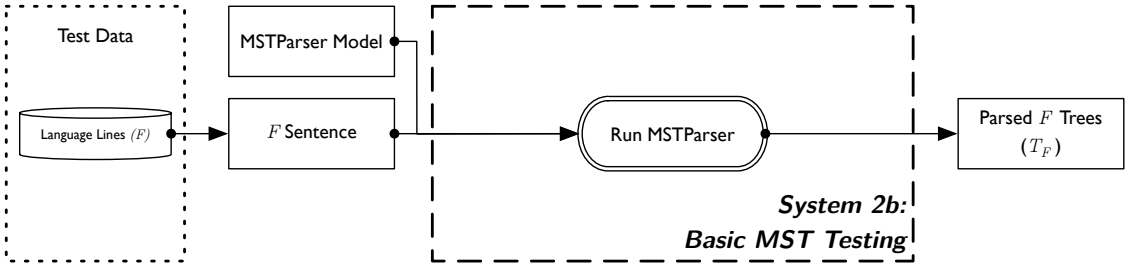
One approach to better handling ambiguity, as previously published in Georgi et al. (2012), is to join the information provided by the projected trees within IGT instances with a small amount of training data for a statistical parser. The projected dependency edges may then be used as a feature in parsing, one which may sometimes have a low weight if its use was shown to not be particularly predictive at the training step. Section 7.3.1 will describe the parser and the modifications that were made to include the projected dependency structures, and Section 7.3.3 will discuss the results of these modifications.

7.3.1 MSTParser

The parser used in these experiments is MSTParser (McDonald et al., 2005), so named for searching for Maximum Spanning Trees (MSTs) in directed graphs, an approach which has an advantage over previous attempts (Eisner, 1996; Collins, 1999) in that, unlike transition-based parsing, the graph traversal approach of this research is able to find a global optimal parse, important for languages with freer word order and long-range dependencies. As the



(a) Flowchart illustrating the training procedure for a given L_F–L_E language pair, where L_F is the foreign (non-English) language, and L_E is English. The "Gold" trees for training the parser can either be manually-created trees, or trees created by projection only.



(b) A flowchart showing an overview of the testing procedure for the L_F–L_E language pair using the L_F parser produced in the training phase, and augmented by the information projected from the parsed L_E portion of the bitext. A Future system will include an augmented statistical aligner to produce the L_F–L_E alignments.

Figure 7.5: Flowcharts describing the end-to-end functioning of the parsing system at both training and testing phases.

approach I present here is geared toward a broad range of languages, this aspect of the algorithm is appealing.

By default, MSTParser operates on unigram, bigram, and trigram features. The unigram features operate on a single word out of the two being examined at any given step by the parser, and use either the word form or part-of-speech tag itself. The bigram features look at both the child and parent words at a given step, while the trigram features add context. According to McDonald et al. (2005), such trigram features help rule out certain unlikely scenarios, such as a noun f_1 depending on a verb f_3 with an intervening noun f_2. For a full description of the feature set used, see §2.4 of McDonald et al. (2005). The learning algorithm used is the Margin Infused Relaxed Algorithm (Crammer and Singer, 2003), which Collins (2002) showed helps avoid over-fitting when training the parser. The unmodified MSTParser is used as a baseline in the subsequent experiments, and a flowchart of the basic MSTParser's training and testing procedures using IGT data is shown in Fig. 7.5.

7.3.2 Adding Features to the Parser

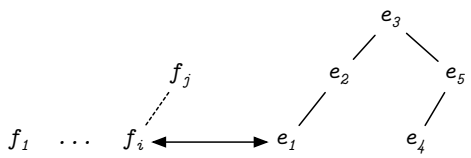
In order for MSTParser to incorporate the information from the projected trees, new features were needed for the parser. I introduced three new types of features. Let E be the English-language sentence, F the foreign-language sentence, T_P is the set of edges that make up the projected tree, and $R_{F \to E}$ and $R_{E \to F}$ are relations that map words from the target language (F) to words in the translation (E) or vice versa. For a summary of all notation used, see Appendix A.

$$ProjBool(f_i, f_j) = \begin{cases} \texttt{TRUE} & \textbf{if}(f_i, f_j) \in T_P \\ \texttt{FALSE} & \text{otherwise} \end{cases} \tag{7.1}$$

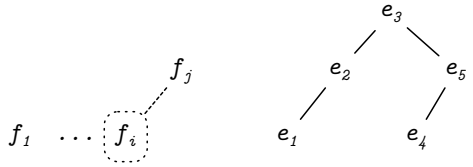
$$ProjTag_{pos_1,pos_2}(f_i, f_j) = \begin{cases} \texttt{TRUE} & \textbf{if}(f_i, f_j) \in T_P \ \wedge \ POS(f_i) = pos_1 \ \wedge \ POS(f_j) = pos_2 \\ \texttt{FALSE} & otherwise \end{cases} \tag{7.2}$$

$$AlignType(f_i, f_j) = \begin{cases} \texttt{IS\_SINGLE} & \textbf{if} \left| R_{F\to E}(f_i) \right| = 1 \\ \hline \texttt{IS\_UNALIGNED} & \textbf{if} \left| R_{F\to E}(f_i) \right| = 0 \\ \hline \texttt{IS\_MATCH} & \textbf{if} \begin{array}{l} \exists e_i, e_j \Big(e_i \in R_{F\to E}(f_i) \wedge e_j \in R_{F\to E}(f_j) \\ \wedge (e_a, e_b) \in T_E \Big) \end{array} \\ \hline \texttt{IS\_LEFTMERGE} & \textbf{if} \begin{array}{l} \exists e_i, e_j \Big(e_i \in R_{F\to E}(f_i) \wedge e_j \in R_{F\to E}(f_i) \\ \wedge \ isParent(e_i, e_j) \Big) \end{array} \\ \hline \texttt{IS\_RIGHTMERGE} & \textbf{if} \exists e_i, f_j \Big(e_i \in R_{F\to E}(f_i) \wedge f_j \in R_{E\to F}(e_i) \Big) \end{cases} \tag{7.3}$$

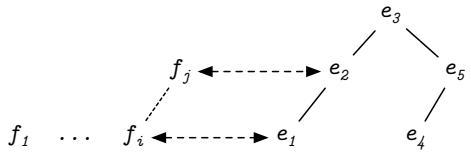
The *ProjBool* feature (Eq. 7.1) is the most basic of the three, simply taking on a `TRUE` value if the edge being considered by the parser also occurs in the projected tree T_P. While this feature covers the most cases, I also wanted to see if subdividing the feature by part-of-speech tag might help reduce errors caused by particular POS tags that project poorly. The *ProjTag* feature (Eq. 7.2) does this by creating separate boolean features for each POS tag



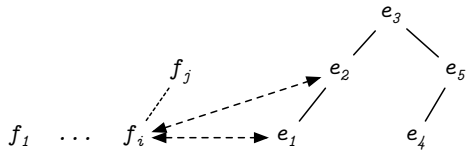
(a) The `IS_SINGLE` alignment feature is fired for token f_i in this configuration.



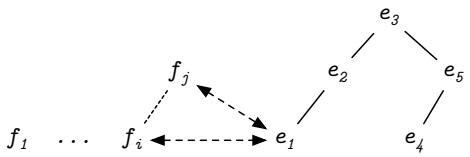
(b) The `IS_UNALIGNED` alignment feature is fired for token f_i in this configuration.



(c) The `IS_MATCH` alignment feature is fired for tokens (f_i, f_j) in this configuration.



(d) The `IS_LEFTMERGE` alignment feature is fired for token f_i in this configuration.



(e) The `IS_RIGHTMERGE` alignment feature is fired for tokens (f_i, f_j) in this configuration.

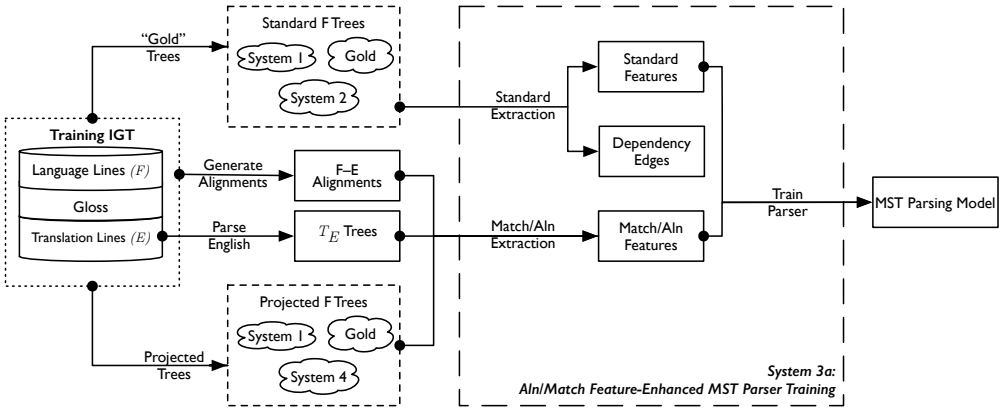
Figure 7.6: Different alignment configurations that trigger the *AlignType* feature. In each case, the dotted alignment arrows indicate that each word may have other words with which it aligns, solid arrows indicate the shown alignment is the only one allowed.

combination of parent and child. This risks exposing the parser to a severe sparsity issue, so it is expected that this feature will be complementary to the *ProjBool* feature. Finally, *AlignType* (Eq. 7.3) is actually a group of binary features used to subdivide agreement with the projection based upon the type of alignment exhibited by the word on the foreign language side. *AlignType* has five sub-features based on possible alignment types, which are illustrated in Fig. 7.6. `IS_SINGLE` [Fig. 7.6a] is `TRUE` when a token f_i has only a single match to the English-language tree in the alignments. `IS_UNALIGNED` [Fig. 7.6b] is triggered for foreign-language words that are not aligned with any English word, that is, the set of English words that f_i aligns to is the empty set.

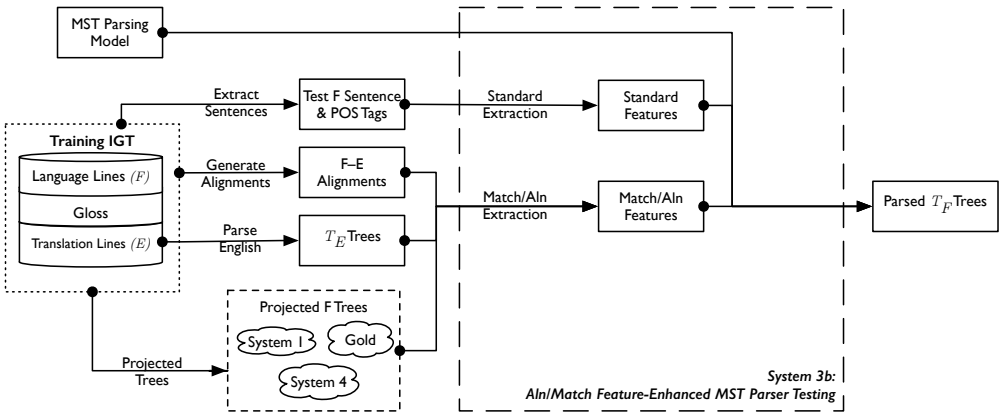
`IS_MATCH` is `TRUE` in the case where the edge (f_i, f_j) is being considered by the parser for edge and the parent and child align with words e_i, e_j in the source (English) side, and have the same parent/child relationship, as seen in Fig. 7.6c. `IS_LEFTMERGE` is `TRUE` when the foreign language word f_i aligns with multiple words on the English tree, and one of those words is an ancestor of the others (Fig. 7.6d). `RIGHT_MERGE` is `TRUE` when the foreign language word f_i is one of multiple foreign words aligned with a single English word e_i (Fig. 7.6e).

While the *AlignType* features and *ProjBool* features may seem similar, they operate in two somewhat different ways. *ProjBool* and *ProjTag* features compare the partially constructed parse tree to the projected version of the same tree during training and testing, and thus learns the weight of the feature based upon how well the projected tree correlates with the training trees. *AlignType*, on the other hand, compares the tokens of the current target tree strictly using alignments to the source tree, and does not look at the projected tree. This way, the *AlignType* features are focused on replicating the structures seen with the alignments between the two languages. My intuition was that combining these two different approaches should result in more robust performance than either feature set alone.

With these modifications made, the result is a parser that can be trained either with trees formed by projection in a sort of self-training, or on a set of manually annotated trees.



(a) Flowchart demonstrating the training process for the modified MSTParser system.



(b) Flowchart demonstrating the testing process for the modified MSTParser system.

Figure 7.7: Flowcharts demonstrating the training and testing process for the modified MSTParser, using additional features provided by projected trees as described in Section 7.3.1. Trees provided at training and testing time can either come from the gold standard, projections (As shown in "System 1", Fig. 7.3) or the corrected trees ("System 4") that will be discussed in Section 7.4.

A depiction of this system can be seen in Fig. 7.7.

7.3.3 Results

The MSTParser was evaluated using a number of different feature combinations, from the baseline features ('B'), to the baseline features plus all the new Bool/Tag/Align features discussed in Section 7.3.1, to a system where the gold-standard trees were provided in the place of projected trees for the additional features ('Oracle' systems). The results of these experiments on the modified MSTParser can be seen in Table 7.2, where Table 7.2a shows the results of using manually corrected trees for the training data with projected trees for the additional features, and Table 7.2b shows the results of using projected trees for both the training data and additional features.

The results in Table 7.2a show what is a fairly low baseline for the MSTParser baseline system, ranging from **51.4%**–**80.8%**, and never outperforming the projection algorithm. This result does show the power of using projection for IGT instances, as despite the very small amount of language data in the XL-IGT and HUTP corpora, a projection-based approach is not constrained by data sparsity, and can produce reasonable structures using only the IGT data and projection. When the parser is trained using the additional features, however, the combined performance of adding the features from the projected trees to the parser improves performance for the languages across the board.

The results for the projection-based systems in Table 7.2b are more modest, rarely performing better than the projection results themselves. This is expected, as the parser is being guided even more strongly by the projected trees in this system formulation.

7.3.4 Summary

The results of these experiments showed that using manually corrected trees in combination with projected trees resulted in a better performing system than either projection alone, or

Using Manually-Corrected English Trees for Training Source

System	Gaelic	German	Hausa	Hindi	Korean	Malagasy	Welsh	Yaqui	Overall
B + Oracle	95.6	98.1	99.3	98.0	99.2	97.9	98.3	96.5	98.0
B + Oracle + POS	89.3	97.9	94.4	97.3	98.3	97.7	94.4	97.3	97.0
B + POS + Bool/Tag/Aln	79.4	91.3	88.0	78.9	89.9	**91.9**	**94.4**	**86.8**	**88.4**
B + POS + Bool/Aln	**80.6**	**91.5**	87.5	**80.2**	90.3	91.5	93.1	86.3	88.3
B + POS + Aln	75.8	90.1	85.2	78.7	89.7	90.9	89.6	86.5	87.1
B + POS + Bool/Tag	70.2	90.2	88.7	79.1	87.8	89.6	88.9	86.5	86.4
B + POS + Tag	65.9	83.5	82.6	76.7	82.4	79.5	72.9	83.8	79.1
B + POS + Bool	69.8	**91.5**	88.4	79.5	87.4	90.3	89.2	86.3	86.5
B + POS	63.1	82.1	83.8	77.2	80.8	81.5	75.0	81.8	79.3
B + Bool/Aln	78.6	89.3	90.3	76.7	91.7	91.5	94.1	86.3	88.1
B + Aln	76.2	89.3	**90.5**	73.0	93.0	89.6	90.6	86.0	86.5
B + Bool	70.2	87.8	87.7	77.3	**92.3**	89.4	91.3	85.5	87.2
MST Baseline (B)	55.2	62.7	72.2	65.2	80.8	73.0	51.4	66.1	67.3
Projection	78.6	87.9	79.4	67.8	89.7	89.6	89.6	84.8	84.3

(a) Results when manually-corrected trees were used as the main training source for the modified parser, combined with the projection-based features.

Using Projected English Trees for Training Source

System	Gaelic	German	Hausa	Hindi	Korean	Malagasy	Welsh	Yaqui	Overall
B + Oracle	79.8	95.1	84.5	75.7	98.6	94.2	87.5	96.3	90.4
B + Oracle + POS	72.6	90.7	80.3	67.0	95.7	90.9	86.1	89.5	85.8
B + POS + Bool/Tag/Aln	78.2	87.9	79.2	67.1	90.7	89.4	88.9	85.3	84.3
B + POS + Bool/Aln	78.2	87.2	79.2	66.9	90.5	90.3	88.9	83.8	84.1
B + POS + Aln	73.8	87.5	76.4	65.2	89.4	87.1	84.7	84.5	82.2
B + POS + Bool/Tag	73.4	87.1	78.7	66.1	90.3	89.2	86.1	84.0	83.1
B + POS + Tag	59.9	77.2	69.9	56.5	81.4	77.8	69.1	78.1	72.6
B + POS + Bool	74.2	87.6	79.4	66.7	89.4	88.6	84.4	85.0	82.8
B + POS	60.7	76.1	70.6	56.0	80.8	77.0	66.0	76.3	71.2
B + Bool/Aln	**79.0**	87.6	**81.0**	66.9	89.4	89.6	**89.6**	85.3	84.2
B + Aln	76.6	**88.2**	79.4	66.1	**91.1**	**91.9**	87.5	86.0	84.5
B + Bool	76.2	87.9	80.6	66.9	90.3	90.3	87.9	**86.5**	**84.4**
MST Baseline (B)	49.2	61.2	51.9	49.5	80.3	73.0	38.5	71.3	62.6
Projection	78.6	87.9	79.4	**67.8**	89.7	89.6	**89.6**	84.8	84.3

(b) Results when projected trees were used as the main training source for the modified parser, combined with the projection-based features.

Table 7.2: UASs for the different feature sets used in the modified parser on the XL-IGT and HUTP corpora. The baseline MST parser features are represented by 'B,' and additional feature sets are combined with this system additively. "Oracle" describes not a feature set, but the result of using the gold-standard trees in place of projected trees at train and test time. The best-performing results for a language on a non-oracle-based system are shown in bold.

a parser trained on the projected trees. The main limitation of this approach, however, is that the parsers created by this system require IGT data at test time, as well as manually-corrected trees in a potentially resource-poor language to receive this boost. These settings are not necessarily appropriate for real-world applications with resource-poor languages; a more generalized approach will be presented in Section 7.5.

7.4 Analyzing and Correcting for Divergence Patterns

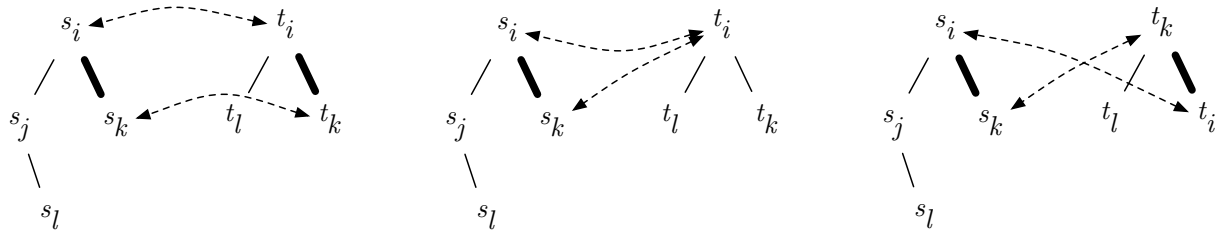
While the approach of the previous section was to train a parser that was able to use features extracted from projected dependency trees at test time, I took a different approach to see if errors that were recurrent in projected trees could be analyzed and systematically corrected in the resulting projections.[3]

An additional goal of this set of experiments was not only to improve parsing performance, but analyze language pairs for ways in which the translationally equivalent sentences might be represented in fundamentally different ways, thus establishing a likely upper bound for any methods that seek to assume direct correspondence between languages.

The concept of linguistic divergence here follows that of Dorr (1994), who describes a number of types of linguistic divergence, using the concept of *lexical conceptual structures* (Jackendoff, 1983)—structures that provide a framework to describe divergence arising from syntactic, lexical, or semantic differences. Here, the focus of divergence will be restricted to only the syntactic, since the systems presented here use predominately shallow processing, and are lexically agnostic as well.

In this approach to automatically detecting divergent structures between language pairs, I first propose a metric to measure the degree of *matched* edges between source and target trees (Section 7.4.1). Second, I define three operations on trees that will rewrite the trees to more closely resemble the other language in the pair and resolve some divergence

[3]Previously published in Georgi et al. (2014).



(a) A *match* alignment (b) A *merge* alignment (c) A *swap* alignment

Figure 7.8: Definition of a *match*, *merge*, and *swap* edges in a tree pair.

patterns (Section 7.4.2). After a discussion of the results between different language pairs (Section 7.4.3), I will discuss how these operations may be learned automatically and applied as post-projection corrections (Section 7.4.4). Finally, the results of the application of the correction rules will be discussed in Section 7.4.5.

7.4.1 Calculating Divergence Across Dependency Structures

One of the key aspects of this method was devising a metric to compare dependency trees cross-linguistically, as most existing tree similarity measures are intended to compare tree representations with the same number of tokens. Comparing between languages, on the other hand, means that the number of tokens can vary. Instead, I look for a method to determine similarity by means of matched edges in the tree, as shown in Fig. 7.8.

Given an IGT, let F be the representation of the language line, consisting of the language words W_F (Eq. 7.5) and parse tree T_F (Eq. 7.6). The parse tree T_F is defined as a set of (parent, child) edges, as shown in Eq. (7.6).

$$F = (W_F, T_F) \tag{7.4}$$

$$W_F = (f_1 \dots f_n) \tag{7.5}$$

$$T_F = \Big\{(f_i, f_k) \dots (f_n, f_m)\Big\} \tag{7.6}$$

E is defined similarly, except words in the translation line are denoted as e_i, not f_i. The alignment A is a set of word pairs:

$$A = \Big\{(f_i, e_k) \dots (f_j, e_l)\Big\} \tag{7.7}$$

An aligned tree pair consists of the 3-tuple (F, E, A). A Corpus, C, in the experiments, is a set of (F, E, A) tuples.

An edge (f_i, f_k) in the foreign-language tree is said to *match* an edge (e_i, e_k) in the English-language tree if f_i is aligned to e_i and f_k is aligned to e_k. Because the alignment between a sentence pair can be many-to-many, I define the following relations $R_{F\to E}$ and $R_{E\to F}$, which map a word from one sentence to the set of words in the other sentence.

$$R_{F\to E}(f_i, A) = \Big\{e|(f_i, e) \in A\Big\} \tag{7.8}$$

$$R_{E\to F}(e_i, A) = \Big\{f|(f, e_i) \in A\Big\} \tag{7.9}$$

I then define the boolean function *match*, as follows:

$$match(f_i, f_j, T_E, A) = \begin{cases} 1 & \textbf{if } \exists e_a, e_b \Bigg(\Big(e_a \in R_{F\rightarrow E}(f_i)\Big) \wedge \\ & \qquad \Big(e_b \in R_{F\rightarrow E}(f_j)\Big) \wedge \Big((e_a, e_b) \in T_E\Big) \Bigg) \\ 0 & otherwise \end{cases} \tag{7.10}$$

That is, an edge (f_i, f_j) in F matches some edge in E according to A if there exists two target words, e_a and e_b in E such that e_a aligns to f_i, e_b aligns to f_j, and (e_a, e_b) is an edge in E.

Given an aligned tree pair (F, E, A), $SentMatch(F, E, A)$ is defined as the percentage of edges in F that match some edge in E. Again, the dependency type (label) is ignored, as for the purposes of this work I wish to focus on the structure alone and save projection of dependency labels and semantic roles for future work. Given a corpus C, $CorpusMatch_{Src\rightarrow Tgt}(C)$ is the percentage of edges in the source trees that match some edges in the corresponding target trees. Similarly, $CorpusMatch_{Tgt\rightarrow Src}(C)$ is the percentage of edges in the target trees that match some edges in the corresponding source trees.

$$CorpusMatch_{Src\rightarrow Tgt}(C) = \frac{\sum_{(F,E,A)\ \in\ C} \left(\sum_{(f_i,f_j)\ \in\ T_F} match(f_i, f_j, T_E, A) \right)}{\sum_{(F,E,A)\in C} \left|T_F\right|} \tag{7.11}$$

7.4.2 Defining Tree Operations to Resolve Divergence

When an edge (f_i, f_k) in a tree does not match any edge in the other tree, it may be caused by one of the following cases:

C1. f_i or f_k are spontaneous (they do not align with any words in the other tree).

C2. f_i and f_k are both aligned with the same node e_i in the other tree (Fig. 7.8b).

C3. f_i and f_k are both aligned with nodes in the other tree, e_k and e_i, but in a reversed parent-child relationship (Fig. 7.8c).

C4. There is some other structural differences not caused by C1–C3.

The first three cases are common. To capture and resolve them, I define three operations on a tree — *remove*, *merge*, and *swap*.

O1 – Remove

The *remove* operation is used to remove spontaneous words. As shown in Fig. 7.9a, removal of the node l is accomplished by removing the link between node l and its parent, j, and adding links between the parent and the removed node's children.

This result of this operation looks can be seen in Fig. 7.9a, using the relation $Children$, which maps a word to the set of all its children in the tree.

Algorithm 7.4.1: Remove a token w from the tree T.

1 **Algorithm:** $Remove(w, T)$
  Input : $T = \{(w_i, w_j) \dots (w_m, w_n)\}$; // Input tree
  Input : w ; // Word to remove.
  Output : T' ; // Modified tree
2 **begin**
3 $T' = T - \{(w, w_i) | w_i = parent(w, T)\}$ // Remove edge between w and parent w_i
4 $-\{(w_j, w) | w = parent(w_j, T)\}$ // Remove edges for children of w
5 $+\{(w_j, w_i) | w_i = parent(w, T), w = parent(w_j, T)\}$;
  /* Finish by ''promoting'' former children of w to now attach to w's parent, w_i. */
6 **return** T'

O2 – Merge

The *merge* operation is used when a node and some or all of its children in one tree align to the same node(s) in the other tree, as can be seen in Fig. 7.8b. The parent j and child l are collapsed into a merged node, as indicated by l+j in Fig. 7.9b, and the children of l are promoted to become children of the new node l+j. The result can be seen in Fig. 7.9b.

Algorithm 7.4.2: Merge a child w_c and parent w_p in the tree T, and "promote" the children of w_c to be children of w_p.

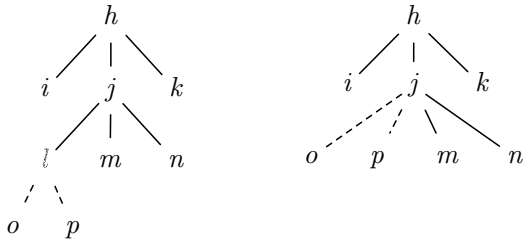
```
1  Algorithm: Merge(w_c, w_p, T)
   Input  : T = {(w_i, w_j) ... (w_m, w_n)} ;          // Input tree
   Input  : w_c ;                                      // Child word to merge.
   Input  : w_p ;                                      // Parent word to merge.
   Output : T' ;                                       // Modified tree
2  begin
3      T' = T − {(w_c, w_p)}
4              −{(w_i, w_c)|w_c = parent(w_i, T)}
5              +{(w_i, w_p)|w_c = parent(w_i, T)};
6  return T'
```

O3 – Swap

The *swap* operation is used when two nodes in one tree are aligned to two nodes in the other tree, but in a reciprocal relationship, as shown in Fig. 7.8c. This operation can be used to handle certain divergence types such as *demotional* and *promotional* divergence, which will be discussed in more detail in Line 25.

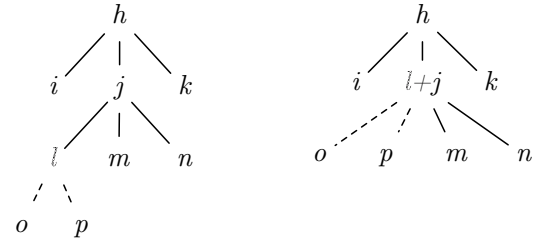
Figure 7.9c illustrates how the swap operation takes place by swapping nodes l and j. Node j, the former parent, is *demoted*, keeping its attachment to its children. Node l, the former child, is *promoted*, and its children become siblings of node j, the result of which can be seen in Fig. 7.9c.

$$Remove(l, T) : \{\ Children(j) = \{o, p, m, n\}$$



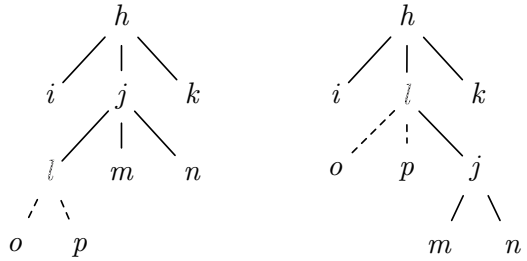
(a) Before and after the node l has been removed (O1).

$$Merge(j, l, T) : \begin{cases} Children(j + l) = \{o, p, m, n\} \\ Children(h) = \{i, j + l, k\} \end{cases}$$



(b) Before and after the nodes l and j have been merged (O2).

$$Swap(j, l, T) : \begin{cases} Children(j) = \{m, n\} \\ Children(h) = \{i, l, k\} \\ Children(l) = \{o, p, j\} \end{cases}$$



(c) Before and after the nodes l and j have been swapped (O3).

Figure 7.9: Trees showing the results of the operations defined in O1–O3. $Children(w)$ returns the set of words that depend on w. Here we show the value of $Children(node)$ after the operations only if its value is changed by the operations.

Algorithm 7.4.3: Swap a child w_c and parent w_p in the tree T.

```
1 Algorithm: Swap(w_c, w_p, T)
  Input  : T_L = {(w_i, w_j) ... (w_m, w_n)} ;                    // Input tree
  Input  : w_c ;                                                  // Child word to swap.
  Input  : w_p ;                                                  // Parent word to swap.
  Output : T' ;                                                   // Modified tree
2 begin
3     T' = T − {(w_c, w_p)} + {(w_p, w_c)}                        // Swap the order in the (w_c, w_p) edge
4            −{(w_p, w_i)|w_i = parent(w_p, T)}                   // Remove edges for parent w_p
5            +{(w_c, w_i)|w_i = parent(w_p, T)} ;                 // Add edges from w_p's parent to w_c
6 return T'_L
```

Calculating Tree Matches After Applying Operations

The operations O1–O3 are proposed to handle common divergence cases in C1–C3. To measure how common C1–C3 is in a language pair, I design an algorithm that transforms a tree pair based on a word alignment.

The algorithm takes a tree pair (F, E) and a word alignment A as input and creates a modified tree pair (S', T') and an updated word alignment A' as output. It has several steps. First, spontaneous nodes (nodes that do not align to any node on the other tree) are removed from each tree. Next, if a node and its parent align to the same node on the other tree, they are merged and the word alignment is changed accordingly. Finally, the swap operation is applied to a node f_i and its parent f_p in one tree if they align to e_i and e_p respectively and e_p is a child of e_i in the other tree. The pseudocode of the algorithm is shown in Algorithm 7.4.4.

Now given a corpus C and word alignment between each sentence pair, I can measure the impact of C1–C3 by comparing $CorpusMatch_{Src \rightarrow Tgt}(C)$ scores before and after applying operations O1–O3. This process can also reveal some patterns of divergence (e.g., what types of nodes are often merged), and the patterns can later be used to enhance existing projection algorithms.

Algorithm 7.4.4: Algorithm for altering an aligned tree pair.

input : $c = (F, E, A)$; // A parallel sentence with alignment
output : $c' = (F', E', A')$; // modified output sentence.

1 **Let** $F = (W_F, T_F)$;
2 **Let** $E = (W_E, T_E)$;
3 **Let** $A = \{(f_i, e_j), \ldots, (f_k, e_l)\}$;
4 **begin**
    // Step 1(a): Remove spontaneous nodes from F
5 **foreach** $f_i \in W_F$ **do**
6 **if** $\nexists\ e_j : (f_i, e_j) \in A$ **then**
7 $T_F = Remove(f_i, T_F)$; // See Algorithm B.1.2
    // Step 1(b): Remove spontaneous nodes from E
8 **foreach** $e_j \in W_E$ **do**
9 **if** $\nexists\ f_i : (f_i, e_j) \in A$ **then**
10 $T_E = Remove(e_i, T_E)$; // See Algorithm B.1.2
    // Step 2(a): Find nodes to merge in F and merge them
11 **foreach** $(f_i, e_j) \in A$ **do**
12 **Let** $f_p = parent(f_i, T_F)$;
13 **if** $(f_p, e_j) \in A$ **then**
14 $T_F = Merge(f_i, f_p, T_F)$; // See Algorithm 7.4.2
15 $A = A - \{(f_i, e_j)\}$;
    // Step 2(b): Find nodes to merge in E and merge them
16 **foreach** $(f_i, e_j) \in A$ **do**
17 **Let** $e_p = parent(e_j, T_E)$;
18 **if** $(f_i, e_p) \in A$ **then**
19 $T_E = Merge(e_j, e_p, T_E)$; // See Algorithm 7.4.2
20 $A = A - \{(f_i, e_j)\}$;
    // Step 3: Find nodes to swap in F and swap them
21 **foreach** $(f_i, e_j) \in A$ **do**
22 **Let** $f_p = parent(f_i, T_F)$;
23 **if** $\exists\ e_c : e_j = parent(e_c, T_E)$ **and** $(f_p, e_c) \in A$ **then**
24 $T_F = Swap(f_i, f_p, T_F)$; // See Algorithm 7.4.3
25 **return** (F', E', A') ;

Relationship to Dorr (1994)

Dorr (1994) lists seven types of divergence for language pairs. While the analysis method presented here is more coarse-grained than the Lexical Conceptual Structure (LCS) that Dorr proposes, it is nonetheless able to capture some of the same cases.

For instance, Fig. 7.10 illustrates an example of what Dorr identified as "promotional" divergence, where *usually*, a dependent of the verb *goes* in English, is "promoted" to become the main verb, *suele* in Spanish. In this case, the direction of the dependency between *usually* and *goes* is reversed in Spanish, and thus the *swap* operation can be applied to the English tree and result in a tree that looks very much like the Spanish tree. A similar operation is performed for *demotional* divergence cases, such as aligning *I like eating* with the German translation *Ich esse gern* ("I eat likingly"). Here, the main verb in English (*like*) is *demoted* to an adverbial modifier in German (*gern*). The *swap* operation is applicable to both types of divergence and treats them equivalently, and so it essentially can handle a superset of promotional and demotional divergence, namely, head-swapping.

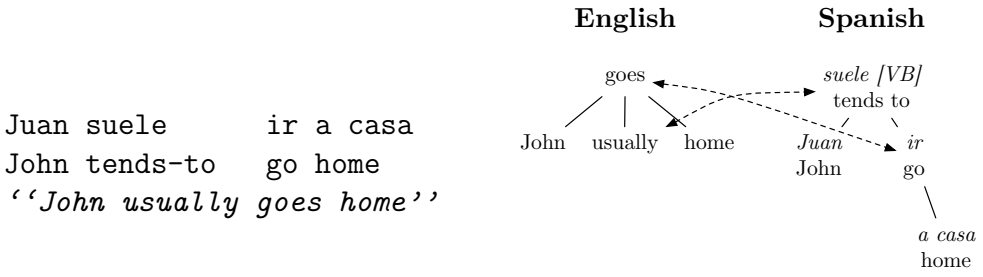


Figure 7.10: An example of *promotional* divergence from Dorr (1994). The reverse in parent-child relation is handled by the *Swap* operation.

Another type of divergence that can be captured by our approach is Dorr's "structural" divergence type, as illustrated in Fig. 7.11. The difference between the English and Spanish structures in this case is the form of the argument that the verb takes. In English, it is a noun

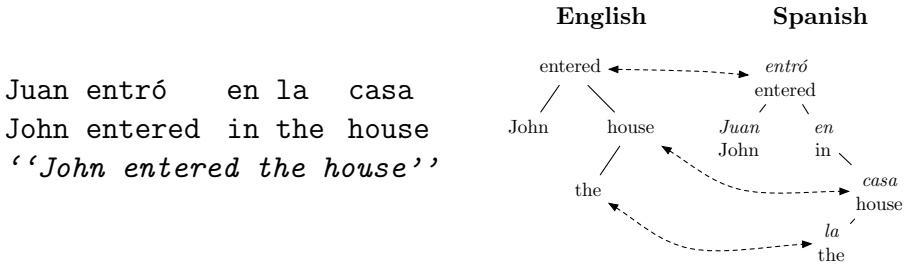


Figure 7.11: Example of structural divergence, which is handled by the *remove* operation.

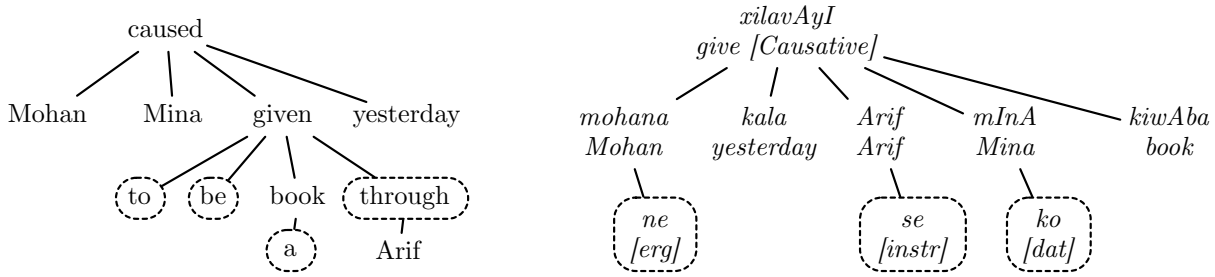
phrase; in Spanish, it is a prepositional phrase. While the tree operations defined previously do not explicitly recognize this difference in syntactic labels, the divergence can be handled by the *remove* operation, where the spontaneous *en* in the Spanish side is removed.

Next, Dorr's description of *conflational* divergence lines up well with the *merge* operation (see Fig. 7.9b). Fig. 7.12 illustrates an example for English and Hindi, where both sides have spontaneous words (e.g. *to* and *a* in English and *ne*, *se* and *ko* in Hindi) and a causative verb in Hindi corresponds to multiple verbs in English. Fig. 7.12(b) shows the original tree pair, Fig. 7.12(c) demonstrates the altered tree pair after removing spontaneous words from both sides. Fig. 7.12(d) shows the tree pairs after the English verbs are merged into a single node. It is clear that the *remove* and *merge* operations make the Hindi and English trees much similar to each other.

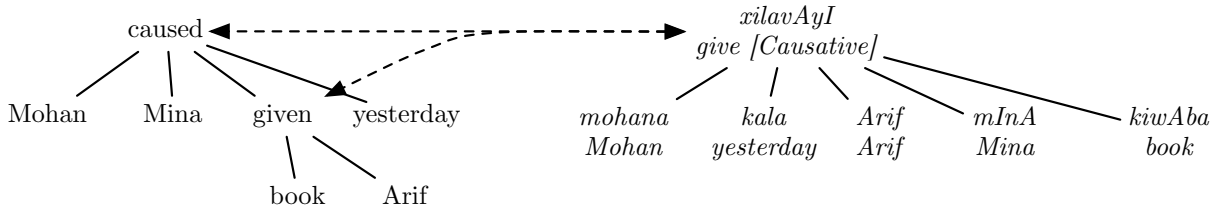
In addition to the four divergence types mentioned above, additional operations could be added to handle other divergence types. For instance, if dependency types (e.g., patient, agent) are given in the dependency structure, we can define a new operation that changes the dependency type of an edge to account for *thematic* divergence, where thematic roles are switched as in *I like Mary* in English vs. *María me gusta a mí* ("Mary pleases me") in Spanish. Similarly, an operation that changes the POS tag of a word can be added to cover *categorial* divergence where words representing the same semantic content have different word categories in the two languages, such as in *I am hungry* in English versus *Ich habe*

```
mohana ne    kala        Arif se      mInA ko     kiwAba     xilavAyI
Mohan  [erg] yesterday   Arif [inst]  Mina [dat]  book       give-caus
```
‘‘Mohan caused Mina to be given a book through Arif yesterday.’’

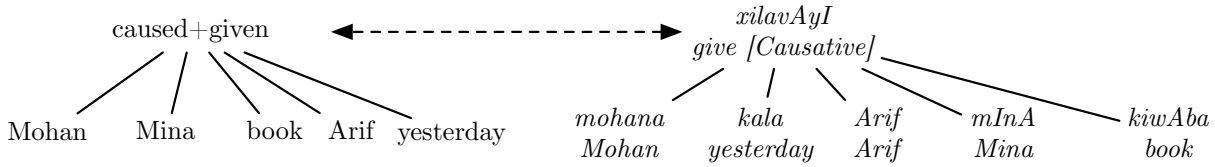
(a) Interlinear text of a sentence pair in Hindi (`hin`).



(b) Initial trees showing spontaneous words on both sides.



(c) Altered trees after removing spontaneous words from both sides, and showing conflational divergence between multiple English words and a single Hindi word.



(d) Altered trees after merging multiple words on the English side.

Figure 7.12: Case of conflational divergence, handled by remove and merge operations.

	English→Hindi					Hindi→English				
	Match	Swap	Unaligned	Merge	Edges	Match	Swap	Unaligned	Merge	Edges
Baseline	47.7	9.1	20.9	1.6	794	46.3	5.6	20.7	1.7	816
+Remove	66.1	11.7	0	2.1	622	63.4	5.3	0	2.2	647
+Merge	69.5	12.3	0	0	586	69.2	4.6	0	0	587
+Swap	90.3	0.3	0	0	586	89.9	2.4	0	0	587

Table 7.3: Breakdown of edges as operations are applied to the English ↔ Hindi language pair (from bottom up) on the HUTP corpus. The "Edges" column represents the number of total edges in the trees of the left hand of the language pair. The numbers given in the other columns are the percentages of those edges that are either in a *match*, *swap*, or *merge* alignment, or the edges for which the child is *unaligned*.

Hunger ("I have hunger") in German.

In contrast to Dorr's divergence types, whose identification requires knowledge about the language pairs, my operations on the dependency structure rely on word alignment and tree pairs and can be applied automatically.

7.4.3 Match Results

By running Algorithm B.2.1, $CorpusMatch_{Src \to Tgt}$ and $CorpusMatch_{Tgt \to Src}$ can be calculated before and after each operation in order to measure how the operation affects the percentage of matched edges in the corpus. As the operations are applied, the percentage of matches between the trees should increase until all the divergence cases that can be handled by operations O1–O3 have been resolved. At this point, the final match percentage can be seen as an estimate of the upper-bound on performance on a projection algorithm, if C1–C3 can be identified and handled by O1–O3. Table 7.3 shows the full results of this process for English and Hindi, while Table 7.4 shows a summary for the results in the remaining ten languages.

	Gaelic	**German**	**Hausa**	**Malagasy**	**Korean**	**Welsh**	**Yaqui**
Baseline	72.0	76.7	54.4	57.4	56.0	75.4	54.4
+Remove	87.8	93.9	95.7	88.9	88.1	95.1	90.9
+Merge	92.5	95.4	97.5	97.4	95.4	97.2	95.9
+Swap	94.1	96.8	97.5	98.0	96.1	98.2	96.2

Table 7.4: Summary of *match* percentages for the remaining seven language pairs of the XL-IGT corpus.

Breakdown By POS

After performing the operations as seen in Section 7.4.3, the breakdown of the operations by language and POS, as is done in Table 7.5 provides a good opportunity to check that the defined operations conform with expectations for specific languages.

For instance, row 1 in Table 7.5a also shows Modals (MD) merging with a parent verb (VB). This is in line with instances such as Figure 7.12c where Hindi states in a single verb a causative meaning that is typically expressed as a separate words in English. This does not appear to be a very frequent occurrence, however, as it only occurs for 42.9% of MD→VB dependencies. Row 3 shows the case in Hindi where auxiliary verbs (VAUX) merge with main verbs (VM). These cases typically represent those where Hindi represents tense as an auxiliary verb, whereas English tense is expressed by inflection on the verb.

Rows 5 and 6 show the English→German pair merging many nouns as multiple English words are expressed as compounds in German. In another case, rows 1–2 in Table 7.5b show that all Hindi nouns undergo *swap* with prepositions, as the Hindi uses postpositions instead of prepositions, and the HUTP corpus places these nouns as the heads of the phrases.

With regard to spontaneous words in English and Hindi, row 3 in Table 7.5c shows that 69.8% of case markers (PSP) were removed from Hindi that were either absent in English or applied as inflections to the noun, while 86% of determiners in English were removed, as they are not seen in Hindi (row 1).

Merges

Lang Pair	Child POS	Parent POS	% Merge
Eng→Hin	MD	VB	42.9
	NN	NN	14.3
Hin→Eng	VAUX	VM	45.4
	NN	VM	5.5
Eng→Ger	NN	NNS	66.7
	NN	NN	65.4

(a) Breakdown of *merge* operations between language pairs by POS tag.

Swaps

Lang Pair	Child POS	Parent POS	% Swap
Hin→Eng	NN	IN	100
	NNP	IN	20
Ger→Eng	NN	APPRART	72.7
	CC	NN	61.5

(b) Breakdown of *swap* operations between language pairs by POS tag.

Removals

Lang Pair	POS Tag	% Remove
Eng→Hin	DT	86.4
	TO	75.6
Hin→Eng	PSP	69.8
	VAUX	18.6
Eng→Ger	POS	57.1
	DT	20.2
Ger→Eng	PRF	85.2
	ADV	43.9

(c) Breakdown of removal operations between language pairs by POS tag.

Table 7.5: Breakdown of significant *merge* and *swap* statistics for various language pairs in the XL-IGT and HUTP corpora, where the language to the left of the arrow is the one being altered.

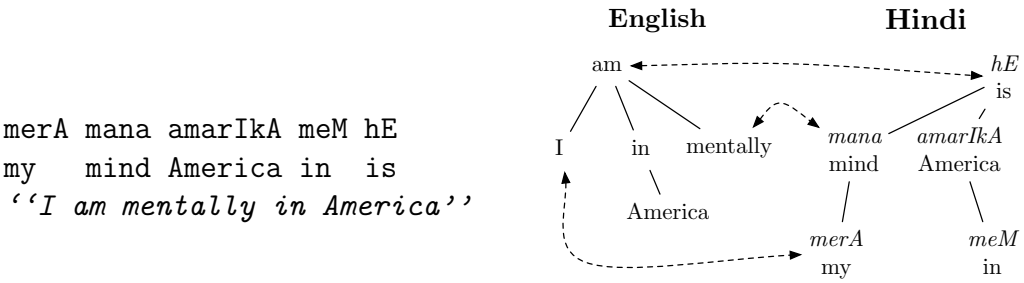


Figure 7.13: A tree pair that still has unmatched edges after applying the algorithm in Table 7.4.4. The dotted line indicates word alignment that would be needed to resolve the divergence with the *extended* merge operation.

Remaining Cases

After applying three operations, there may still be unmatched edges. An example is given in Figure 7.13.[4] The dependency edge *(in, America)* can be reversed by the *swap* operation to match the Hindi counterpart. The difficult part is the adverb *mentally* in English corresponds to the noun *mana (mind)* in Hindi. If the word alignment includes the three word pairs as indicated by the dotted lines, one potential way to handle this kind of divergence is to extend the definition of *merge* to allow edges to be merged on both sides simultaneously – in this case, merging *am* and *mentally* in the English side, and *hE (is)* and *mana (mind)* on the Hindi side.

7.4.4 Learning Correction Patterns

After looking closely at how frequent particular patterns occurred in Section 7.4.3, the next step was to devise a series of simple corrections that could be applied to the trees as a post-processing step. By examining the projected and gold-standard tree pairs, when and how these corrections needed to be applied could be learned. Using the three alignment types

[4]It is a topic of debate whether *mentally* in English should depend on *in* or *am*. If it depends on *in*, handling the divergence would be more difficult.

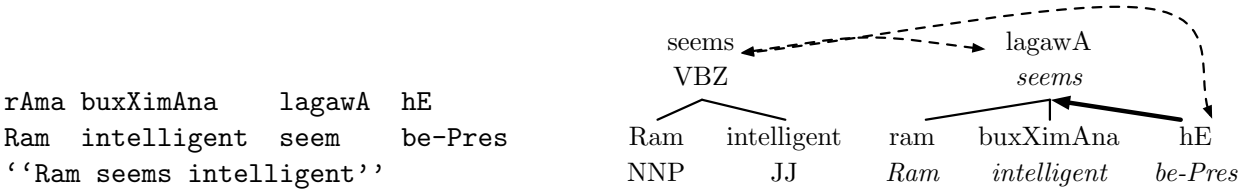


Figure 7.14: An example of merged alignment, where the English word *seems* aligns to two Hindi words *hE* and *lagawA*. Beside the IGT instance are the dependency trees for English and Hindi. Dotted arrows indicate word alignment, and the solid arrow indicates that *hE* should depend on *lagawA*.

discussed in Section 7.4.2, the following improvements could be made to the resultant trees:

O1. **Merge:** better informed choice for head for multiply-aligned words.

O2. **Swap:** post-projection correction of frequently swapped word pairs.

O3. **Spontaneous:** better informed attachment of target spontaneous words.

The details of the enhancements are explained below.

Merge Correction

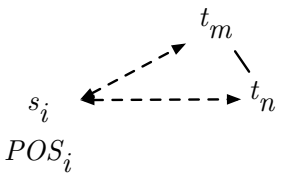
“Merged” words, or multiple words on the target side that align to a single source word, are problematic for the projection algorithm because it is not clear which target word should be the head and which word should be the dependent. An example is given in Figure 7.14, where the English word *seems* aligns to two Hindi words *hE* and *lagawA*.

In these merge cases, the training trees can be used to analyze which word is most likely to be the head and which the dependent. This learning process can be generalized away from the specific words by using the POS tags of the words in the *merge* alignment. The process is illustrated in Fig. 7.15. In this example, the target words t_m and t_n are both aligned with the source word s_i whose POS tag is POS_i, and t_m appears before t_n in the target sentence. Going through the examples of merged alignments in the training data, a count is kept for

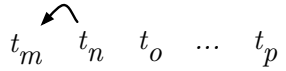
the POS tag of the source word and the position of the head on the target side.[5] Based on these counts for a given language, the system will generate rules such as the ones in Figure 7.15(c), which says if a source word whose POS is POS_i aligns to two target words, the probability of the right target word depending on the left one is 75%, and the probability of the left target word depending on the right one is 25%. Maximum likelihood estimation (MLE) is used to calculate this probability.

The projection algorithm will use those rules to handle merged alignment; that is, when a source word aligns to multiple target words, the algorithm determines the direction of dependency edge based on the direction preference stored in the rules. In addition to rules for an individual source POS tag, this method also keeps track of the overall direction preference for all the merged examples in that language. For merges in which the source POS tag is unseen or there are no rules for that tag, this language-wide preference is used as a backoff.

[5]The position of the head is used, not the POS tag of the head, because the POS tags of the target words are not available when running the projection algorithm on the test data.



(a) Alignment between a source word and two target words, and one target word t_m is the parent of the other word t_n.

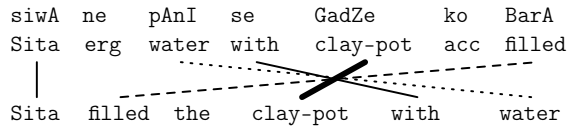


(b) Target sentence showing the "left" dependency between t_m and t_n.

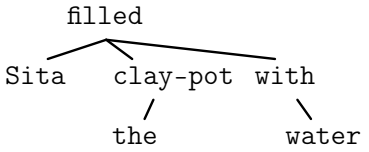
$\texttt{POS}_i \rightarrow \texttt{left} \quad 0.75$
$\texttt{POS}_i \rightarrow \texttt{right} \quad 0.25$

(c) Rules for handling merged alignment

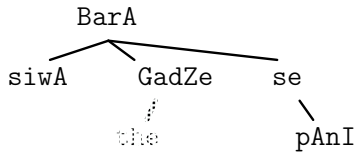
Figure 7.15: Example of *merge* alignment and derived rules.



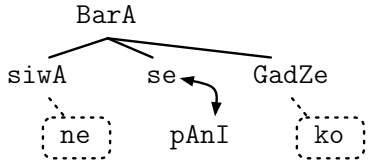
(a) An Interlinear Glossed Text (IGT) instance in Hindi and word alignment between the gloss line and the English translation.



(b) Dependency parse of English translation.



(c) English words are replaced with Hindi words and spontaneous word "the" are removed from the tree.



(d) Siblings in the tree are reordered based on the word order of the Hindi sentence and spontaneous Hindi words are attached as indicated by dotted lines. The words *pAnI* and *se* are incorrectly inverted, as indicated by the curved arrow.

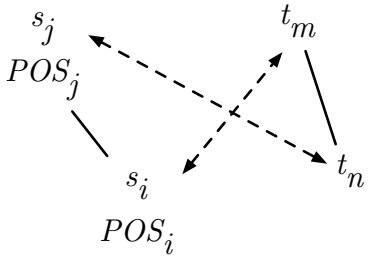
Figure 7.16: An example of projecting a dependency tree from English to Hindi.

Swap Correction

An example of swapped alignment is in Fig. 7.17a, where (s_j, s_i) is an edge in the source tree, (t_m, t_n) is an edge in the target tree, and s_j aligns to t_n and s_i aligns to t_m. Figure 7.16d shows an error made by the projection algorithm due to swapped alignment. In order to correct such errors, I count the number of (POS_{child}, POS_{parent}) dependency edges in the source trees, and the number of times that the directions of the edges are reversed on the target side. Figure 7.17b shows a possible set of counts resulting from this approach. Based on the counts, we keep only the POS pairs that appear in at least 10% of training sentences and the percentage of swap for the pairs are no less than 70%.[6] We say that those pairs trigger a swap operation.

At test time, swap rules are applied as a post-processing step to the projected tree. After the projected tree is completed, the swap handling step checks each edge in the source tree.

[6]These thresholds are set empirically.



(a) A swapped alignment between source words s_j and s_i and target words t_m and t_n.

POS Pair		Swaps	Total	%
$(\texttt{POS}_i, \texttt{POS}_j)$	$\rightarrow$	16	21	76
$(\texttt{POS}_k, \texttt{POS}_l)$	$\rightarrow$	1	1	100
$(\texttt{POS}_n, \texttt{POS}_o)$	$\rightarrow$	1	10	10

(b) Example set of learned swap rules. `Swaps` counts the number of times the given (child, parent) pair is seen in a swap configuration in the source side, and `total` is the number of times said pair occurs overall.

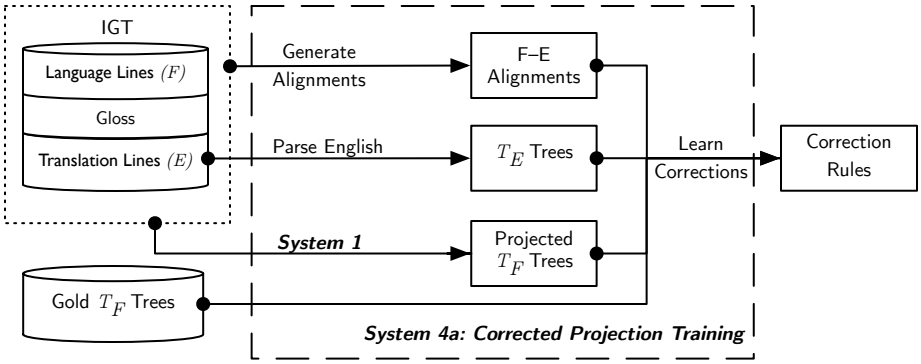
Figure 7.17: Example swap configuration and collected statistics.

If the POS tag pair for the edge triggers a swap operation, the corresponding nodes in the projected tree will be swapped (see Fig. 7.9c).

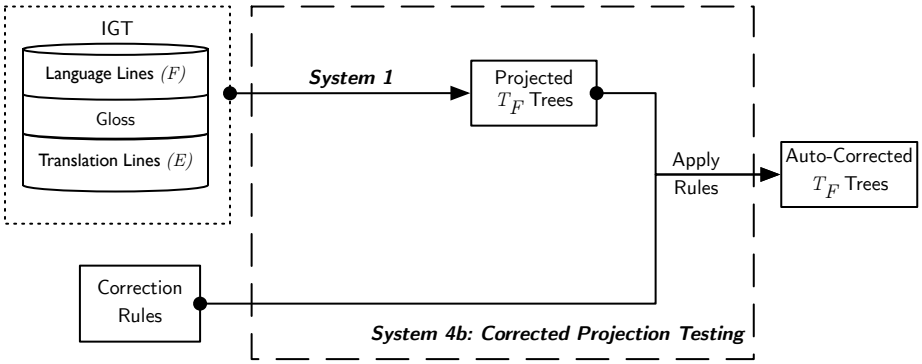
Spontaneous Reattachment

Target spontaneous words are difficult to handle because they do not align to any source word and thus there is nothing to project to them. To address this problem, two types of information are collected from the training data. First, I keep track of all the lexical items that appear in the training trees, and the relative position of their head. This lexical approach may be useful in handling closed-class words which account for a large percentage of spontaneous words. Second, we use the training trees to determine the favored attachment direction for the language as a whole.

At test time, for each spontaneous word in the target sentence, if it is one of the words for which statistics have been gathered from the training data, it is attached to the next aligned word in the preferred direction. If the word is unseen, it is attached using the overall language preference as a backoff.



(a) Flowchart illustrating the training phase of learning the correction patterns.



(b) Flowchart demonstrating the testing phase of applying correction patterns to test IGT data.

Figure 7.18: Flowcharts demonstrating the training and testing steps of the enhanced projection system (System 4).

Resulting System

The system that results from integrating these correction rules into the projection algorithm is illustrated in Fig. 7.18. In addition to above enhancements to the projection algorithm itself, I ran experiments training a dependency parser on the improved projections, reusing the additional parser features from Section 7.3.1. The resulting system is a combination of the one shown in Fig. 7.18 feeding into the training and testing trees provided to "System 3" shown in Fig. 7.7.

7.4.5 Correction Rule Results

I ran two different experiment settings using these automatically applied correction rules; the first was to use the correction rules directly on projected sentences. These results are shown in Table 7.6a. The second setting was to use the improved projections at training and testing for the projection-guided parser described in Section 7.3.1. These results are shown in Table 7.6b.

In both of the tables, the "Best" row uses the enhanced projection algorithm. The "Baseline" rows use the original projection algorithm from Section 7.2, where the word in the parentheses indicates the direction of merge. The "Error Reduction" row shows the error reduction of the "Best" system over the best performing baseline for each language. The "No Projection" row in the second table shows parsing results when no features from

	Gaelic	German	Hausa	Hindi	Korean	Malagasy	Welsh	Yaqui	Overall
Best	87.7	88.7	90.1	77.4	91.8	93.1	94.9	88.0	88.0
Baseline (Right)	86.9	88.0	79.3	57.5	90.3	89.6	89.8	87.3	87.3
Baseline (Left)	77.0	88.0	79.5	68.1	88.9	89.6	89.8	84.3	84.3
Error Reduction	6.1	5.7	51.7	29.3	14.6	33.4	50.0	5.9	5.9

(a) The accuracies of the original projection algorithm (the *Baseline* rows) and the enhanced algorithm (the *Best* row) on eight language pairs. For each language, the best performing baseline is in italic. The last row shows the error reduction of the Best row over the best performing baseline, which is calculated by the formula $ErrorRate = \frac{Best - BestBaseline}{100 - BestBaseline} \times 100$

	Gaelic	German	Hausa	Hindi	Korean	Malagasy	Welsh	Yaqui	Overall
Best	81.4	92.9	88.7	81.4	93.0	93.1	94.9	89.3	89.3
Baseline (Right)	81.0	90.5	87.6	78.0	92.4	92.4	94.2	88.3	88.3
Baseline (Left)	81.0	90.5	89.2	79.6	91.0	92.4	94.2	87.9	87.9
No Projection	55.2	62.7	72.2	65.2	80.8	73.0	91.3	66.1	66.1
Error Reduction (BestBaseline)	2.1	25.7	-4.3	8.4	8.0	8.2	11.8	8.5	8.5
Error Reduction (No Projection)	58.4	81.0	59.5	46.5	63.4	74.2	41.2	68.4	68.4

(b) The parsing accuracies of the MSTParser with or without new features extracted from projected trees. There are two error reduction rows: one is with respect to the best performing baseline for each language, the other is with respect to *No Projection* where the parser does not use features from projected trees.

Table 7.6: Unlabeled Attachment Scores of enhanced projection system (System 4) on the eight languages of the XL-IGT and HUTP corpora.

the projected trees are added to the parser, and the last row in that table shows the error reduction of the "Best" row over the "No Projection" row.

The results in Table 7.6 show that using features from the projected trees provides a big boost to the quality of the statistical parser, reinforcing what was previously demonstrated in Section 7.3.1. Furthermore, the enhancements laid out in Section 7.4.4 improve the performance of both the projection algorithm and the parser that uses features from projected trees. The degree of improvement may depend on the properties of a particular language pair and the labeled data for that language pair. For instance, the *swap* rule is used frequently for the Hindi–English pair because of the choice of headedness for adpositions between Hindi and English. As a result, the enhancement for the swapped alignment alone results in a large error reduction, as in Table 7.7. This table shows the projection accuracy on the Hindi data when each of the three enhancements is turned on or off. The rows are sorted by descending overall accuracy, and the row that corresponds to the system labeled "Best" in Table 7.6b is in bold.

Spont	Swap	Merge Direction	Accuracy
✓	✓	Left	78.1
✓	✓	**Informed**	**77.4**
	✓	Left	76.7
	✓	Informed	76.1
✓		Left	69.5
✓		Informed	69.0
		Left	68.1
		Informed	67.6
✓	✓	Right	66.3
	✓	Right	65.0
✓		Right	58.8
		Right	57.5

Table 7.7: Projection accuracy on the Hindi data, with the three enhancements turning on or off. The "spont" and "swap" columns show a check mark when the enhancements are turned on. The merge direction indicates whether a left or right choice was made as a baseline, or whether the choice was *informed* by the rules learned from the training data.

Discussion

These sets of experiments have two important takeaways: first, that using projection to transfer DSs between languages is indeed more complicated than just finding the optimal word alignments between language pairs – there are some fundamental differences between how the same sentences are structured in different languages. Secondly, the results of these experiments show that there is indeed room to improve upon the projection algorithms, and that the projection-guided parser performs even better with these improved projections.

In the end, however, this approach is still fundamentally limited by the requirement of having gold-standard parse trees to achieve optimal results, as well as requiring parallel data at testing time. In the next set of experiments, I examine what possibilities there are for using these IGT resources for producing a system capable of parsing monolingual input data.

7.5 Training Dependency Parsers for Monolingual Data

While the previous approaches at improving upon DS projection did indeed result in improvements, they assumed an idealized set of conditions, requiring IGT data as input, manually-corrected DS trees, and gold-standard POS tags.

Given the target of resource-poor languages, however, it would be ideal if the parser produced by the system would have the ability to parse monolingual sentences in the target language without any further intervention. This system would be even more powerful if no language-specific knowledge was required, aside from that which was provided by the IGT found in ODIN.

This end-to-end pipeline from ODIN data to monolingual DS parser is what will be described in this section. I will start with a discussion of how the parser will be trained (Section 7.5.1), followed by a the different scenarios that were used for the experiments (Section 7.5.2) before presenting the results (Section 7.5.3).

7.5.1 Training the Monolingual Parser

The previous dependency parsing systems 1–4 were systems which required IGT data at test time to produce parses. Although these systems were able to leverage additional information through the use of IGT test data, here I would like to produce a system capable of training on IGT data, but running only on monolingual test data. With this goal, I now present the system shown in Fig. 7.19. Here, the IGT instances in ODIN are used as training data for the MSTParser, as described in Section 7.3, but this time without the assumption of any manual DSs or POS tags. Testing is carried out the same as the example shown in Fig. 7.5a.

This system will represent the end of a pipeline of enrichment that includes multiple word alignment methods, POS tag information added by leveraging the unique gloss-line features of IGT, and the ability to draw this data from any of the 1,487 languages in ODIN.

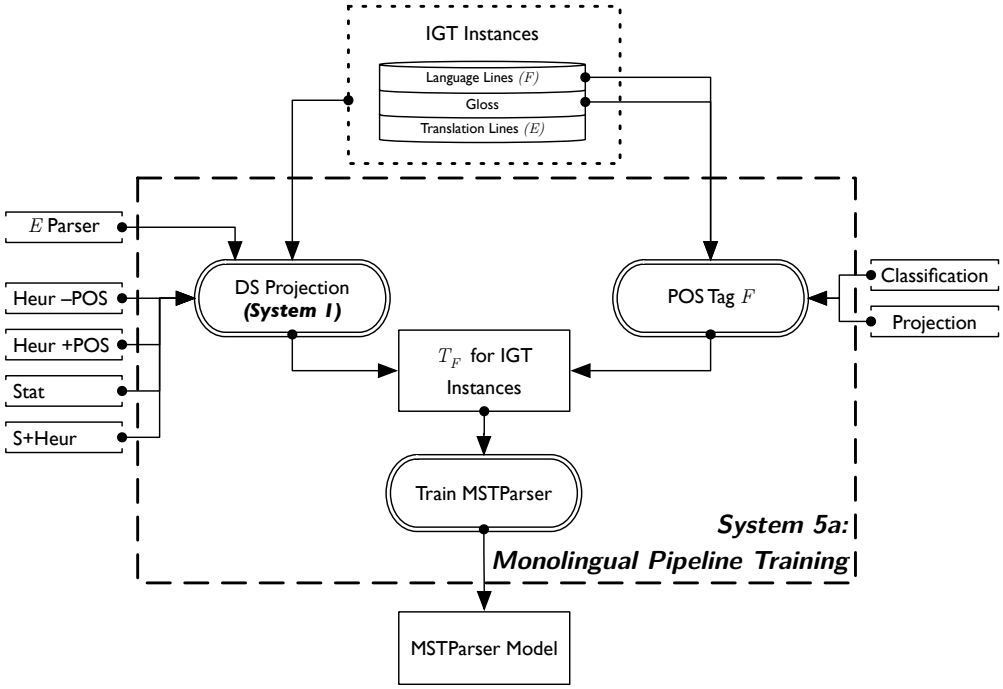


Figure 7.19: Flowchart illustrating the pipeline for training a monolingual DS parser from IGT. This is essentially the same process as the system shown in Fig. 7.5, except that POS tags are not assumed of the IGT instances, and a choice between the classification and projection approaches described in Section 6.3 and Section 6.2 is used to add POS tags to the training T_F trees.

7.5.2 Monolingual Parsing Settings

				Fully Aligned Instances			
Language	**ISO**	**All Instances**	**Filtered Instances**	**Stat**	**Stat +Heur**	**Heur**	**Heur +POS**
Bulgarian	bul	1,802	659	302	302	158	292
French	fra	7,412	1,360	479	479	216	394
German	deu	10,814	3,233	1,659	1,658	813	1,464
Gaelic	gle	954	354	176	174	65	125
Hausa	hau	2,504	1,149	398	398	50	221
Indonesian	ind	1,699	1,101	524	524	131	427
Italian	ita	3,513	964	439	439	166	363
Korean	kor	5,383	2,287	1,496	1,496	575	1,181
Malagasy	plt	1,402	660	146	147	113	242
Portuguese	por	742	251	118	118	65	115
Spanish	spa	3,390	1,340	593	593	249	451
Swedish	swe	1,628	351	180	179	87	159
Welsh	cym	404	196	95	95	54	81
Yaqui	yaq	664	415	303	303	124	277

Table 7.8: Breakdown of ODIN instances, with all available instances; instances filtered for having Language, Gloss, Translation lines, and 1-to-1 language↔gloss alignment; and number of instances with every language line token aligned with one or more translation line tokens for each of the languages tested against the XL-IGT and UD-2.0 corpora.

With the monolingual parsing approach, there are several options to consider for training the parser. Since the parser is being trained on projected IGT instances as in Section 7.2, there are the four available sources for word alignment. An additional setting to consider, given the reliance upon projected DSs, is whether or not to use IGT instances for which there is not complete alignment between translation and gloss lines.[7] Table 7.8 gives a breakdown of the data that will be available for each of these settings. This table shows the number of IGT instances that were available in total for each language under "All

[7]As is the case in all the experiments using the raw ODIN data, IGT instances that do not have an equal number of whitespace-delimited tokens on the gloss and translation lines (ignoring punctuation) are discarded. That is, one-to-one alignment between gloss and language tokens is required, to attempt to cut down on misaligned projection.

Instances," the number of languages for which Language, Gloss, and Translation Lines are all present, and the Language and Gloss lines have a 1-to-1 alignment ("Filtered Instances"), and then the number of instances for which every language line token is aligned with one or more translation line token ("Fully Aligned Instances") for each alignment method. Each additional filtering step greatly reduces the number of available instances, so there is a trade-off of quantity for quality of annotation the more aggressive the filtering.

Finally, I will be using the UD-2.0 corpus (McDonald et al., 2013), which was created from "newswire, weblogs and/or consumer reviews" (p. 94) for evaluation. This means both a substantial difference both in length and domain of the text, so I also evaluate against held-out IGT data with gold-standard DSs from the XL-IGT corpus. In addition, I also perform evaluation on the UD-2.0 corpus using only sentences with a length of 10 words or fewer.

7.5.3 Monolingual Parsing Results

In presenting the results of these experiments, I will first start with the most favorable settings, testing the trained parsers on IGT language data that has been stripped of annotation and excluded from the training data.

Results on XL-IGT Corpus

The tables in Table 7.9 show the Unlabeled Attachment Scores for four settings. Tables 7.9a and 7.9b show the scores when the parser is trained with projected tags and classifier-produced tags, respectively, and when all instances are used for training, whether or not all words on the language line are aligned with a gloss token. Tables 7.9c and 7.9d show the scores when the parser is trained only with instances for which every language line token is aligned with a gloss-line token.

Though the scores are similar between settings in average, they can vary wildly between

Projected Tags – All Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
German	68.7	70.3	59.0	72.4
Hausa	55.8	56.5	36.3	41.3
Korean	73.4	72.6	50.7	68.7
Welsh	55.8	50.8	49.2	60.4
Yaqui	53.6	54.4	39.2	59.4
Overall	63.0	62.9	48.2	61.9

(a) UAS results using projected POS tags and all instances, regardless of how many words in the instance were aligned.

Classifier Tags – All Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
German	70.6	69.9	57.3	65.8
Hausa	51.9	54.7	38.3	41.5
Korean	65.8	65.8	48.7	65.8
Welsh	52.8	55.5	49.8	53.8
Yaqui	55.1	52.9	33.4	59.4
Overall	61.0	61.3	46.7	58.5

(b) UAS results using classifier-produced POS tags and all instances, regardless of how many words in the instance were aligned.

Projected Tags – Fully Aligned Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
German	71.5	71.5	69.6	70.0
Hausa	52.6	52.4	18.8	49.2
Korean	76.9	77.7	75.0	73.8
Welsh	41.9	40.6	39.0	47.0
Yaqui	56.9	56.6	53.9	58.6
Overall	62.9	62.8	54.9	62.1

(c) UAS results using projected POS tags and only instances in which an aligned token was found for every word in the language line.

Classifier Tags – Fully Aligned Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
German	70.4	70.4	70.6	66.1
Hausa	50.1	50.3	23.4	37.6
Korean	75.0	76.3	75.3	73.2
Welsh	40.6	40.3	43.1	45.1
Yaqui	59.1	58.4	57.1	60.6
Overall	61.9	62.1	57.2	58.8

(d) UAS results using classifier-produced POS tags and only instances in which an aligned token was found for every word in the language line.

Table 7.9: UASs for parsers trained on the automatically enriched data from all available instances in Odin for that language and tested on the language lines of the XL-IGT corpus data, using POS tags that were either projected using the given alignment method or produced by the gloss-line classification method.

languages. For instances, although Korean shows results of 50.7% and 48.7% using heuristic alignments when all instances are used, this jumps to 75% and 75.3% when only fully aligned instances are used. Conversely, scores for Welsh drop roughly 10% across all alignment methods when only fully-aligned instances are used. As can be seen in Table 7.8, while Korean drops from 2,287 filtered instances to 575 when filtering for those that are heuristically fully-aligned, Welsh drops from 196 to only 54. As might be expected, using fully-aligned instances tends to improve performance by increasing the quality of data, but quantity can become a limiting factor.

With regards to the alignment methods, while there is again variation between specific languages, overall the Stat and Stat+Heur alignment methods outperform the heuristic-only approaches for this task. For the POS tagging methods, it appears that using projected POS tags seems to outperform the classification-based approach overall, as well.

With these three observations, it appears that for this task, a higher-recall alignment method is preferred for optimal results, in combination with the higher precision, lower recall POS tagging methods.

Results on UD Data, Short Sentences ($\leq$ 10 Words)

The tables in Table 7.10 give a breakdown of the different settings for this experiment similarly to the previous table, but for this setting, on sentences from the Universal Dependency Treebank, v2.0 consisting of ten words or fewer. The languages used from the Universal Dependency treebank vary from those used in the XL-IGT corpus, though German is found in both.

By switching domains from the simple, illustrative sentences used in the IGT data from the XL-IGT corpus to the far freer newswire and blog domain of the UD-2.0 corpus, we see the UAS results drop substantially. This change in domain is accompanied by a substantial change in vocabulary from the IGT instances in ODIN. Furthermore, there are likely syntactic

Projected Tags – All Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
French	36.4	36.6	30.0	40.1
German	40.5	39.0	31.2	44.4
Indonesian	24.1	24.1	13.8	28.2
Italian	35.5	36.7	33.5	37.0
Portuguese	35.7	34.2	25.1	36.3
Spanish	35.4	34.6	28.9	34.5
Swedish	27.2	27.2	18.8	26.9
Overall	34.5	34.0	26.5	36.5

(a) UAS results using projected POS tags and all instances, regardless of how many words in the instance were aligned.

Classifier Tags – All Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
French	36.2	35.2	32.9	37.2
German	42.6	42.9	33.1	41.9
Indonesian	23.9	23.9	20.3	27.3
Italian	34.0	34.2	33.3	35.5
Portuguese	37.0	35.3	34.2	36.2
Spanish	30.9	32.3	31.4	32.0
Swedish	26.2	26.2	26.1	35.5
Overall	34.2	34.1	30.7	36.0

(b) UAS results using classifier-produced POS tags and all instances, regardless of how many words in the instance were aligned.

Projected Tags – Fully Aligned Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
French	38.6	38.6	34.8	36.7
German	41.0	41.0	41.7	42.0
Indonesian	25.4	25.4	26.8	30.8
Italian	36.2	36.2	33.5	34.0
Portuguese	30.1	30.9	32.9	30.8
Spanish	33.8	33.3	36.2	34.1
Swedish	32.0	32.0	35.7	31.4
Overall	34.8	34.8	35.5	35.2

(c) UAS results using projected POS tags and only instances in which an aligned token was found for every word in the language line.

Classifier Tags – Fully Aligned Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
French	38.6	38.6	37.4	37.4
German	40.3	40.3	44.4	40.8
Indonesian	26.0	26.0	27.3	28.2
Italian	35.7	35.7	39.7	35.7
Portuguese	33.0	32.8	32.0	34.7
Spanish	29.9	30.0	28.6	30.6
Swedish	29.4	29.4	35.7	35.5
Overall	34.1	34.1	35.8	35.5

(d) UAS results using classifier-produced POS tags and only instances in which an aligned token was found for every word in the language line.

Table 7.10: UASs for parsers trained on the automatically enriched data from all available instances in Odin for that language, using POS tags that were either projected using the given alignment method or produced by the gloss-line classification method. Evaluation was done on sentences from the UD-2.0 evaluation corpus that were ten words in length or shorter.

differences between the IGT instances and the longer newswire sentences. As discussed in Lewis and Xia (2008), there could be an IGT-Bias; that is, given IGT's purpose in illustrating unusual phenomena, the IGT data may be biased toward unrepresentative examples. The longer newswire sentences may also simply differ by virtue of their length—longer sentences may include subordinate clauses with different syntactic structures than are encountered in the typically short IGT instances.

Whatever the cause of the degraded performance, it appears that, for the case when evaluation is performed on UD-2.0 data instead of XL-IGT data, selecting instances which are fully aligned between language and gloss lines results in improvements for every alignment method when averaged across the languages. Among these results, the highest performing system uses heuristic alignments, projected tags, and only fully aligned instances – all methods that favor precision over recall in other settings.

Results on UD Data, All Sentences

Table 7.11 shows the results for the final monolingual parsing experiment, again using the UD-2.0 corpus for evaluation, but this time using all sentences, regardless of length. Here we see the lowest UASs of any of the monolingual experiments, as we have moved not only out-of-domain, but also into a corpus where the average sentence length is now 21.8 tokens, rather than the 4.8 of the Odin corpus. The system that performs the best among these is again the system that achieved the best results for the short sentences, fully aligned instances using heuristic alignment and projected tags, though the UAS of 22.4% is much lower.

7.5.4 Analysis

While the results achieved by these monolingual dependency parsing approaches may be seen as somewhat disappointing, particularly for sentences of unconstrained length, such multilingual dependency parsing is still very much an unsolved problem. Recent work such

Projected Tags – All Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
French	23.1	23.4	16.7	20.0
German	28.8	28.8	23.0	29.4
Indonesian	14.9	14.9	8.2	17.9
Italian	20.5	22.0	16.8	20.6
Portuguese	19.3	22.6	13.3	20.4
Spanish	26.0	25.7	21.9	21.1
Swedish	16.2	16.2	9.7	17.3
Overall	22.3	23.0	16.8	20.8

(a) UAS results using projected POS tags and all instances, regardless of how many words in the instance were aligned.

Classifier Tags – All Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
French	22.1	22.3	21.4	20.6
German	28.4	28.3	25.6	27.7
Indonesian	9.9	9.9	16.0	17.5
Italian	22.4	21.9	19.6	20.8
Portuguese	18.9	17.5	17.9	17.4
Spanish	22.2	24.1	17.9	18.4
Swedish	18.4	18.4	16.6	17.5
Overall	20.8	21.0	19.3	19.5

(b) UAS results using classifier-produced POS tags and all instances, regardless of how many words in the instance were aligned.

Projected Tags – Fully Aligned Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
French	23.2	23.2	22.7	20.9
German	28.4	28.4	29.3	28.6
Indonesian	13.6	13.6	20.1	18.3
Italian	20.8	20.8	19.3	18.8
Portuguese	22.5	21.6	22.0	20.5
Spanish	24.4	26.8	27.9	26.5
Swedish	19.1	19.1	20.5	17.5
Overall	22.6	23.1	23.9	22.4

(c) UAS results using projected POS tags and only instances in which an aligned token was found for every word in the language line.

Classifier Tags – Fully Aligned Instances

	Stat	**Stat +Heur**	**Heur −POS**	**Heur +POS**
French	24.2	24.2	24.6	21.3
German	26.7	26.7	28.2	28.2
Indonesian	14.9	14.9	19.0	17.6
Italian	19.9	19.9	21.1	19.9
Portuguese	17.6	17.6	17.1	15.7
Spanish	19.0	18.6	19.2	19.1
Swedish	18.6	18.6	20.3	18.4
Overall	20.3	20.2	21.0	19.6

(d) UAS results using classifier-produced POS tags and only instances in which an aligned token was found for every word in the language line.

Table 7.11: UASs for parsers trained on the automatically enriched data from all available instances in Odin for that language, using POS tags that were either projected using the given alignment method or produced by the gloss-line classification method. Evaluation was done on all sentences from the UD-2.0 evaluation corpus.

as Xiao and Guo (2015), using millions of parallel sentences available in the Europarl corpus (Koehn, 2005) to produce word alignments and project DSs still achieves a best result of 59.7% average over the eight language pairs covered for sentences of all lengths, and 65.8% for sentences of fewer than ten words. While this is still a great deal higher than the 25.0% and 40.6% seen in the equivalent experiments above, the approach given here is also working with far fewer resources. Additionally, if it is possible to limit the input to more simple sentences such as those found in IGT, the 63.0% UAS achieved for this system is actually fairly competitive, and available for such rare and resource-poor languages as Yaqui and Hausa. In the latter cases, the choice otherwise would be to develop no system at all.

7.6 Summary

In this chapter, I presented five dependency parsing approaches using IGT as training data. The first projection-based approach in System 1 was improved on by subsequent Systems 3 and 4. While these systems outperformed either a parser trained solely on the projected trees or the projection alone, they both relied upon some manually corrected trees to achieve this boost, and required IGT data to parse. System 2, the baseline statistical parser trained directly on IGT data performed extremely poorly, and thus was similarly improved by the enhanced parsers in Systems 3 and 4. System 5 presented a parser which utilized the word alignment system from Chapter 5 and the POS tagging systems from Chapter 6 to generate training data from IGT instances that could then be used on monolingual input data. The results of System 5 were disappointing, but this system is the result of sparse data as well as noisy data, which causes compounding errors through the model pipeline. Future work on other parsing approaches, as well as the data cleaning and expansion of ODIN under the R*i*PL*es* project may improve the performance of these methods substantially.

7.7 Further Work

As the monolingual parsing results show, there is still clearly room for improvement. Below are a few possible paths to follow for future work.

7.7.1 Additional Data Cleaning

Although it will possibly sound a bit repetitive at this point, the dependency parsing results demonstrated in this chapter are the result of a pipeline that combines word alignment, part-of-speech tagging, and dependency projection, all of which are independently affected by pdf-to-text corruption found in the original source data. At the end of the pipeline, the compounding errors are certain to have a substantial impact on the resulting performance of the system. Reducing noise by discarding instances also sacrifices the amount of available training data. Better approaches to reconstructive cleaning could help reduce noise while maintaining the maximum amount of training instances.

At the time of writing, the R*i*PL*es* project is currently working on improving and expanding the source data for ODIN by reprocessing the data with an improved pdf to text extractor on both the old documents, and a large number of new documents. When that work is complete, it is possible that the systems outlined here may see substantial gains just by use of the new data. Given that the software I created for this thesis will be available online, this expanded ODIN data should be able to bootstrap new, improved systems immediately once available.

7.7.2 Use Modified Parser Model

As shown in previous sections, lack of alignments can be a cause of error when attempting to project DSs, and heuristic methods to reattach unaligned words do not appear to result in trees that are particularly well suited to training subsequent parsers. One possible avenue would be to create a modified DS parser, following Spreyer and Kuhn (2009), where the

trees created by the initial projection need not be complete. Instead, the modified parsers use a modified score for calculating likely dependency links during training such that only dependencies that arise from sure alignments are used. Such a modification could reduce errors caused by unaligned words that were incorrectly reattached during projection.

7.7.3 Similarity-Based Approaches

Finally, one of the weaknesses of IGT remains its sparsity. Another path for future research would be a variety of approaches that focus on expanding the coverage of category identifiers for unseen data, whether through clustering based on part-of-speech tags (Koo et al., 2008) or more general lexical similarity features (Mirroshandel et al., 2012). In particular, these approaches would help supplement a failing of the systems produced by projection with IGT, in that there is a large data sparsity issue for the open-class words. As Mirroshandel et al. noted, even for supervised parsers trained on several thousand sentences, performance is greatly improved on open-class words. In conjunction with improved POS induction and word alignment, OOV rates for POS tagging these approaches could be greatly reduced, which is one of the key weaknesses of IGT-based systems. This improved tagging for previously unknown words would in turn inform the parser, leading to improved parsing accuracies as well.

Chapter 8

INTENT: THE INTERLINEAR TEXT ENRICHMENT TOOLKIT

The **IN**terlinear **T**ext **EN**richment **T**oolkit, or INTENT, is the code package I wrote that implements the IGT enrichment methods presented in this thesis. The following resources are available:

Code	`https://github.com/rgeorgi/intent`
Annotated IGT Data	`https://github.com/rgeorgi/xigt-data`

As I will be distributing this package for any researcher who wishes to use it under the MIT license, I will use this chapter to give an overview of the software and the main commands that can be used, though more complete and up-to-date documentation can be found at the URLs above.

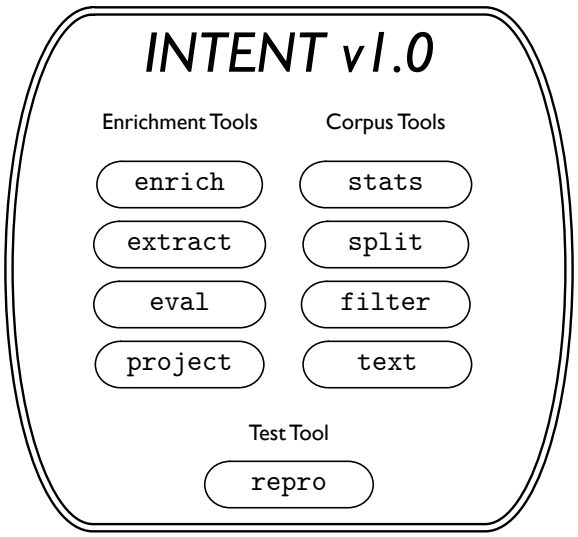


Figure 8.1: Overview of the modules of the INTENT software package.

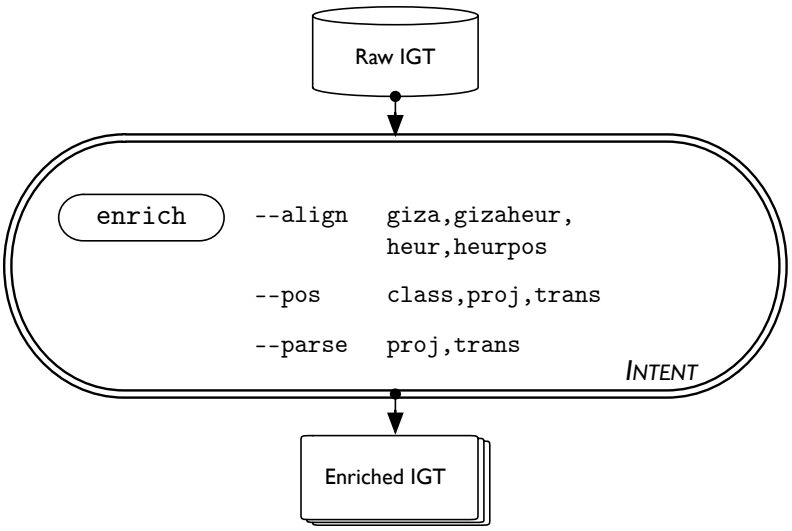


Figure 8.2: Flowchart demonstrating the primary arguments to the `enrich` command and their options.

Figure 8.1 gives a brief overview of the primary modules contained within the overall INTENT package, while Sample 8.1 shows the main commandline prompt for INTENT. These subcommands are split over two primary tasks. The first group deals with enrichment of IGT instances, while the second are utilities designed to deal with *Xigt-XML* corpora (Goodman et al., 2014). I will describe the function of each module in the following sections.

8.1 `enrich`

Perhaps the most important command in the INTENT package is the `enrich` command. The `enrich` command, as illustrated in Fig. 8.2, provides three arguments to provide the enrichment for word alignment, POS tagging, and dependency parsing and projection described in Chapters 5 to 7. In addition to DSs, the `--parse` argument also supports phrase structure parsing and projection.

The example in Sample 8.2 gives a commandline for running enrichment on a `cmn.xml` *Xigt-XML* file, which would correspond to Mandarin Chinese in the ODIN-2.1 data.

```
usage: intent.py [-h]
                 {enrich,stats,split,filter,extract,eval,text,project} ...

This is the main module for the INTENT package.

positional arguments:
  {enrich,stats,split,filter,extract,eval,text,project}
                        Valid subcommands
    enrich              Enrich igt data.
    stats               Get corpus statistics for a set of XIGT files.
    split               Command to split input file(s) into train/dev/test
                        instances.
    filter              Command to filter input file(s) for instances
    extract             Command to extract data from enriched \xigtxml{} files
    eval                Command to eval INTENT functions against a gold-
                        standard \xigtxml{}.
    text                Command to convert a text document into \xigtxml{}.
    project             Command that will (re)project pos/ps/ds using the
                        specified pos source and alignment type.

optional arguments:
  -h, --help            show this help message and exit
```

Code Sample 8.1: Main commandline usage prompt for INTENT, with a list of all available subcommands.

```
intent.py enrich --align heurpos --pos proj,class --parse proj \
cmn.xml cmn-enriched.xml
```

Code Sample 8.2: Example enrichment command providing Heur+POS alignment, both classifier-based and projection-based POS tagging, and DS and PS projections.

The `--align` argument selects a word alignment method to use in aligning the gloss and translation lines from among `giza`, `gizaheur`, `heur` or `heurpos`. These settings correspond to the **Stat** (Section 5.3.2), **Stat+Heur** (Section 5.4), **Heur −POS** and **Heur +POS** methods (Section 5.2) discussed previously. The `--pos` argument selects a POS tagging method between `class` for classification (Section 6.3) , `proj` for projection-based (Section 6.2), or `trans`, which tags only the translation line. The `--parse` argument similarly offers an option of `proj` to parse and project the DS/PS trees (Section 7.2), or `trans` to only parse the translation line.

After the `enrich` command is run, it generates an output *Xigt-XML* document with the requested enrichment, if possible. Some issues with the IGT data, such as lack of 1-to-1 gloss–language line alignment will cause enrichment involving the language line to fail, and the instance will be left unenriched in the output.

For a user who would like only POS tags on Welsh (`cym`) with maximal precision, potentially sacrificing recall, an appropriate command would be to enrich using only projected tags aligned using the heuristic alignment:

```
intent.py enrich --pos proj --align heur cym.xml cym_tagged.xml
```

Alternatively, a user who wants the maximum number of POS-tagged language line tokens on Hausa (`hau`), and cares less about precision might use simply:

```
intent.py enrich --pos class hau.xml hau_tagged.xml
```

8.2 `extract`

The `extract` subcommand is the second-most important command in the INTENT package, in that it provides the mechanism by which to extract data from the enriched IGT. The flowchart in Fig. 8.3 shows the `extract` subcommand's various arguments, and the output products that each generate. Unlike the `enrich` command, the `extract` command expects that the provided *Xigt-XML* file already contains enrichment. This enrichment may be generated from INTENT itself, or from other sources, such as XIGTEDIT (Xia et al., 2016).

The `--classifier-prefix` argument uses POS tags found on the gloss-line tokens in the enriched IGT instance to train a MALLET (McCallum, 2002) MaxEnt classifier (Berger et al., 1996). The `--tagger-prefix` argument will use the POS tags found on language-line tokens to train a Stanford Tagger POS tagging model (Toutanova et al., 2003). The `--dep-prefix` will train a Stanford Parser model (Klein and Manning, 2003; de Marneffe and MacCartney, 2006) on projected dependency structures found on the language line.

The last two arguments are not used for purposes explored in this thesis, but would be useful for other tasks. The `--sent-prefix` argument produces either language–translation

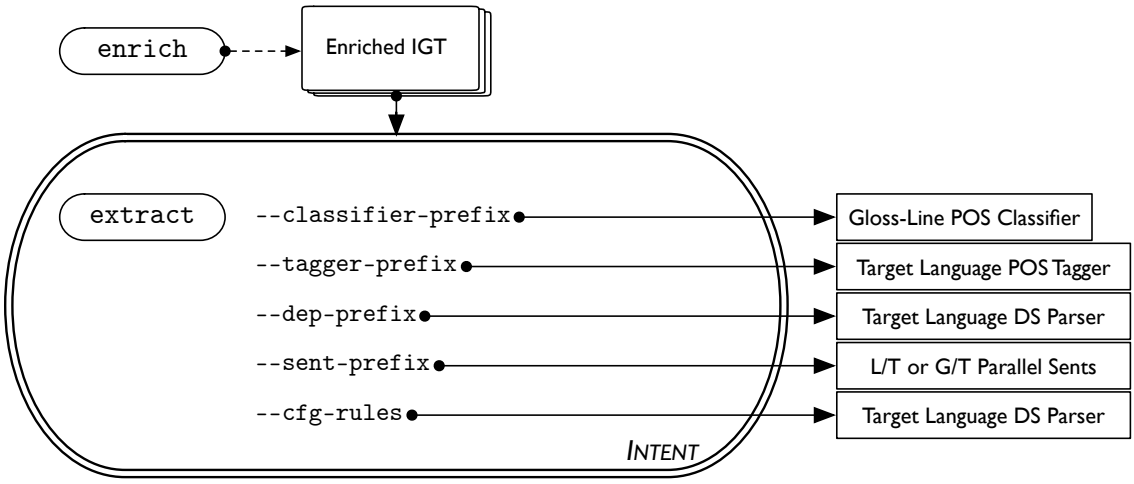


Figure 8.3: Flowchart illustrating the `extract` command, and the possible outputs.

or gloss–translation parallel sentences, with or without heuristic word-level matches, which could potentially be useful in producing or improving a word alignment system between the target language and English. The `--cfg-rules` will produce a list of CFG rules extracted from phrase structures that have been projected to the language line, which could potentially be analyzed to infer syntactic phenomena from a language. While I have not described the phrase structure projection process in this paper, my projection algorithm follows that of Xia and Lewis (2007).

Sample 8.3 gives an example of how the `extract` subcommand might be used to extract a product for a target language from an enriched *Xigt-XML* file. In this case, the `--tagger-prefix` argument is given a `german*` filename prefix in the `./taggers` subdirectory, and will extract the language-line POS tags from the file `deu-enriched.xml`.

```
intent.py extract --tagger-prefix ./taggers/german deu-enriched.xml
```

Code Sample 8.3: An example commandline for the `extract` subcommand, which will produce a German POS tagger from the `deu-enriched.xml` file.

8.3 `eval`

The `eval` subcommand is used for evaluating performance of various INTENT methods on a *Xigt-XML* document containing gold-standard annotations. If gold-standard POS tags are present, it can evaluate a gloss-line classifier using `--classifier` and a specified classifier file, as well as `--pos-projection`. If gold-standard word alignments are available, `--alignment` can be specified to give a report of how the different alignment methods perform. If gold-standard DSs are present in the file, the `--ds-projection` argument will evaluate ds projection against those.

The commandline in Sample 8.4 shows an example where the `my_gloss.maxent` file—which can be extracted from an IGT with the `extract` subcommand—is evaluated against

```
intent.py eval --classifier my_gloss.maxent fra-gold.xml
```

Code Sample 8.4: Example of an `eval` commandline.

```
intent.py project --aln-method heur --completeness 0.75 \
jpn-enriched.xml jpn-projected.xml
```

Code Sample 8.5: An example commandline for the `project` subcommand.

the `fra-gold.xml` file containing gold-standard gloss-line POS tags.

8.4 `project`

In many ways, the `project` subcommand is similar to the `enrich` subcommand in that it produces enriched IGT. However, unlike `enrich`, `project` expects an already-enriched *Xigt-XML* file that has word alignments and translation-line POS tags or DSs. Similar to the `extract` subcommand, the input enriched *Xigt-XML* file need not have been generated by INTENT. This subcommand can be used for testing the projection algorithms with files that provide gold-standard POS tags or word alignment.

The `project` subcommand takes two arguments, `--aln-method`, which selects the alignment method from the file to use in case there are multiple methods specified. Aside from the four INTENT methods, this argument may also select for `manual` word alignments or `any`, which selects the first available alignment. Selecting `any` alignment may be required if the metadata specifying the alignment type is not present in the file. The second argument, `--completeness`, provides the option to only project the annotations from the translation line if a given proportion of the words are aligned. This proportion is represented as a decimal between 0 and 1, inclusive.

The example in Sample 8.5 will use the `heur` alignment method to project from the translation line of the `jpn-enriched.xml` file, but only when at least 75% of the language

line tokens can be aligned. The output will be written to `jpn-projected.xml`.

8.5 Corpus Management Subcommands

The `stats`, `split`, `filter`, and `text` subcommands are used as simple utilities for working with *Xigt-XML* corpora, and I will discuss their functions only briefly here. The `stats` command provides simple statistics on an *Xigt-XML* corpus file, consisting of counts of instances and tokens as well as counts of unique words. The `split` command offers the ability to split a corpus into train, dev, and test sets, with the ability to specify the estimated proportion of tokens that should appear in each.

The `filter` subcommand can be used to filter *Xigt-XML* documents according to whether the instances they contain do or do not have gloss, language, or translation lines (as some instances are missing one or more), or whether or not they have one-to-one alignment between whitespace-delimited language and gloss tokens. Although INTENT will attempt to fail gracefully when an instance fails to meet the requirements for enrichment or extraction, INTENT will attempt to maximize the utility of every instance available. For example, only a POS tagged gloss line is required to extract a gloss-line classifier. This may mean that certain extracted systems may use more instances than others, depending on the approach. For comparing systems based on a shared set of instances, it may be more appropriate to pre-filter the *Xigt-XML* document.

Finally, the `text` subcommand can be used to convert plain text of Language, Gloss, Translation lines separated by whitespace into a *Xigt-XML* document.

8.6 Reproducing The Experiments (`repro`)

An all-too common problem in the sciences is the difficulty in reproducing a set of experimental results. The `repro` subcommand seeks to address this problem by providing a built-in set of scripts that reproduce the experiments detailed in this thesis, when used in combination

Component	Version	Purpose
Python	≥3.4	Main codebase
Mallet	≥2.0	Classifier support
NLTK	≥3.0	Tree structure support
Xigt	≥1.0	Xigt Support
Java	≥8	Support for Mallet, Stanford Parser/Tagger
mgiza	≥0.7	Statistical Alignment
Stanford Parser	≥3.6.0	DS/PS parsing
Stanford POS Tagger	≥3.6.0	POS tagging

Table 8.1: Software requirements for INTENT

with the data made available at the link above. The five options for the `repro` command are `ds-igt`, `ds-mono`, `pos-igt`, `pos-mono`, and `align`. These commands reproduce the experiments of the dependency structures and POS tagging experiments, both on IGT and monolingual data, and the alignment methods.

8.7 Interfaces to INTENT

While INTENT was initially designed as a commandline tool, it quickly became apparent that, due to the requirements of installing a python interpreter and several third-party software packages for full functionality, this commandline interface might be too cumbersome for many end-users to install, given the platform-specific dependencies (see Table 8.1 for a list of required packages.)

In order to simplify usage for end-users, and additionally offer a basic query and submit functionality, INTENT-WEB was developed as a web-based interface for common enrichment tasks. Figure 8.6 shows a screenshot of INTENT-WEB using the Upload/Enrichment mode to enrich an instance of German, with verbose output. This interface can be accessed at `http://intent.xigt.org/`.

8.8 INTENT *For Cleaning and Annotation*

Another tool that makes use of INTENT is a web-based IGT editor tool, that enables the creation of manually-reviewed, gold-standard IGT data. This tool helps support the R*i*PL*es* project (Xia et al., 2016)[1] in providing a universal, browser-based interface for annotators to easily work with Xigt-based IGT data. At the time of writing, cleaned and normalized IGT data is the end product, with a visualization of the INTENT-produced annotations, but it is hoped that future modifications will add the ability for annotators to correct these annotations. The current cleaning process improves IGT usability in the computational approaches outlined in this thesis, as well as providing a means to create gold-standard data that can be used to evaluate different automatic cleaning approaches.

Figure 8.7 gives an example of the basic editor interface, and its ability to provide access to multiple corpora, as well as select different instances within each corpus. Additionally, annotators may change the automatically detected `TAG` for a line, which identifies whether an instance is a **L**anguage line, **G**loss line, **T**ranslation line, or contains non-IGT **M**etadata. Users can also tag each line with multiple labels, such as the `LN` shown in Fig. 8.7 that indicates the **L**anguage **N**ame.

Figure 8.8 shows where INTENT is used more extensively in the interface, to perform the automatic segmentation and some initial enrichment for the user to verify that the instance has been cleaned appropriately for the automated methods to be functioning correctly. In future releases, users will have the ability to correct the automatically generated enrichment, in order to create gold standard corpora with minimal additional human effort.

8.9 Summary

While a great deal of the work involved in this thesis was creating the methods with which the problems would be approached, INTENT represents the bigger task of implementing those

[1] `http://faculty.washington.edu/fxia/riples/`

methods, and making them available in a tool that users can apply to their own data. As a researcher with interest in rare and endangered languages, my hope is that the release of this software, along with the ODIN-2.1 data and subsequent versions, will provide tools with which other researchers may find the means to answer questions about or build tools for many languages that have, up until this point, had little to no other resources available. While INTENT is by no means a comprehensive tool for all possible IGT-related tasks, I hope that it makes the data format more accessible to a broader audience.

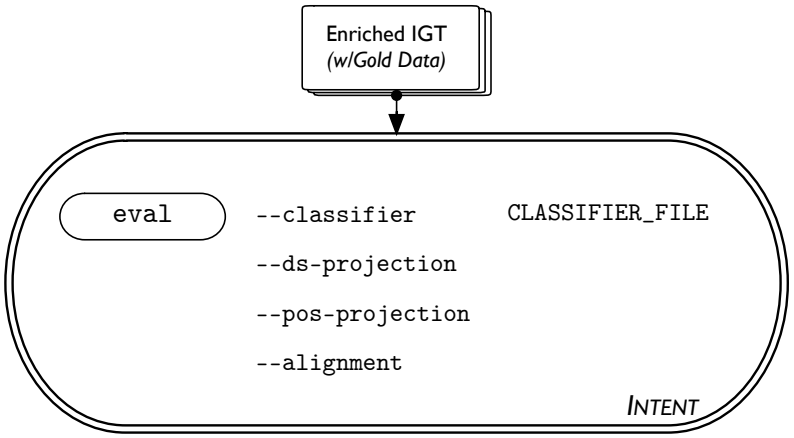


Figure 8.4: Flowchart illustrating the `eval` command and its arguments.

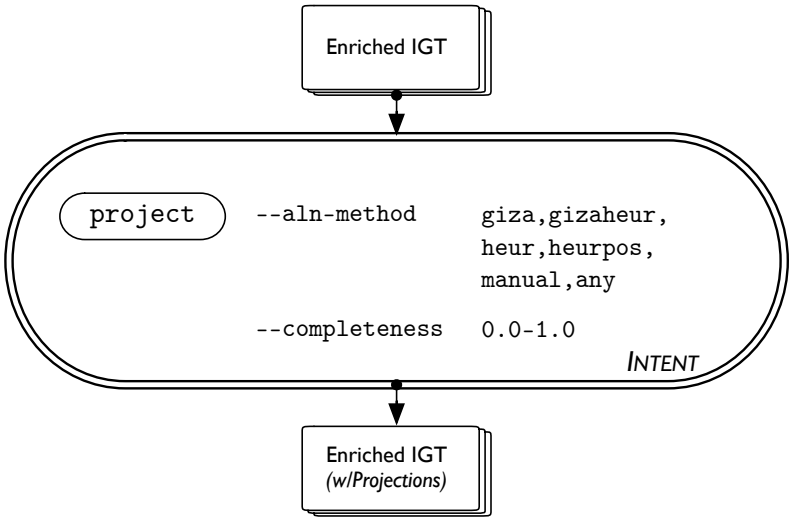


Figure 8.5: Flowchart illustrating the `project` subcommand and its arguments.

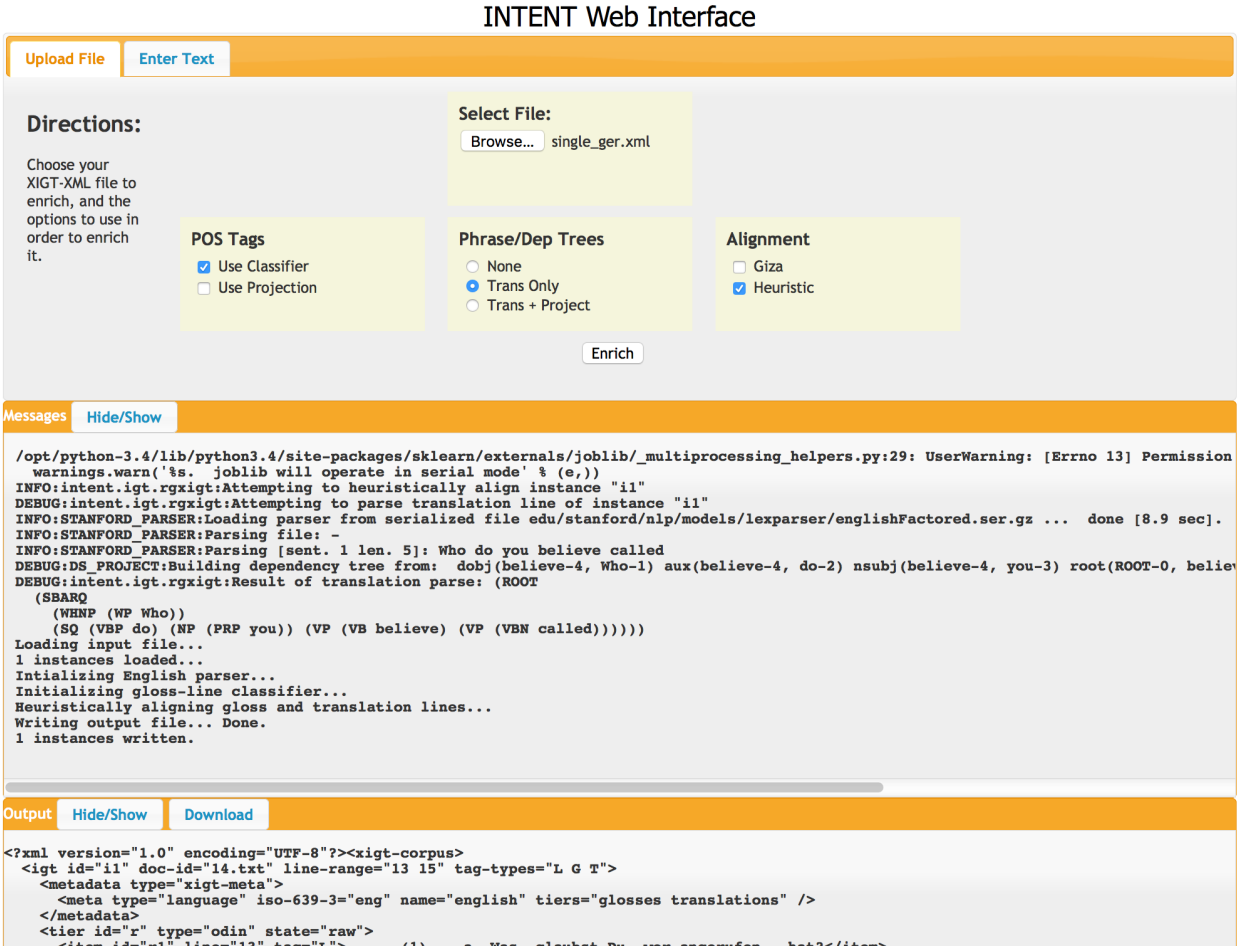


Figure 8.6: The INTENT Web Interface running at the University of Washington's Department of Linguistics server.

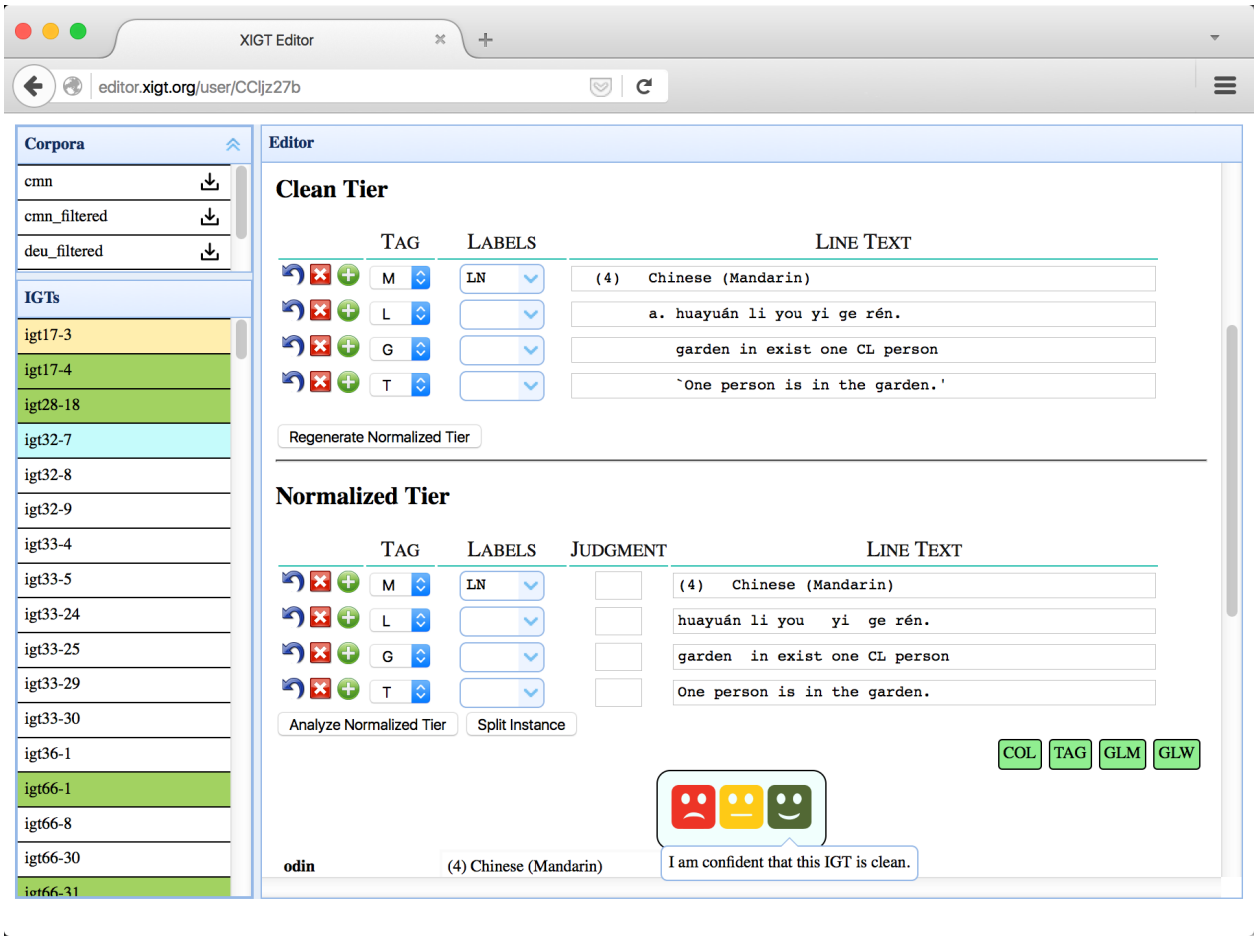


Figure 8.7: Overview of the XIGT editor interface for cleaning IGT instances. The editor allows annotators to clean corruption from XIGT instances, normalize the data so that it is ready for the automated methods of INTENT and other systems, and provide a cleanliness rating and other comments that are saved in the IGT instance metadata.

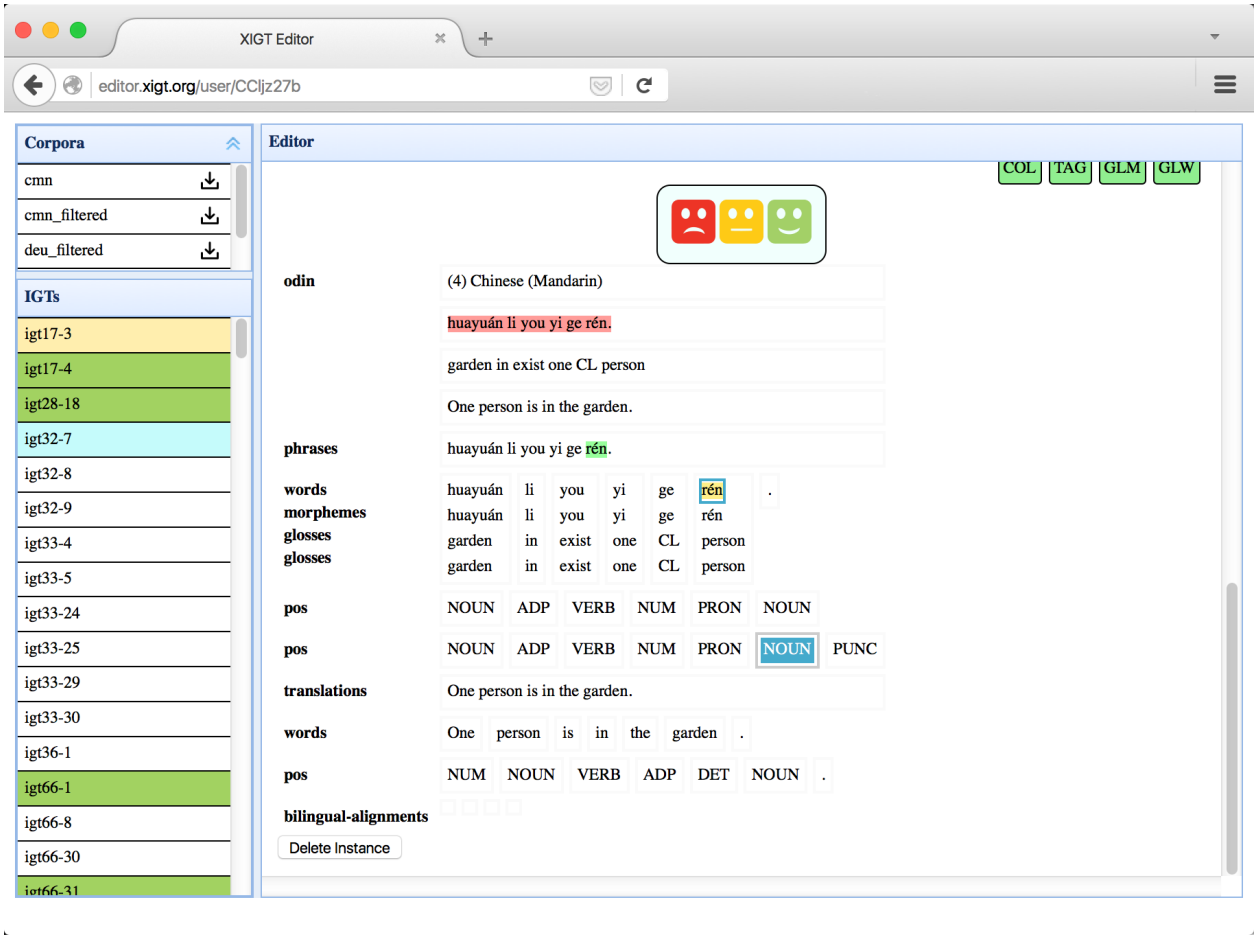


Figure 8.8: INTENT-enriched information is automatically generated by the editor interface for inspection by the annotator. Future releases will include the ability for annotators to correct the enriched data, as well as view other information such as word alignment.

Chapter 9

CONCLUSION AND FUTURE WORK

In this thesis, I sought to answer the question: *can the existing language knowledge contained in Interlinear Glossed Text be harnessed to perform basic NLP tasks on resource-poor languages in a repeatable and broadly applicable manner?* For my conclusions, I will examine the answers that I found to this question, and suggest pathways for future studies.

9.1 Summary of Results

Word Alignment

Chapter 5 proposed two primary approaches to using the unique format of IGT instances to align words between translation and language lines for the purposes of enabling many of the subsequent methods through projection-based methods. The first of these methods was a purely heuristic approach, taking advantage of the unique gloss line that IGT provides to find tokens that matched between translation and gloss line. That "matching" could be defined as literal string matches, a mapped set of grams, or words with matching POS tags.

The second approach utilized statistical alignment methods, but leveraged the thousands of instances in the ODIN database that reuse translation words and gloss words, regardless of the language the instance is annotating. Thus, a language with only 10 instances would still potentially have access to 151,623 other language-gloss sentence pairs with which to inform alignment. This statistical method could be informed by the higher-precision, lower-recall heuristic method to add heuristic matches to the alignment data.

Table 9.1 Experiments on these methods showed that the best-performing heuristic

System	Settings	Precision	Recall	F-Measure
Heuristic	− POS	0.96	0.75	0.85
	+ POS	0.86	0.83	0.85
Statistical	G-T (+ODIN +Heur)	0.89	0.78	0.83
	L-T	0.47	0.51	0.49

Table 9.1: Summary of word alignment results on IGT instances in the RG-IGT corpus. Heuristic alignment is shown with or without the POS tag matching heuristic, while the statistical approaches are shown using either the augmented gloss/translation alignment (G-T), or the language/translation alignment(L-T).

method achieved an F_1-score of 0.88 on the RG-IGT corpus, and 0.86 on the XL-IGT corpus. The best statistical approach showed results of 0.86 and 0.80 on the same datasets. While the heuristic approach appears to outperform the statistical approach, these evaluation corpora are very small, and both approaches appear to be good options for aligning IGT instances. Furthermore, these results show that IGT instances are able to produce high-quality word alignments even with very little data to align.

Finally, both alignment methods might be used to inform one another. The heuristic approach was used to add high precision alignments to guide the statistical approach, while a future path of research could be investigating using high-reliability statistical alignments to create a translation $\leftrightarrow$ gloss line translation dictionary as an additional alignment heuristic.

POS Tagging

Chapter 6 focused on two main tasks for obtaining POS tags for IGT instances. The first used word alignment and a POS tagger trained on English to tag the translation line and project those tags to the target language. This was the standard approach for projecting a POS tagger in previous work, although this the work in this thesis was able to show substantially higher performance than previous work when projecting POS tags within IGT

Projection	Classifier	
	Automatic	Manual
66.8	84.3	92.9

Table 9.2: Summary of POS tagging accuracies on the RG-IGT corpus. Projected tags are evaluated without special handling for unaligned words. The classifier approach is evaluated either using tokens labeled by automatic projection or manually, with a 90/10 split and tenfold cross-validation.

instances, due to the availability of high-quality word alignment.

The second method focused instead on the particular qualities of IGT, namely, the information encoded in the gloss line. Despite even high quality word alignment, unaligned words or incorrectly aligned words present a problem for projection methods, I posited the approach of looking instead at the gloss line's grammatical markers such as `3.SG.FEM` and using such markers as features in a classifier to predict the POS tag of the gloss token directly. The results of this classifier could even be used as a preprocessing step to the word aligner to find unaligned words with matching POS tags and attempt to align them to improve recall.

One further addition to the classifier-based approach was that rather than training the classifier with a small collection manually-labeled gloss tokens, the classifier could be trained with all of the gloss tokens for which heuristic alignments were available, thus greatly improving coverage, if at the cost of introducing noise.

Table 9.2 shows the summary of the projection and classifier-based approach POS tagging methods on the RG-IGT corpus. The projection approach achieved an accuracy of 66.8% when using automatic alignments. The classifier approach on the same data achieved 92.9% for the manually-labeled data and 84.3% for the automatically-labeled data. On IGT test data, the classifier-based approach thus reduces errors over the projection-based approach on by 79% with manually-labeled data and 53% with automatically-labeled data.

After either of these methods were run, the resulting IGT instances with POS-tagged

	Projection		Classifier		Supervised	
	Partial	Full	Automatic	Manual	1K	All Tokens
All Sentences	56.3	63.4	66.3	69.2	77.7	95.4
Short Sentences	62.7	65.5	69.5	70.4	74.9	94.1

Table 9.3: Table summarizing the overall POS tagging accuracies evaluated against the UD-2.0 corpus. Tagger results using projected POS tags are shown allowing partially aligned sentences, or only fully aligned sentences. Classifier approaches are shown with tokens labeled via automatic projects as well as manually. Supervised approaches are shown using 1,000 tokens and all available tokens.

language-lines could thus be used as a source of training data for POS taggers in the target language, the results of which are summarized in Table 9.3. Although it represented a shift of domain from IGT, these approaches were used to tag IGT instances for the languages in the UD-2.0 treebank. The best performing projection-based method and classifiers using either automatically or manually labeled data achieved 63.4%, 66.3%, and 69.2% respectively. While a fully supervised approach was able to achieve 95.4% accuracy overall, a supervised approach utilizing a number of tokens comparable to those used in the IGT instances achieved only 77.7% overall. Evaluating on sentences of only ten words or fewer, and thus more comparable to the short sentences in IGT closed the performance gap even further.

This thesis presents a unique approach of training a classifier to work on the gloss line of IGT, providing a method to obtain POS tags on the language line without requiring alignment with the translation line. The system that results shows that while not directly competitive with fully supervised methods, POS taggers can be created for over a thousand languages with nothing more than IGT data that is freely available in ODIN. Although performance of the resulting systems are limited by the amount of IGT data available and the cleanliness of said data, the R*i*PL*es* project is currently engaged in extracted more and cleaner data for subsequent versions of ODIN.

	Alignment				
Source DS	G-T (+ODIN)	G-T (+ODIN +Heur)	Heuristic	Heuristic (+POS)	Manual
Parser	63.8	63.7	55.5	64.1	69.1
Manual	72.6	72.6	61.9	71.2	81.0

Table 9.4: Summary of dependency structure projection UASs on the XL-IGT corpus with various alignment sources, and with source DSs generated either manually or by a parser.

Dependency Parsing

In Chapter 7, I addressed the task of dependency parsing in a number of potential settings. The first was to project DS trees using IGT-based word alignments similar to the initial approach used for the POS tagging experiments. When projections were performed using a parser and automatic alignments, as Table 9.4 shows, the Unlabeled Attachment Score (UAS) was 64.1% on the XL-IGT corpus. Using gold-standard trees and manual alignments, the results were as high as 81.0%.

The next approaches looked at two different ways of using a small set of manually corrected trees to improve the dependency parses produced by these projections, the results of which are summarized in Table 9.5. The first of these extended approaches took a set of projected trees for the target language and had an annotator manually correct them. Then, a parser was trained that had been modified to use the suggested projected edges as a fea-

	Unimproved Projections	Improved Projections
Parser + Projection	88.4	89.3
Projection	84.3	88.0
Parser	67.3	

Table 9.5: Summary of Unlabeled Attachment Scores (UASs) for dependency parsing on the XL-IGT corpus.

ture, and tested against unseen IGT instances. In this setting, the combined parser was able to improve from the projection-only 81.0% to 88.4%, a 39% reduction in error, and a 66% reduction in error over the 65.8% of the baseline parser that did not use the projected trees.

The second approach took the same set of manually corrected trees and compared them against the English parse trees using a set of defined tree comparison operations. I used these tree comparison operations to measure the structural differences between languages, and used these statistics to develop a method for rewriting the dependency structures post-projection.

Finally, as with the POS tagging task, once an IGT instance has projected dependency structures available for the target language, these structures can be used to train a monolingual dependency parser. While it is exciting to potentially be able to generate a monolingual dependency parser using nothing more than IGT data, the results on this monolingual parsing task were less promising. When the language-line sentences in the IGT were used for evaluation, the best-performing system, using statistical alignments, and POS tags projected with all available instances, achieved 63% UAS. Heuristic alignment performed uniformly poorly, though heuristic alignment combined with POS tags was competitive with the statistical methods. This result suggests that for the dependency projection task a higher recall method is preferable to a system that emphasizes only precision.

When the monolingual parsing target was moved from IGT instances to the newswire text in the UD-2.0 corpus, however, performance dropped such that the best performing system achieved only 36.5% UAS on sentences of ten words or fewer, and 23.9% overall. While disappointing, these figures are still for a system dealing with a great many compounding errors and noisy data. While it does not yet compete against state-of-the art systems for resource-rich languages with high-quality annotated training data, my system is able to produce parsers for languages for which existing state-of-the-art systems do not have sufficient data to train against. Even with less than optimal performance, the produced parses can still serve to reduce labor in treebank creation by producing parses which can be corrected

Corpus	IGT	UD-2.0	
Sentences		All	≤10
All Alignments	63.0	20.8	34.5
Full Alignments	62.9	22.6	34.8

Table 9.6: Summary of dependency parsing results, both evaluated against the combination of RG-IGT and XL-IGT corpora (IGT) and on the UD-2.0 corpus. For the UD-2.0 corpus, the parsers were evaluated against all sentences, and sentences with ten words or fewer.

with less effort than starting from scratch.

In answer to the research question for the dependency parsing task, I found that IGT can indeed produce parse trees repeatably and broadly for resource-poor languages, though the quality of these trees is of more usefulness in an active-learning system than as end products.

9.2 Contributions of This Work and Potential Uses

Overall, this thesis constitutes a major step in providing tools for languages for which previously no electronic resources were available. With IGT data available for approximately 1,500 languages currently in ODIN, the methodology presented here provides the means to quickly and automatically enrich these raw IGT instances. This is a substantial development, and opens up a number of tasks that can now be accomplished on languages for which these tasks previously would have been impossible.

Word Alignment and Machine Translation Although I discussed obtaining word alignment for IGT instances using the INTENT software, the word pairs that are extracted from the IGT instances using the various alignment methods the package provides can be used to seed the translation dictionaries for languages for which some amount of parallel data is available. While this approach will not guarantee high-quality word alignment, it might help improve alignment quality and thus translation quality for languages where a

small amount of parallel data exists, just not the amounts typically used for MT systems. For languages where data is scarce, even modest improvements in MT output can be highly impactful.

Typological Studies Also implemented in the INTENT software is the ability to project and extract CFG rules from phrase structures. While the projection process might be noisy, the CFG rules, if filtered appropriately, or analyzed for statistically significant patterns, could be used to infer interesting syntactic properties over the 1,500+ languages in the ODIN database. Additionally, using the POS tags generated on the gloss line by the classification POS-tagging approach to identify nouns and verbs could be used along with gloss-line grammatical markers to attempt to detect whether a language has accusative or ergative alignment, or is something else entirely. Indeed, this type of typological inquiry is one of the goals of the AGGREGATION project[1], as discussed in Bender et al. (2014).

In addition to the editor interface described in Section 8.8, the enriched data provided by INTENT for ODIN v2.1 will soon be included in a search interface that will allow linguists to search for typological characteristics of a language as represented in the enriched data.

Treebank/Resource Creation While the automated approaches described in this thesis are able to function independently of additional expert knowledge, data that has been cleaned and vetted by a language expert are valuable not only for training higher-quality systems than automated approaches are capable of, but also for evaluating systems in a given language. One of the immediately useful uses for which the systems in this thesis may be used is for improving the speed and ease of treebank and resource creation. Though not currently implemented in the editor interface, the ability for users to correct the INTENT-produced annotations would essentially turn IGT data into the source for an active-learning system,

[1]Automatic Generation of Grammars for Endangered Languages from Glosses and Typological Information, `http://depts.washington.edu/uwcl/aggregation/`

potentially producing word-aligned and POS-tagged parallel treebanks for previously unavailable languages. It is even possible that were the correction task to made simple enough for native, non-linguist speakers, this INTENT-aided correction task could be crowdsourced, creating new gold-standard resources for future studies.

Other Impacts These are only a few ways in which INTENT can be used at present. My research also had a number of findings about IGT as a resource more generally.

Significantly, my research also shows that although a single resource-poor language may only have a handful of IGT instances available in ODIN or other sources, INTENT can use IGT instances from other languages to improve improve performance on word alignment and POS tagging tasks. Thus, as ODIN is expanded with more, and cleaner, data, the resulting INTENT systems should improve, both by incorporating more training instances for languages in ODIN, but also by obtaining more cross-linguistically useful data.

Furthermore, the work done in this thesis contains, as far as I am aware, the first published results of a part-of-speech tagging system for a number of rare or endangered languages, including Chintang, Hausa, Welsh, and Yaqui. INTENT can generate similar systems for any language with available IGT data. Only lack of the ability to evaluate such systems prevented more results from being reported here.

This thesis has affirmed that IGT can indeed be harnessed as a computational resource for a number of basic NLP tools. While these tools are simple, they may serve as the basis for many other related tasks and provide a foundation to form typological queries over a data source that covers over a thousand languages.

9.3 Future Work

This thesis represents a major advance in the domain of resource-poor language tool development, though there naturally remain many paths yet available for further research.

Word Alignment For the word alignment task, there are two promising experiments that could be taken as the next steps to the work given here. First, the work of Gao et al. (2010) uses high-precision "known" word alignments within a sentence pair to constrain the statistical alignment search to be consistent with those known links. Using the high-precision heuristic alignments with this aligner to align IGT instances would impose a stronger constraint on the statistical alignment approach than is currently done by adding the extracted heuristic matches as additional sentence pairs, hopefully improving precision of the statistical approach while benefiting from the improved precision.

With respect to non-IGT data, the parallel text that INTENT is able to extract from IGT instances in the form of both the parallel sentences and translation lexicons could potentially be used to boost performance of an existing statistical alignment system for which parallel data is available, but perhaps limited.

POS Tagging For the part-of-speech tagging task, while the coverage of IGT appears limited, the potential for high-quality, but sparse POS-tagged data seems to be well-suited for a number of semi-supervised approaches. The prototype approach of Haghighi and Klein (2006a) seems particularly appropriate for this task, although the noise of IGT data and its sparsity might make extracting prototypes that are truly representative of the word class difficult. Similarly, attempting to model ambiguity classes as in Toutanova and Johnson (2007) sounds promising, but again the noise and sparsity in IGT might be a limiting factor in how viable the extracted distributions for such ambiguity classes might be.

Dependency Parsing Dependency parsing was the task that was found to be the most difficult. One of the possible reasons for this was the nature of attempting to project complete DSs when alignment information was imprecise, and unaligned words were attached using best-guess heuristics. One potential path to take to solve this issue is to follow the work of Spreyer and Kuhn (2009), and focus exclusively on precision rather than completeness.

Using Spreyer and Kuhn's approach, partially projected trees could be used to train a parser, and thus the parser trained on the projected IGT data need only consider correctly projected edges as ground truth, rather than those attached heuristically, which are far more likely to be erroneous.

Cleaning Ultimately, one of the biggest limitations in working with the version of IGT data that was available was data cleanliness. All the enrichment-related tasks depend on clean IGT instances, and while effort was made to automatically clean and normalize the instances as much as possible, a great deal of noise remains. As the enrichment progresses from basic word alignment to shallow POS tagging, to deeper dependency parsing, errors caused by this noise compound, negatively and increasingly impacting performance. Current work with the R*i*PL*es* is focusing on producing new, cleaner data for ODIN. With future, cleaner versions of IGT instances, many of the methods detailed here may see substantial improvements without any further investment.

9.4 Conclusion

This thesis has presented an integral step toward a much greater broadening of language resources involved in the field of natural language processing. While the tasks presented here are limited, the methods used here could easily be adapted. Chapter 6 focuses on the task of POS tagging, but the methods described could be used to project Named Entity Recognition (NER) in the same manner. Similarly, the dependency structures from Chapter 7 represent one form of shallow parsing that could also be used for Semantic Role Labeling (SRL). The word alignment methods from Chapter 5 focus on obtaining alignments within the IGT instances to assist the other tasks in the thesis, but these alignments could be used to extract translation lexicons that would be useful for a wide variety of bilingual tasks, including machine translation.

It is hard to picture a time when computational resources are available for all the lan-

guages of the world. In the intervening time, broadly applicable methods such as those presented here will remain valuable tools in the field of computational linguistics.

Appendix A

TERMS AND VARIABLES

E English-language sentence

F Foreign-language sentence

C Corpus

Each corpus C contains pairs of sentences, and alignments between the words in each sentence.

$C = \{(F_1, E_1, A_1) \ldots, (F_n, E_n, A_n)\}$

Each sentence contains a single pair of sentences, and an alignment.

$c = (F, E, A)$

Each sentence contains a set of words W, a set of edges T, and a set of POS tags P.

$F = \{W_F, T_F, P_F\}$ $\qquad$ $E = \{W_E, T_E, P_F\}$

The set of words is just the tokens for each word in the sentence, while the edges are parent-child pairs (with a special token w_0 for the root node). The POS tags are represented also as a set of pairs with the

word and associated tag, where $p \in$ Tagset.

$W_F = \{f_1, f_2 \ldots f_n\}$ $\qquad$ $W_E = \{e_1, e_2 \ldots e_n\}$

$T_F = \{(f_i, f_j) \ldots (f_k, f_l)\}$ $\qquad$ $T_E = \{(e_i, e_j) \ldots (e_k, e_l)\}$

$P_F = \{(f_i, p_j) \ldots (f_k, p_l)\}$ $\qquad$ $P_E = \{(e_i, p) \ldots (e_k, p)\}$

Each alignment contains pairs of edges between the English and foreign-language words.

$A = \{(f_i, e_j), \ldots, (f_k, e_l)\}$

Finally, I define a relationship R that maps a word from one language onto a set of nodes from the other.

$R_{F \to E}(f_i) = \{e_i \ldots e_n | \forall e, f((e, f) \in A)\}$

$$R_{E \to F}(e_i) = \{f_i \dots f_n | \forall e, f((e, f) \in A)\}$$

Appendix B

PSEUDOCODE

B.1 Projection

Algorithm B.1.1: Algorithm for projecting DSs from one language to another, following Quirk et al. (2005).

```
   input  : Source Sentence W_E = {e_1, e_2, ... e_n}
            Source Tree T_E = {(e_{c_i}, e_{p_k}) ... (e_{c_j}, e_{p_l})}
            Target Sentence W_F = {f_1, f_2 ... f_n}
            Word Alignment A_{E→F} = {(e_i, f_k) ... (e_j, f_l)}
   output : Projected Tree T_F
 1 begin
 2     Let T_F ← Copy(T_E);
 3     foreach (e_c, e_p) ∈ T_F do
           /* If the child is unaligned, delete it and promote its children, if it has any. */
 4         if IsNotAligned(A_{E→F}, e_c) then
 5             Remove(e_c, T_F);                                   // See Algorithm B.1.2
           /* Otherwise, replace the source word with the target word it aligns with. If there are multiple target words that align with
              this source word, each target word will be inserted as a sibling. */
 6         else
 7             foreach f_c ∈ GetAlignedWords(A_{E→F}, e_c) do
 8                 DeleteFromTree(T_F, (e_c, e_p));
 9                 InsertIntoTree(T_F, (f_c, e_p));
       /* After the initial replacement, target words that aligned with multiple source words may appear multiple times in the tree; remove
          all but the shallowest. */
10     foreach f ∈ W_F do
11         f_nodes ← {(f_c, f_p) ∈ T_F | f_c = f};                 // The set of nodes with f in them.
12         f_depths ← {Depth(T_F, f_c) ∀ (f_c, f_p) ∈ f_nodes};    // The set of node depths.
13         foreach {(f_c, f_p) ∈ f_nodes} do
14             if Depth(T_F, f_c) > min(f_depths) then
15                 Remove(T_F, f_c);                               // Remove nodes deeper than the shallowest.
       /* Finally, unaligned target words will be reattached. */
16     foreach {f_i < f_j < f_k | IsAligned(A_{E→F}, f_i) ∧ IsAligned(A_{E→F}, f_k) ∧ IsUnaligned(A_{E→F}, f_j)} do
           /* Attach to leftmost aligned node if no indices to the left of j are aligned. */
17         if |f_i| = 0 then
18             InsertIntoTree(T_F, (f_j, min(f_k)))
           /* Attach to rightmost aligned node if no indices to the right of j are aligned. */
19         else if |f_k| = 0 then
20             InsertIntoTree(T_F, (f_j, max(f_i)));
           /* If f_i depends on f_k or vice versa, attach to the lower of the two. */
21         else if (f_k, f_i) ∈ T_F then
22             InsertIntoTree(T_F, (f_j, f_k));
23         else if (f_i, f_k) ∈ T_F then
24             InsertIntoTree(T_F, (f_j, f_i));
25     return T_F;
```

Algorithm B.1.2: Remove a token w from the tree T.

```
1 Algorithm: Remove(w, T)
  input  : T = {(w_i, w_j) ... (w_m, w_n)} ;                          // Input tree
  input  : w ;                                                        // Word to remove.
  output : T' ;                                                       // Modified tree
2 begin
3     T' = T − {(w, w_i)|w_i = parent(w, T)}                        // Remove edge between w and parent w_i
4            −{(w_j, w)|w = parent(w_j, T)}                         // Remove edges for children of w
5            +{(w_j, w_i)|w_i = parent(w, T), w = parent(w_j, T)} ;
      /* Finish by ''promoting'' former children of w to now attach to w's parent, w_i.  */
6 return T'
```

B.2 Dependency Tree Manipulations

Algorithm B.2.1: Calculating the percentage of matched edges in a corpus C.

input : A corpus C
output : $CorpusMatch_{Src \rightarrow Tgt}(C)$
$CorpusMatch_{Tgt \rightarrow Src}(C)$

begin
  Let $F \rightarrow E$ matches $= 0$;
  Let $E \rightarrow F$ matches $= 0$;
  foreach $(F, E, A) \in C$ **do**
    Let $F = (W_F, T_F)$;
    Let $E = (W_E, T_E)$;
    Let $A = \{(f_i, e_j), \ldots, (f_k, e_l)\}$;
    `// Get matches for` $F \rightarrow E$
    foreach $(f_c, f_p) \in T_F$ **do**
      `/* If the child and parent in this edge align with the child→parent edge of the other tree... */`
      if $\exists\ e_c, e_p : e_p = parent(e_c, T_E)$
        and $(f_p, e_p) \in A$
        and $(f_c, e_c) \in A$
      then
        `// Increase the match count.`
        $(F \rightarrow E$ matches$)$++;
    `// Get matches for` $E \rightarrow F$
    foreach $(e_c, e_p) \in T_E$ **do**
      `/* If the child and parent in this edge align with the child→parent edge of the other tree... */`
      if $\exists\ f_c, f_p : f_p = parent(f_c, T_F)$
        and $(f_p, e_p) \in A$
        and $(f_c, e_c) \in A$
      then
        `// Increase the match count.`
        $(E \rightarrow F$ matches$)$++;
  return $Match(F \rightarrow E) = 100 \times \frac{(F \rightarrow E matches)}{|T_F|}$;
  return $Match(E \rightarrow F) = 100 \times \frac{(E \rightarrow F matches)}{|T_E|}$;

Appendix C

POS TAGSETS

C.1 Chintang

Tag	Universal	Description
n	NOUN	a word that can denote a referent without derivation
pro	PRON	various words with deictic reference (presently not used consistently)
v, vi, vt	VERB	a word that can serve as a predicate without derivation and that has one agreement slot (vi) or two (vt)
adj	ADJ	a word that can modify a head noun without derivation (only occurs in Nepali)
predadj	ADJ	a word that is mainly used as a predicate noun but can neither form the head of an NP nor modify a noun without further derivation
adv	ADV	a word that can modify a predicate without derivation
num	NUM	a noun used for counting that can take numeral classifiers
gm	PRT	any kind of grammatical marker, dependent or independent
v2	VERB	a dependent verb marking a grammatical function
interj	PRT	a word that regularly constitutes an utterance on its own
sound	ADV	an adverb necessarily followed by =mo [CIT]; mostly but not always indicates a sound
NoPOS	X	No POS tag provided

Table C.1: POS Tags used for the Chintang data (Section 4.1.6), (Bickel et al., 2009), and their mapping to the universal POS tags given in Table 4.9.

C.2 Hindi

Original	**Universal**	**Description**
NNP, NNPC	NOUN	Proper Noun (+Compound)
NN, NNC	NOUN	Common Noun (+Compound)
NST	NOUN	Noun denoting spatial relationship
PRP, WQ	PRON	Personal Pronoun (+Question)
DEM	PRON	Demonstrative
PSP	ADP	Postposition
VM	VERB	Main Verb
VAUX	VERB	Auxiliary Verb
RB	ADV	Adverb
RP	PRT	Particle
CC	CONJ	Conjunction (Subordinating & Coordinating)
QF	ADJ	Quantifier (e.g. lots, some)
QC	NUM	Cardinal (two, three)
QO	NUM	Ordinals (first, third)
INTF	ADJ	Intensifier
INJ	PRT	Interjection
RDP	PRT	Reduplication
ECH	PRT	Echo Words
NEG	ADV	Negative
UT	VERB	Quotative
SYM	X	Special Symbol
JJ	ADJ	Adjective
PUNC	.	Punctuation

Table C.2: POS tags used in the Hindi-Urdu-Treebank Project (Section 4.1.5) (Bhatt et al., 2009), and their mapping to the universal POS tags given in Table 4.9.

Appendix D

DEFINITIONS OF TERMS AND INDEX

Definition of Terms

gram A token of gloss-line annotation used to mark a grammatical feature of a token, such as inflection or case. e.g. `'3sg'` for third-person singular, or `NOM` for nominative case, following Bybee and Dahl (1989). 26, 28, 58, 72, 73, 191

projection Using annotation from tokens on language and alignment with those from another to "project" the annotation from the source language onto the other. 25, 71

subword Any element, gram or word, that is contained within a single whitespace-delimited gloss-line token. iii, 58, 73, 84, 88, 91

BIBLIOGRAPHY

Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Jan Hajič, Anders Trærup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Krister Lindén, Nikola Ljubešić, Teresa Lynn, Christopher D Manning, Héctor Alonso Martínez, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Piitulainen, Barbara Plank, Prokopis Prokopidis, Sampo Pyysalo, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 2.0. `https://github.com/ryanmcd/uni-dep-tb`.

Alfred V Aho and Jeffrey D Ullman. 1969. Syntax directed translations and the pushdown assembler. *Computer And System Sciences* 3(1):37–56.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*. pages 547–550.

Mark Baker. 1996. *Phrase Structure and the Lexicon*, Springer Netherlands, Dordrecht, chapter On the Structural Positions of Themes and Goals, pages 7–34.

Dorothee A. Beermann and Lars Hellan. 2002. VP-chaining in Oriya. In *Proceedings of the 2002 Lexical Functional Grammar Conference*. Stanford Linguistic Association and CSLI, Stanford, CA, USA.

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

Emily M. Bender, Joshua Crowgey, Michael Wayne Goodman, and Fei Xia. 2014. Learning grammar specifications from IGT: A case study of Chintang. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*. Baltimore, MD, USA, pages 43–53.

Emily M. Bender, Michael Wayne Goodman, Joshua Crowgey, and Fei Xia. 2013. Towards creating precision grammars from interlinear glossed text: Inferring large-scale typological properties. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*. Sofia, Bulgaria, pages 74–83.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics* 22(1):39–71.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for Hindi/Urdu. In *ACL-IJCNLP '09: Proceedings of the Third Linguistic Annotation Workshop*. Singapore, pages 186–189.

Balthasar Bickel, Goma Banjade, Toya N. Bhatta, Martin Gaenzle, Netra P. Paudyal, Manoj Rai, Novel Kishore Rai, Ichchha Purna Rai, and Sabine Stoll. 2009. *Audiovisual corpus of the Chintang language, including a longitudinal corpus of language acquisition by six children, plus a trilingual dictionary, paradigm sets, grammar sketches, ethnographic descriptions, and photographs*. DoBeS, Universität Leipzig, Nijmegen, Leipzig. `http://www.mpi.nl/DOBES`.

Balthasar Bickel, Bernard Comrie, and Martin Haspelmath. 2008. The Leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses. Max Planck Institute for

Evolutionary Anthropology and Department of Linguistics, University of Leipzig. `https://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf`.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Thorsten Brants. 2000. TnT: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*. Seattle, Washington, pages 224–231.

Thorsten Brants, Wojciech Skut, and Hans Uszkoreit. 2003. Syntactic annotation of a German newspaper corpus. In *Treebanks*, Springer Netherlands, Dordrecht, pages 73–87.

M. M. Bravmann. 1977. *Studies in Semitic philology*. Brill.

Peter F Brown, Peter V deSouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics* 18(4):467–479.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19(2):263–311.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*. New York, NY, USA, pages 149–164.

Joan L Bybee and Östen Dahl. 1989. The creation of tense and aspect systems in the languages of the world. *Studies in Language* 13(1):51–103.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's mechanical turk. In *Proceedings of the NAACL HTL Workshop on Creating*

Speech and Language Data with Amazon's Mechanical Turk. Los Angeles, CA, USA, pages 1–12.

Nicoletta Calzolari, Riccardo Del Gratta, Gil Francopoulo, Joseph Mariani, Francesco Rubino, Irene Russo, and Claudia Soria. 2012. The LRE map: Harmonising community descriptions of resources. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics* 33(2):201–228.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 Workshop on Computational Natural Language Learning*. Toulouse, France.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Budapest, Hungary.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA, pages 1–8.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research* 3:951–991.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)* 4(1):3:1–3:34.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*. Portland, OR, USA, pages 600–609.

Hal Daumé III. 2009. Unsupervised search-based structured prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*. Montreal, Canada, pages 209–216.

Marie-Catherine de Marneffe and Bill MacCartney. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the fifth International Conference on Language Resources and Evaluation*. Genoa, Italy.

Marie-Catherine de Marneffe and Christopher D Manning. 2008. The Stanford typed dependencies representation. In *Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*. Association for Computational Linguistics.

Peter de Swart. 2003. Dative pronouns, binding, and empty prepositions. In *Mini Conference on Binding*. Nijmegen.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* 39:1–38.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of Recent Advances in Natural Language Processing*. Hissar, Bulgaria, pages 198–206.

Bonnie Jean Dorr. 1994. Machine translation divergences: a formal description and proposed solution. *Computational Linguistics* 20:597–633.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, volume 1 of *COLING '96*, pages 340–345.

Anna Feldman, Jirka Hana, and Chris Brew. 2006. Experiments in cross-language morphological annotation transfer. In *Computational Linguistics and Intelligent Text Processing: 7th International Conference*. Springer Berlin Heidelberg, pages 41–50.

Foo Labs. 2014. Xpdf: A PDF viewer for X. `http://www.foolabs.com/xpdf/`.

Alexander Fraser and Daniel Marcu. 2006. Semi-supervised training for statistical word alignment. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 769–776.

Alexander Fraser and Daniel Marcu. 2007. Getting the structure right for word alignment: LEAF. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic, pages 51–60.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia.

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

Qin Gao. 2013. mgiza. `https://github.com/moses-smt/mgiza`.

Qin Gao, Nguyen Bach, and Stephan Vogel. 2010. A semi-supervised word alignment algorithm with partial manual alignments. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics (MATR)*. Uppsala, Sweden, pages 1–10.

Ryan Georgi. 2009. *Grammar induction with prototypes derived from interlinear text*. Master's thesis, University of Washington, Seattle, WA.

Ryan Georgi, William D Lewis, and Fei Xia. 2014. Capturing divergence in dependency trees to improve syntactic projection. *Language Resources and Evaluation* 48(4):709–739.

Ryan Georgi, Fei Xia, and William D Lewis. 2012. Improving dependency parsing with interlinear glossed text and syntactic projection. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*. Mumbai, India.

Ryan Georgi, Fei Xia, and William D. Lewis. 2013. Enhanced and portable dependency projection algorithms using interlinear glossed text. In *The 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria.

Ryan Georgi, Fei Xia, and William D Lewis. 2015. Enriching interlinear text using automatically constructed annotators. In *Proceedings of the 9th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH-2015), in conjunction with ACL 2015*. Beijing, China.

Michael Wayne Goodman, Joshua Crowgey, Fei Xia, and Emily M Bender. 2014. Xigt: extensible interlinear glossed text for natural language processing. *Language Resources and Evaluation* 49(2):455–485.

João V Graça, Kuzman Ganchev, Luísa Coheur, Fernando Pereira, and Ben Taskar. 2011. Controlling complexity in part-of-speech induction. *Journal of Artificial Intelligence Research* 41.

Joseph H Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. *Universals of Language* pages 73–113.

Aria Haghighi and Dan Klein. 2006a. Prototype-driven grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 881–888.

Aria Haghighi and Dan Klein. 2006b. Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, New York, NY, USA, pages 320–327.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel

Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*. Boulder, CO, USA.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2016. *Glottolog 2.7*, Max Planck Institute for the Science of Human History, chapter Yukaghir. Available online at `http://glottolog.org`, Accessed on 2016-08-16.

Jiri Hana, Anna Feldman, and Chris Brew. 2004. A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Rodopi Bv Editions, pages 222–229.

David G Hays. 1964. Dependency theory: A formalism and some observations. *Language* 40(4):511.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2004. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering* 1(1):1–15.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering* 11(03):311–325.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, PA, USA.

Ray Jackendoff. 1983. *Semantics and Cognition*, volume 8 of *Current Studies in Linguistics Series*. MIT Press, Cambridge, MA, USA.

Douglas Jones and Rick Havrilla. 1998. Twisted pair grammar: Support for rapid development of machine translation for low density languages. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup*. Springer-Verlag.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, PA, USA, pages 128–135.

Dan Klein and Christopher D Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15*. Cambridge, MA, USA, pages 3–10.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*. Association for Computational Linguistics, Morristown, NJ, USA. Article No. 478.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*. Phuket, Thailand.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*. Columbus, OH, USA.

Tibor Laczkó. 2002. Control and complex event nominals in Hungarian. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG02 Conference*. CSLI Publications.

M. Paul Lewis. 2009. *Ethnologue: Languages of the World*. SIL International, 16th edition.

Philip M. II Lewis and Richard E Stearns. 1968. Syntax-directed transduction. *Journal of the ACM (JACM)* 15(3):465–488.

William D Lewis. 2006. ODIN: a model for adapting and enriching legacy infrastructure. In *2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*. IEEE, Amsterdam, Netherlands, pages 137–137.

William D Lewis and Fei Xia. 2008. Automatically identifying computationally relevant typological features. In *Proceedings of the Third International Joint Conference on Natural Language Processing*. Hyderabad, India.

William D Lewis and Fei Xia. 2009. Parsing, projecting & prototypes: repurposing linguistic data on the web. In *12th Conference of the European Chapter of the ACL*. Association for Computational Linguistics, Athens, Greece, pages 41–44.

William D Lewis and Fei Xia. 2010. Developing ODIN: A multilingual repository of annotated language data for hundreds of the world's languages. *Literary and Linguistic Computing* 25(3):303–319.

Marco Lui, Jey Han Lau, and Timothy Baldwin. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics* 2:27–40.

Gideon S Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. Columbus, OH, USA, pages 955–984.

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

Andrew Kachites McCallum. 2002. MALLET: a machine learning for language toolkit. `http://mallet.cs.umass.edu`.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* pages 523–530.

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

Ilya Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. Morristown, NJ, USA, pages 79–86.

Seyed Abolghasem Mirroshandel, Alexis Nasr, and Joseph Le Roux. 2012. Semi-supervised dependency parsing using lexical affinities. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. pages 777–785.

Wataru Nakamura. 1997. *A Constraint-Based Typology of Case Systems*. Ph.D. thesis, State University of New York at Buffalo.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal

linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. pages 1234–1244.

Norio Nasu. 2001. Towards a theory of non-cyclic A-movement. In *Essex Graduate Student Papers in Language and Linguistics*. volume 3, pages 133–160.

Igor Nedjalkov. 1998. Converbs in the languages of eastern Siberia. *Language Sciences* 20(3):339–351.

Ad Neeleman and Kriszta Szendrői. 2007. Radical pro drop and the morphology of pronouns. *Linguistic Inquiry* 38(4):671–714.

Peter Nilsson. 2015. *A Snowball Sampling Approach for Studying Digital Minority Languages*. Ph.D. thesis, Swarthmore College.

Rachel Nordlinger and Louisa Sadler. 2000. Tense as a nominal category. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the 2000 Lexical Functional Grammar Conference*. CSLI Publications, Stanford, CA, USA.

Franz Josef Och and Hermann Ney. 2003a. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

Franz Josef Och and Hermann Ney. 2003b. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

PDFLib GmbH. 2015. PDFLib TET 5 – text and image extraction toolkit. `https://www.pdflib.com/products/tet/`.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

Martin F Porter. 2001. Snowball: A language for stemming algorithms. http://snowballstem.org/.

Princeton University. 2010. *About WordNet*. Princeton University. http://wordnet.princeton.edu.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, pages 271–279.

Lawrence R Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*. volume 77, pages 257–286.

Christian Rishøj and Anders Søgaard. 2011. Factored translation with unsupervised word clusters. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. pages 447–451.

Edward Sapir. 1921. *Language: An introduction to the study of speech*. Harcourt, Brace and company, New York.

Kevin Scannell. 2014. Indigenous tweets. http://indigenoustweets.com/.

Kevin P. Scannell. 2007. The Crúbadán project: Corpus building for under-resourced languages. In *Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop*. Louvainla-Neuve, Belgium, pages 5–15. http://crubadan.org/.

Kevin P. Scannell. 2016. An gramadóir. http://borel.slu.edu/gramadoir/index.html.

SIL International. 2006. ISO 639-3. http://www-01.sil.org/iso639-3/.

Noah A Smith and Jason Eisner. 2005a. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for*

Computational Linguistics. Association for Computational Linguistics, Ann Arbor, MI, USA, pages 354–362.

Noah A Smith and Jason Eisner. 2005b. Guiding unsupervised grammar induction using contrastive estimation. In *Proceedings of IJCAI Workshop on Grammatical Inference Applications*. pages 73–82.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*. Columbus, OH, USA.

Kathrin Spreyer and Jonas Kuhn. 2009. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*. Boulder, CO, USA, pages 12–20.

Anthony Stone. 2004. Transliteration of Indic scripts: How to use ISO 15919.

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

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris.

Jörg Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics*. Budapest, Hungary, pages 339–346.

Kristina Toutanova, Haluk Tolga Ilhan, and Christopher D Manning. 2002. Extensions to HMM-based statistical word alignment models. In *Proceedings of the ACL-02 Conference*

on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Morristown, NJ, USA, pages 87–94.

Kristina Toutanova and Mark Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, volume 20, pages 1521–1528.

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

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics*. Copenhagen, Denmark.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics* 23(3):377–403.

Fei Xia and William D Lewis. 2007. Multilingual structural projection across interlinear text. In *Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Rochester, NY, USA, pages 452–459.

Fei Xia, William D Lewis, Michael Wayne Goodman, Glenn Slayden, Ryan Georgi, Joshua Crowgey, and Emily M Bender. 2016. Enriching a massively multilingual database of interlinear glossed text. *Language Resources and Evaluation* pages 1–29.

Min Xiao and Yuhong Guo. 2015. Annotation projection-based representation learning for

cross-lingual dependency parsing. In *Proceedings of the 2015 Conference on Computational Natural Language Learning*. Beijing, China.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second meeting of the North American Association for Computational Linguistics*. Johns Hopkins University, Stroudsburg, PA.

Daniel Zeman. 2008. Reusable tagset conversion using tagset drivers. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*. Marrakech, Morocco.

Jan-Wouter Zwart. 2002. The antisymmetry of Turkish. In *Generative Grammar in Geneva*. volume 3, pages 23–26.