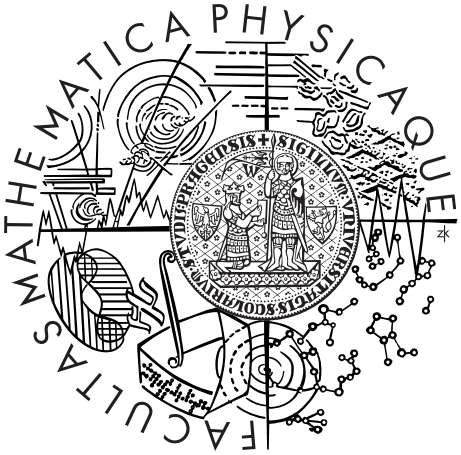
Charles University in Prague
Faculty of Mathematics and Physics

DOCTORAL THESIS



Loganathan Ramasamy

Parsing under-resourced languages: Cross-lingual transfer strategies for Indian languages

Institute of Formal and Applied Linguistics

Prague 2014

Supervisor **assoc. prof. (doc.) Ing. Zdeněk Žabokrtský, Ph.D.**
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25
118 00 Prague 1
Czech Republic

Opponents **RNDr. Daniel Zeman, Ph.D.**
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25
118 00 Prague 1
Czech Republic

RNDr. Otakar Smrž, Ph.D.
Seznam.cz
Radlická 3294/10
150 00 Praha 5
Czech Republic

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Acknowledgements

I would like to express my sincere gratitude to my advisor Zdeněk Žabokrtský for keeping me focused on my dissertation topic all along with great care. It is his constant nudge on the importance of writing that helped me to shape up my thesis, without which I would be nowhere near to finishing my dissertation.

I would like to thank other ÚFAL members who have directly or indirectly contributed to my research in various ways. I would like to thank Martin Popel for all his help in Treex and for general discussion on various topics. I would like to thank David Mareček and Rudolf Rosa for reading my dissertation and providing many valuable inputs. Again, thanks to Rudolf for making his translation data available for my experiments. I would also like to thank Daniel Zeman for making available many tools and data, without which a part of my thesis would not have been possible.

I am very thankful to prof. Markéta Lopatková for all her help right since my arrival to ÚFAL.

I would like to thank the funding agencies who through CLARA (a Marie Curie ITN under FP7) and LINDAT/CLARIN supported my research all these years.

Finally, I would also like to thank my family members and friends for their gracious support in everything I do.

Abstract

Key to fast adaptation of language technologies for any language hinges on the availability of fundamental tools and resources such as monolingual/parallel corpora, annotated corpora, part-of-speech (POS) taggers, parsers and so on. The languages which lack those fundamental resources are often referred as under-resourced languages.

In this thesis, we address the problem of cross-lingual dependency parsing of under-resourced languages. We apply three methodologies to induce dependency structures: (i) projecting dependencies from a resource-rich language to under-resourced languages via parallel corpus word alignment links (ii) parsing under-resourced languages using parsers whose models are trained on treebanks of other languages, and do not look at actual word forms, but only on POS categories. Here we address the problem of incompatibilities in annotation styles between source side parsers and target side evaluation treebanks by harmonizing annotations to a common standard; and finally (iii) we add a new under-resourced scenario in which we use machine translated parallel corpora instead of human translated corpora for projecting dependencies to under-resourced languages.

We apply the aforementioned methodologies to five Indian languages (ILs): Hindi, Urdu, Telugu, Bengali and Tamil (in the order of high to low availability of treebank data). To make the evaluation possible for Tamil, we develop a dependency treebank resource for Tamil from scratch and we use the created data in evaluation and as a source in parsing other ILs. Finally, we list out strategies that can be used to obtain dependency structures for target languages under different resource-poor scenarios.

Contents

List of Tables

List of Figures

Abbreviations

DS Dependency Structure. 2

HMM Hidden Markov Model. 96

IL Indian Language. 5, 6, 73, 77, 83, 97, 141

MT Machine Translation. 3, 7, 83, 141

NER Named Entity Recognition. 1

NLP Natural Language Processing. 1, 6, 9

PDT Prague Dependency Treebank. 2

POS Parts of Speech. 1

SRL Semantic Role Labeling. 1

UAS Unlabeled Attachment Score. 97

Chapter 1

Introduction

This chapter provides an overview of the thesis.

- the chapter starts with the examination of current scenario in treebank development and treebank availability in general in Section 1.1
- provides an overview of current approaches to languages that do not have treebanks and limitations of those approaches in Section 1.3 and 1.4 respectively
- defines thesis goals and contributions in Section 1.5
- provides a logical overview to what follows after this chapter in Section 1.6

1.1 Treebanks Availability: A Brief Overview

Natural language data with linguistic annotations is indispensable in natural language processing (NLP) research. NLP tasks such as part-of-speech (POS) tagging, syntactic parsing, semantic role labeling (SRL), named entity recognition (NER) to name a few – heavily make use of annotated data to obtain state of the art results. The applications of linguistically annotated data are two-fold: (i) linguistic annotation helps to study various linguistic phenomena and allows to make generalizations in a theoretical perspective (ii) linguistic annotation helps machine learning algorithms to learn mathematical models of linguistic phenomena of interest. All types of linguistic description/annotation on the natural language data require linguistic expertise and domain expertise. For a few languages, we will be able to find linguistic/domain experts easily and create annotated data. For most of the languages it's not an easy task. Depending on the linguistic annotation (whether shallower or deeper), the annotation task itself becomes very complex, time consuming and more costly with the increasing of richness in the annotation.

The focus of this thesis is treebank annotation. Treebanks are usually corpora of sentences annotated with syntactic structures and they can vary according to the chosen formalism. Two such formalisms for which treebanks are available and are

used in various shared tasks are: (i) treebanks that follow phrase structure grammars and (ii) treebanks that follow dependency grammars. Treebanks are developed for languages with many years of effort in manual or semi-automatic annotation. Two such popular large scale treebanks available today are: (i) the Penn Treebank (PTB) [Taylor et al., 2003] and (ii) the Prague Dependency Treebank (PDT) [Hajič et al., 2006]. The PTB (1989-1999, from the beginning to the release of PTB 3) has 3 million tokens annotated with syntactic structures and 7 million tokens annotated with POS tags. The PDT (1999-2013, from the beginning to the release of PDT 3.0) has primarily 3 layers of annotation: morphological (m-layer), analytical (a-layer) and tectogrammatical (t-layer). m-layer is an equivalent of POS annotation; a-layer is an equivalent of surface dependency annotation (syntactic structure + dependency relations); t-layer corresponds to semantic layer. The size of the PDT 2.0 is: 2 million tokens tagged with morphological tags in m-layer, 1.5 million tokens with syntactic structures in a-layer and 0.8 million tokens with semantic structures t-layer. Annotation of PDT extends to (i) multiword expressions, clause segmentation, etc in PDT 2.5 (← PDT 2.0), (ii) bridging anaphora, discourse relations, etc in PDTiT 1.0 (← PDT 2.5), and (iii) genres of documents, updated discourse relations and other minor extensions in the most recent version of PDT 3.0.

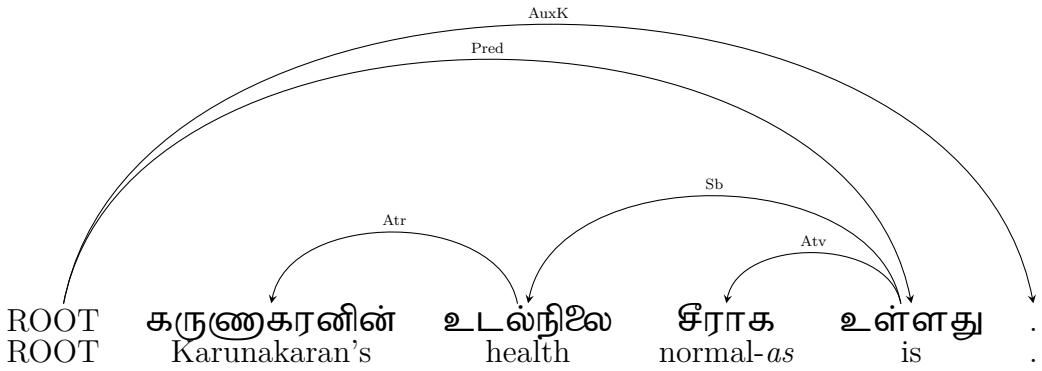
[Zeman et al., 2012] identifies the existence of about 30 treebanks with size varying from as few as 10000 tokens to large scale treebanks.

1.2 Dependency Treebanks

In this thesis, we explore treebanks that use dependency structure for their syntactic representation. The dependency structure (DS) of a sentence consists of *lexical elements* [Nivre, 2005] which correspond to word forms in the surface structure and *dependencies* which establish a relationship between a pair of lexical elements. The lexical elements pair involved in this relationship are called *head/governor* and *modifier/dependent*. The dependency relationship is not strictly restricted to syntactic relationships, the relationship may sometimes can be interpreted as morphological or semantic. We chose DS annotation for our experiments, because:

- treebanks are increasingly made available with DS annotation
- simple syntactic representation than phrase structure trees (compact in terms of number of nodes in the syntactic tree)
- predicate-argument nature of dependencies make DSs somewhat structure neutral across languages; and this makes dependency representation all the more suitable for multilingual applications especially multilingual parsing

The following figure shows the dependency representation of a Tamil sentence.



Directed edges go from *governor* to *dependent*. The entire dependency structure is governed by a special node called ROOT which is not part of the sentence. Dependency trees are directed acyclic graphs. But edges can cross other edges in a dependency tree (in that case, the dependency tree is said to be containing *non-projective* edges).

1.3 Current Approaches to Languages Without Treebanks

Many languages including some of the popular languages (according to the number of native language speakers, see Table 1.1) do not have treebanks. The reasons for the absence of treebanks for languages that have sizeable population are mainly extraneous (such as lack of NLP research in those languages, research funding, etc.). The availability of treebanks for these languages will have tremendous practical value given the fact that treebanks are useful in many practical applications such as machine translation (MT), relation extraction, and so on. Other category of languages that lack treebanks include languages that are not in use or languages with only few speakers. [Krauwer, 2003] argued that commercially important languages (such as English, French and German) would continue to dominate the language technology field and had introduced a notion called *Basic Language Resource Kit (BLARK)* to identify the kind of resources a language must adequately have in order to develop practical language applications for languages that do not have enough resources.

In the case of treebanking, research in recent years focuses on developing methodologies that could reduce years of manual annotation effort by automatically inducing treebanks for resource-poor languages with varying levels of success. Those methodologies can be broadly categorized into: (i) unsupervised methods which can parse any language data without any annotated treebank for training (ii) semi-

#	Language	# Speakers	Size	Source
1	Chinese	1197M	large	[Xue et al., 2010]
2	Spanish	406M	large	[Marimon et al., 2012]
3	English	335M	large	[Taylor et al., 2003]
4	Hindi	260M	large	[Bhatt et al., 2009]
5	Arabic	223M	large	[Maamouri et al., 2012]
6	Portugese	202M	medium	[Afonso et al., 2002]
7	Bengali	193M	small	[Husain et al., 2010]
8	Russian	162M	large	[Boguslavsky et al., 2000]
9	Japanese	122M	medium	[Kawata and Bartels, 2000]
10	Javanese	84.3M	-	-
11	German	83.8M	large	[Brants et al., 2002]
12	Lahnda	82.7M	-	-
13	Telugu	74.0M	small	[Husain et al., 2010]
14	Marathi	71.8M	-	-
15	Tamil	68.8M	small	[Ramasamy and Žabokrtský, 2012]
16	French	68.5M	large	[Abeillé et al., 2003]
17	Vietnamese	67.8M	medium	[Nguyen et al., 2009]
18	Korean	66.4M	medium	[Han et al., 2006]
19	Urdu	63.4M	medium	[Bhat and Sharma, 2012]
20	Italian	61.1M	medium	[Montemagni et al., 2003]
21	Malay	59.4M	-	-
22	Persian	56.6M	medium	[Rasooli et al., 2011]
23	Turkish	50.7M	medium	[Atalay et al., 2003]
24	Oriya	50.1M	-	-

Table 1.1: Treebanks (of all formalisms) availability for languages by number of speakers. $>$ 250000 tokens = *large*; 50000-250000 tokens =*medium*; $<$ 50000 tokens = *small*. Source: http://www.ethnologue.com/statistics/size as on June 7, 2013.

supervised methods which make use of small amount of treebank annotation to bootstrap the annotation further and (iii) cross-lingual syntactic annotation transfer through projection (or simply syntactic projection) of annotation from resource-rich source languages to resource-poor target languages. Depending on the resource availability (for the languages in question), all three frameworks mentioned above can be combined in a hybrid fashion to obtain better parsing results albeit supervised parsing is most likely to give superior performance.

In our research work, we mainly focus on cross-lingual transfer-based techniques. This approach has been initially studied by [Hwa et al., 2005], which induced treebank/parser for Spanish and Chinese by projecting syntax from English via alignment links in the English-Spanish and English-Chinese parallel corpus. Most of the earlier transfer-based approaches heavily rely on bitext[1] or some other target language resources (such as POS taggers) which may not be available for many resource-poor languages. Recent works on cross-lingual transfer-based techniques [McDonald et al., 2011], [Täckström et al., 2012] and [Durrett et al., 2012] decouple this target language resource requirement by directly transferring the syntax from source language parsers via *delexicalization*. *Delexicalized parsing* is a method of using source language parser directly to parse target language sentences. This method requires only POS sequences of source side training data to train delexicalized parsers. It has been shown in [McDonald et al., 2011] that training a parser with POS tags alone (leaving wordforms from training data) achieves parser accuracy comparable to that of supervised parsers. Extending that to parser transfer, delexicalized transfer parsers [McDonald et al., 2011] can give surprisingly better performance than state-of-the-art unsupervised parsers given that source and target languages use the same POS tagset.

1.4 Limitations of the Current Approaches

In the thesis, we mainly focus on transfer-based approaches.

1. *Current approaches to transfer-based dependency structure prediction do not cover transfer of dependency structure for Indian language (IL) treebanks.* The simple reason was the unavailability of IL treebanks even for evaluation. But the situation is improving with Hindi treebank being the largest treebank now among other Indian language treebanks (Tamil, Bengali, Telugu and Urdu).

2. Parsers transferred from other language treebanks that do not have the same annotation style as the target language treebank suffer due to annotation differences. The annotation differences alone can systematically underestimate the predicted (transferred) structure accuracy. This is still an open problem

[1]Bitext is a sentence aligned parallel corpus (the number of sentences in language A and its translation equivalents in language B should be equal)

which has not been adequately addressed so far in the transfer-based approaches. However, there are interesting works emerging in this area, such as [Zeman et al., 2012, McDonald et al., 2013] which strive for common annotation of treebanks, and [McDonald et al., 2011] which try to use common POS tagset to transfer parsers to other languages.

3. Previous works have shown the usability of transfer-based approaches to parsing. However, unlike other shallow NLP tasks, such as POS annotation where an attempt has been made recently to get quick POS annotated corpora in a time-bound manner (two hours of annotation) [Garrette and Baldridge, 2013] for 2 new languages Kinyarwanda and Malagasy—or acquiring parallel corpora for new set of languages through crowdsourcing [Post et al., 2012], no such attempts have been made for the more complex task of dependency annotation or treebank annotation in general (except the more recent work by [McDonald et al., 2013]). Also, most of the earlier works simulate the resource-poor scenario mostly on reasonably well developed treebanks.

1.5 Thesis Goals and Contributions

The main of goal of the thesis is to identify best cross-lingual transfer strategies for Indian languages (ILs). Toward that goal, we also try to deviate from the existing cross-lingual transfer-based approaches in two respects: (i) the thesis culminates in a new treebank resource plus parallel corpus for Tamil language and (ii) explores new under-resourced scenarios such as using machine translated parallel texts instead of human translated texts for Indian languages. The thesis makes three novel contributions:

1. **Creation of dependency treebank and parallel corpus for a new under-resourced language:** In this thesis, we share our experience in creating a dependency treebank from scratch for a new language. The language we chose was Tamil. The intended goal is to create a high quality treebank using the existing annotation standards. We also create a new parallel corpus resource for Tamil.

2. **Cross-lingual dependency transfer results for Indian languages (ILs):** We apply transfer-based approaches to major Indian languages. Most of the existing NLP resources for Indian languages are morphological analyzers and POS taggers and to some extent parallel corpora between English and Indian languages. Syntactic resources such as treebanks are quite rare and are still at the preliminary level of development even for major Indian languages. In the thesis, we show experimental results for languages that have at least evaluation treebank data.

3. **Cross-lingual dependency transfer using machine translated parallel texts:** One of the most crucial requirement for projection based dependency transfer is the availability of parallel corpora or bitexts. In the thesis, we explore a new under-resourced scenario, where we have access to machine translation (MT) systems but not to parallel corpora.

1.6 Thesis Organization

The thesis is structured as follows:

- Chapter 2 presents the related work in the field. The chapter gives the survey of existing work in cross-lingual dependency induction, semi-supervised and unsupervised parsing.
- Chapter 3 presents the current scenario of data availability for Indian languages in the context of cross-lingual dependency transfer. The chapter surveys the availability parallel corpus, POS taggers and treebanks for ILs.
- Chapter 4 describes our effort in the development of Tamil dependency treebank (TamilTB). The chapter explains various design aspects of the TamilTB. The manually annotated treebank for one of the Indian languages is used as pivotal treebank in various experiments.
- Chapter 5 describes the data and evaluation measures used in our experiments.
- Chapter 6 presents theory and experimental results for cross-lingual dependency transfer methods. Results are presented for 5 Indian languages: Bengali, Hindi, Tamil, Telugu and Urdu.
- Chapter 7 provides error analysis for projection based approach. Some common error patterns are identified from the parsed data.
- Chapter 8 concludes the thesis.

Chapter 2

Related Work

In this chapter, we briefly survey the existing body of work related to this thesis. Sections 2.2 and 2.3 present earlier transfer-based approaches. Sections 2.4 and 2.5 describe related works from other methodologies, namely semi-supervised and unsupervised techniques. Section 2.6 present efforts in standardizing POS and dependency annotations.

2.1 Introduction

Treebanks and statistical parsers for resource-poor languages are still a highly focused area of research in natural language processing (NLP). Over the years various approaches have been evolved which may or may not require target[1] language tools and resources. Those approaches can be broadly classified into three categories,

1. **Transfer-based methods:** Usually require bilingual or multilingual resources. The grammar or parser is induced from resource-rich languages to resource-poor languages. Earlier approaches (such as [Hwa et al., 2005]) require certain amount of target language resources such parallel corpus (resource-rich - resource-poor combination). Recent works such as [McDonald et al., 2011] do not require any large parallel corpus, but rely mainly on resource-rich monolingual tools and resources.

2. **Semi-supervised methods:** Usually bootstrap the treebank [Steedman et al., 2003] from small amount of annotated data using self-training or other semi-supervision strategies. Semi-supervised methods also try to incorporate various target language features/resources to boost the accuracy of the induced statistical parsers. The seed annotated treebank is essential for training semi-supervised models.

3. **Unsupervised methods:** Do not assume any existence of target language resources for parsing the text. However, the accuracy of these methods such

[1]Target in the context of dependency projection or cross-lingual dependency transfer means the language for which we want to obtain dependency trees from some other *source* language

as [Klein and Manning, 2004] are still largely low compared to the above methods. Yet these methods are very attractive considering least target language prerequisites.

In the following sections, we briefly examine related works (based on their method) of dependency grammar induction for resource-poor languages.

2.2 Transfer Based on Bitexts

The availability of bilingual resources has found usability in variety of applications besides MT being the most popular example. One other important application is transferring annotation from one language to another language via bilingual resources. The annotation can be anything from morphological, POS, syntactic to semantic structures. Bilingual resources thus act as a bridge for such annotation transfer. This transfer strategy is also known as *projection*. Consider for example, we have a list of words from Badaga language (only words from dictionary) for which we want to guess all possible POS tags. Let us assume that we do not have monolingual tools such as a morphological analyzer or a POS tagger for Badaga, but we do have an English-Badaga dictionary (a bilingual resource) for which POS tags of the English side entries are known. Applying projection to this problem consists of simply copying the POS tags of English dictionary entries to the corresponding Badaga entries. In this way, we can reasonably guess the POS tags of Badaga word list even if they are not 100% accurate. Similarly, we can also guess other linguistic annotations such as Named Entities (NE), dependency structures and semantic roles and so on. However, the quality of annotation obtained through projection depends on the complexity of the annotation, the quality of bilingual resources as well as the properties of languages themselves.

Projection based methods are being successfully exploited by researchers as a reasonable alternative to induce annotations for resource-poor languages. Annotations are usually projected from resource-rich languages to resource-poor languages to reap the maximum benefit. [Yarowsky et al., 2001] is the first to demonstrate the use of annotation projection to induce shallow parsing tools such as morphological analyzers, POS taggers, noun phrase bracketers and named entity taggers using parallel corpora and resource-rich monolingual tools. For POS tagging, [Das and Petrov, 2011a] used cross-lingual transfer to obtain more than 10% higher absolute percentage points over the state-of-the art unsupervised method [Berg-Kirkpatrick et al., 2010]. Similarly, cross-lingual transfer has been applied to other applications such as semantic role transfer [Bentivogli and Pianta, 2005, Mukund et al., 2010, Padó and Lapata, 2009].

For syntactic parsing, the same idea has been successfully exploited by [Hwa et al., 2005] for bootstrapping parsers for Spanish and Chinese. The outcome of the above work was dependency treebanks for Spanish and Chinese projected from En-

glish dependency structures. They used *Direct Projection Algorithm (DPA)* which took into account various alignment possibilities between source (English) and target languages (Spanish and Chinese). They implemented minimal linguistically aware rules that they applied on the projected trees to improve the performance. They also showed that the accuracy of the projected treebank when tested against unseen sentences is similar to that of the commercial rule based parser. Other notable thing from their work was, they showed similar performance for Chinese and Spanish parsers trained on the projected treebank. This also implies that despite the differences in the languages (English and Chinese), their results impart optimism for considering more diverse set of languages.

[Ganchev et al., 2009] incorporated projection constraints (such as how much percentage of *conserved* edges should be preserved in the transfer) to guide the training of both generative and discriminative models. In addition to that, they have also used small number of rules to tackle annotation style issues. The projection experiments were carried out for English to Spanish and English to Bulgarian. [Ganchev et al., 2009] showed that their approach outperforms standard unsupervised methods and supervised methods on limited training data.

[Smith and Eisner, 2009] tried to address the task in two modes: (i) parser adaptation for a new annotation style given a parser trained with another annotation style (ii) parser for a new language through projection. For both the tasks, they used *quasi synchronous grammar* (QG) features for learning the target language trees.

[Spreyer, 2010] tried to address some of the issues (mainly annotation style difference between the source and the target) in projection based annotation transfer via parallel corpora. They showed that it is useful to transform the projected annotations to source annotation style rather than to target style annotation. They have also shown that annotation scheme itself has an impact on parsing algorithms i.e., for example, an observation that MST parsers are good at parsing flat structures.

Another related work was by [Mareček, 2011], which essentially improved the dependency tree projection by combining various alignment symmetrization. The idea is that, by combining various GIZA++ alignment symmetrization (such as *intersection, grow-diag-final-and*), the projection algorithm makes sure that all nodes on the target side language are linked. The work also showed almost similar performance for four languages (Bulgarian, Czech, German and Dutch) projected from English trees.

The work by [Mannem and Dara, 2011] involved syntactic projection onto Hindi and two other languages (Bulgarian and Spanish) from English. They developed a parsing algorithm which was able to learn from partial parses, they also improved upon baseline which was trained using fully connected trees. No other attempt was made for any other Indian languages except Hindi. Also, the parallel corpora is still an issue even for major Indian languages as they possess very little or no data. Since syntactic projection depends entirely on the parallel data, the task will be really challenging.

Though projection based methods can be competitive in terms of accuracy, they do make an assumption of plenty of parallel corpora and give better results mainly under ideal scenarios. Other challenges in syntactic projection are alignment errors as well as different annotation styles [Mareček, 2011]. The alignment errors in the parallel corpora could lead low quality projections whereas different annotation styles of source and target treebanks could add to the complexity to evaluating the projection results. Some kind of *transformation rules* or making the treebanks adhere to single annotation can solve the annotation style issue.

2.3 Delexicalized Transfer

[Zeman and Resnik, 2008] showed that it is possible to transfer the dependency annotation from a resource-rich language to another closely related resource-poor language without the need of a parallel corpora word alignment. The dependency annotation was transferred through the process called *delexicalization*. In delexicalized transfer, parsers are transferred from source to target languages using POS taggers alone. Source language parser is trained with only POS tag sequences (without including source wordforms). Target language sentences are tagged by a POS tagger that use the same POS tagset as the source language. Then, target language sentences are parsed by the source language parser trained with only POS tag sequences. Target wordforms should be removed before parsing the target sentences. Only prerequisite for this kind of transfer is the availability of source and target POS tagger with a common POS tagset. This approach [Zeman and Resnik, 2008] has been shown to work remarkably well for languages that are typologically similar or languages within the same family. In their experiments, they mapped both source and target treebanks to common annotation standards to avoid annotation differences. The source and target language in their experiments were Danish and Swedish respectively.

[McDonald et al., 2011] conducted delexicalized direct transfer (similar to [Zeman and Resnik, 2008]) and projected transfer experiments across 8 Indo-European languages and showed significant performance improvements over unsupervised approaches. In the projected transfer, they used target side delexicalized parser as a starting point. Then they used constraint driven learning algorithm to learn better target parsing model by using parallel corpora world alignment as constraints. During training, the target dependencies that are aligned to source dependencies are favored. In this way, they guided the supervision with alignment as additional information. All their experiments were conducted for gold POS and more realistic (no gold POS and no restrictions on sentence length) scenarios. [McDonald et al., 2011] did not use any target language POS tagger, rather they induced POS tags using POS projection [Das and Petrov, 2011a].

[Søgaard, 2011] performed cross language parser adaptation similar to [Zeman and Resnik, 2008] but with experiments conducted for more diverse languages as

well as choosing the source data that are as closer to target language. In other words, during delexicalization, instead of choosing all the source data for training, the work ranked the source data according to their closeness with the target then only top group of data were chosen. [Søgaard, 2011] demonstrated that choosing the source data according to their closeness with the target improves the target language dependency parser performance.

Word cluster features have been shown to aid variety of learning tasks in NLP including dependency parsing [Koo et al., 2008]. Recently, [Täckström et al., 2012] exploited the idea of word cluster features to both monolingual and cross-lingual structure prediction tasks. For cross-lingual dependency structure prediction, they extended delexicalized direct transfer approach of [McDonald et al., 2011] with cross-lingual word cluster features. The results have shown to improve over direct transfer approach. However, inducing cross-lingual word cluster require parallel corpora and word alignment which could be still problematic for many resource-poor languages.

[Durrett et al., 2012] considered adding lexical features to delexicalized parsers via bilingual dictionaries. This approach does not require parallel corpus for target language parsing, however it assumes the availability of a bilingual dictionary constructed manually or automatically via word aligners. They applied their approach for eight languages and shown competitive results similar to [McDonald et al., 2011] and [Täckström et al., 2012]. They have also shown the application of their approach by training a few hundred (from 100 to 400 trees) target language trees and noted that constructing as few as 100 trees would tremendously helpful in learning target language parsers.

2.4 Semi-supervised Methods

Supervised approaches work well when there is plenty of labeled data for training, but they are often ineffective in situations where labeled training data is scarce or not available. This scenario has been well studied in the Machine Learning (ML) research community. A class of algorithms generally known as semi-supervised algorithms can be suitable in these scenarios. Semi-supervised approaches [Zhu, 2005] normally use small amount of labeled data with large amounts of unlabeled data to learn parameters. Semi-supervised learning can be viewed as either *inductive* (where the main aim is to predict labels for the *test set* in which unlabeled data will be part of the learning process along with the labeled data) or *transductive* (where the main aim is to predict labels for the unlabeled data of our interest, this approach mainly focuses on the task at hand, i.e., predicting labels for the unlabeled data rather than building generic models that can make predictions for unseen data). An interesting analogy for difference between transductive and inductive learning is similar to in-class and take-home exam [Zhu, 2005].

Variety of algorithms [Zhu, 2005] have been developed under the tag of semi-supervised approaches. Some of the most popular methods in semi-supervised ap-

proaches include self-training, co-training, Expectation-Maximization (EM) with generative mixture models, transductive support vector machines (TSVMs) and graph based algorithms.

Generative models can be expressed in the generic form: $p(x, y) = p(x|y)p(y)$. Some of the well-known generative models include Gaussian Mixture Models (GMM), Hidden Markov Models (HMM) and Naïve Bayes classifier. Underlying algorithms often use EM like iterative procedure to determine model parameters. Generative models have been successfully used in many real-time applications including text-classification, speech recognition, image recognition and so on. Generative models are very effective if the model assumptions are correct, but the addition of unlabeled data can hurt the model predictions if the model assumptions are wrong. Self-training and co-training are the other two simple and popular approaches in semi-supervised learning. Self-training works in scenarios where there's a small amount of labeled data. Initially a learner is trained with the small amount of available labeled data. Then the learner is tasked with the prediction of unlabeled data. Unlabeled data which was predicted with high confidence is added to the training set. Then the learner is retrained with the additional self-labeled data in addition to the original labeled data. This procedure, i.e., predicting labels for unlabeled data and adding high confidence predictions to the training set is repeated many times. Self-training approach helps to bridge the gap in the lack of labeled data. In *co-training*, two independent classifiers (with two different views of the same labeled data) are trained on a small labeled dataset and used to label large quantities of unlabeled data. High confidence predictions from both the classifiers are then added to the training set. This way each classifier produces labeled data for the other and learn together. Co-training assumes that two independent view of the labeled data exists.

In NLP, it is easy to find large amounts of unlabeled data [Majliš and Žabokrtský, 2012a] for most of the natural languages. [Steedman et al., 2003] used co-training based approach to bootstrap parsers from small datasets. [Koo et al., 2008] incorporated cluster features from unlabeled data in addition to baseline features. They observed that when cluster based features are included in the feature design, the performance they obtained was similar to the results that used baseline features and twice the amount of training data. That means, cluster based features reduces the training data requirements by half.

For Tamil, the work by [Green et al., 2012] used small amount of Tamil treebank data to improve parsing accuracy. They trained various dependency models and used model agreements to train a support vector machine (SVM) based classifier. SVM decisions were made on an edge by edge basis to build the target structure. The work showed statistically significant results for Tamil data.

2.5 Unsupervised Methods

Unsupervised learning approaches are quite popular in every domain. In NLP, unsupervised methods are being used in almost every NLP tasks, in some tasks unsupervised methods are indispensable or most preferred way to attack the problem, for example, tasks such as word alignments [Liang et al., 2006, Och and Ney, 2003] and sentence alignments [Varga et al., 2005].

Unlike other NLP tasks which require only prediction over simple output space (such as binary class problems or even multi-class problems), syntactic representations rather have complex structured output spaces. Given the complexity of the parsing task, unsupervised methods rather have a mixed success in achieving better prediction of syntactic structures. Early works such as [Klein and Manning, 2004] has generated huge interest in unsupervised methods. This is an active area of research and most notable works in this area include: [Snyder et al., 2009, Mareček and Žabokrtský, 2012, Spitkovsky et al., 2011].

2.6 Annotation Standardization Efforts

There has not been many attempts in the past to standardize annotation for natural languages at different levels. Though there has been a convergence of opinion in developing NLP tools (such as taggers, parsers, etc.) in a language independent manner, no such conformity exists on the annotation standards. Moreover, most of the language independent tools do not impose any restrictions on the annotation standards, thus accommodating flexibility in having desired annotation styles. If one were to look at annotation standards in a holistic view, having a same annotation style across languages for a similar task has many advantages,

- helps to make generalization of various linguistic properties across languages and language families.
- comparative studies of various linguistic phenomena will become much more easier.
- useful in multilingual POS, grammar induction and other NLP tasks.
- common annotation standard can be helpful in developing analysis tools for languages that do not have any.

Treebank annotations can diverge in terms of formalism level (phrase structure vs. dependency) or various levels within the same formalism (for example: differences in annotation styles of POS, dependency relations and marking of structures in the dependency framework). Especially in the context of transferring analysis from a source language to a target language, having a common annotation standard

would be much more useful in inducing tools and annotation for target languages. In this section, we list down three works in the area of evolving standardized annotation in POS and treebank schemes.

2.6.1 Universal part of speech tagset

The work by [Petrov et al., 2012] concerns with the standardization of POS tagset across languages. They proposed a tagset of 12 coarse-grained POS categories which they called universal POS tags. In addition to the universal POS tags, they also developed a POS mapping for 25 treebank POS tagsets (covers 22 languages) to the proposed universal POS tagset. Their selection of tagset is based on the principle (*linguistic universal*) that most languages across different language families share certain POS categories and can occur widely across different language families.

To illustrate the benefits of the universal tagset, they have used the tagset in three use case scenarios.

1. **POS tagging:** They mapped 25 different treebank tagsets to universal tagset and measured the POS accuracy. They trained POS taggers with both the original tagset and the universal tagset. In the case of supervised dependency parsing, they observed that the parser looses only about 0.6% accuracy when training with the universal tags instead of the PennTreebank tagset with 45 tags. This observation is significant in a sense that it shows the usage of coarse-grained tagset in the place of fine-grained tagset does not result in a huge drop of POS accuracy.

2. **Grammar induction:** They used the universal tags in the grammar induction system of [Naseem et al., 2010] to induce grammar for eight Indo-European languages. Two step process were used to achieve this. Initially the tagger projection system is used to project universal tags across the languages and then a set of universal syntactic rules were used to guide the parsing model. They showed that even without using gold POS tags, their system performed better than other unsupervised models.

3. **Parser transfer:** They used the universal tagset in the delexicalized English parser. Target language sentences that are tagged with the universal tags are parsed directly using the delexicalized English parser. They obtained better accuracies than most state-of-the-art unsupervised parsers.

Tag	Description
NOUN	nouns
VERB	verbs
ADJ	adjectives
ADV	adverbs
PRON	pronouns
DET	determiners and articles
ADP	prepositions and postpositions
NUM	numericals
CONJ	conjunctions
PRT	particles
.	punctuation marks
X	a catch-all for other categories such as abbreviations and foreign words.

Table 2.1: Universal POS tagset [Petrov et al., 2012]

2.6.2 HamleDT

HamleDT - *HArmonized Multi-LanguagE Dependency Treebank* [Zeman et al., 2012] is a recent attempt at harmonizing various treebanks (phrase structure as well as dependency) into single annotation style. [Zeman et al., 2012] harmonizes treebanks at structural level (transforming various syntactic structures to conform to PDT[2] style annotation), morphological level (*Interset* [Zeman, 2008] approach is used to transform treebank POS tags to PDT style tags and Interset feature structures) and relational level (original dependency relations are converted to dependency relations used in PDT).

Interset - is the approach [Zeman, 2008] used in HamleDT to transform treebank POS/morphological information into PDT style morphological tag format. Universal set or Interset is a feature structure capable of storing all kinds of POS and morphological information belonging to any tagset. The tagset can be a simple coarse-grained tagset or can be a fine-grained tagset. Interset acts as an intermediate step in the tagset conversion. Through *tagset drivers*, conversion between arbitrary tagsets in both the directions are possible, i.e., $A \rightarrow B$ and $B \rightarrow A$. To do that, one has to write a tagset driver per tagset. Tagset driver can do two things: (i) *decoding* a given tag into Interset features and (ii) *encoding* a interset feature structure into a tag.

[2]Inspired by the annotation style used in Prague dependency treebank (PDT)

Interset features
pos, subpos, prontype, numtype, numform, numvalue, accommodability, advtype, punctype, puncside, synpos, morphpos, poss, reflex, negativeness, definiteness, foreign, gender, possgender, animateness, number, possnumber, possednumber, case, prepcase, degree, person, possperson, politeness, subcat, verbform, mood, tense, subtense, aspect, voice, abbr, hyph, echo, style, typo, variant, tagset & other

Table 2.2: Interset features [Zeman, 2008]

Table 2.2 shows the components of Interset feature structure that are used to capture POS/morphological information present in a given tagset. First part of the Table 2.3 shows what values (fixed) can be stored under the feature `pos`. The second part of the Table 2.3 shows the coarse-grained tagset in PDT style [3]. Each PDT style tag or Interset tag is a fixed length string (15), and each character position signifies a particular morphological feature. The first position corresponds to main POS category (or coarse-grained). HamleDT 1.0 contains 29 treebanks, and a tagset driver is written for each treebank's native POS tagset. Harmonized treebanks contain POS/morphological information in both Interset feature and tag formats.

Harmonizing the structure of various treebanks into HamleDT annotation was done by: (i) converting dependency relation of the original treebank into PDT style dependency relation, and (ii) transforming the original tree structure to PDT style tree structure. Tree transformation was applied to various syntactic phenomena, such as coordinations, ad-positional phrases, subordinated clauses, verb groups, determiner heads and punctuations. The harmonization of treebanks did not include tokenization and sentence segmentation, so as to allow easier comparison between original and harmonized treebanks. Harmonized treebanks are shown in Table 5.3.

2.6.3 Universal dependency annotation

Similar to HamleDT, [McDonald et al., 2013] developed treebanks for six languages - German, English, Swedish, Spanish, French, and Korean with common syntactic annotation style. For POS, they use universal POS tagset [Petrov et al., 2012], and for dependencies, they use Stanford dependency [de Marneffe and Manning,

[3]PDT 2.0 documentation - `http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/m-layer/html/ch02s02s01.html`

Tagset	Tag	Description
Interset (`pos`)	noun	noun
	adj	adjective
	num	numeral, number
	verb	verb
	adv	adverb
	prep	adposition
	conj	conjunction
	part	particle
	int	interjection
	punc	punctuation or symbol
PDT coarse-grained tag (1^{st} position in the morphological tag)	A	adjective
	C	numeral
	D	adverb
	I	interjection
	J	conjunction
	N	noun
	P	pronoun
	V	verb
	R	preposition
	T	particle
	X	unknown
	Z	punctuation

Table 2.3: Interset feature `pos` [Zeman, 2008] and PDT coarse-grained tag

2008] representation as the annotation style for their treebanks. They do automatic conversion to Stanford style for English and Swedish, and for the other languages, they constructed the treebanks manually in Stanford style. In the case of manual annotation, annotators were allowed to add new labels for the language specific aspects. Thus, in the harmonization step they made the language specific labels into more general labels. They also applied other operations such as renaming of dependency labels to make the annotation more uniform across treebanks. They demonstrate the effectiveness of common annotation strategy via cross-lingual parser transfer case study. Unlike in earlier attempts (only unlabeled attachment score is normally reported), they also report both labeled and unlabeled attachment scores for their parser transfer. One interesting outcome of this study was that, cross-lingual parser performs better if the treebanks (both source and target) are mapped to a common annotation style than cross-lingual parsers that use native treebank annotations.

2.6.4 Remarks

We have listed three frameworks for common annotation scheme of POS (universal tagset) and dependency treebanks (HamleDT and universal treebank) that are emerging as a common standard for evaluating cross-lingual technologies. Apart from the works described above, a well-known related work is MULTEXT-East [Erjavec, 2010], a multilingual standardized dataset for 16 central and eastern European languages. It defines uniform encoding and common annotation format at morphosyntactic level. Core to their approach is *morphosyntactic descriptions (MSD)* [4] which consist of 12 categories (corresponding to major POS types) and comprehensive list of valid *attribute-value* pairs each category can take. The dataset contains MSD specifications (categories, valid *attribute-value* pairs for each category ordered by types and a table showing whether a particular attribute can occur across languages), MSD lexicon (wordforms enriched with MSD), and other corpora (such as parallel corpus) since the initial development in 1995.

There's a clear trend emerging in multilingual dependency parsing, i.e., the use of simplified/standardized annotation styles during both transfer and evaluation. In this thesis, in one of our dependency transfer experiments, we train delexicalized parsers from HamleDT treebanks (harmonized version as given in Table 5.3) and parse our ILs directly from using delexicalized HamleDT parsing models. Please refer Section 6.2 for more details.

[4] MULTEXT-East Morphosyntactic Specifications - http://nl.ijs.si/ME/Vault/V3/msd/html/

Chapter 3

Survey of Available Resources

In this chapter, we outline data resources that are already available for Indian languages and are relevant to our work. We pay attention to three kinds of resources, namely parallel corpora involving Indian languages, treebanks and POS taggers.

3.1 Languages of India

India is one of the linguistically diverse country in the world; and it is home to hundreds of languages (447 living)[1] spanning four language families. The language families found in India are: *Indo-Aryan*, *Dravidian*, *Austroasiatic*, and *Tibeto-Burman*. The first 2 families are the major ones and together they account for more than 95% of all speakers [2] in India. *Indo-Aryan* family is a subbranch of Indo-European and are spoken in most of northern region of India. Some major languages in this family include *Hindustani (Hindi-Urdu), Bengali, Marathi, Gujarati, Oriya,* etc. The Figure 3.1 depicts the geographical spread of languages of Indo-Aryan branch. Speakers of Dravidian language family can be found mostly in southern India and northern Sri Lanka. Major languages of the Dravidian family are: *Tamil*, *Telugu*, *Kannada*, and *Malayalam*. One notable exception is *Brahui*, a Dravidian language-speakers (about 4 million) of which are mostly in Pakistan. Similar kind of exception for Indo-Aryan is *Sinhalese* which is spoken mainly in Sri Lanka. The geographical spread of Dravidian languages is depicted in Figure 3.2. The speakers of the other 2 families are mostly found in north-eastern regions of India. Major languages of Tibeto-Burman family within India include *Bodo* and *Manipuri*; and an example for Austroasiatic family within India is *Santali*. Major languages (such as Tibetan, Burmese, and Vietnamese) making up Tibeto-Burman and Austroasiatic family are widely spoken in Tibet and southeast Asian regions.

Hindi and English are the two official languages of India; however, individual states can legislate their own official languages. At the moment, India recognizes 22 languages with official status. Rich body of literature exists for many of those

[1] Ethnologue: Languages of the World - `http://www.ethnologue.com/country/IN`

[2] Languages of India - `http://en.wikipedia.org/wiki/Languages_of_India`

languages.

3.2 Are Indian Languages Under-resourced?

Though as many as 29 languages are spoken by at least 1 million people, we restrict our focus to top 10 Indian languages according to number of speakers [3]. Computational approach to Indian languages have been active for at least two decades. Pāṇinian approach [Bharati et al., 1996] is being widely used as a fundamental framework for analyzing Indian languages. Pāṇinian approach is actually a grammar system introduced by an early (about 4^{th} century BCE) Sanskrit grammarian Pāṇini for the Sanskrit language. As opposed to phrase structure grammars which are quite popular for analyzing languages such as English, Pāṇinian model of grammar emphasizes on extracting *syntactico-semantic* relations (also known as *karaka* relations). These relations often express relations between verbs and other constituents in a given sentence. The *karaka* relations closely resemble dependency relations in dependency grammars. Also, this framework relies mainly on case endings and postpositions to determine *karaka* relations. This form of analysis makes it suitable for free word order languages (languages that make use of rich morphology). Computational grammars such as treebanks for Hindi, Bengali, and Telugu – built with the Pāṇinian framework, is a case in point.

Can we determine whether ILs are under-resourced at the present scenario? This question can be partially answered through the concept of Basic Language Resource Kit (BLARK) as defined by [Krauwer, 2003]. It defines BLARK as *"the minimal set of language resources that is necessary to do any precompetitive research and education at all"*. BLARK includes [Krauwer, 2003]: spoken/written language corpora, grammars, modules (taggers morphological analyzers, parsers, speech recognizers, text-to-speech), annotation standards and tools, and so on.

We did a simple survey of availability of tools and resources that are essential for cross-lingual dependency transfer and are also defined as necessary for language research according BLARK. The methodology for survey is a simple web search for those resources and check whether they are accessible through the web. We did a survey for top 10 Indian languages and assessed the accessibility of resources such as parallel corpora, POS taggers, treebanks and so on. The Table 3.1 shows our survey results. Treebanks and parallel corpora[4] are available for only 5 out of 10 Indian languages.

[3] Census of India (2001): `http://www.censusindia.gov.in/Census_Data_2001/Census_Data_Online/Language/Statement4.aspx`

[4] There are many Indian language tools and resources made available via *Technology Development for Indian Languages (`http://tdil-dc.in/`)*. The tools and resources are accessible only within India due to licensing issues. So we were not able to access them even though they are free of cost.

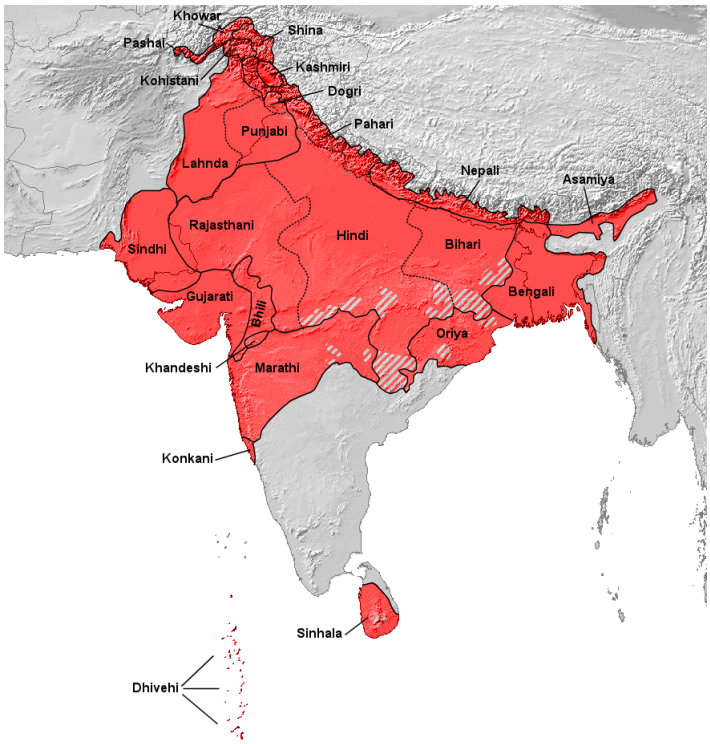


Figure 3.1: Indo-Aryan languages (Image source: Wikipedia)

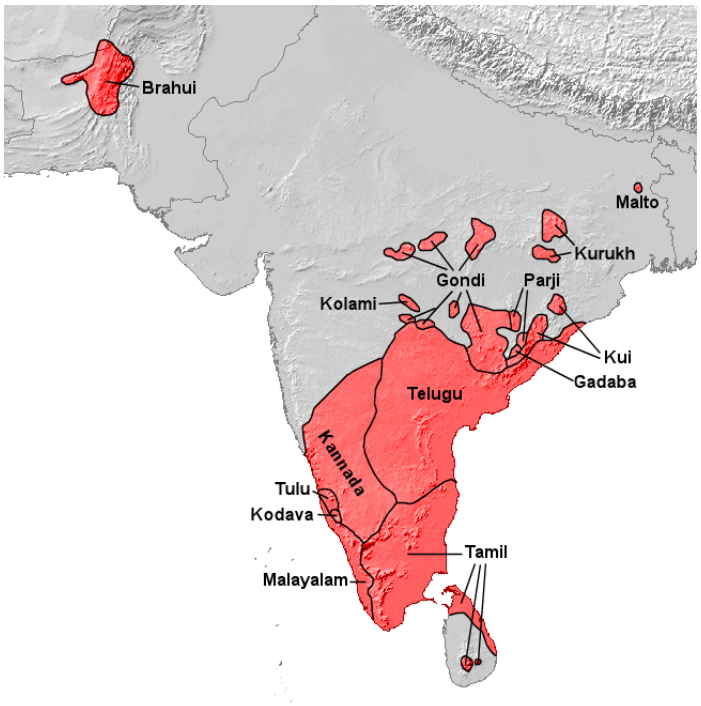


Figure 3.2: Dravidian family (Image source: Wikipedia)

Lang.	Treebank	Parallel corpora	POS tagger	Web corpora	GT
Hindi	HyDT	JHU HindEnCorp EMILLE	✓	W2C EMILLE	✓
Bengali	HyDT	JHU EMILLE	✓	W2C EMILLE	✓
Telugu	HyDT	JHU	✓	W2C EMILLE	✓
Marathi	×		✓	W2C EMILLE	✓
Tamil	TamilTB	EnTam JHU	✓	W2C EMILLE	✓
Urdu	UDT	UMC005 JHU EMILLE	✓	W2C EMILLE	✓
Gujarati	×	EMILLE	✓	W2C EMILLE	✓
Kannada	×		✓	W2C EMILLE	✓
Malayalam	×	JHU	✓	W2C EMILLE	×
Oriya	×			EMILLE	×

Table 3.1: Resource availability for Indian languages. The column header *GT* means online Google machine translation system. The cell entry ✓indicate the resource is accessible through web. The cell entry × means the unavailability of resource. Empty cells indicate the resource may be available, but we were not able to verify. *W2C* - is a monolingual corpora for number of languages made available by [Majliš and Žabokrtský, 2012a]. *HyDT* - 3 IL treebanks released as part of ICON 2010 NLP tools contest [Husain et al., 2010]. *TamilTB* - is a Tamil dependency treebank developed by [Ramasamy and Žabokrtský, 2011]. *UDT* - is a Urdu treebank developed by [Bhat and Sharma, 2012]. *JHU* - is a parallel corpora for six Indian languages by [Post et al., 2012]. *HindEnCorp* - is an English-Hindi parallel corpus made available by [Bojar et al., 2014]. *EnTam* - is an English-Tamil parallel corpus made available by [Ramasamy et al., 2012]. *EMILLE* - is corpus developed by [Baker et al., 2002] for many south Asian languages.

3.3 IL Treebanks

Syntactic annotation for Indian languages is rather a recent development (late 2000) when compared to resources such as creation of parallel corpora, POS tools/corpora and morphological analyzers and so on. However, there have been efforts to create a manually annotated dependency treebanks for major Indian languages. Table 3.1 shows for how many Indian languages possess treebanks at the moment.

- Dependency treebanks for three Indian languages (Hindi, Telugu and Bengali) were released as part of the ICON-2010 tools contest [Husain et al., 2010]. The annotation was based on *Paninian* approach [Begum et al., 2008a].
- The dependency treebank for Tamil [Ramasamy and Žabokrtský, 2012] was released in 2011 and Prague dependency treebank (PDT) style multilayer annotation was used to annotate Tamil data.
- Urdu treebank was made available by [Bhat and Sharma, 2012]. The treebank is still in development.

Treebanks are not available for the rest of the Indian languages.

3.4 IL Parallel Corpora

The parallel corpora have been the single most important resource in practical applications such as Machine Translation (MT). They have also been used in other tasks such as grammar induction [Hwa et al., 2005, Ganchev et al., 2009], semantic role labeling [Padó and Lapata, 2009] and POS tagging [Das and Petrov, 2011a] by transferring annotation from one language (usually resource-rich language) to another language (usually resource-poor). For Indian languages, the most likely scenario will be the availability of parallel corpora between English and Indian languages than a multilingual corpora comprising Indian languages (with or without English).

Many Indian languages still lack parallel corpora, also the size of the parallel corpora is not mature enough to get good quality MT outputs. However, an available bilingual/multilingual corpora would be a valuable resource for inducing treebanks for ILs using cross-lingual syntactic projection. At present, IL parallel corpora is available for major Indian languages with varying size. Table 3.1 shows the availability of parallel corpus seven ILs. Since treebanks are available for only five ILs, we will make use of only 5 ILs parallel corpus for our experiments. The individual parallel corpus size we use for our experiments are given in Table 5.1. Following sub sections describe more about the nature of the parallel data.

3.4.1 JHU parallel corpora

The parallel corpus for six Indian languages from [Post et al., 2012] were another important source of parallel data for ILs. The parallel data contains Wikipedia articles from six Indian languages (Bengali, Hindi, Urdu, Malayalam, Tamil and Telugu) translated into English through crowd sourcing. The data for each language consists of top 100 most viewed Wikipedia documents in that language, so the data is not multilingual parallel corpora.

3.4.2 English-Tamil parallel corpora

Apart from the manually constructed English-Tamil parallel data (JHU) as part of [Post et al., 2012], there is a decent amount of parallel English-Tamil parallel data available in the web that are largely unnoticed or unexploited. We thus collected our own corpus (EnTam) [Ramasamy et al., 2012] quickly and at no cost for translation.

We mainly collected parallel corpora from three sources: (i) `www.wsws.org` (*News* - news website) (ii) `www.cinesouth.com` (*Cinema* - Tamil cinema articles) and (iii) `http://biblephone.intercer.net/index.php` (*Bible*). The above three sources are either multilingual or contain exclusive English and Tamil contents. To collect the *News* corpus, we downloaded only URLs that have matching *file names* on both English and Tamil sides. The collection of *Cinema* corpus was simple: all the English articles had a link to the corresponding Tamil translation on the same page. The collection of *Bible* corpus followed a similar pattern.

After downloading the English URLs and the corresponding Tamil URLs, we stripped all the HTML tags. At this stage, the *Bible* corpus was already sentence aligned. The *News* and *Cinema* articles had similar paragraph structures but they were not sentence aligned. We used *hunalign* [Varga et al., 2005] to sentence align them.

3.5 IL POS Taggers

Table 3.1 shows that POS taggers are available for 9 out of 10 ILs. However, we have not verified the functioning of each tool. In our dependency transfer experiments we train POS taggers ourselves for ILs to have the tagset compatibility with IL treebanks.

3.6 IL Web Corpora

Web Corpus (W2C) [Majliš and Žabokrtský, 2012b] is a compilation of textual data for 106 languages. Table 3.1 shows W2C is available for 9 out of 10 ILs (except Oriya). EMILLE [Baker et al., 2002] is another collection of monolingual/parallel

and annotated data for many south Asian languages. EMILLE contains monolingual corpora for all ILs in Table 3.1.

3.7 IL Google Translation

At present, *Google translation*[5] is available for 8 out of 10 ILs in Table 3.1. In our experiments we make use of Google online MT system as well as Google Translate API to obtain machine translated bitexts for one of experiments.

[5]Online Google MT system: `https://translate.google.com/`

Chapter 4

Tamil Dependency Treebank

This chapter provides a detailed look into our efforts in creating a dependency treebank for Tamil from scratch. Sections 4.6 and 4.7 provide a complete overview of morphological and syntactic annotation standard used to annotate Tamil dependency treebank. So far, the Tamil dependency treebank (TamilTB) has seen two versions:

- *TamilTB.v0.1* - An initial release which was made available in May, 2011.
- *TamilTB 1.0* - The ongoing latest version which has been available since Nov, 2013.

This chapter explains the annotation scheme on the basis of TamilTB.v0.1. However, most of the annotation descriptions explained here are valid for the latest version too.

4.1 Why New Treebank?

As we have already mentioned in Sections 1.4 and 1.5, one of our main goal besides finding best cross-lingual strategies for ILs is to create a treebank for an under-resourced language with reasonable quality. In this respect, the created resource can be used in evaluation of cross-lingual strategies as well as to gain insights into how much work will be needed in case we want to create a reliable treebank for a new under-resourced language.

4.2 Background and Objectives

Treebank is an important resource in many natural language processing (NLP) applications including parsers, language understanding, machine translation (MT) and so on. Treebanks are often manually developed and are available for only handful of languages such as English, German and Czech. Most of the world's other languages (irrespective of contemporariness and the number of speakers) do not have

treebanks due to various reasons: (i) high development cost (ii) long development time (iii) lack of expertise to name a few. In this project, our main aim is develop a dependency treebank for Tamil language similar to Prague Dependency Treebank (PDT) annotation style.

We briefly introduce the work carried out on the subject prior to this work. There is an active research on dependency parsing [Bharati et al., 2009], [Nivre, 2009] and developing annotated treebanks for other Indian languages such as Hindi and Telugu. One such effort is, developing a large scale dependency treebank [Begum et al., 2008b] (aimed at 1 million words) for Telugu, as of now the development for which stands [Vempaty et al., 2010] at around 1500 annotated sentences. For Tamil, previous works which utilized Tamil dependency treebanks are: [Dhanalakshmi et al., 2010] which developed dependency treebank (around 25000 words) as part of the grammar teaching tools, [Selvam et al., 2009] which developed small dependency corpora (5000 words) as part of the parser development. Other works such as [Janarthanam et al., 2007] focused on parsing the Tamil sentences. Unfortunately, none of the treebanks have been published in the web. One of our general aim is to create annotated resources and make it publicly accessible. In technical aspects, the current treebank annotation task differs from previous works with respect to the following objectives:

- annotating data at morphological level and syntactic level
- in each level of annotation, try for maximum level of linguistic representation
- building large annotated corpora using automatic annotation process

4.3 Data

The data used for the creation of Tamil dependency treebank (TamilTB) annotation comes from news domain. We decided to use the news data for two reasons: (i) huge amount of data is available in digital format and can be easily downloaded and (ii) the news data can be considered as representative of written Tamil. At present, the data for the annotation comes from `www.dinamani.com`, and we downloaded pages[1] randomly covering various news topics. The data we use for annotation is described in Table 4.1

4.4 Text Preprocessing

Before the actual annotation takes place, the raw text data is preprocessed in three steps in sequential order,

[1]We downloaded news paper articles that appeared during year 2010-2011.

Description	Value
Source	www.dinamani.com
Source transliterated	Yes
Number of words	9581
Number of sentences	600
Morphological annotation (sen)	600
Syntactic annotation (sen)	600
Tectogrammatical annotation	–

Table 4.1: The data for annotation

- Transliteration
- Sentence segmentation
- Tokenization

Each preprocessing step is explained in the following subsections.

4.4.1 Transliteration

The UTF-8 encoded Tamil raw text was transliterated to Latin format for ease of processing during all levels of annotation. The raw UTF-8 encoded text can be obtained by applying reverse transliteration to the Latin-transcribed text. The transliteration scheme for Tamil script is given in Figure 4.1. The Figure shows the transliteration for vowels, consonants, Sanskrit characters and an example transliterated sequence of consonant ('k')-vowel combination. Tamil has separate character representation for each consonant-vowel combination.[2]

Tamil is a phonetic language and the transliteration scheme is designed to match the Tamil sounds as much as possible. In this documentation, wherever possible, both Tamil script and transliterated form are used in examples. In few places, only transliterated format is used due to difficulties (mainly rendering problems) in embedding Tamil scripts.

4.4.2 Sentence segmentation

Before the annotation, the raw corpus downloaded from the source is sentence segmented automatically. This step can be performed before or after the transliteration.

[2]For full list of transliteration map, please refer this URL: https://ufal.mff.cuni.cz/~{}ramasamy/tamiltb/0.1/utf8_to_latin_map.txt (change the encoding of the browser to UTF-8 to view the contents properly)

Tamil Vowels	அ ஆ இ ஈ உ ஊ எ ஏ ஐ ஒ ஓ ஔ ஃ
Transliteration	a A i I u U e E ai o O au q
Tamil Consonants	க் ங் ச் ஞ் ட் ண் த் ந் ப் ம் ய் ர் ல் வ் ழ் ள் ற் ன்
Transliteration	k ng c nj t N T w p m y r l v z L R n
Sanskrit Characters	ஷ் ஹ் ஜ் ஸ்ரீ ஸ்
Transliteration	sh h j sri S
'k' + vowel combination	க கா கி கீ கு கூ கெ கே கை கொ கோ கௌ
Transliteration	ka kA ki kI ku kU ke kE kai ko kO kau

Figure 4.1: Transliteration scheme

Like English, Tamil can also be ambiguous at various places that may look like sentence boundaries, but in reality they are not. Those ambiguous sentence boundaries (such as dots at decimal numbers, initials in names and dates) are detected through heuristics and sentences are segmented only at appropriate places. As an example, in Tamil, the proper names are written using (*'surname name'*) format. But in usage the format is shortened to (*'initial. name*') where a dot is placed between *initial* and *name*. *Initial* is the first letter of the *surname*. For example, the full-name *'Palaniappan Chidambaram'* in English will be written as *'P. Chidambaram'* & ப. சிதம்பரம்/pa. ciTamparam in English and Tamil respectively. The heuristics for this problem is straightforward[3]. We listed down all possible Tamil characters (letters) and whenever a sentence termination symbol (*dot*) occurs after an initial will not be treated as a sentence boundary.

4.4.3 Tokenization

Tokenization is one of the important module that helps the annotation task. This module splits the sentence into words. Tamil uses spaces to mark word boundaries. But yet, a lot of Tamil wordforms are agglutinative in nature, meaning they glue together at least two words (in majority of cases). Those cases can be identified as determiners+nouns, nouns+postpositions, verbs+particles or nouns+particles. Except the first pattern (determiners+nouns), in all other cases, the second part of the wordforms are restricted and can be listed. So it is possible to split certain Tamil agglutinative wordforms into separate tokens. Certain particles (also called clitics) such as உம்/um/'also', ஓ/O/'or' are not treated as separate tokens in Tamil. But for the purpose of annotation we treat them as separate tokens. The same module will be used for tokenization when parsing the raw Tamil text.

[3]Tamil words with a single letter are quite rare. So a Tamil letter followed by a dot is most likely to be an initial. Since Tamil initials are always one/two letters in length, we use the whole Tamil alphabets and different combinations to check whether the token is an initial or not. Moreover, Tamil does not have lower or upper case distinctions.

Before splitting	Tamil	புதிய சட்டத்தின்படி , பாதுகாக்கப்பட்ட நினைவுச் சின்னத்திலிருந்து 1000 அடி வரை எந்த கட்டுமானமும் கட்ட அனுமதி இல்லை .
	Trans.	puTiya cattaTTin**pati** , pATukAkkap**patta** winaivuc cinnaTT**iliruwTu** 1000 ati varai ewTa kattumAnam**um** katta anumaTi illai .
After splitting	Tamil	புதிய சட்டத்தின் படி , பாதுகாக்கப் பட்ட நினைவுச் சின்னத்தி லிருந்து 1000 அடி வரை எந்த கட்டுமான மும் கட்ட அனுமதி இல்லை .
	Trans.	puTiya cattaTTin **pati** , pATukAkkap **patta** winaivuc cinnaTT **iliruwTu** 1000 ati varai ewTa kattumAnam **um** katta anumaTi illai .

Figure 4.2: Tokenization example

In Figure 4.2, படி/pati/'manner' and இலிருந்து/iliruwTu/'from' are postpositions, பட்ட/patta is an auxiliary verb and உம்/um is a clitic. This kind of agglutination is very prevalent in Tamil, and it would be useful to tagging process if we are able to reduce the vocabulary size by splitting the known combinations as separate tokens.

clitics	உம், ஏ, ஏயே, ஆவது
postpositions	கூட, உடன், படி, குறித்து, இலிருந்து, அன்று, உள், ஆறு, தவிர, போது, போல, பின்னர், பின், அருகே, அற்ற, இன்று, இல்லாத, மீது, கீழ், மேல், முன்பே, ஒட்டி, பற்றி, பற்றிய, போன்ற, மூலம், வழியாக
auxiliary verbs	பட்ட, பட்டு, உள்ள, பட, மாட்டாது, படுவார்கள், உள்ளார், உள்ளனர், இல்லை, இருந்தார், இருந்தது, பட்டது, பட்டன, முடியும், கூடாது, வேண்டும், கூடும், இருப்பின், உள்ளன, முடியாது, படாது, கொண்டு, செய்
particles	ஆக, ஆன and their spelling variants ஆகச், ஆகத், ஆகப், ஆகக்
demonstrative pronouns	அப், அச், இச், அந், இ as prefixes

Table 4.2: List of suffixes and words for tokenization

Some of the most commonly occurring (from the corpus) words and suffixes which participate in agglutination are given in Table 4.2 above. Except demonstrative pronouns, all other words and suffixes are added after the stem. Among the categories,

clitics and particles are the most participated in the agglutination. The tokenizer[4] makes use of this list and try to separate these words from the original wordform. Even after the tokenization, it would be possible to reconstruct the original sentence by making use of the attribute called `no_space_after'[5]. The `no_space_after' will be set to 1 if the following token is not separated from the current token. Whenever the splitting takes place this attribute will be set to 1 for the first token. For example, The `no_space_after' attribute for பாதுகாக்கப்/pATukAkkap will be 1. Whereas the `no_space_after' attribute for உம்/um will be 0. The splitting for the corpora has been done semi-automatically using some of the most commonly occurring combination from the above list and edited manually during the annotation process. At present, the tokenizer includes only few commonly occurring combinations such as clitics, particles and very few postpositions.

We evaluated how much such combinations have been splitted from the original corpora. We found that 953 splits took place out of 9581 words. We simply did this by counting how many `no_space_after' attributes have been set to 1. We can say that almost 10% of the additional corpus size is due to splitting some wordforms into separate tokens.

4.5 Layers of Annotation

The annotation scheme used for TamilTB is similar to that of Prague Dependency Treebank 2.0 (PDT 2.0) [Hajič et al., 2006]. PDT 2.0 uses the notion *'layers'* to distinguish annotation at various levels (linguistic) such as word level and syntactic level. Precisely, PDT 2.0 is annotated at 3 levels or layers: (i) morphological layer (m-layer), (ii) analytical layer (a-layer) and (iii) tectogrammatical layer (t-layer). At present, TamilTB is annotated at only two layers: m-layer and a-layer.

Tamil	பண்பாட்டு அடையாளங்களைப் பாதுகாக்க தொல்பொருள் ஆய்வுத் துறை உருவாக்கப் பட்டு , தனிச் சட்டங்கள் இயற்றப் பட்டு உள்ளன .
Trans	paNpAttu ataiyALangkaLaip pATukAkka TolporuL AyvuT TuRai uruvAkkap pattu , Tanic cattangkaL iyaRRap pattu uLLana .
English	Having created Arachelogical Department, separate laws have been enacted to protect cultural symbols .

Figure 4.3: Tamil sentence example

In the the example shown in Figure 4.3, the actual sentence is given in Tamil

[4]At the moment, we only make use of the list in Table 4.2. So we don't handle compounds that comprise more than two words.

[5]We use Tred editor for annotating the treebank. In Tred, each wordform/token in a sentence is represented by a node, and each node has set of attributes such as `form, tag, afun, is_member, no_space_after,` and so on. Those attributes are filled/set in the graphical editor during annotation as and when required.

script (indicated as Tamil:) in the 1st row, the transliterated (indicated as *Trans*) version in the 2nd row, and the actual English translation in the 3rd row. The same format is used to illustrate sentence examples elsewhere in the document. There are 15 words in the Tamil sentence (including punctuations), each word will be treated as a node in each annotation layer. Please note that the term *node* will be used interchangeably with other terms *wordform* or *word* to represent a vertex in a-layer or m-layer.

Each node will have general attributes and layer specific attributes. For ex: a node in morphological layer will have attributes such as, 'lemma', 'form', 'tag' and `` `no_space_after' `` corresponding to lemma, wordform, POS tag of a particular wordform and whether the following wordform is part of the current wordform. A node in analytical layer will have attributes such as dependency label ('afun') of the current node, whether the current node is an element in the coordination conjunction (`` `is_member') ``. These attributes will be set automatically during parsing or editing attributes manually using TrEd[6]. Also, the lower layer (m-layer) attributes are visible to upper layers (a-layer or t-layer).

Only transliterated version of the text will be used in all layers of annotation for the ease of processing. Examples in Tamil script are shown only for display purposes.

The following subsections briefly describe the annotation layers of TamilTB with an example.

4.5.1 Morphological layer (m-layer)

The purpose of m-layer is to assign Parts of Speech (POS) tag or more refined morphological tag to each word in the sentence. This is accomplished by setting the *tag* attribute of the node (corresponds to word) to the POS or morphological tag. The *lemma* attribute will store the conceptual root or the word listed in dictionary as the lemma of the wordform. Figure 4.4 illustrates m-layer annotation.

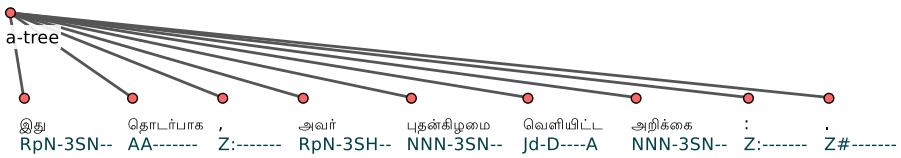


Figure 4.4: An example for morphological layer annotation

In the figure, there are two tokens displayed under each node. The text at the top of the node is the *form* or the exact word which appeared in the text. The text at the bottom (for example: AA - - - - - - -) of each node is the morphological tag of the wordform. The length of each morphological tag is 9 characters and each character position corresponds to some feature of a wordform. The first 2 positions

[6]Tree Editor (TrEd): `http://ufal.mff.cuni.cz/tred/`

in the morphological tag correspond to main POS and refined POS. Together they provide finer information than a main POS category. Thus, it is possible to train a POS tagger for fine-grained tagset or coarse-grained tagsets. This kind of tagging is known as positional tagging. Positional tagging is suitable for morphologically rich languages and has been successfully applied to languages such as Czech. Section 4.6 gives a detailed description about positional tagging and the tagset used to perform annotation for TamilTB.

4.5.2 Analytical layer (a-layer)

Analytical layer (a-layer) is used to annotate sentences at syntactic level. There are two phases in a-layer annotation: (i) capture dependency structure of a sentence in the form of tree and (ii) identify relationship between words or nodes in the tree. From m-layer, we know that each wordform corresponds to a node in the tree but they are without their parents assigned. The dependency structure is captured by hanging dependent nodes (words) under their governing nodes (words). Visually, dependent nodes are hanged as children of their governing nodes. There will be one extra node called *technical root* to which the predicate node and the terminal node (end of the sentence) will be attached. The sole purpose of the *technical root* is to have some tree level attributes such as tree identifier. Figure 4.5 illustrates the a-layer annotation of a sentence shown in Figure 4.3.

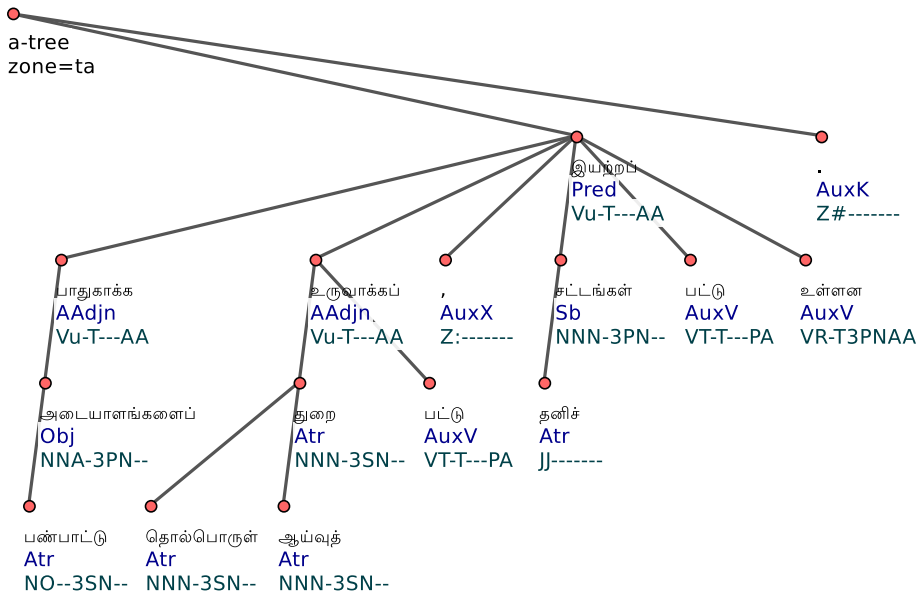


Figure 4.5: An example for analytical layer annotation

Edges between nodes indicate the relationship with which they are connected. In linguistic terms, it is called syntactic relation between governor and dependent. The dependency relationship between two nodes are stored in the attribute called

afun. In PDT style annotation, for technical reasons, the edges do not have *afun* attribute, instead the dependent nodes store the *afun* attribute. For example, the *afun* value of the word சட்டங்கள்/cattangkaL/laws is Sb, meaning subject, is the subject of the verb இயற்றப்/iyaRRap/'to enact'.

More detailed treatment of various syntactic relationships and a-layer annotation scheme is given in Section 4.7.

4.6 Morphological Annotation

This section gives a detailed description of annotation at word level. The annotation at this layer is roughly equivalent to part of speech (POS) tagging. This section begins with the introduction of positional tag, a format for tagging wordforms. Then the section introduces positional tagset for tagging Tamil data with examples.

4.6.1 Positional tagging

For m-layer annotation (aka POS tagging in general), we chose to annotate separate word tokens with morphological features in addition to single main POS. Having morphological features would be ideal and necessary for morphologically rich languages such as Tamil. Just by knowing those features it may be possible to identify certain syntactic phenomena (such as case markers can identify syntactic relations). So to include morphological information (such as case, person, number, etc.) in addition to main POS, we decided to adopt the positional tagging scheme which has been successfully applied for the Czech language.

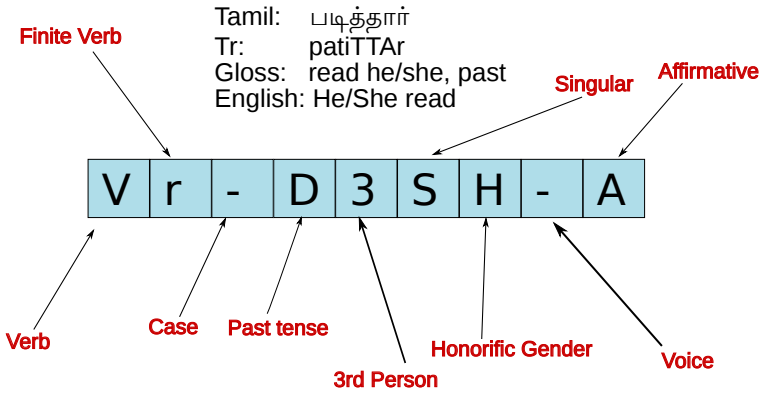


Figure 4.6: Positional tag

In positional tagging, each token (word) is tagged with a fixed length string. Each position or character in the tag represents a particular morphological feature of the token. The first position or character of the tag indicates the broad word category (such as noun, verb and so on) to which the token belongs and the 2nd position indicates the detailed POS (for example, what kind of verb? whether finite

or non-finite). The original Czech positional tagset includes 15 positions, and we designed a 9 positional tagset for Tamil. Figure 4.6 illustrates the tagging of a Tamil word using positional tagging scheme.

Position	Feature	#Possible Values
1	POS	14
2	Sub POS	42
3	Case	10
4	Tense	05
5	Person	04
6	Number	03
7	Gender	06
8	Voice	02
9	Negation	02

(a) Each position & num of possible values

Value	Description
A	Adverbs
C	Conjunctions
D	Determiners
I	Interjections
J	Adjectives
N	Nouns
P	Postpositions
Q	Quantifiers
R	Pronouns
T	Particles
U	Numerals
V	Verbs
X	Unknown
Z	Punctuations

(b) Main POS tags

Figure 4.7: Positional tagset

Figure 4.7 illustrates our positional tagset. The table in Figure 4.7(a) describes what each position occupies and the number of possible values they can take. The table in Figure 4.7(b) enumerates the possible values for *main POS* (first position). The tagset has been inspired from [Lehmann, 1989]. Our tagset includes separate entries for pronouns, numeral, interjections, particles and punctuations.

Position 2 - Sub POS

The *sub POS* corresponds to more finer version of *main POS*, i.e., it can give some more information about the *main POS*. For example, the *main POS* may just indicate that the wordform belongs to verb, the *sub POS* will further indicate that whether the verb is finite or non-finite. The *main POS* and *sub POS* together indicate the finer version of a part of speech. The other positions describe morphological details such as *tense, case, person, gender*, etc.

Table 4.3 sub POS values for adverbs, conjunctions, determiners and interjections.

Sub POS	Description	Example	Tag
A	adverbs, general	மறுபடியும்/maRupatiyum	AA- - - - - - -
C	conjunctions	மற்றும்/maRRum	CC- - - - - - -
		அல்லது/allaTu	CC- - - - - - -
D	determiners	இந்த/iwTa	DD- - - - - - -
		அந்த/awTa	DD- - - - - - -
		எந்த/ewTa	DD- - - - - - -
I	interjections	ஆஹா/AhA	II- - - - - - -

Table 4.3: SubPOS for adverbs (A), conjunctions (C), determiners (D) and interjections (I)

Table 4.4 shows sub POS values for adjectives.

Sub POS	Description	Example	Tag
J		ஆன/Ana	JJ- - - - - - -
d	adjectival participle	ஓடுகிற/OtukiRa	Jd-P- - - - A
		ஓடிய/Otiya	Jd-D- - - - A
		ஓடும்/Otum	Jd-F- - - - A

Table 4.4: SubPOS for adjectives (J)

Table 4.5 shows sub POS values for nouns.

Sub POS	Description	Example	Tag
N	common nouns	காற்று/kARRu	NNN-3SN- -
		காற்றை/kARRai	NNA-3SN- -
		காற்றல்/kARRAl	NNI-3SN- -
E	proper nouns	கௌதமா/kautamA	NEN-3SH- -
		பிராக்/pirAk	NEN-3SH- -
P	participial nouns	வாழ்ந்தவர்கள்/vAzwTavarkaL	NPN-3PA- -
O	oblique nouns	திருமண/TirumaNa	NO- - 3SN- -

Table 4.5: SubPOS for nouns (N)

Table 4.6 shows sub POS values for postpositions and quantifiers.

Sub POS	Description	Example	Tag
P	postpositions	மீது/mItu	PP- - - - - - -
		இடமிருந்து/itamiruwTu	PP- - - - - - -
Q	quantifiers	அதிகம், கொஞ்சம், மிகவும், முழு/aTikam, konjcam, mikavum, muzu	QQ - - - - - - -

Table 4.6: SubPOS for postpositions (P) and quantifiers (Q)

Table 4.7 shows sub POS values for pronouns.

Sub POS	Description	Example	Tag
p	personal pronouns	அவர்/avar	RpN-3SH- -
		இது/iTu	RpN-3SN- -
		நாம்/wAm	RpN-1PA- -
h	reflexive pronouns	தனது/TanaTu	RhG-3SA- -
		தன்னை/Tannai	RhA-3SA- -
B	general referential pronouns	யாரும்/yArum	RBN-3SA- -
		எதுவும்/eTuvum	RBN-3SN- -
		யாருக்கும்/yArukkum	RBD-3SA- -
F	specific indefinite referential pronouns	யாரோ/yArO	RFN-3SA- -
		எதுவோ/eTuvO	RFN-3SN- -
I	inerrogative pronouns	யார்/yAr	RiN-3SA- -
		எது/eTu	RiN-3SN- -
G	non specific indefinite pronouns	யாராவது/yArAvaTu	RGN-3SA- -
		எதாவது/eTAvaTu	RGN-3SN- -

Table 4.7: SubPOS for pronouns (R)

Table 4.8 shows sub POS for particles.

Sub POS	Description	Example	Tag
b	comparative particle	காட்டிலும், விட/kAttilum, vita	Tb - - - - - - -
d	adjectival participle, adjectivalized verbs	என்கிற, என்ற/enkiRa, enRa	Td-P- - - - A
e	interrogative particle	ஆ/A	Te - - - - - - -
g	adverb and Adjectival suffix	ஆக, ஆன/Aka, Ana	Tg - - - - - - -
k	intensifier particle	ஏ, ஏயே, தான்/E, EyE, TAn	Tk - - - - - - -
l	clitic	ஆவது/AvaTu	Tl - - - - - - -
m	clitic (*limit*)	மட்டும்/mattum	Tm - - - - - - -
n	complementizing nouns	போது/pOTu	Tn- - - - - - -
		உடன்/utan	Tn- - - - - - -
o	particle of doubt	ராமனே/rAmanO	To - - - - - - -
q	emphatic particle	ஏ,ஏயே/E, EyE and தான்/TAn	Tq - - - - - - -
Q	complementizer	என்பது/enpaTu	TQ - - - - - - -
s	concessive particle	ஓடியும்/Otiyum	Ts - - - - - - -
S	immediacy particle	வந்ததும்/vawTaTum	TS - - - - - - -
t	complementizer in verbal participle	என, என்று/ena, enRu	Tt - T - - - - A
v	inclusive particle	உம், கூட/um, kUta	Tv - - - - - - -
w	complementizer in conditional form	என்றால்/enRAl	Tw - T - - - - A

Table 4.8: SubPOS for particles (T)

Table 4.9 shows sub POS for numerals.

Sub POS	Description	Example	Tag
x	cardinals	ஒன்று, இரண்டு/onRu, iraNtu	Ux - - - - - - -
y	ordinals	ஒன்றும், இரண்டாம்/onRAm, iraNtAm	Uy - - - - - - -
=	digits	10, 20 ...	U = - - - - - - -

Table 4.9: SubPOS for numerals (U)

Table 4.10 shows sub POS for verbs.

Sub POS	Description	Example	Tag
j	imperative verb (lexical)	ஓடு, உதவு/Otu, uTavu	Vj - T2PAAA
r	finite verb (lexical)	ஓடுகிறவன், உதவுகிறவன்/OtukiRAn, uTavukiRAL	Vr - P3SMAA
t	verbal participle (lexical)	ஓடி, வந்து/Oti, vawTu	Vt - T - - - AA
u	infinitive (lexical)	ஓட, உதவ/Ota, uTava	Vu - T - - - AA
w	conditional verb (lexical)	ஓடினால், வந்தால்/OtinAl, vawTAl	Vw - T - - - AA
z	verbal nouns (lexical)	பிடிப்பது/pitippaTu	VzNF3SNAA
R	finite verb (auxiliary)	/iruwTAr	VR - D3SHAA
T	verbal participle (auxiliary)	கொண்டு/koNtu	VT - T - - - AA
U	infinitive (auxiliary)	பட்ட/patta	VU - T - - - AA
W	conditional (auxiliary)	பட்டால்/pattAl	VW - T - - - AA
Z	verbal nouns (auxiliary)	இருப்பது/iruppaTu	VZNF3SNAA

Table 4.10: SubPOS for verbs (V)

Table 4.11 shows sub POS for punctuations.

Sub POS	Description	Example	Tag
#	terminal symbol	.(dot)	Z# - - - - - - -
:	other symbols	*, , ? , -, (,), !*	Z: - - - - - - -

Table 4.11: SubPOS for punctuations (Z)

Position 3 - Case

Tamil case occupies third position in the positional tag. Table 4.12 lists Tamil *case markers* and the corresponding values for the 3^{rd} position of the positional tag.

Val.	Description	Example	Tag
A	Accusative	கட்சியை/katciyai/‘party’	NNA - 3SN - -
D	Dative	வீட்டுக்கு/vIttukku/‘to/for the house’	NND - 3SN - -
I	Instrumental	முயற்சியால்/muyaRciyAl/‘by the efforts’	NNI - 3SN - -
G	Genitive	அரசின்/aracin/‘of government’	NNG - 3SN - -
L	Locative	போரில்/pOril/‘in the war’	NNL - 3SN - -
N	Nominative	ஆண்டு/ANtu/‘year’	NNN - 3SN - -
S	Sociative	துணையோடு/TuNaiyOtu/‘with the help’	NNS - 3SN - -

Table 4.12: Tamil case

Position 4 - Tense

Table 4.13 shows values for tense.

Val.	Description	Example	Tag
D	past	கட்டினார்/kattinAr/‘built he’	Vr - D3SHAA
F	future	உதவும்/uTavum/‘it will help’	Vr - F3SNAA
P	present	செல்கிறார்/celkiRAr/‘he is going’	Vr - P3SHAA
T	tenseless	இல்லை/illai/‘exist not’	Vr - T3PNAA

Table 4.13: Tense

Position 5 - Person

Table 4.14 shows values for person.

Val.	Description	Example	Tag
1	1^{st} person	மேற்கொண்டேன்/mERkoNtEn/‘I undertook’	Vr - D1SAAA
2	2^{nd} person	அஞ்சுகிறீர்கள்/anjcukiRIrkaL/‘you fear’	Vr - P2PAAA
3	3^{rd} person	விவாதிக்கும்/vivATikkum/‘it will discuss’	Vr - F3SNAA

Table 4.14: Person

Position 6 - Number

Table 4.15 shows values for number.

Val.	Description	Example	Tag
P	plural	விவரங்கள்/vivarangkaL/‘details’	NNN - 3PN - -
S	singular	நியூஸிலாந்து/nyUSilAwTu/‘New Zealand’	NEN - 3SN - -

Table 4.15: Number

Position 7 - Gender

Table 4.16 shows values for gender.

Val.	Description	Example	Tag
F	feminine	வருவாள்/varuvAL/‘she will come’	Vr - F3SFAA
M	masculine	ஆடவனின்/Atavanin/‘man’s’	NNG - 3SM –
N	neuter	எடுத்தது/etuTTaTu/‘it took’	Vr- D3SNAA
H	honorific	அவர்/avar/‘he/she [polite]’	RpN - 3SH - -
A	animate (humans)	யார்/yAr/‘who?’	RiN - 3SA - -
I	inanimate (non humans)	*unused*	*unused*

Table 4.16: Gender

Position 8 - Voice

Table 4.17 shows values for voice.

Val.	Description	Example	Tag
A	active	எடுத்தது/etuTTaTu/‘it took’	Vr- D3SNAA
P	passive	படுகிறது/ patukiRaTu /‘being [*verb*]...’	VR - P3SNPA

Table 4.17: Voice

Position 8 - Negation

Table 4.18 shows values for negation.

Val.	Description	Example	Tag
A	affirmative	பின்பற்ற/pinpaRRa/‘to follow’	Vu - T - - - AA
N	negation	முடியாது/mutiyATu/‘cannot’	VR - T3SN -A

Table 4.18: Negation

4.6.2 Annotation of pronouns

Tamil pronouns are one of the closed but in combination with clitics produce various derived pronouns. In this section, we list all possible pronouns (in their combination) with their tags. More information about Tamil pronouns can be obtained from [Lehmann, 1989]. The listing of tags for all possible pronouns will be useful in annotation task.

Singular referential definite (personal) pronouns

Table 4.19 shows values for personal pronouns.

Per./Num.	Pronoun	Tag
1^{st}/sg	நான்/wAn/I)	RpN - 1SA - -
1^{st}/pl	நாம்/wAm/‘we, exclusive’	RpN - 1PA - -
	நாங்கள்/wAngkaL/‘we, inclusive’	RpN - 1PA - -
2^{nd}/sg	நீ/wI/you	RpN - 2SA - -
	நீங்கள்/wIngkaL/‘you, honorific, singular’	RpN - 2SH - -
2^{nd}/pl	நீங்கள்/wIngkaL/‘you, plural’	RpN - 2PA - -
3^{rd}/sg	அவன்/avan/‘that one - he’	RpN - 3SM - -
	இவன்/ivan/‘this one - he’	RpN - 3SM - -
	அவள்/avaL/‘that one - she’	RpN - 3SF - -
	இவள்/ivaL/‘this one - she’	RpN - 3SF - -
	அது/aTu/‘that one - it’	RpN - 3SN - -
	இது/iTu/‘this one - it’	RpN - 3SN - -
	அவர்/avar/‘that one - he/she hon.’	RpN - 3SH - -
	இவர்/ivar/‘this one - he/she hon.’	RpN - 3SH - -
3^{rd}/pl	அவை/அவைகள் /avai/avaikaL/‘those ones’	RpN - 3PN - -
	இவை/இவைகள்/ivai/ivaikaL/‘these ones’	RpN - 3PN - -
	அவர்கள்/avarkaL/‘those people’	RpN - 3PA - -
	இவர்கள்/ivarkaL/‘these people’	RpN - 3PA - -

Table 4.19: Personal pronouns

Non-referential (interrogative) pronouns

Table 4.20 shows values for interrogative pronouns.

Pronoun	Tag
யார்/hyAr/who	RiN - 3SH - -
என்ன/enna/what	RiN - 3SN - -
எவன்/evan/'which male person'	RiN - 3SM - -
எவள்/evaL/'which female person'	RiN - 3SF - -
எது/eTu/'which thing'	RiN - 3SN - -
எவர்/evar/'which male/female person'	RiN - 3SH - -
எவர்கள்/evarkaL/'which persons'	RiN - 3PA - -
எவைகள்/evai(kaL)/'which things'	RiN - 3PN - -

Table 4.20: Interrogative pronouns

General referential pronouns

In the case of general referential pronouns, the particle *-um* is added to interrogative pronouns which results in the addition of referential property to interrogatives. The resultant words will have a meaning of 'anyone, anybody or anything' in English. Table 4.21 lists general referential pronouns.

Pronoun	Tag
யாரும்/yAr**um**/anyone	RBN - 3SH - -
எவனும்/evan**um**/'anyone, male person'	RBN - 3SM - -
எவளும்/evaL**um**/'anyone female person'	RBN - 3SF - -
எதுவும்/eTuv**um**/anything	RBN - 3SN - -
எவரும்/evar**um**/'anyone, male/female person'	RBN - 3SH - -
எவர்களும்/evarkaL**um**/'any persons'	RBN - 3PA - -
எவைகளும்/evaikaL**um**/'anything, plural'	RBN - 3PN - -
எவையும்/evaiy**um** /'anything, plural'	RBN - 3PN - -

Table 4.21: General referential pronouns

Specific indefinite referential pronouns

Table 4.22 shows values for specific indefinite referential pronouns.

Pronoun	Tag
யாரோ/yAr**O**/someone	RFN - 3SH - -
எவனே/evan**O**/'some male person'	RFN - 3SM - -
எவளோ/evaL**O**/'some female person'	RFN - 3SF - -
எதுவோ/eTuv**O**/something	RFN - 3SN - -
எவரோ/evar**O**/'some male/female person'	RFN - 3SH - -
எவர்களோ/evarkaL**O**/'someone, plural'	RFN - 3PA - -
எவையோ/evaiy**O**/'something, plural'	RFN - 3PN - -

Table 4.22: Specific indefinite referential pronouns

Non-specific indefinite referential pronouns

Table 4.23 shows values for non-specific indefinite referential pronouns.

Pronoun	Tag
யாராவது/yAr**AvaTu**/'someone or other'	RGN - 3SH - -
எவனாவது/evan**AvaTu**/'some male person or other'	RGN - 3SM - -
எவளாவது/evaL**AvaTu**/'some female person or other'	RGN - 3SF - -
எதாவது/eT**AvaTu**/'something or other'	RGN - 3SN - -
எவராவது/evar**AvaTu**/'some male/female person or other'	RGN - 3SH - -
எவர்களாவது/evarkaL**AvaTu**/'someone, or other (plural)'	RGN - 3PA - -
எவைகளாவது/evaikaL**AvaTu**/'something, or other (plural)'	RGN - 3PN - -
எவையாவது/evaiy**AvaTu**/'something, other (plural)'	RGN - 3PN - -

Table 4.23: Non-specific indefinite referential pronouns

4.7 Syntactic Annotation

This section will give a detailed description about how annotation takes place at the syntactic level. The syntactic annotation consists of two phases: (i) identifying the structure (dependency) of the sentence in the form of dependency tree and (ii) identifying the dependency relations and assigning those relations to edges in the dependency tree structure.

4.7.1 Identifying the structure

The structure of the sentence is identified manually by attaching the dependent nodes to the governing nodes. In the sentential structure, the head of the sentence

will be predicate, and the predicate will have arguments (noun phrases, adverbials) as their children. The objective of this step would be, identifying the predicate rooted structure and attaching to the technical root (AuxS, defined below) of the tree. The end of the sentence will also be attached to the technical root. Once the structure is identified, all the edges have to be labeled with their relations. For technical reasons, the relation between the dependent and the governing node is stored as an a-layer attribute of the dependent node. The attribute is called *'afun'*. The following sections explains each dependency relation in detail.

4.7.2 Dependency relations

According to PDT naming convention, dependency relations are also called as *analytical functions* or *afuns*. The documentation uses these names interchangeably.

#	afun	description	comments
1	AAdjn	Adverbial Adjunct	optional adverbs or adverbial phrases
2	AComp	Adverbial Complement	obligatory adverbs or obligatory adverbial phrases
3	Apos	Apposition	heads of apposition clauses - clauses attaching to என்ற/enRa
4	Atr	Attribute	noun modifiers
5	AdjAtr	Adjectival participial	adjectivalized verbs or relative clauses
6	AuxA	Determiners	demonstrative pronouns (இந்த/iwTa/`this')
7	AuxC	Subordinating conjunctions	subordinating conjunctions - என்று/enRu, என/ena, ஆக/Aka
8	AuxG	Symbols other than comma	-, ", ', $, rU., (,), [,]
9	AuxK	Sentence termination symbols	:, . , ?
10	AuxP	Postpositions	மீது/mITu/`on', பற்றி/paRRi/`about'
11	AuxS	Technical root	technical root
12	AuxV	Auxiliary verb	உள்/uL, கொண்டு/koNtu, இரு/iru, etc.
13	AuxX	Comma (not coordination)	,
14	AuxZ	Emphasis words or particles	தான்/TAn, உம்/um, ஏ/E
15	CC	Part of a word	கிளர்ந்து எழுந்து/kiLarwTu ezuwTu/'having risen'. *kiLarwTu* will be labeled as *CC*.
16	Comp	Non verbal complements	obligatory attachments to non verbs. Example: "belongs to the batch of 1977"
17	Coord	Coordination node	மற்றும்/maRRum/`and', உம்/um
18	Obj	Object (both direct and indirect)	usually nouns with accusative case
19	Pnom	Predicate nominal	nominals that act as main verbs
20	Pred	Predicate	main verb (usually finite) of a sentence
21	Sb	Subject	subject

Table 4.24: Dependency relations *(afuns)*

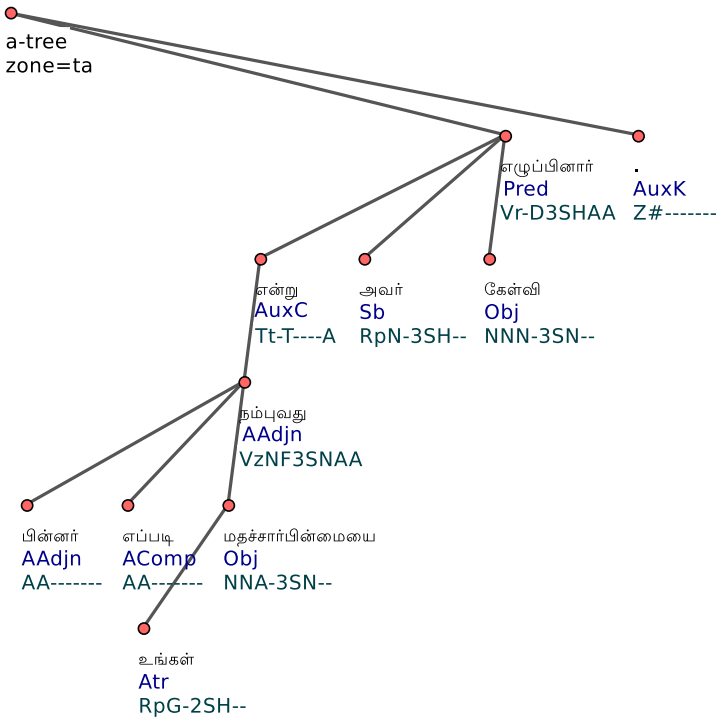
4.7.3 Detailed description of *afuns*

In this sub section, we describe each dependency relation in detail with an example annotation. For the list of dependency relations or *afuns*, please refer Table 4.24. Each dependency relation is explained with an analytical tree and the sentence it represents. The sentence is shown in Tamil script and it's transliteration, gloss and

the actual English translation in that order. The word shown in bold receives the dependency relation that is being explained.

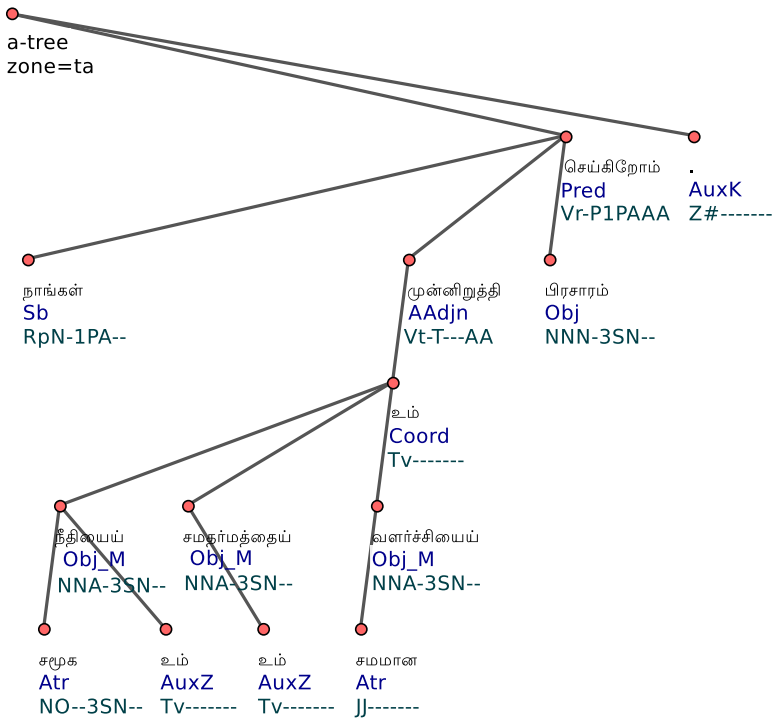
Afun: *AAdjn*

The *AAdjn* relation is used to mark adverbial adjuncts. Adverbial adjuncts are optional adverbial phrases, prepositional phrases, clauses or simple adverbs modifying the verbs. Figures 4.8 & 4.9 show examples for *AAdjn* relation. In Figure 4.8, the adverb பின்னர்/pinnar/'later, after' has been labeled with *AAdjn* relation.



Tamil: பின்னர் எப்படி உங்கள் மதச்சார்பின்மையை நம்புவது என்று கேள்வி எழுப்பினார் .
Tr: **pinnar** eppati ungkaL maTaccArpinmaiyai wampuvaTu enRu kELvi ezuppinAr .
Gloss: **then** how your secularism to-believe - question raised-he .
English: "**Then** how to believe your secular credentials ", he raised a question .

Figure 4.8: *AAdjn:* Adverbial adjunct



Tamil: நாங்கள் சமூக நீதியை உம் சமத்துவத்தை உம் சமமான வளர்ச்சியை உம் முன்னிறுத்தி பிரசாரம் செய்கிறோம் .
Tr: wAngkaL camUka wlTiyai um camaTarnaTTai um camamAna vaLarcciyai um **munniRuTTi** piracAram ceykiROm .
Gloss: we social justice - equality - equal development - **by-putting-forward** campaign doing-we .
English: We are doing campaign **by putting forward** social justice, equality and equal development .

Figure 4.9: *AAdjn:* Adverbial adjunct

In Figure 4.9, the adverbial phrase **முன்னிறுத்தி**/munniRuTTi/'by putting forward ...' has been labeled with *AAdjn* relation. The *AAdjn* label is determined by whether excluding the adverbial adjunct affects the meaning of the sentence. If it does not (only provides extra information about the sentence), the head of the phrase is assigned *AAdjn* relation.

Afun: *AComp*

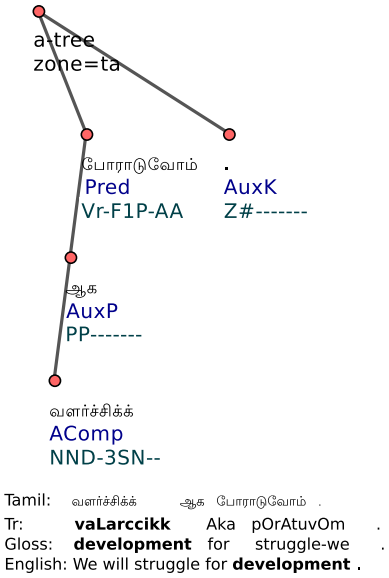


Figure 4.10: *AComp:* Adverbial complement

The *AComp* relation is used to mark obligatory adverbials or adverbial complements in the dependency structure. The context of occurrence of *AComp* relation is same as that of *AAdjn* relation. The only difference is that the *adverbial adjuncts (AAdjn)* are optional elements in the sentential structure whereas *adverbial complements (AComp)* are obligatory elements in the sentence structure. While doing annotation, this relationship is determined by whether the adverbial structure is required to complete the sentence. If the removal of the adverbial structure does not affect the sentence as a whole, then it is labeled as *AAdjn* otherwise it will be labeled as *AComp*. Figure 4.10 an example annotation for *AComp*.

Afun: *Apos*

The adjectival clauses headed by *enRa* are appositional clauses. The entire finite clause will be attached to *enRa* which will act as a modifier to the following noun phrase. The clausal head which is attached to *enRa* will receive Apos label. The following Figure 4.11 illustrates the labeling of *Apos*.

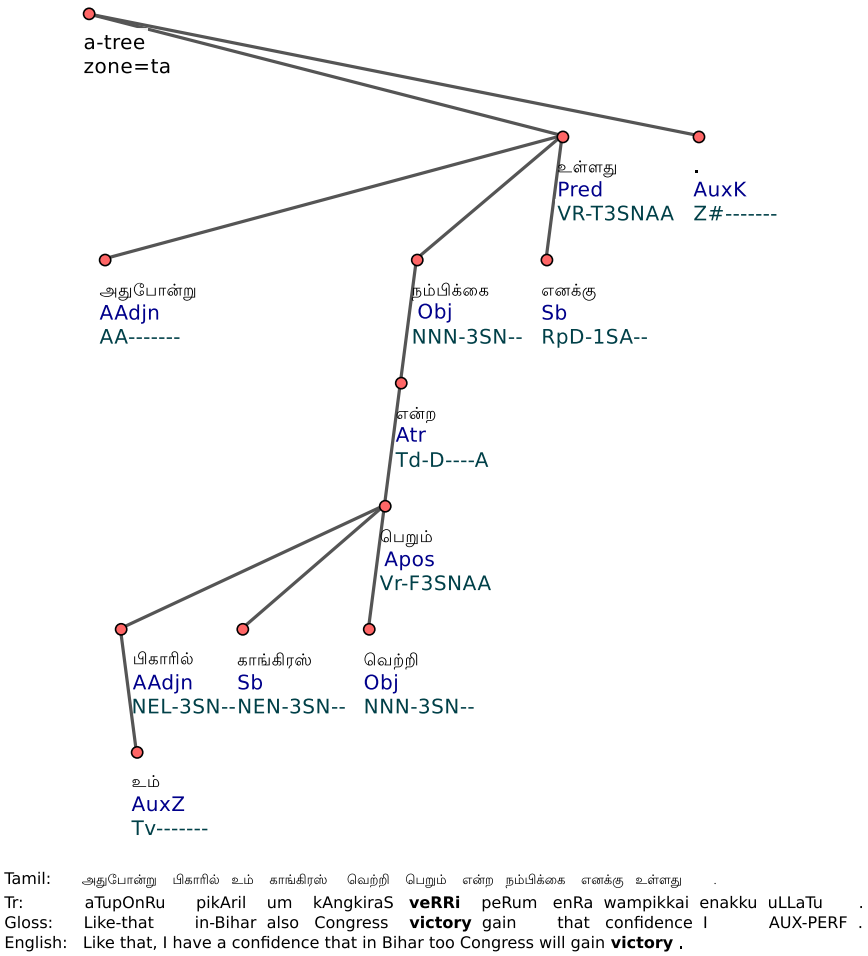
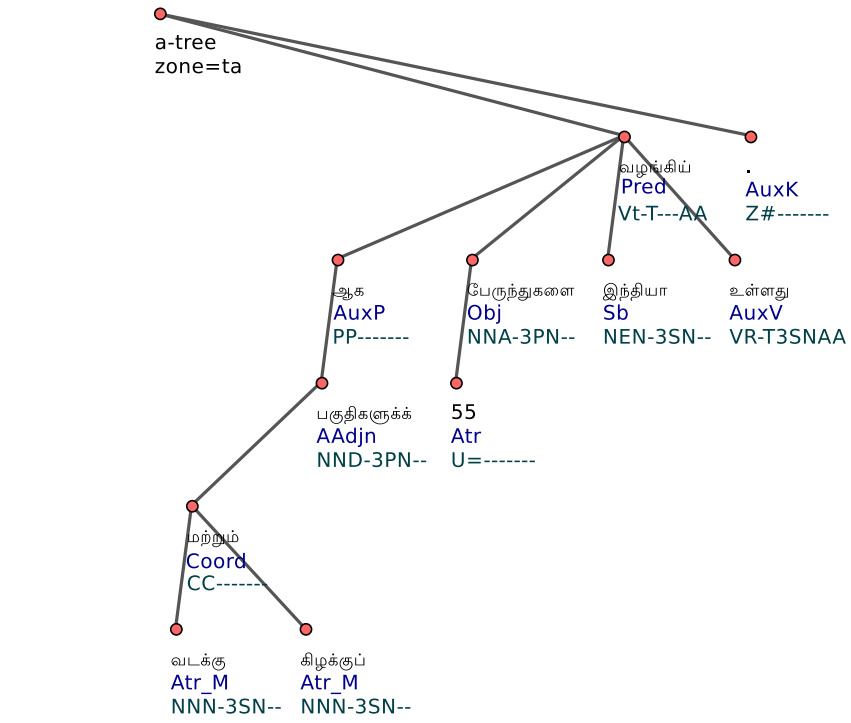


Figure 4.11: *Apos:* Apposition

Afun: *Atr*

Attribute [Hajič et al., 2006] is a sentence member which depends on noun and closely determines its meaning. Original PDT annotation differentiates "agreeing" and "non-agreeing" attribute. But, since Tamil does not have any agreement between nouns and their modifiers, all noun modifiers receive the afun label *Atr*. The noun modifiers include nouns (except the head noun) in noun compounds, adjectives, numerals and adjectival participles. Figure 4.12 shows the usage of afun *Atr*.



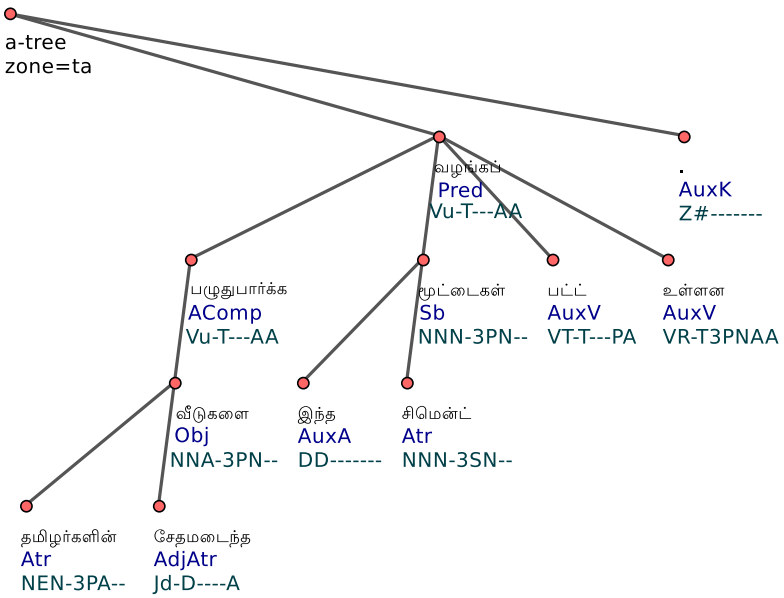
Tamil: வடக்கு மற்றும் கிழக்குப் பகுதிகளுக்க் ஆக 55 பேருந்துகளை இந்தியா வழங்கிய் உள்ளது .
Tr: **vatakku** maRRum **kizakkup** pakuTikaLukk Aka **55** pEruwTukaLai iwTiyA vazangkiy uLLaTu .
Gloss: **north** and **east** regions for **55** buses India give AUX-PERF .

English: India has provided 55 buses to north and eastern regions .

Figure 4.12: *Atr:* Attribute

Afun: *AdjAtr*

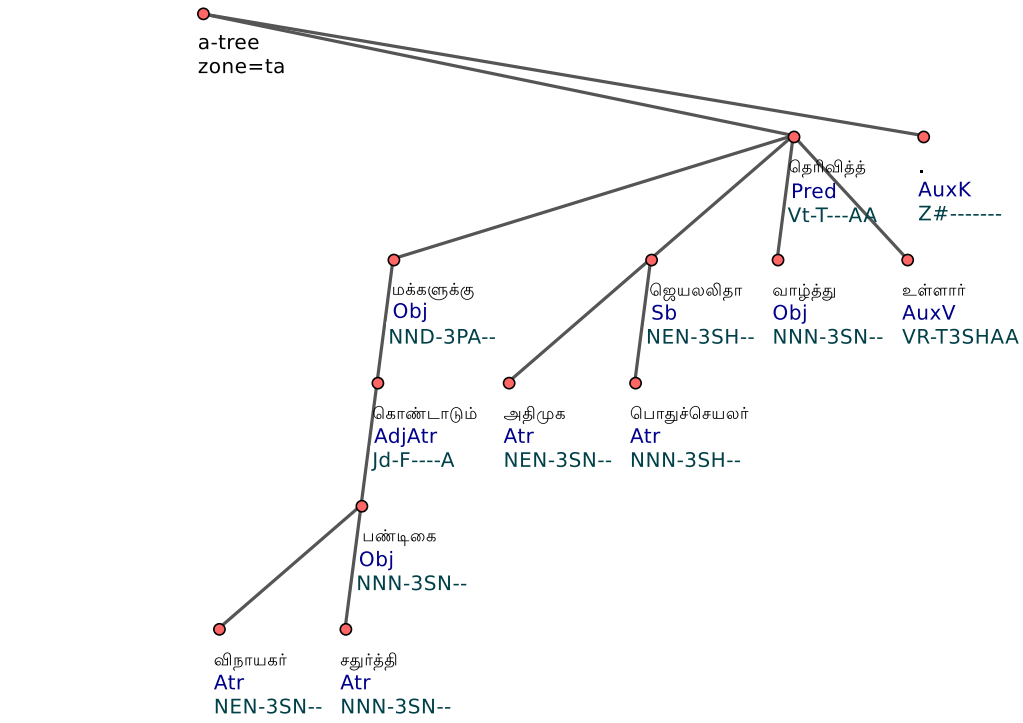
AdjAtr label is used to mark the adjectival clauses, adjectivalized verbs or adjectival participials. They are equivalent to -ing, -ed (singing girl, departed train) forms in English. Verbs in Tamil can be adjectivalized for all three tenses, and they take appropriate tags depending on the word form features. Figures 4.13 and 4.14 show examples for *AdjAtr* labeling.



Tamil:	தமிழர்களின்	சேதமடைந்த	வீடுகளை	பழுதுபார்க்க	இந்த	சிமெண்ட்	மூட்டைகள்	வழங்கப்	பட்ட்	உள்ளன	.	
Tr:	TamizarkaLin	**cETamataiwTa**	vItukaLai	pazuTupArkka	iwTa	cimeNt	mUttaikaL	vazangkap	patt	uLLana	.	
Gloss:	Tamil's	**ruined**	houses	to-repair	these	cement	-	to-provide	AUX-PASS	AUX-PERF	.	

English: These cement stocks have been provided to repair the ruined houses of Tamils.

Figure 4.13: *AdjAtr:* Adjectival attribute



Tamil:	விநாயகர்	சதுர்த்தி	பண்டிகை	கொண்டாடும்	மக்களுக்கு	அதிமுக	பொதுச்செயலாளர்	ஜெயலலிதா	வாழ்த்து	தெரிவித்த்	உள்ளார்	.	
Tr:	viwAyakar	caTurTTi	paNtikai	**koNtAtum**	makkaLukku	aTimuka	poTucceyalALar	jeyalaliTA	vAzTTu	TeriviTT	uLLAr	.	
Gloss:	Vinayakar	Chaturthi	festival	**celebrating**	people-DAT	ADMK	general-secretary	Jeyalalitha	wishes	express	AUX-PERF	.	

English: ADMK General Secretary Jeyalalitha expressed her wishes to people who are celebrating Vinayakar Chaturthi festival .

Figure 4.14: *AdjAtr:* Adjectival attribute

Afun: *AuxA*

Tamil has two demonstrative determiners corresponding to 'this' and 'that' in English. At present, the question word *ewTa* ('which') is also tagged as determiner. Sometimes determiners occur as a prefix in a contracted form to the following noun or noun phrases. During the tokenization phase, these demonstrative suffixes are separated from noun or noun phrases, and thereafter they are considered separate tokens. The determiners are *iwTa* ('this'), *awTa* ('that') and *ewTa* ('which'), and the corresponding contracted determiners are *i, a* and *e*. Due to orthographic rules, the first letter of the following noun phrase is added to the contracted form. Figure 4.15 illustrates the usage of all determiners.

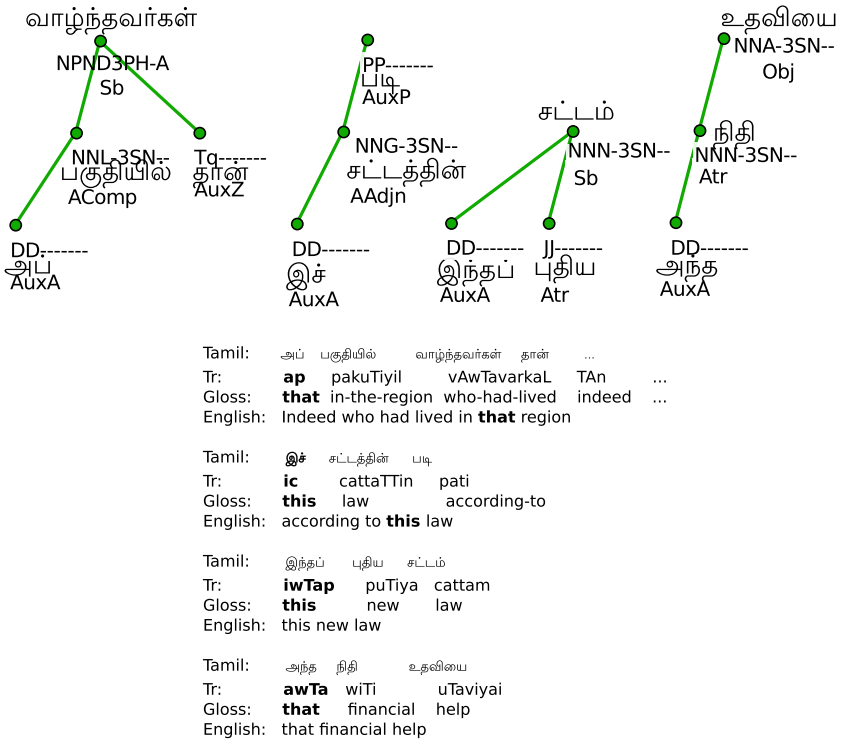
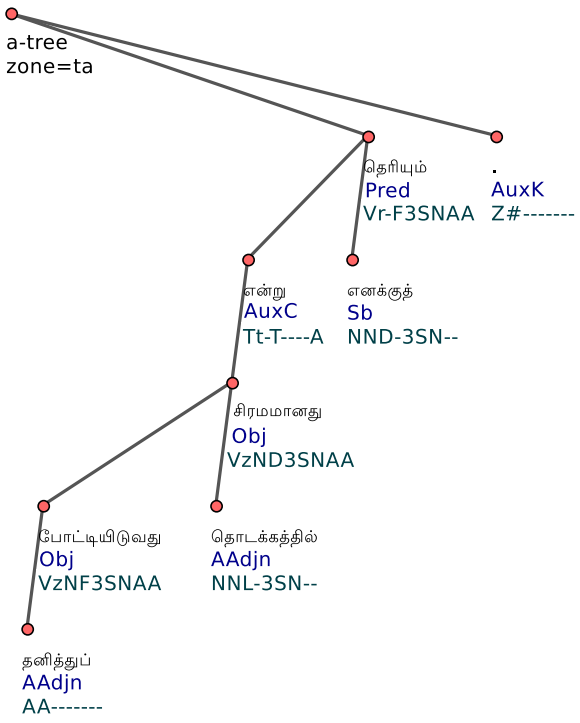


Figure 4.15: *AuxA:* Determiners

Afun: *AuxC*

In Tamil, embedding or adjoining of clauses are performed either by morphologically marking the clause or by using separate words. When separate words are used, they function similar to that of subordinating conjunction words in other languages such as English. These separate words are called complementizers in Tamil. Complementizers can be verbs, nouns or postpositions after nominalized clauses. There are three complementizing verbs - என்/en/'say', போல்/pOl/'seem' and ஆகு/Aku/'become'. They have grammatical function during embedding of clauses, otherwise they retain their lexical meanings. The following list provides some of the noun complementizers - போது/pOTu/'time', முன்/mun/'before', பிறகு/piRaku/'after', உடன்/utan/'immediacy' and வரை/varai/'as long as'. The postpositions can also be interpreted as subordinating conjunction words when they are preceded by nominalized clauses. Refer [Lehmann, 1989] for detailed treatment of how complementizers work in Tamil. *AuxC* annotations are shown in Figures 4.16 and 4.17.



Tamil:	தனித்துப்	போட்டியிடுவது	தொடக்கத்தில்	சிரமமானது	என்று	எனக்குத்	தெரியும்	.
Tr:	TaniTTup	pOttiyituvaTu	TotakkaTTil	ciramamAnaTu	**enRu**	enakkuT	Teriyum	.
Gloss:	alone	competing	in-the-beginning	difficult	**that**	I	know	.

English: I know **that** it is difficult to compete alone in the beginning .

Figure 4.16: *AuxC:* Subordinating conjunctions

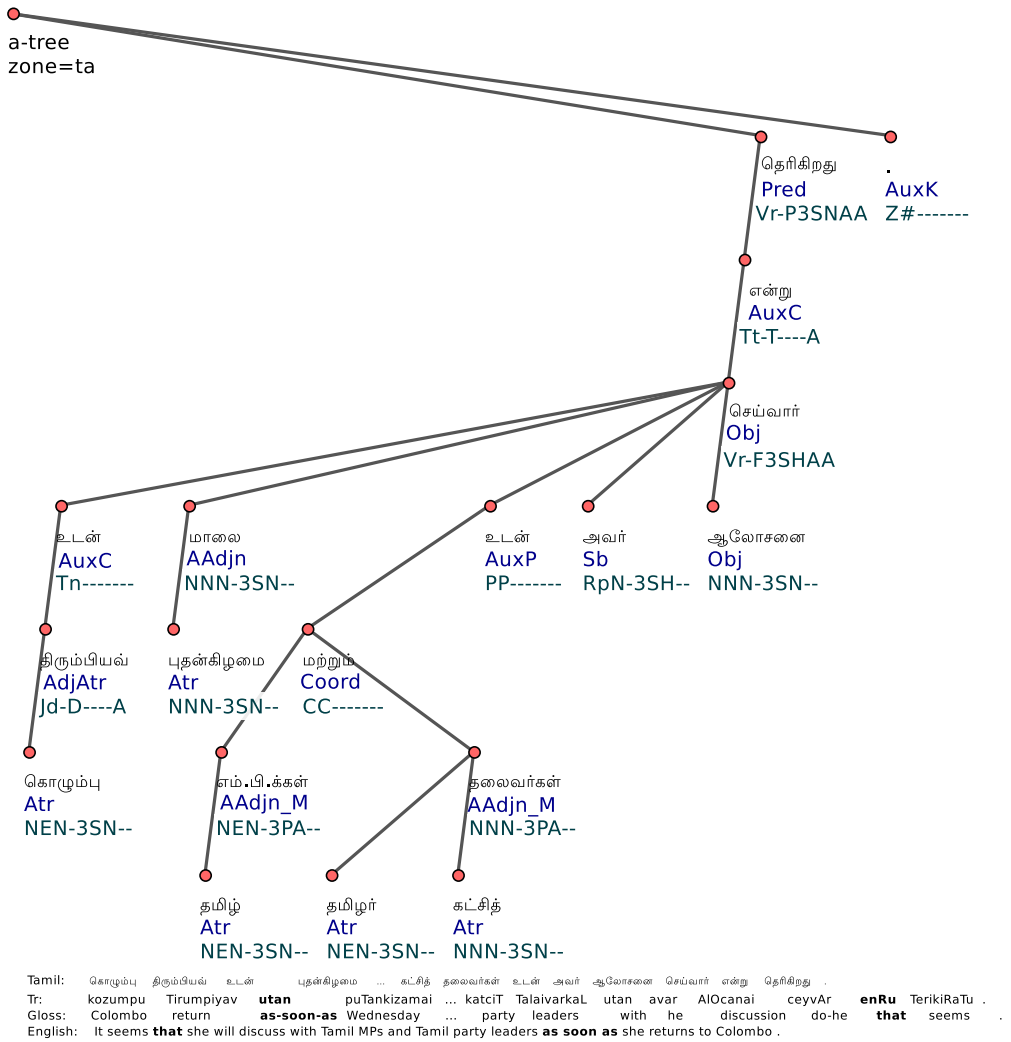


Figure 4.17: *AuxC:* Subordinating conjunctions

Afun: *AuxG*

The symbols other than sentence boundary and comma are labeled with *AuxG* afun. The *AuxG* symbols include pairs of symbols such as *(, }, [, “, ‘* and other symbols such as !, , #,$. Figure 4.18 shows an example for *AuxG*.

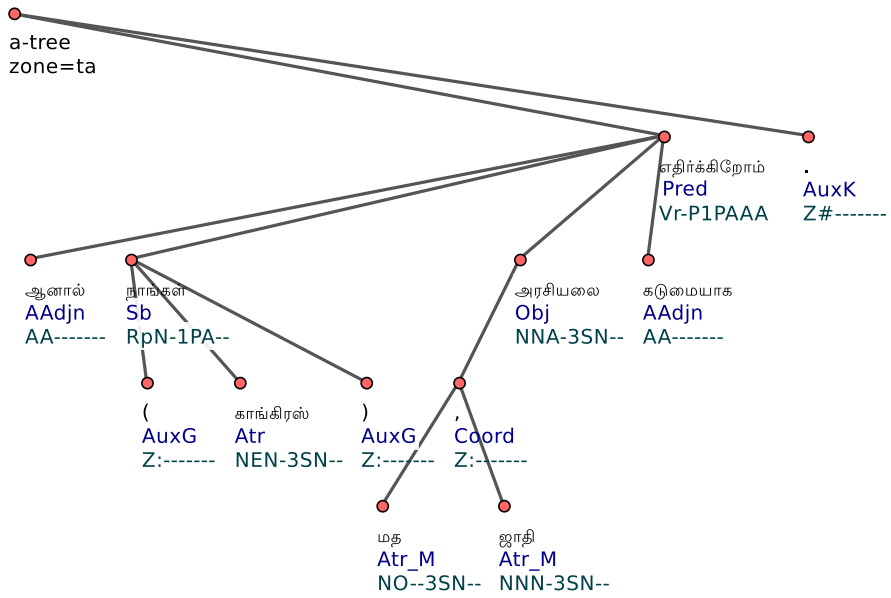


Figure 4.18: *AuxG:* Symbols

Afun: *AuxK*

The *AuxK* afun is assigned to sentence termination symbols. The symbols *.*, *?* and *:* are considered as sentence terminals and they are expected at the end of a sentence. *AuxK* relation is shown in Figure 4.19.

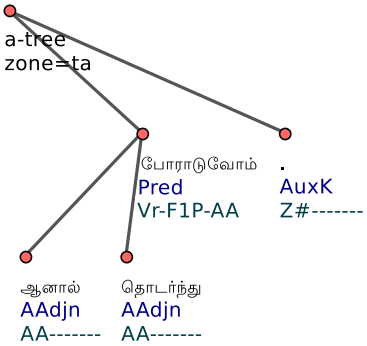


Figure 4.19: *AuxK:* Sentence termination symbol

Afun: *AuxP*

AuxP is used to mark the postpositions (heads) of the postpositional phrases. The postposition will recieve the *AuxP* label and the element attached to *AuxP* will receive the afun *(AAdjn , AComp, Atr, Comp)* according to their context of occurrence. Figures 4.20 & 4.21 show examples for *AuxP*.

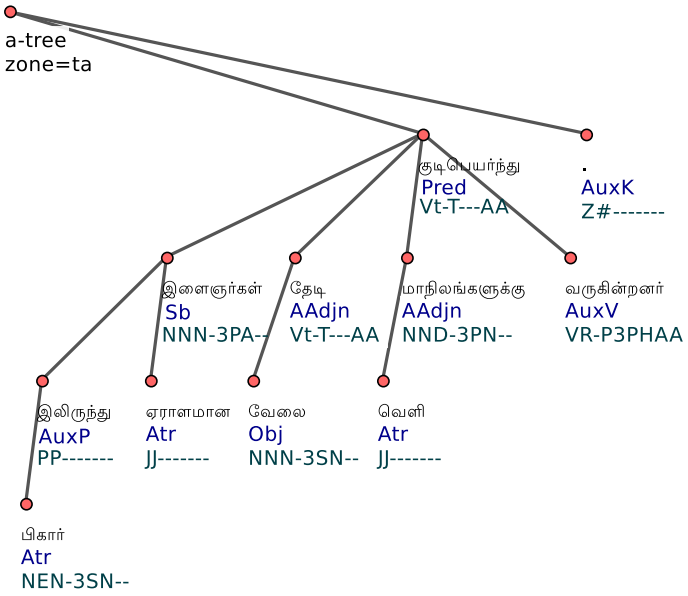


Figure 4.20: *AuxP:* Postpositions

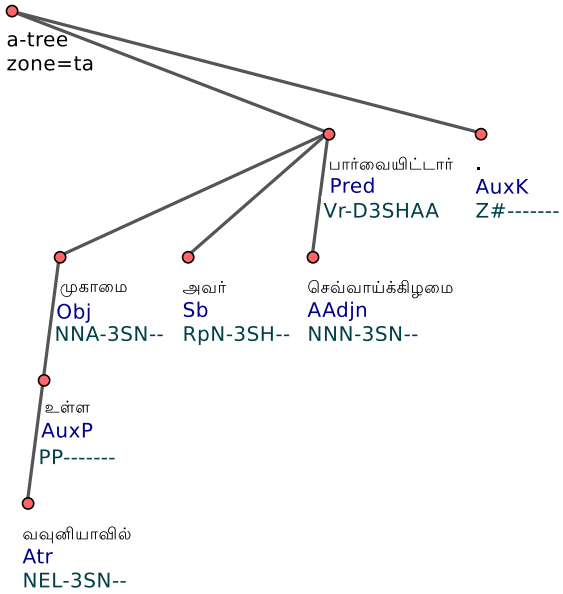


Figure 4.21: *AuxP:* Postpositions

Afun: *AuxS*

AuxS is used to label the technical root of a tree. In Figure 4.21, the technical root (shown as *StaA*) is the root of the tree structure. To this node, the predicate and the sentencing ending node will be attached.

Afun: *AuxV*

Auxiliary verbs are assigned the afun *AuxV*. In compound verb constructions, the auxiliary verb will be hanged under lexical verb. All auxiliary words including passive constructions will receive the afun *AuxV*. In Figure 4.22, there are two auxiliary verbs படு/patu/'experience' and உள்/uL/'exist'. The auxiliary படு/patu/'experience' and உள்ளது/uLLaTu/'exist' are labeled with *AuxV*. The lexical verb receive the label *Pred* if there is only one clause in the sentence, otherwise the label of the lexical verb depends on the upper clauses.

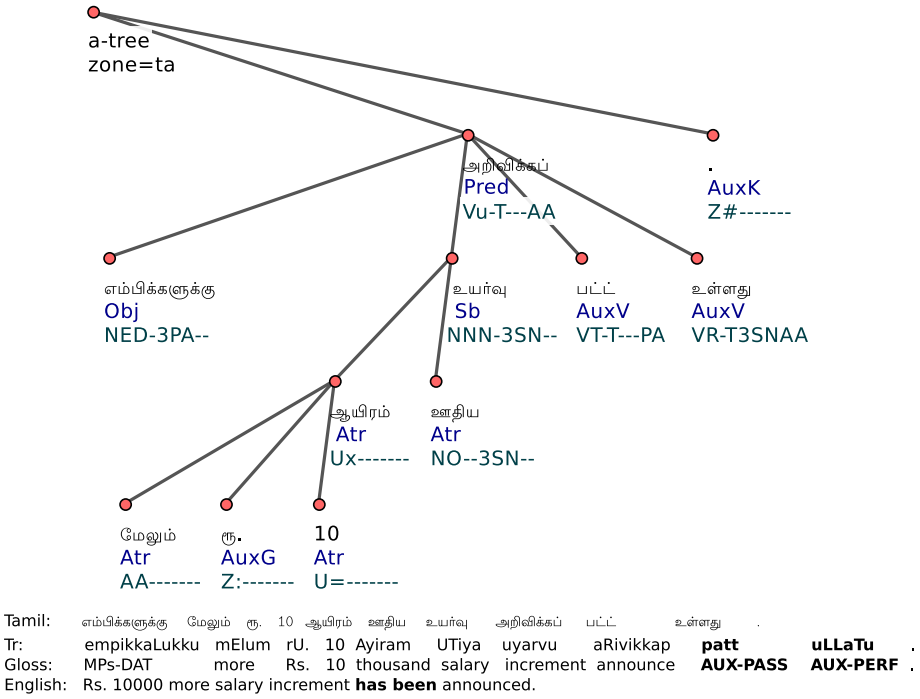


Figure 4.22: *AuxV:* Auxiliary verb

Afun: *AuxX*

AuxX afun is used to label commas. All commas except the commas which act as coordination head is labeled with *AuxX*. Figure 4.23 shows how *AuxX* has been annotated in the coordination structure.

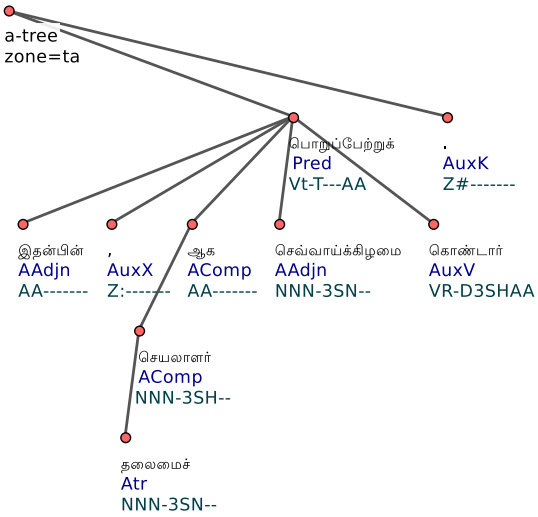


Figure 4.23: *AuxX:* Commas

Afun: *AuxZ*

The clitics *um, E, TAn, mattum* will receive the afun *AuxZ*. Figures 4.24 & 4.25 illustrate the labeling of *AuxZ*. The afun *AuxZ* corresponds to emphasizing words in English such as ('just', 'only', 'indeed'). These are not separate words but clitics in Tamil. Among the clitics, *um* has various semantics functions including acting as *coordination head*. In Figure 4.24, the clitic *um* ('also') adds an inclusive meaning to the sentence. In Figure 4.25, the clitic *TAn* ('only') emphasizes the entire clause.

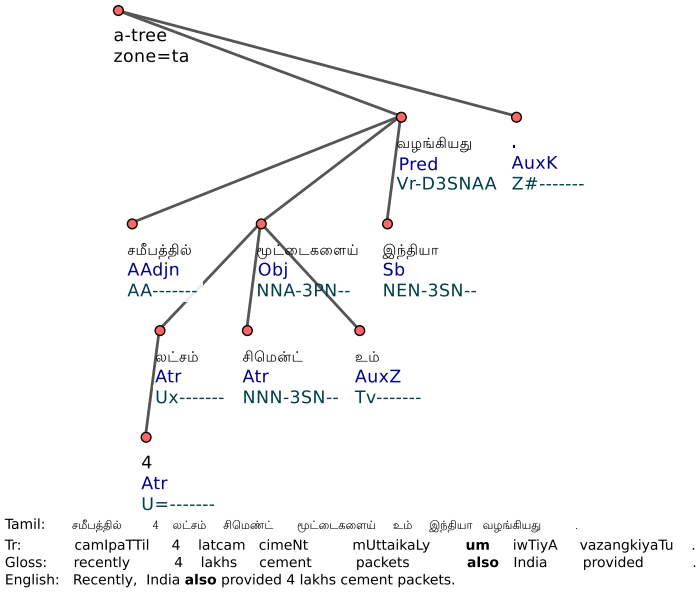


Figure 4.24: *AuxZ:* Emphasis

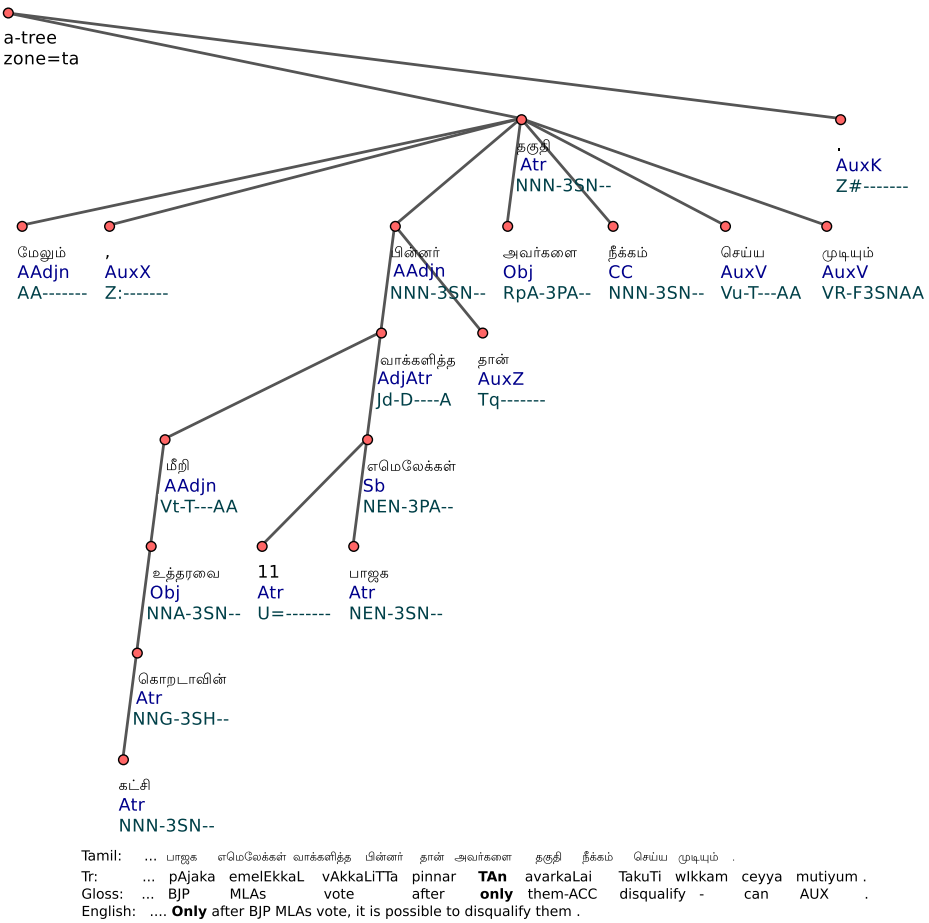


Figure 4.25: *AuxZ:* Emphasis

Afun: *CC*

The label *CC* is used to mark in places where a single lexical unit is composed of multiple words. The *CC* relation of a word would indicate that the current word together with it's parent word form a single lexical unit. In Tamil, a single action verb can be split into multiple words. But during annotation, the problem arises as to which part of the word sequence the arguments of that lexical unit should be attached. To resolve this issue, we treat the first word as having the lexical meaning and the remaining to be children of the first word. In that case, the first word would receive the afun corresponding to the entire lexical unit and the remaining words would receive the label *CC*. For example, Figure 4.26 indicates that the Tamil verb *veRRipeRu* ('to win') is splitted into 2 words - *veRRi peRu*. The first word *veRRi* receives the afun which is meant for the entire lexical unit comprising the following words with afun *CC*.

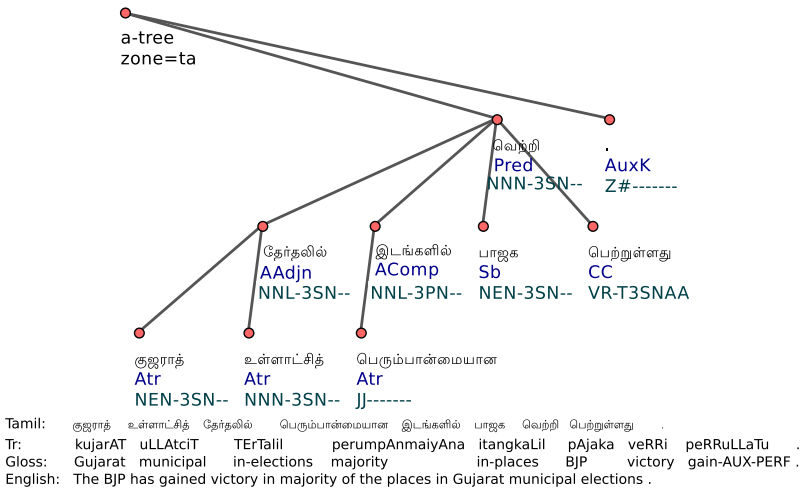


Figure 4.26: *CC:* Marker for a multi word sequence

Afun: *Comp*

Comp label is used to mark the obligatory element not attaching to verbs. For example, consider the phrase 1200க்க் உம் மேற்பட்ட பொதுமக்கள் உயிரிழந்த் உள்ளனர்/1200kk um mERpatta poTumakkaL uyirizawT uLLanar/(‘more than 1200 people have been died’), in that phrase, *1200kk* occurs as an obligatory argument to *mERpatta* (‘more than’). So, *1200kk* will be labeled with *Comp*. Even nouns (not modifiers) which obligatorily attach to other nouns are labeled with *Comp* afun. Other occurrences of *Comp* is when postpositional phrase (PP phrase) attaches to a noun phrase. The postpositional head will receive *AuxP* label whereas the head noun phrase of the PP phrase will receive *Comp* label. Figure 4.27 illustrates the labeling of *Comp* relation.

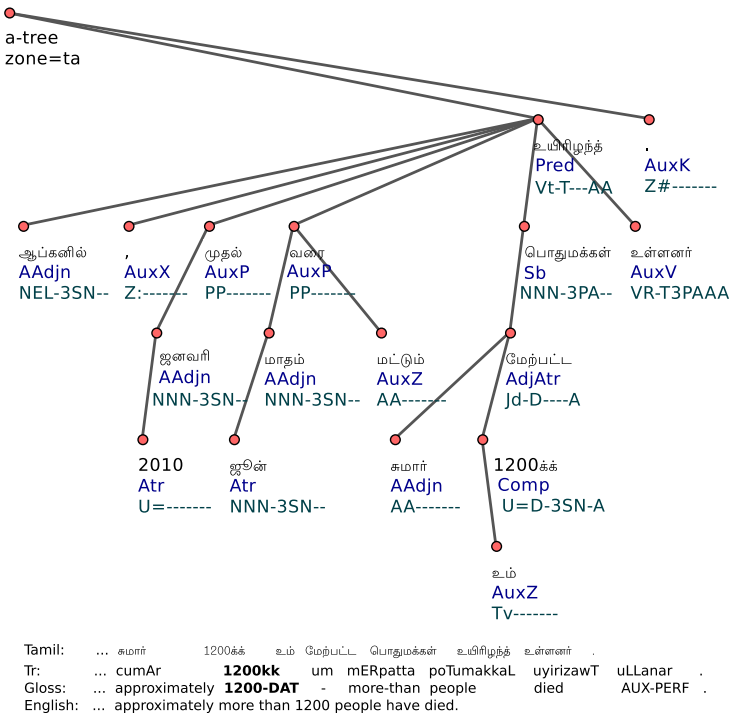


Figure 4.27: *Comp:* Complement (other than verbs)

Afun: *Coord*

Coordination is one of the complex phenomena in Tamil. Coordination conjunction in Tamil can be performed using at least 2 different ways. In the first method (for 'and' coordination), all conjoining elements adds the inclusive particle *um* ('also') at the end of the word form. Thus in this method, all conjoining elements possess the suffix (um) which would indicate the coordination is taking place. Moreover, the `` `is_member' `` attribute of the conjoining elements will be set to 1. The separator (comma) between elements is optional. It is perfectly legitimate if there is no comma between any of the conjoining elements.

Second method for Tamil 'and' coordination is similar to English style 'and' coordination. The conjunction word *maRRum* ('and') is added between conjoining elements. If there are more than 2 elements, then *maRRum* ('and') will be added just before the last conjunct. The other elements will be separated by comma. Again, the `` `is_member' `` attribute of the conjoining elements will be set to 1.

The 'or' coordination is performed in a similar way for both the methods. For the first method, the suffix *-O* is added to all conjuncts, and for the second method, the conjunction word *allaTu* ('or') is added between the last 2 conjoining elements. The remaining elements will be separated using comma.

Apart from the above two main methods, the coordination can be done via with just commas. In that case, the comma between conjuncts will act as coordination

head. Figures 4.28, 4.29 and 4.30 illustrates how coordination head is marked in the above mentioned scenarios.

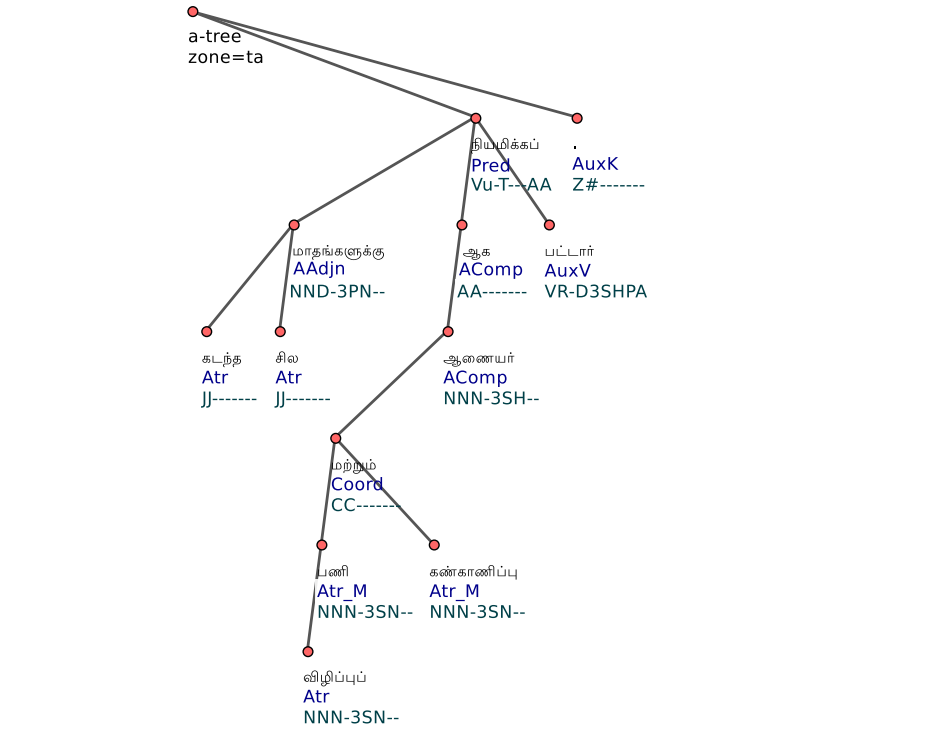


Figure 4.28: *Coord:* Coordination head (English style)

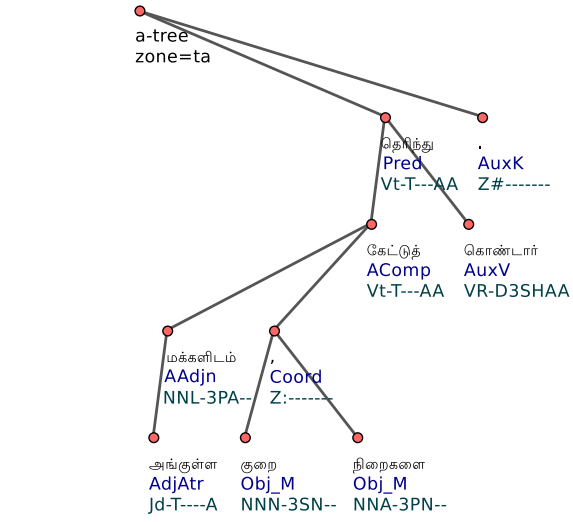


Figure 4.29: *Coord:* Coordination head (Comma)

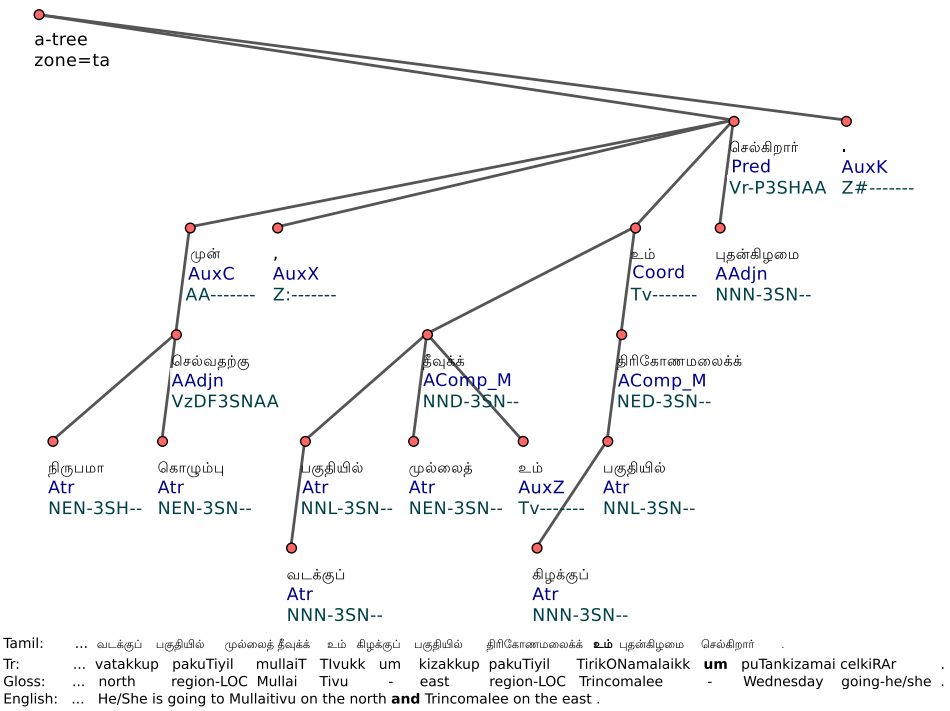


Figure 4.30: *Coord:* Coordination head (Morphological marker -*um*)

Afun: *Obj*

Direct and indirect objects receive the relation *Obj*. Figure 4.31 shows an example annotation of sentence with *Obj* relation. If there are both direct and indirect objects in a same sentence, then both will be labeled with *Obj*.

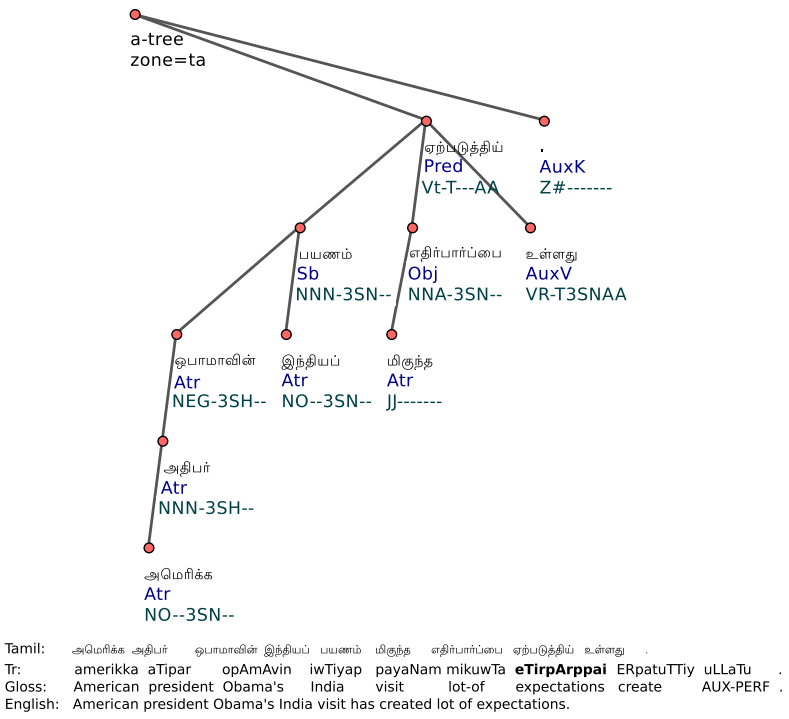


Figure 4.31: *Obj:* Object

Afun: *Pnom*

Nominal predicate occurs in the copula (be) constructions or verbless constructions. In these constructions, the sentence will not have any lexical verb. Instead, the predicate will contain only noun phrase. The noun phrase will be the predicate, and the afun label *Pnom* will be assigned to the noun phrase. Figure 4.32 shows the usage of *Pnom*.

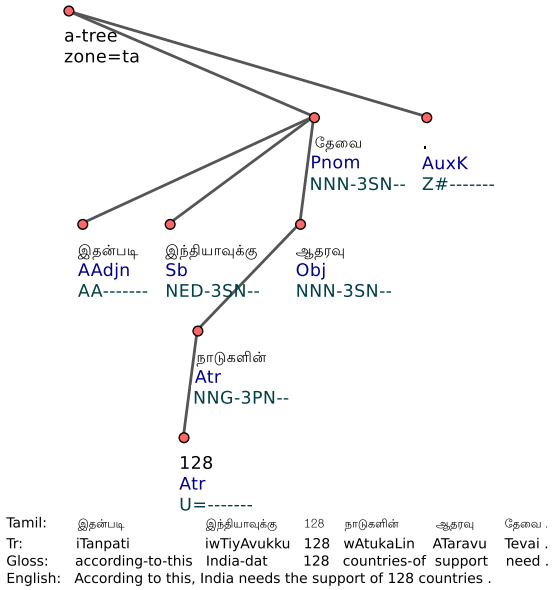


Figure 4.32: *Pnom:* Pnom

Afun: *Pred*

Predicate of the main clause will be given Pred. Only finite verbs in Tamil can be the predicate of the sentence. In Tamil, finite verbs at the end of the sentences are main predicates. So they receive Pred afun. In some cases, finite verbs will be absent at the end of the sentence. In that case, if there is a nominal predicate, it will receive Pnom afun or there won't be any Pred for that sentence. Figure 4.33 shows an example annotation for *Pred* relation.

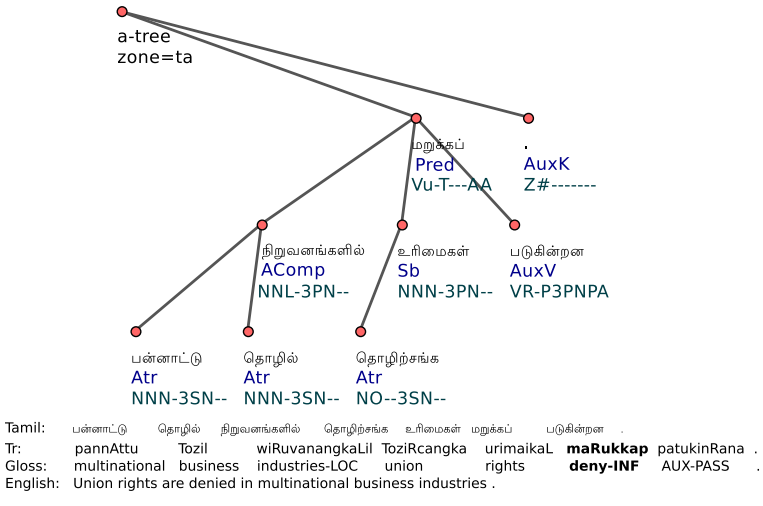


Figure 4.33: *Pnom:* Pred

Afun: *Sb*

The label *Sb* is assigned to the subject of the sentence. If there are more than one subjects in the sentence (i.e in the case of multiple clauses), then the label *Sb* will be assigned to all of them. The subject of passive verbs will also be labeled with *Sb* relation. Figures 4.34 and 4.35 illustrate how *Sb* annotated in dependency structures.

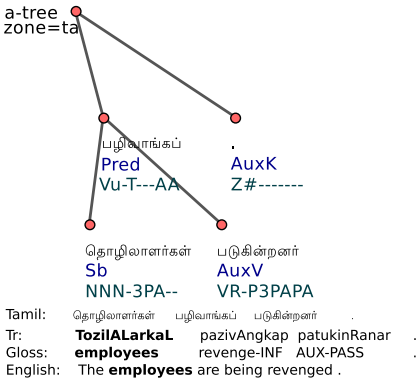


Figure 4.34: *Sb:* Subject

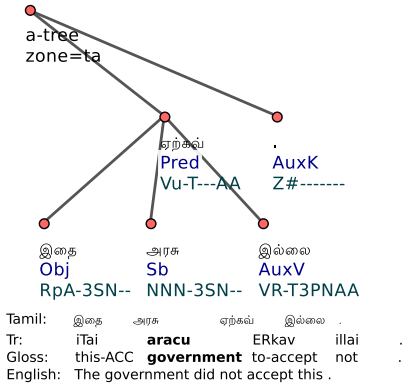


Figure 4.35: *Sb:* Subject

Chapter 5

Data and Evaluation

This chapter describes the data (Section 5.1) used for cross-lingual dependency transfer experiments in Chapter 6.

5.1 Data

For our experiments, depending on the requirements, we use labeled and unlabeled data from both the source and target languages. The datasets are described in the following subsections.

5.1.1 Parallel corpora

We use parallel corpus for bitext based projection transfer. Since the projection based experiments use English as the source language, we collected parallel corpus for English-Indian language (EN-IL) pairs. The parallel corpus for each EN-IL pair is described in Table 5.1. We have 8 parallel datasets involving 5 ILs. We divide the parallel datasets into two groups. We distinguish datasets in terms of the source (column one) from which the other side of the parallel corpus was obtained. The primary source for the first 3 datasets (translation direction: EN→IL) is English, and for the rest of the datasets IL the primary source (translation direction: IL→EN). In addition to that, most of the data from the first part (EN→IL) were translated by professional translators, whereas for the datasets in IL→ EN, the English side of the ILs were obtained by *crowdsourcing* using Amazon's Mechanical Turk (MTurk) in a short span of time. The purpose of experimenting dependency transfer for two different datasets for the same language pair serves the following,

- how the translation direction of the parallel corpus affects the projection.
- how inexpensive translations obtained in a relative quick time (using MTurk) can be useful in dependency transfer.

The above aspects will be studied in later sections when the experimental results are presented.

Parallel corpora: EN→IL

The following paragraphs in this subsection describe the parallel datasets for which the IL side of the texts are translated from English.

English-Hindi (`en-hi`): HindEnCorp[1] [Bojar et al., 2014] is the biggest collection of parallel corpora for `en-hi`[2] language pair made available in the web recently. The corpora was collected from various web sources and was made available in sentence-aligned plain text format. Some parts of the original corpora were not sentence-aligned, thus, they applied automatic sentence aligner - Hunalign [Varga et al., 2005] to obtain sentence-aligned data. In addition to that, they also applied some cleaning and normalization on the Hindi side of the data. The original HindEnCorp contains 274K sentence pairs. However, for our experiments, we use only part of that data (only first 50K sentence pairs). Before extracting the 50K sentence pairs, we remove the parallel sentences from [Post et al., 2012] which was also included as part of HindEnCorp.

English-Tamil (`en-ta`): We use only 50K sentence pairs from the *news* section of the EnTam[3] corpus [Ramasamy et al., 2012]. The news section of the parallel corpus contains professional translation of English news articles in Tamil. Like the `en-hi` pair, automatic sentence level alignments for the dataset were obtained using Hunalign [Varga et al., 2005].

English-Urdu (`en-ur`): We use Penn Treebank and Emille sections of the UMC005[4] corpus [Jawaid and Zeman, 2011] for projection experiments. `en-ur` pair has less corpus size (around 13K excluding 2K for development and test sets) than the above other two pairs.

We restricted the corpus size of `en-{hi,ta,ur}` under EN→IL for the following reasons,

- Reduced the corpus size of `en-{ta,hi}` to 50K to avoid slow training of parsing models obtained from the parallel corpus.
- Removed religious sections of `en-ta` and `en-ur`. The characteristic of language in religious texts are vastly different from the contemporary language.

[1] HindEnCorp - `http://hdl.handle.net/11858/00-097C-0000-0023-625F-0`

[2] We use the shorthand notation as this to refer the specific parallel corpus (Table 5.1) for this language pair.

[3] EnTam corpus - `http://ufal.mff.cuni.cz/~ramasamy/parallel/html/`

[4] UMC005 corpus - `http://ufal.ms.mff.cuni.cz/umc/005-en-ur/`

Parallel corpora: IL→EN

The parallel datasets (`en-{bn,hi,ta,te,ur}`) under IL→EN [Post et al., 2012] were constructed manually through crowdsourcing (via Amazon's Mechanical Turk - MTurk) with ILs as source. The datasets were obtained in a span of a month for six languages (Malayalam is not shown in the Table 5.1). For our experiments, we use the tokenized part of their training sections of the corpora. Note that the texts of IL are different from each other, so the individual parallel corpus under IL→EN does not overlap with each other. We didn't restrict the size of parallel datasets in IL→EN.

As far as the sentence length among the parallel corpora is concerned, most of the parallel datasets have the average sentence length of < 20 (except `en-{ta,ur}`).

Tr. dir.	lang. pair	source	corpus size	#tokens		#avg. sen. length	
				src (en)	tgt (IL)	src (en)	tgt (IL)
EN→IL	English-Hindi (en-hi)	[Bojar et al., 2014]	50.0K	636.4K	12.7	668.8K	13.4
	English-Tamil (en-ta)	[Ramasamy et al., 2012]	50.0K	1431.0K	28.6	1155.2K	23.1
	English-Urdu (en-ur)	[Jawaid and Zeman, 2011]	12.8K	267.6K	20.9	323.9K	25.2
IL→ EN	English-Bengali (en-bn)	[Post et al., 2012]	20.8K	323.7K	15.6	264.9K	12.7
	English-Hindi (en-hi)	[Post et al., 2012]	37.6K	622.9K	16.6	692.5K	18.4
	English-Tamil (en-ta)	[Post et al., 2012]	35.0K	455.9K	13.0	417.9K	11.9
	English-Telugu (en-te)	[Post et al., 2012]	43.0K	583.0K	13.6	480.4K	11.2
	English-Urdu (en-ur)	[Post et al., 2012]	33.8K	530.9K	15.7	658.4K	19.5

Table 5.1: Parallel corpus for projection

lang.	source	# sentences		avg. sen. len.		tagset size		
		train	test	train	test	dep.	CPOSTAG	POSTAG
Bengali (bn)	ICON2010 [Husain et al., 2010]	979	150	6.6	5.4	42	14	21
Hindi (hi)	MTPIL (COLING2012) [Husain et al., 2010]	12041	1233	22.3	21.4	119	36	20
Tamil (ta)	TamilTB 1.0 [Ramasamy and Žabokrtský, 2011]	480	120	15.1	15.8	25	12	465
Telugu (te)	ICON2010 [Husain et al., 2010]	1300	150	3.9	4.0	41	16	24
Urdu (ur)	UDT [Bhat and Sharma, 2012]	2808	313	13.1	12.4	71	15	29

Table 5.2: Treebank statistics

5.1.2 IL treebanks

We use treebanks for five ILs, namely Bengali (bn[5]), Hindi (hi), Tamil (ta), Telugu (te) and Urdu (ur). We use the existing (see Table 5.2) treebank data for two purposes: (i) to train IL POS taggers and (ii) to evaluate projected and various parser transfer models.

For training IL POS taggers, we use the training part of the treebank data. We extract sentences with their corresponding POS information and train Stanford tagger [Toutanova et al., 2003] on them. Though this step relies mainly on the POS annotated data of the training part, it avoids mapping to treebank tags in case the external POS taggers use different tagset. However, the size of the training data will be small for some of the treebanks (Bengali, Tamil and Telugu). We use the test data for all evaluation tasks. In cases where the evaluation should be done on harmonized treebanks, we use harmonized version of both the train and test data.

Among the five IL treebanks (from Table 5.2), the treebank for Hindi is the only large scale treebank available for ILs as of now. Hindi is definitely not an under-resourced language with respect to treebanking, however, it would still be interesting to see how transfer-based methods work for Hindi. For Bengali, Hindi, and Telugu, we use the version of the treebanks supplied during shared tasks (ICON 2009-2010 and MTPIL 2012) [6] [7]. For Hindi, we use the MTPIL 2012 version of the treebank. Treebank for Hindi appeared in earlier shared tasks too, but they were much smaller than the current version. Treebanks for Bengali and Telugu didn't appear after ICON 2010 shared task, so we use the ICON 2010 version.

For Tamil, we use TamilTB[8] [Ramasamy and Žabokrtský, 2011] developed at Charles University in Prague. We use the ongoing version of TamilTB 1.0 rather than the earlier TamilTB.v0.1. TamilTB 1.0 differs from the earlier version in many respects including the use of Tamil orthography directly in the treebank instead of Latin transliteration, tagset revisions under POS and dependency categories and some other corrections (mainly with respect to tokenization) in the data. The broad guidelines outlined in the Chapter 4 are from the earlier TamilTB.v0.1(most of the guidelines will be valid for newer version too). However, all our experimental results will be shown for the ongoing version. In cases where it is necessary to make a distinction between the current and the previous version of the TamilTB,

[5]We use language code as this to refer to a particular resource for the language (as indicated by the language code). The resource can be a POS tagger, parser or annotated data such as treebank or POS tagged data. For example, if we say 'bn parser', it means the parser trained from the treebank that language code indirectly refers to. In the same manner, 'Bengali (bn) treebank' or 'bn treebank' refer the specific treebank for Bengali language listed in Table 5.2. When we make general observation about languages or language properties, we use full language name instead of language codes.

[6]ICON 2010 shared task - http://ltrc.iiit.ac.in/icon/2010/nlptools/

[7]MTPIL 2012 (COLING) shared task - http://ltrc.iiit.ac.in/mtpil2012/sharedtask.php

[8]TamilTB - http://ufal.mff.cuni.cz/~ramasamy/tamiltb/1.0/html

we explicitly indicate the version to avoid any confusion.

Treebanking for Urdu is an ongoing effort [Bhat and Sharma, 2012, Bhatt et al., 2009] and we obtained the treebank through personal communication as it was not available through web or shared tasks. The Urdu (`ur`) treebank is relatively larger than Tamil (`ta`), Bengali (`bn`) or Telugu (`te`) treebanks.

In terms of language relatedness, Tamil and Telugu belong to *Dravidian* family, and Bengali, Hindi and Urdu belong to *Indo-Aryan* family. In terms of annotation style, there's a fundamental difference between Tamil (`ta`) treebank and other IL treebanks. Tamil (`ta`) treebank uses PDT style annotation, whereas the other IL treebanks use annotation style based on the Pāṇinian grammar theory. POS tagging style too differs between Tamil (`ta`) and other IL treebanks. Tamil (`ta`) treebank uses positional tags whereas the other IL treebanks use PennTreebank style tagset.

Some of the treebanks were not in UTF-8. For example: Bengali (`bn`) and Telugu (`te`) were supplied with wordforms in `` `wx' `` notation (uses only Latin characters). The treebanks were transliterated[9] to use their native scripts in UTF-8.

Part of the Tamil (`ta`) treebank (only test data) has English translations and manual word alignments between English and Tamil wordforms. However, we don't have gold syntactic trees for the translated English sentences, so we obtain English trees by parsing using MSTParser while doing projection. We use the manual word alignments (between English and Tamil) in the test data to perform error analysis for the projection experiments in Chapter 7.

5.1.3 HamleDT

We use HamleDT treebanks [Zeman et al., 2012] for our delexicalized parser transfer experiments. HamleDT contains original treebanks in their native annotation style and their POS and dependency annotation (both labels and edges) harmonized into PDT style annotation. Harmonization refers to mapping/conversion of one annotation style to another annotation style. Harmonization on dependencies include mapping some of the most common syntactic units, such as coordination, subordinate clauses, punctuations, prepositional phrases, etc. into PDT style annotation. HamleDT 1.0 contains 29 harmonized treebanks and HamleDT 2.0 contains 30 harmonized treebanks. We use HamleDT 2.0 for our experiments. HamleDT 2.0 [Rosa et al., 2014] also includes Stanfordized dependencies obtained from the harmonized versions of the treebanks. For our experiments, we use only harmonized treebanks (without Stanfordization). We use the following HamleDT related terminologies in the thesis,

- **HamleDT treebank:** refers to the fully harmonized (POS + dependency labels + dependency structure) version of the original treebank.

[9]Transliteration credit: Dan Zeman

- **Interset tag**[10]**:** refers to the fine-grained harmonized tag in HamleDT treebanks. For example, the normal POS tag of the English word *'Heavy'* is *JJ* which means the word is adjective. The harmonized tag for *JJ* is *AOXX - - - - - 1 - - - - -*. Each Interset tag is a positional tag and encodes variety of information (mainly pertaining to morphology) inside each tag. An Interset tag is a fixed length string of 15 characters similar to positional tags used in Prague dependency treebank (PDT) annotation.

- **Coarse-grained Interset tag:** The first position of a given Interset tag. The first position corresponds to main POS category. For example, the coarse-grained Interset tag of *AOXX - - - - - 1 - - - - -* is *'A'*. *'A'* is given the meaning *adjective* in the Interset tagset. Coarse-grained Interset tagset size[11] is 12.

Data composition of individual treebanks[12] that are harmonized is shown in Table 5.3. Four (`bn, hi, ta, te`) of the five treebanks described in Table 5.2 are harmonized and are already part of the HamleDT distribution. The Tamil (`ta`) treebank described in Table 5.2 uses TamilTB 1.0. However, the HamleDT distribution uses the earlier version of the Tamil (`ta`) treebank (TamilTB.v0.1). Since all our Tamil related tasks are carried out for TamilTB 1.0, we harmonize TamilTB 1.0 and use it in place of TamilTB.v0.1.

[10]The terminology *fine-grained Interset tag* is also used in places where emphasis is needed. Otherwise, *Interset tag* refers to full length positional tag.

[11]See the values for POS: `https://ufal.mff.cuni.cz/pdt/Morphology_and_Tagging/Doc/hmptagqr.html`

[12]Table 5.3 shows only part of the information. Additional information about HamleDT 2.0 can be obtained from here: `http://ufal.mff.cuni.cz/hamledt/hamledt-treebanks`

Lang.	Train		Test		Total		
	sen.	tokens	sen.	tokens	sen.	tokens	avg. sen. len.
Arabic (ar)	6776	249600	771	27823	7547	277423	36.8
Basque (eu)	12823	190217	398	5934	13221	196151	14.8
Bengali (bn)	979	6440	150	812	1129	7252	06.4
Bulgarian (bg)	13200	390302	1724	53015	14924	443317	29.7
Catalan (ca)	77765	1330152	10148	173586	87913	1503738	17.1
Czech (cs)	5190	94386	322	5852	5512	100238	18.2
Danish (da)	36020	648677	2000	32033	38020	680710	17.9
Dutch (nl)	2705	65419	197	4804	2902	70223	24.2
English (en)	18577	446573	214	5003	18791	451576	24.0
Estonian (et)	14329	427442	1655	50368	15984	477810	29.9
Finnish (fi)	1184	8535	131	956	1315	9491	07.2
German (de)	10104	137309	1122	14295	11226	151604	13.5
Greek (el)	12126	182878	329	6694	12455	189572	15.2
Greek, Ancient (grc)	3877	53151	430	5425	4307	58576	13.6
Hindi (hi)	20632	302957	528	5925	21160	308882	14.6
Hungarian (hu)	12041	268093	1233	26416	13274	294509	22.2
Italian (it)	6034	131799	390	7344	6424	139143	21.7
Japanese (ja)	3110	71199	249	5096	3359	76295	22.7
Latin (la)	17044	151461	709	5711	17753	157172	08.9
Persian (fa)	3157	48354	316	4789	3473	53143	15.3
Portuguese (pt)	13349	195069	386	5585	13735	200654	14.6
Romanian (ro)	9071	206678	288	5867	9359	212545	22.7
Russian (ru)	3776	33510	266	2640	4042	36150	08.9
Slovak (sk)	34493	494007	402	3458	34895	497465	14.3
Slovenian (sl)	51941	815313	5494	85985	57435	901298	15.7
Spanish (es)	1534	28750	402	6390	1936	35140	18.2
Swedish (sv)	11042	191467	389	5656	11431	197123	17.2
Tamil (ta)	480	7592	120	1989	600	9581	16.0
Telugu (te)	1300	5125	150	597	1450	5722	04.0
Turkish (tr)	5635	65182	300	4513	5935	69695	11.7

Table 5.3: HamleDT 2.0 data composition

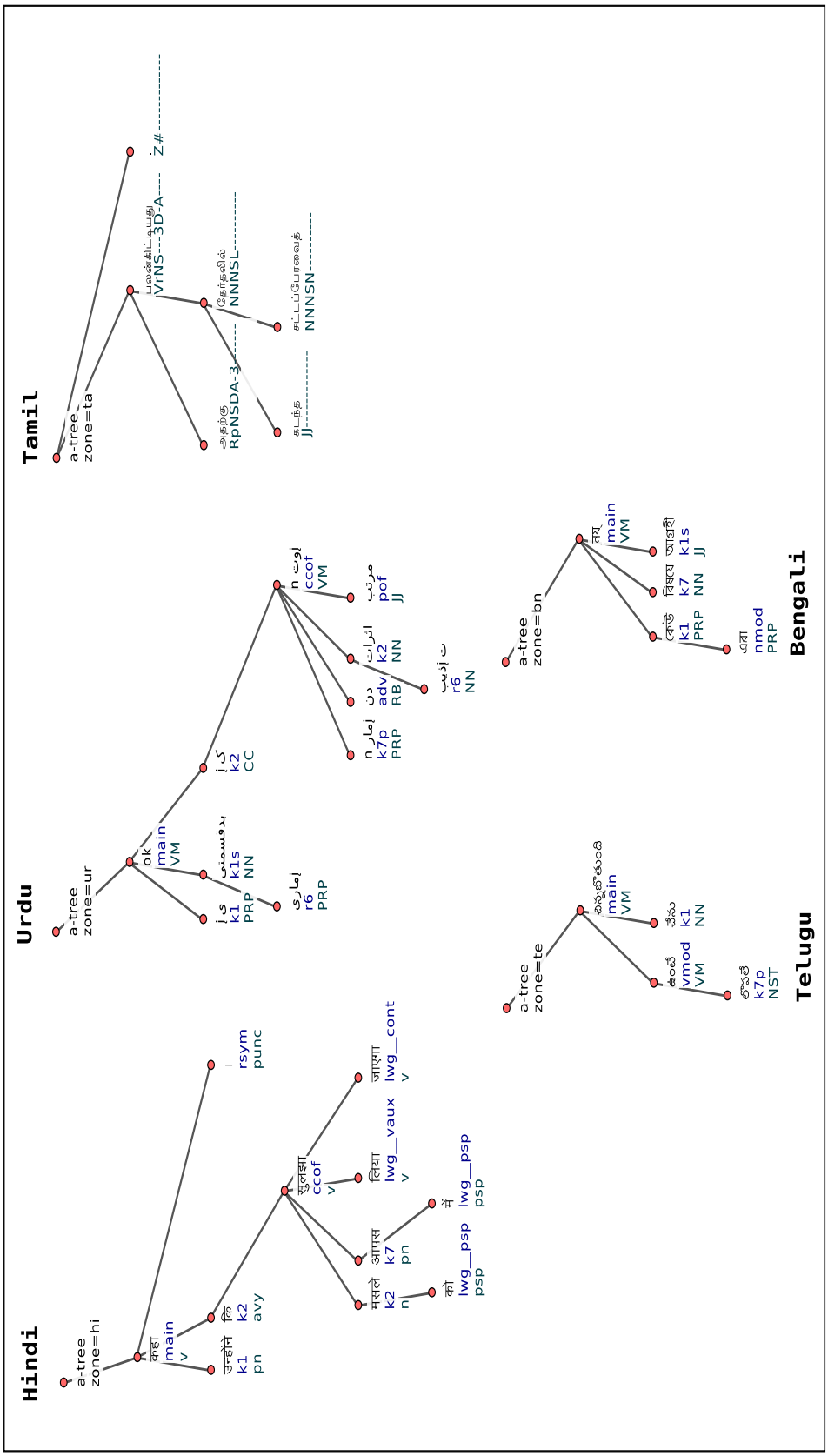


Figure 5.1: Example trees from original treebanks

5.2 Evaluation

We report *Unlabeled Attachment Score (UAS)* for all our experiments. UAS score provides the estimate of proportion of nodes that have correct parents against the total number of nodes. We keep our evaluation to UAS score only as we do not have mappings of dependency relations between English and IL. Recent works such as [Zeman et al., 2012] and [McDonald et al., 2013] offer some hope for evaluations based on dependency relations. Moreover, sticking to UAS alone can provide maximum insight into the kind of structures that can be transferred across languages using transfer approaches.

$$Unlabeled\ Attachment\ Score\ (UAS) = \frac{\#number\ of\ nodes\ with\ correct\ parents}{\#number\ of\ nodes} \tag{5.1}$$

Chapter 6

Cross-Lingual Dependency Transfer

In this chapter, we experiment with various cross-lingual transfer approaches for Indian languages.

- Section 6.1 introduces projection based transfer approach from resource-rich English to Indian languages. We experiment with the traditional projection approach (Section 6.1.1) for 5 major Indian languages (ILs) for which we have access to treebanks.
- In Section 6.2, we apply delexicalization based transfer approach to ILs. We parse ILs using delexicalized parsers trained from 30 treebanks. Experiments are carried out for settings where the source treebanks follow common annotation standards.
- Section 6.3 introduces the transfer approach in which machine-translated bitexts are used to transfer parsers from English to target languages. The experimental results are given for non-IL and IL treebanks. Since the machine translated texts are generated by restricted access machine translation (MT) system and due to time constraints, we experimented only for a subset of language treebanks that appeared in [Zeman et al., 2012]. However, we perform experiments in resource-poor scenarios and the conclusions we draw from those experiments can well be generalized to other languages.

6.1 Transfer using bitext projection

[Hwa et al., 2002, Hwa et al., 2005] was the first to use projection based approach to transfer syntactic dependencies from one language to another. This approach relies on parallel texts. Fundamental to their approach was *Direct Correspondence Assumption (DCA)* which says that, if we have a sentence and its translation, then the dependency structure of the sentence and its translation is most likely to be similar. More formally, the DCA [Hwa et al., 2005] is defined as,

Direct Correspondence Assumption (DCA): Given a pair of sentences E and F that are (literal) translations of each other with syntactic structures $Tree_E$ and $Tree_F$, if nodes x_E and y_E of $Tree_E$ are aligned with nodes x_F and y_F of $Tree_F$, respectively, and if syntactic relationship $R(x_E, y_E)$ holds in $Tree_E$, then $R(x_F, y_F)$ holds in $Tree_F$.

$R(x_F, y_F)$ is a directed edge in the graph. It doesn't tell which node is parent and which node is child. This is inferred from $R(x_E, y_E)$. In the case of DCA, if x_E is the parent of y_E in the structure $Tree_E$, then x_F becomes the parent of y_F in $R(x_F, y_F)$ when the projection algorithm makes relations in the target sentence.

6.1.1 Algorithm

We present a slightly modified *Direct Projection Algorithm (DPA)* of [Hwa et al., 2005]. [Hwa et al., 2005]'s algorithm considers all types of alignments during the projection. In general, the algorithm we present here has two main differences with respect to DPA: (i) the projected tree in the target language is always connected; we achieve this by projecting source dependencies to the initialized target dependency structure, and (ii) we do not add any additional empty nodes in the target dependency structure as it may complicate the evaluation procedure.

Unlike [Hwa et al., 2005]'s DPA where they did not initialize the target tree, we start with the parsed source tree and left-branched[1] target tree. The projection based experiments are carried out for ILs. Since we project English trees to ILs and IL treebanks are predominantly left-branching (Table 6.1), we decided to initialize target trees to left-branching structures.

As the target tree is already connected, we only adjust edges; in other words, we rehang them to appropriate nodes in the target tree when the projection algorithm processes source nodes. We do not remove any edges during the application of the projection algorithm, so the target tree remains connected before and after the projection.

- We project *1-to-1* alignments in the same manner as in DPA. That is, if e_i is aligned to f_i, e_j is aligned to f_j, and there exists a relation $R(e_i, e_j)$, then we make a relation $R(f_i, f_j)$ in the target. The only difference is, the DPA makes a new relation in the target whereas we replace the existing (default) edge in the target.

- In the case of *1-to-M* alignments, if e_i is aligned to $f_i f_j ... f_n$, we first find the head node within the M-word chunk, i.e., $f_i f_j ... f_n$ and connect remaining members of the chunk to the head of the chunk. For instance, if the head node is f_m, then the other members of the chunk $f_i f_j f_k f_l f_n$ are children to f_m. Then we use *1-to-1* strategy above to find the relation for f_m.

[1] In a left-branching tree, each word is the modifier of the next word.

- We treat *M-to-1* alignments as *M{1-to-1}* alignments. We again, use the *1-to-1* strategy to find relation in the target. However, we do not use all the *M{1-to-1}* alignments. Once we find a target relation using any of the *M{1-to-1}* alignments, we do not pursue further finding the target relation using other *(M-1){1-to-1}* alignments. Once the recursive algorithm finds a target relation for the first node in the M-word group, then the algorithm marks the target relation as found, so it does not try to find a relation again when the algorithm reaches other *M-to-1* nodes.

- *M-to-M* alignments are treated as *1-to-M* and *1-to-1* alignments. Since the projection algorithm processes source nodes one-by-one, it can only know whether the source node it currently processes has *1-to-M* or *1-to-1* alignment. In that case, the algorithm applies *1-to-M* or *1-to-1* strategy.

- The algorithm skips *unaligned nodes* on the source side. When an unaligned source node is encountered, [Hwa et al., 2005]' DPA adds an empty node on the target and treats it with the procedure for projection of *1-to-1* alignments.

The pseudocode for the projection procedure described above is given in Algo-

rithm 6.1.1.

Algorithm 6.1.1: PROJECTIONALGORITHM(s_{root}, t_{root})

INITIALIZETARGETTREE($'left', t_{root}$)
PROJECT(s_{root}, t_{root})
$t_{proj} \leftarrow t_{root}$
return (t_{proj})

procedure PROJECT(s_{par}, t_{par})
$SC \leftarrow s_{par}.get\_children()$
for each $c \in SC$
do
- $alignedNodes \leftarrow c.get\_aligned\_nodes()$
- $chunkHead \leftarrow get\_chunk\_head(alignedNodes)$
- **if** t_{par} **not** $descendent\_of(chunkHead)$
  - **then**
    - $chunkHead.set\_parent(t_{par})$
    - **comment:** make nodes other than $chunkHead$ to ...
    - **comment:** ... be the children of $chunkHead$
    - PROJECT($c, chunkHead$)
  - **else** PROJECT(c, t_{par})

The Algorithm 6.1.1 takes as input a parsed source tree and a flat target tree. The algorithm runs recursively from the root node of the source tree in a pre-order fashion. At each source node, the algorithm tries to find the relation for the aligned target node.

The function `get_children()` returns ordered[2] children of a given node. Not processing the source children in the order they appear in sentence may likely to result in a different projected structure. For the sake of simplicity, we decided to use the first approach (ordered children).

The function `get_aligned_nodes()` returns target nodes (if any) that are aligned to a given source node. Given a set of nodes, the function `get_chunk_head()` returns the head among them. Internally, the function considers those nodes as a small syntactic structure and makes a guess about the head of that structure. In

[2]Let us assume a node n_2 in a tree represents a word at 2^{nd} position in a sentence. If n_2 has three children, namely n_5, n_1 and n_7, then the procedure `get_children()` returns the children in this order: {$n_1 n_5$,n_7}

experiments, for simplicity, we consider the last node (based on its position in the sentence) as the head of the group. Other nodes in the group are considered as the children of the head node.

The algorithm does not allow cycles[3], and when the proposed edge for a target node leads to cycle, it skips adjusting the edge and proceeds to the next source node.

Figures 6.3 and 6.4 illustrate the projection algorithm (Algorithm 6.1.1) for the aligned sentence pair as given in Figure 6.1. The illustrations in the figures are self-explanatory. After the target initialization to left-branching, each row in Figures 6.3 and 6.4 project a source node (circular nodes) to the target side. Thus each row in the right column is the result of projecting a source node to the target. The currently projected node has edges going from source to target. If there is no edge(s), then the source node is unaligned to the target side. Squared nodes are not projected since they serve only the purpose of a technical-root, and they are always aligned.

Figure 6.1 is in fact taken from a real English-Tamil gold word-aligned sentence pair from the Tamil (`ta`) treebank test data. The actual sentence pair and the output of running the projection algorithm on this sentence pair is shown in Figure 6.2.

[3]It is also partly due to practical considerations in visualizing tree structures in the tree editor Tred. Tred editor does not allow cyclic structures.

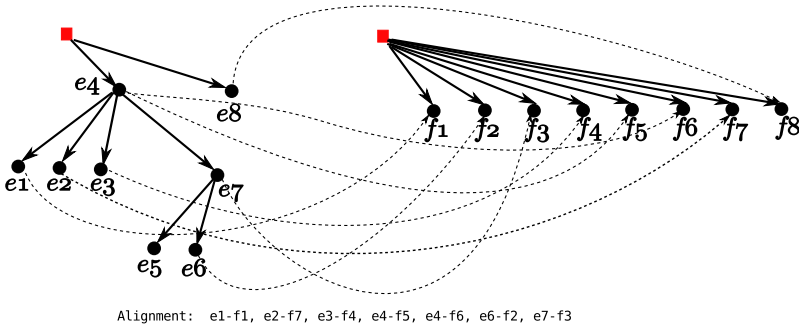


Figure 6.1: Aligned sentence pair

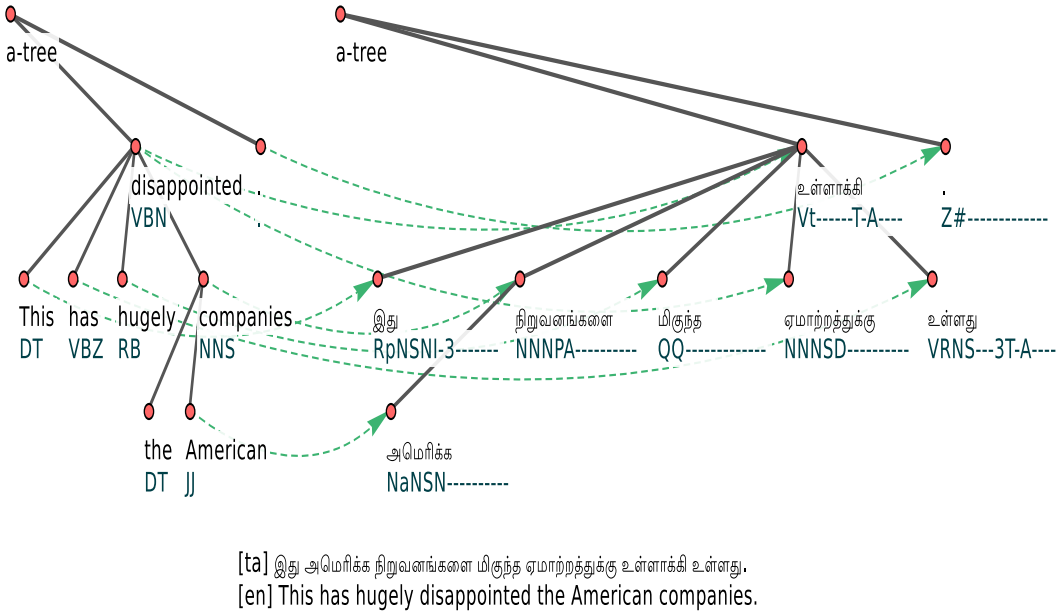


Figure 6.2: Projection algorithm output on a real sentence pair

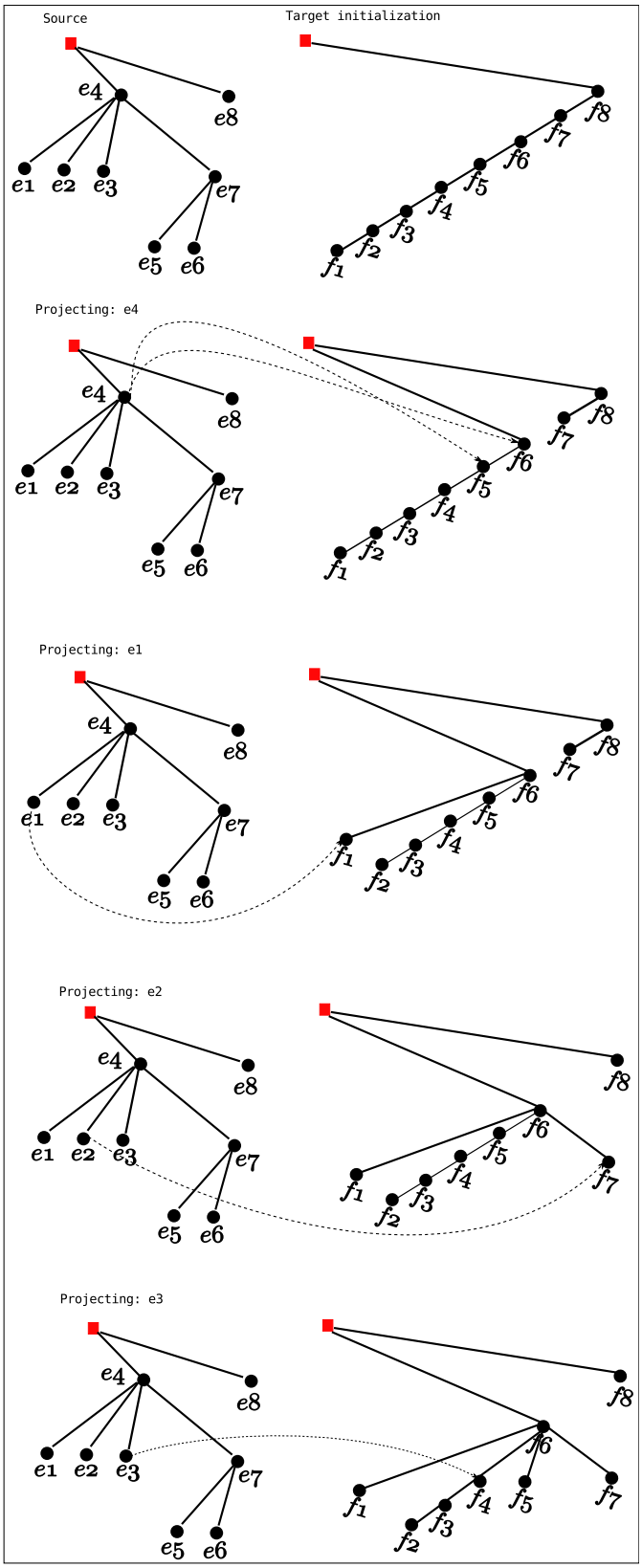


Figure 6.3: Projecting source onto target: a step-by-step example

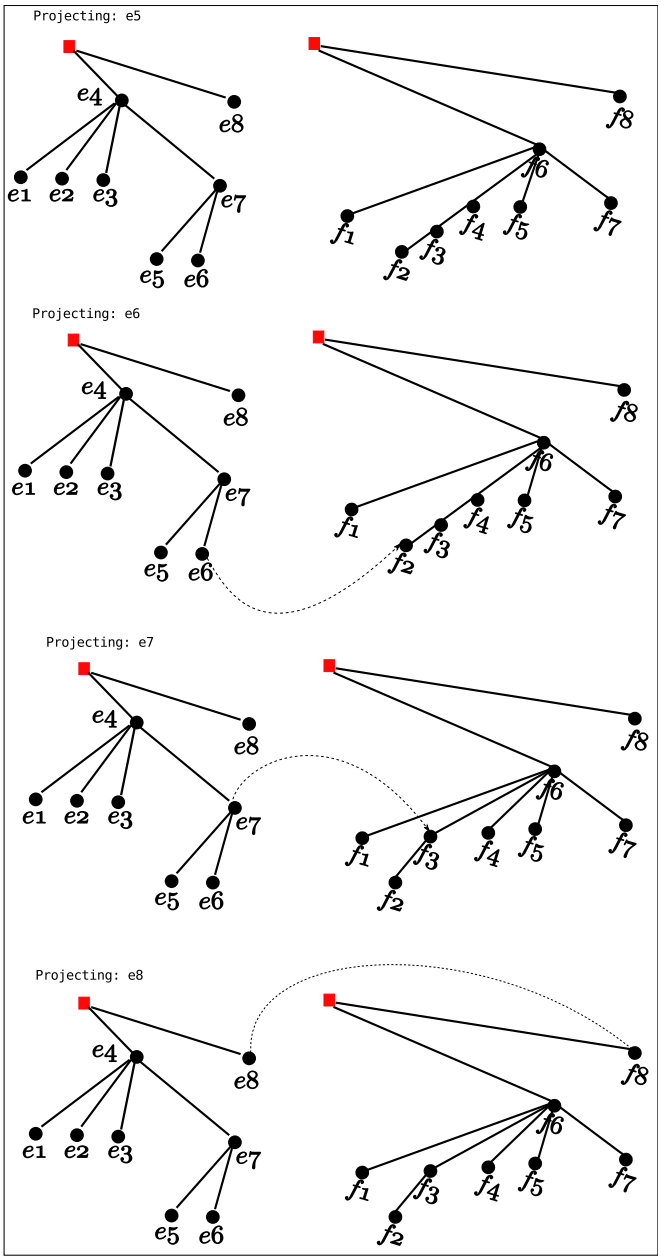


Figure 6.4: Projecting source onto target: a step-by-step example (...continued from Figure 6.3)

6.1.2 Challenges

Direct projection based approach has some key assumptions that could well be a challenge when we try to apply projection to a new set of languages. In general, language differences (mainly syntactic difference in syntax transfer) and annotation differences can limit the type of target structures we would desire from the transfer.

Although syntactic differences do not result in greater impact on the transfer, annotation differences do make a big difference in syntactic transfer. Those generalities apart, in order for the dependency syntax transfer to be successful, the following key points need to be satisfied.

High quality source structures

Projection accuracy to a larger extent depend on the quality of the source side analysis tools. In the entire projection pipeline, the following two scenarios from source language processing that have the potential to affect the projection accuracy,

- inaccurate POS tagging
- inaccurate parsing

Resource-poor target languages can largely benefit from resource-rich source languages if they are also closely related.

High quality word alignments

Correct word alignment links between source and target sentences are crucial to obtain better target structures. Manual alignments are impossible to obtain, even if they are available they may not exceed few hundred sentences. It is unrealistic to obtain manual word alignments for larger bitext. Thus, for projecting larger data in a target language, the word alignments are usually obtained via automatic word alignment programs such as GIZA++ or Berkeley aligners. However, the quality of the word alignment highly depends on the size of the bitext as well as language divergences between the source and the target language.

Annotation differences

One important aspect that requires further inquiry when contemplating transfer of syntactic structures from one language to another is: "How much the annotation differences alone can make the dependency transfer difficult?" Annotation differences can have deeper impact than the differences that were due to variations in morphological, syntactic or semantic structures between languages (provided alignments are of high quality). Earlier works such as [Hwa et al., 2005] tried to address this important challenge by including language specific transformation rules after projecting dependency trees from a source to a target language. The transformation rules converted the projected structures from a source annotation to a target annotation. The rules might look like, as in [Hwa et al., 2005],

- ...
- The aspectual marker should modify the verb to its left.

- Some prepositions appear in pairs (such as from ... to ...). In these cases, the first preposition should modify the second.
- ...

Such rules as listed above were employed on the Chinese structures after the projection from English to Chinese. In their comparison with projection with no transformation rules, the projection performance jumped from 33.9% to 65.7% for Spanish and from 26.3% to 52.4% for Chinese when language specific transformation rules are applied. This simply shows how important the annotation differences are and how much impact it can have on the projection performance. One obvious disadvantage with this approach is that it requires some knowledge about target language annotation and syntax.

6.1.3 Experiments

In this subsection, we conduct projection experiments for the 8 parallel datasets shown in Table 5.1. English is the source language for all the experiments, i.e., we project English trees to the ILs side to get approximate dependency structures for target ILs. We then create parsing models from the projected trees by training a regular parser on them. Finally we evaluate these parsing models against the original treebank test data available for ILs.

We provide a simple baseline comprising left-branching, right-branching and standard supervised parsing results. The following subsections explain the individual experiments with greater details.

Baseline

We consider left and right-branching trees as one of our baseline. We also provide supervised parser results for comparison, this could in fact indicate the upper limit that other cross-lingual approaches can aim for. Indian languages are head-final in many respects and have a tendency to be left-branching. So, one can expect the left-branching accuracy to be higher than the right-branching accuracy.

Language	Left modifiers (%)	Right modifiers (%)
bn	86.10	13.90
en	48.92	51.08
hi	57.01	42.99
ta	88.40	11.60
te	94.58	05.42
ur	83.94	16.06

Table 6.1: Modifier direction in treebanks. % - indicates how often nodes are attached in a particular direction.

The Table 6.1 shows the percentage of nodes that are left and right modifiers to nodes in the treebanks. We calculated these numbers from the training section of the non-harmonized treebanks from HamleDT [Zeman et al., 2012]. For `ta` treebank we evaluated from the latest version the Tamil dependency treebank (TamilTB 1.0). Except `hi`, all other IL treebanks are predominantly left-branching. The reason that `hi` has more right-branching structures than the other IL treebanks is simply because of some annotation choices rather than any syntactic divergence from the other ILs.

As a second baseline, we train supervised parsers on the training section of the treebanks in Table 5.2. We train our models with MSTParser [McDonald et al., 2005a] (version-0.5.0, with default MSTParser feature templates). We train both *projective* and *non-projective* parsers with *second-order* features. For training the parsers, we are not using any features other than POS tags to make sure the training procedure is uniform across different treebanks.

We present our test results on the trained parser models with two settings: (i) parsing test data with gold POS[4] tags and (ii) parsing test data with POS tags obtained from a supervised POS tagger. For the task of target IL POS tagging (needed in projection, delexicalization and supervised parsing experiments), we train a supervised POS tagger (using Stanford tagger [Toutanova et al., 2003])for each ILs from the training section of the IL treebanks. We extracted wordforms and POS information from the IL treebanks (Table 5.2). The IL treebanks in Table 5.2 are in CoNLL format. The CoNLL format has coarse-grained (CPOSTAG - 4^{th} column) and fine-grained (POSTAG - 5^{th} column) distinctions. The Table 6.2 lists (2^{nd} column) which CoNLL column we are using to train the Stanford tagger.

For all IL datasets, the Stanford tagger is trained with the feature template `ARCH='generic,unicodeshapes(-5,5),suffix(5)'`[5].

[4]The primary objective of providing parsing results with gold POS tags is to observe how sensitive are the parsers to POS tags in the case of ILs.

[5]`'generic'` is a shortcut for `words(-1,1),order(2),biwords(-1,0),wordTag(0,-1)`.

Dataset	POS tag	Open tags
bn	POSTAG	NN VM
hi	CPOSTAG	NN NNP
ta	POSTAG	NN
te	POSTAG	VM NN
ur	POSTAG	NN NNP

Table 6.2: Training settings for target POS taggers

Further, the Stanford tagger requires open class tags (property: `'openClassTags'`) or closed class tags (property: `'closedClassTags'`) to be set during the training procedure. We identify open class tags from the training data through the following procedure,

1. Identify singleton wordforms, i.e., the wordforms that occur only once in the training data.
2. Get the POS tag for the singleton wordforms and count those tags.
3. Any POS tag (from the above step) that has a count of >= 15%[6] of the total count (or simply total number of singletons) is an *open class tag*.
4. If none of the tag reaches 15% of the total count, then we simply take the most frequently occurring tag among singletons to be the open class tag.

Open class tags obtained through the aforementioned procedure is given in Table 6.2. Note that for `ta`, we do not use the full length positional tag, instead we restrict the tag length to first two positions. This is reasonable given the fact that the training data for `ta` is small (about 7.2K tokens). We also enable *normalization function* during training which maps the numbers present in the training data to a unique symbol, so that the model can identify the numbers during testing phase (by normalizing the test data again before tagging) too. Baseline results are discussed in Section 6.1.4 (Table 6.3).

Bitext projection

Bitext projection is the projection of syntactic structures from one side of a parallel corpus to another side via word alignment links. In this experiment, we apply the projection algorithm described in Section 6.1.1 (Algorithm 6.1.1) to the parallel

[6]Choosing of this threshold is arbitrary. We tried >= 20%, but no tag was found for Tamil POS tagged data. So we tried >= 15% and we stick to that number. Another criteria while considering this threshold is not to have too many opentags that could slow the training speed of the Stanford tagger.

datasets described in Table 5.1. The source language for this experiment is English and the target is ILs, i.e., for each language pair (`en-bn`, `en-hi` and so on), we project syntactic structures from English side of the parallel corpus to the IL side of the parallel corpus. Bitext projection for any language pair consists of the following steps,

1. word alignments in parallel corpus
2. POS tagging + parsing of the source language
3. projection of dependency trees (source → target)
4. POS tagging + parser training of the target language

Word alignments in parallel corpus The main aim of this step is to make word alignments in the parallel corpus. There are at least three popular tools available to accomplish this task: (i) GIZA++ (ii) mgiza and (iii) Berkeley aligner. *GIZA++* [Och and Ney, 2003] is a *de facto* choice of word aligner in MT systems such as Moses. *Mgiza* is a multi-threaded version of GIZA++ with additional features such as incremental training. *Berkeley aligner* [Liang et al., 2006] is also a word aligner designed for tasks such as machine translation. All these word aligners work in an unsupervised fashion, i.e., they only require parallel corpus.

Our choice of word aligner for this alignment task is Berkeley aligner. One of the features of the Berkeley aligner is making use of syntactic structure of one side of the parallel corpus to improve overall word alignment quality. The actual mechanics of this feature of the aligner is described in [DeNero and Klein, 2007]. They proposed a method that used constituent structure of sentences in the parallel corpus. Syntax-aware distortion model that they introduced conditioned on those constituent structures to guide the alignments in the parallel corpus. In their comparison with GIZA++, they showed that their syntax-aware alignment model provided better alignment results than GIZA++. [Post et al., 2012] in their MT experiments has also shown that the Berkeley aligner produced better alignments and BLEU scores in a parallel corpora evaluation involving six Indian languages (we are using 5 of their parallel datasets). Our comparison with GIZA++(through mgiza) also brings the same conclusion (Figure 7.3).

We first make word alignments in the parallel corpus for each language pair for the respective parallel datasets as given in Table 5.1. We use syntax-based Hidden Markov Model (HMM) [DeNero and Klein, 2007] for our alignment task instead of GIZA++ for the reasons given in the previous paragraph. The word alignment task is performed as given below,

1. *parsing:* we parse the English side of the parallel corpus for each `en-{bn, hi, ta, te, ur}` language pair (for all datasets given in the Table 5.1) using the

constituency parser[7] [Klein and Manning, 2003].

2. *word alignments:* we run the word aligner for each `en-{bn, hi, ta, te, ur}` language pair using the Berkeley aligner [8]. We run two *MODEL1*s (one in each direction) jointly for 3 iterations, and the obtained parameters are used to run one *syntactic HMM* and one *classic HMM* for 3 iterations.

When running the word aligner, we set the alignment direction as target language `{bn, hi, ta, te, ur}`[9] → `en` since the syntactic alignment model requires this setting. The alignment direction (`{bn, hi, ta, te, ur}` → `en`) is not a concern for the projection algorithm to work properly since the algorithm does not depend on the direction of alignments.

POS tagging + parsing of the source language After the word alignment step for each parallel dataset (`en-{bn, hi, ta, te, ur}`) , we perform POS tagging and parsing on the source side (`en`) of the parallel corpus. POS tagging of `en` corpus is done with the Morce tagger [Spoustová et al., 2007]. We parse the English source using the MSTParser [McDonald et al., 2005a]. We use *second order, non-projective* model trained on the CoNLL version of the English treebank [Nivre et al., 2007] for parsing the English data. The English data we use for parsing is same as the one we used in the word alignment.

Projection of dependency trees (source → target) Once we have word alignments and parsed source sentences, we project the source dependency structure (as identified by the source side parser) onto the target side using the algorithm described in Section 6.1.1. The target trees obtained in this way will obviously contain many syntactic errors. However, many common syntactic sub-structures that had correct word alignments would have been transferred successfully.

POS tagging + parser training of target Following the projection, we obtain target parsers by training parsing models out of the projected data. When we have large amount of target tress, the parser training will have a positive effect by learning consistent patterns. We train both *projective* and *non-projective second-order* MST parsing models. Before training the parsing models, we tag the target sentences using the target language POS tagger.

[7]Stanford parser - `http://nlp.stanford.edu/software/lex-parser.shtml`, Model - English PCFG.

[8]Berkeley Aligner - `http://code.google.com/p/berkeleyaligner/`

[9]Here the *target language* still has the same meaning as the previous usages, i.e., the language for which we would like to induce dependencies. In [Klein and Manning, 2003], when aligning from $A \rightarrow B$, the alignment model makes use of constituent structures from B

6.1.4 Results

The Table 6.3 presents baseline results for five ILs. The numbers are reported for the test dataset given in Table 5.2.

Expectedly, like we have already shown in Table 6.1, left-branching trees produce reasonably good unlabeled attachment score (UAS) without much effort. Right-branching results measure exactly the opposite. The average accuracy of left-branching trees alone (excluding `hi`) is around 53.0%. For `hi`, results are almost equally poised for left and right-branching trees. One reason why `hi` behaves differently in the case of left-branching results is because of the annotation choice, for instance, in postpositional phrases (usually a noun phrase followed by a postposition), postpositions are governed by immediately preceding nouns (head of the noun phrase; in a noun phrase, the head will be the final in position), that actually makes them right-branching. About 19.83% of the wordforms in the `hi` treebank (training section) is tagged with postpositions and most of them (17.73%) are attached to immediately preceding noun heads. Had the annotation been made the other way, i.e., making postpositions the governors, the left-branching accuracy of `hi` would have seen a big jump (at least 17.73%). Treebanks `{bn, te, ur}` do not mark postpositions directly in their CoNLL data instead they are marked as features for the nouns they govern.

Lang.	Left	Right	Supervised parser	
			pred. POS	gold POS
bn	53.6	04.6	70.5/72.2	76.2/78.3
hi	24.4	27.3	80.3/80.2	85.5/85.4
ta	50.9	09.5	59.8/60.7	76.4/76.8
te	65.8	02.4	83.1/83.1	87.5/87.0
ur	42.9	06.3	63.4/66.3	66.7/68.2

Table 6.3: Baselines: Left/right-branching and supervised parsing (projective/non-projective) results.

The Table 6.3 also shows the accuracy of supervised parsers under two settings: (i) test data tagged with a supervised tagger (sub column *Pred. pos* in the table) and (ii) test data tagged with gold POS tags. Using gold POS tags obviously increases the supervised parser accuracy. For Tamil, gold POS makes a big difference possibly due to the tagset complexity (with very limited training data).

Bitext projection

Results for target language (ILs) parsers trained on the dependency trees obtained from the parallel corpus projection (English to ILs) are given in Tables 6.4 and 6.5.

Projection results are shown for two word alignment training settings: (i) parallel corpus word alignment by HMM syntax and (ii) parallel corpus word alignment by default HMM training[10]. Apart from this, results are also shown for parsing test data with predicted and gold POS, and parsing test sentences of all length and test sentences (column *reparse*) of length 10 or less (column $reparse_{10}$). Numbers in **bold** (only in Table 6.4) indicate that the results are higher than the left/right baseline given in Table 6.3.

We interpret the results based on two parameters: (i) alignment strategy used for parallel corpus word alignment and (ii) translation direction in the parallel corpus.

From Table 6.4, projection accuracy is slightly better for all target IL treebanks (except `hi`) when the parallel corpus alignment is performed with HMM (syntax) strategy.

When we look at the results based on the corpus type, `{en-hi, en-ur}` pairs of IL-EN type performs slightly better than those in EN-IL counterpart. `en-ta` projection results are better in EN-IL category than in the IL-EN category. Especially in the corpus type, we cannot make any precise generalization for the reason that the individual parallel datasets in EN-IL and IL-EN are not same (and their size too). But what could be interesting based on the results is that, the parallel corpus obtained as quickly as IL-EN (in one month or so through crowdsourcing) can be useful in projection based dependency transfer. Out of five language pairs in this corpus type, projection accuracy for two language pairs (`{en-te, en-ur}`) outperform the left/right baseline. In the EN-IL category, only `en-ta` outperforms the left/right baseline.

[10]Both word alignment settings are available in Berkeley aligner.

Corpus type	Lang. pair	HMM (syntax)		HMM	
		reparse	reparse_{10}	reparse	reparse_{10}
EN-IL	en-hi	25.3	28.4	26.9	32.2
	en-ta	**54.2**	52.9	**52.9**	52.0
	en-ur	42.9	49.0	42.9	48.4
IL-EN	en-bn	52.3	52.8	52.2	52.8
	en-hi	26.1	31.8	26.7	34.6
	en-ta	49.8	47.7	46.8	45.7
	en-te	**70.3**	71.6	**66.8**	68.0
	en-ur	**43.7**	49.2	**43.2**	48.4

Table 6.4: Accuracy of projected parsers (target) when tested with predicted POS tags. We distinguish two projection settings with respect to parallel corpus word alignment: (i) alignment using HMM (syntax) and (ii) alignment using HMM. *reparse* - test sentences of all length; $\textit{reparse}_{10}$ - test sentences of length 10 or less.

Corpus type	Lang. pair	HMM (syntax)		HMM	
		reparse	reparse_{10}	reparse	reparse_{10}
EN-IL	en-hi	25.7	29.0	27.3	32.2
	en-ta	53.8	55.4	53.2	55.7
	en-ur	42.6	48.8	42.9	48.4
IL-EN	en-bn	52.6	53.0	51.0	51.3
	en-hi	26.3	32.6	27.0	35.3
	en-ta	51.0	50.9	47.4	50.0
	en-te	70.2	71.1	65.4	66.5
	en-ur	43.9	50.0	43.2	48.7

Table 6.5: Accuracy of projected parsers (target) when tested with gold POS. We distinguish two projection settings with respect to parallel corpus word alignment: (i) alignment using HMM (syntax) and (ii) alignment using HMM. *reparse* - test sentences of all length; $\textit{reparse}_{10}$ - test sentences of length 10 or less.

So, how these results can actually fare against supervised parsers? Given the fact that we did not implement any transformation rules to transform projected trees or parsed outputs, the numbers are far below than the supervised counterparts. Figure 6.5 shows the accuracy of IL supervised parsers trained with different training data size (from the training section of the original treebanks). We trained supervised

parsers (*second-order* MSTParser in both *projective* and *non-projective* setting) for each ILs for training data size ranging from 10 sentences to full available training data. One can see from Table 6.6 that supervised IL parsers trained with just 10 labeled sentences achieve around 87% accuracy (on average) of supervised parsers trained with full training data (See Table 5.2).

Lang.	Full-model	10-sen. model	(%) Retain by 10-sen. model
bn	72.2	57.3	79.4
hi	80.2	68.8	85.8
ta	60.7	57.3	94.4
te	83.1	74.2	89.3
ur	66.3	57.9	87.3

Table 6.6: Comparison of IL supervised parsers trained with just 10 labeled sentences against parsers trained with full training data.

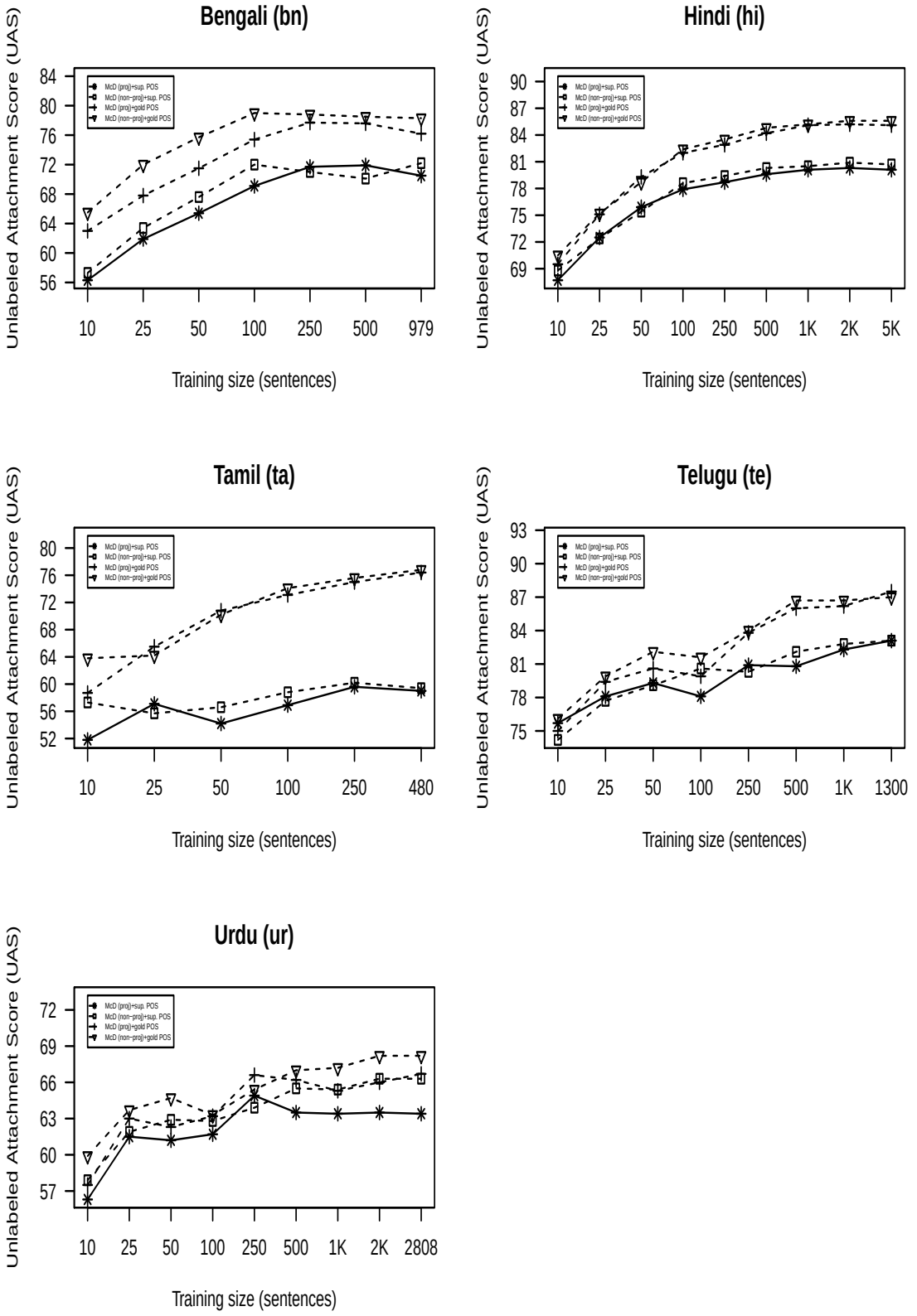


Figure 6.5: Supervised parsing accuracy for various training data size

6.2 Transfer using delexicalized parsers

In the projection based approach we have discussed in the previous section (Section 6.1), the projected structures (as a result of projection alone or from parsers trained from projected treebanks) for the target language sentences will have some systematic differences with respect to the target treebank annotation style since the projected structures follow source treebank annotation style. One must implement some transformation rules similar to [Hwa et al., 2005] to overcome the drop in evaluation accuracy due to annotation differences. Other alternative is to map both source and target annotation styles to adhere to a common standard [Zeman et al., 2012, McDonald et al., 2013].

Delexicalized parsing is the method of parsing target language sentences using a parser trained on a source language treebank. The parser trained on the source treebank does not use source wordforms (hence the name delexicalized), but relies only on POS tags associated with the source trees. Such a parser can be used to parse target language sentences provided target language sentences too are tagged with the same POS tagset as that of the one used in the source treebank.

It has been shown in earlier works such as [Petrov et al., 2012, McDonald et al., 2013] that mapping the annotation to common annotation style helps various tasks including POS tagging, grammar induction and cross-lingual dependency transfer. [Zeman and Resnik, 2008] showed that harmonizing POS helps to induce parsers for closely related languages. To our best knowledge, this approach has not been tried out for ILs. Thus, the goal of this section is to map source treebanks and target IL treebanks to a common annotation style, so that to make delexicalization based transfer viable for ILs.

Section 6.2.1 briefly talks about differences in annotation in treebanks; Section 6.2.2 explains the mapping of POS tagsets used in IL treebanks to a common POS tagset; and Section 6.2.3 discusses mapping dependency structures in IL treebanks to a common annotation style.

6.2.1 Different annotation styles

In this subsection, we analyze annotation style for two commonly found syntactic phenomena across different treebanks. All references to treebanks in this subsection actually points to original treebanks in HamleDT (refer Table 5.3).

Adposition-noun pattern Prepositional or postpostional phrases are found in almost every languages, and they also occur very frequently. Sometimes their representation at the structural level may vary according to the annotation style. Most treebanks treat prepositions/postpositions as the head of the phrases. There are treebanks such as `hi` in which adpositions are governed by nouns. Telugu tree-

bank (`te`)[11] did not have any occurrence of adposition-noun combination in the data. Most treebanks (except `fi, hi`) in HamleDT annotate adpositions (pre and postpositions) as the head of adpositional phrases. Table 6.7 lists the way various treebanks annotate adpositional phrases. Also, Table 6.8 quantifies the fraction of times adpositions occur in treebanks and their status, i.e., whether they occur as parent or a child.

Coordination pattern Coordination structures are usually found in all treebanks. Coordinations are one of the complex syntactic phenomena as far as treebanks are concerned. Coordinations are annotated in at least 3 major styles (each having its own variations) as listed by [Popel et al., 2013],

- PDT style (Prague family)
- Me'lčuk style (Moscow family)
- Stanford style (Stanford family)

These variations in annotation styles would certainly impact projection or direct use of other language parsers to parse resource-poor languages. Table 6.7 lists different coordination styles used by various treebanks in HamleDT.

[11] In the treebank data, adpositions are part of lexical features and they are not shown as separate wordforms.

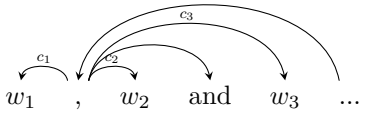
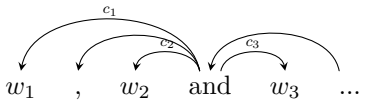
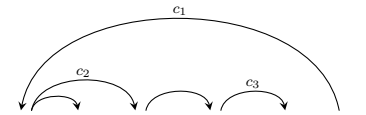
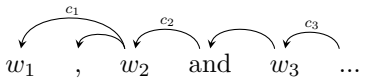
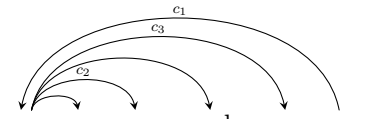
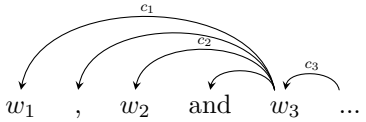
Pattern	Treebanks	Structure
adposition-noun	ar, bg, ca, cs, da, de, el, en, es, et, eu, fa, grc, he, hu, is, it, ja, la, nl, pl, pt, ru, sl, sk, sv, ta, tr, zh	
	fi, hi	
coordination-structure [Popel et al., 2013] (Prague family)	ar, bn, cs, nl, en, el, eu, grc, hi, la, ro, sl, ta, te	 
(Moscow family)	de, fa, ru, sv, tr	 
(Stanford family)	bg, da, fi, it, pt, es	 

Table 6.7: Annotation differences in treebanks

6.2.2 POS harmonization

One of the requirements in delexicalized parsing is the common POS tagset. Common POS tagset is essential for both source side parser training and target side parsing. HamleDT harmonizes treebanks by mapping POS, dependency relations and dependency structure to PDT style annotation. Luckily, four of the five treebanks in Table 5.2 (except `ur`[12] treebank) are already POS and dependency harmonized in HamleDT. So, we make use of them for our delexicalized parser experiments. For the clarity of purpose, we discuss how the POS harmonization is done for ILs.

Treebank	Orig. size	Interset fine	Interset coarse
bn	21	29	12
hi	36	344	12
ta	219	79	10
te	23	58	12
ur	29	10	10

Table 6.9: Tagset size: original vs. harmonized

Table 6.9 shows the POS tagset size of original and harmonized treebanks (measured only from the training section). Tagset size of original treebanks `{bn, hi, te}` and `ur` are relatively lower than the `ta` treebank. *AnnCorra* is a POS/chunk tagset standard [13] for Indian languages. AnnCorra POS tagset is similar to PennTagset in terms of tags (most of them are reused) and its compactness. Treebanks `{bn, hi, te}` and `ur` use AnnCorra scheme whereas `ta` uses PDT style positional tags for the treebank POS annotation. In addition to POS tags, AnnCorra also includes chunk level tags in the annotation. These tags (such as noun chunk, verb chunk, etc.) are included in treebanks `{bn,te}` as coarse-grained tags (CPOSTAG column in the CoNLL data) and in `hi` as one of the feature (through `chunkId, chunkType` in FEATS column) in the CoNLL data. The Table 6.9 also shows the tagset size of treebanks after harmonization (columns 3 and 4). For all the treebanks, harmonized fine-grained Interset tagset is larger than the original POS tagset. Since each fine-grained Interset tag is a detailed morphological tag and each position represents a particular morphological phenomenon, it is easy to extract coarse-grained tags to tailor to the needs of different NLP tasks. In HamleDT treebanks, coarse-grained Interset tags are extracted from the 1^{st} position of fine-grained Interset tags.

[12]We harmonized original `ur` POS tagset to Interset POS tagset. We mapped only major POS information to Interset tagset, at the moment, the POS harmonization does not make use of features supplied by the `ur` treebank.

[13]AnnCorra: POS tagging guidelines for Indian languages - `http://ltrc.iiit.ac.in/tr031/posguidelines.pdf`

treebank	parent/child tag	adposition status	frequency
ar	prep / noun	parent	10.91%
bg	R / N	parent	12.40%
bn	**NNP / PSP**	**child**	**00.02%**
ca	s / n	parent	12.61%
cs	R / N	parent	08.12%
da	SP / N	parent	06.00%
de	APPR / NN	parent	07.57%
el	AsPp / No	parent	06.59%
en	IN / NN	parent	07.12%
es	s / n	parent	12.07%
et	prp / n	parent	01.75%
eu	-	parent	-%
fa	PREP / N	parent	09.96%
fi	**N / PP**	**child**	**00.11%**
grc	r / n	parent	03.08%
hi	**NN* / PSP**	**child**	**17.14%**
hu	S / N	parent	01.61%
it	E / S	parent	12.28%
ja	P / N	parent	09.25%
la	r / n	parent	04.32%
nl	Prep / N	parent	08.17%
pt	prp / n	parent	10.14%
ro	-	parent	-%
ru	PR / S	parent	11.26%
sk	-	parent	-%
sl	Adposition / Noun	parent	06.02%
sv	PR / NN	parent	06.01%
ta	P / N	parent	02.40%
te	**-**	**child**	**-%**
tr	Postp / Noun	parent	01.26%

Table 6.8: Adposition-noun patterns in various treebanks

Harmonized fine-grained Interset tagset size of `hi` is larger than `bn,te`. This could be attributed to more detailed inclusion of morphological features in `hi` treebank annotation with in comparison to `bn,te`. Moreover, `hi` treebank has been evolving and the treebank comes from 2012 shared task whereas both `{bn,te}` come from 2010 ICON shared task. For all harmonized treebanks, the coarse-grained Interset tagset size is similar to universal POS tagset.

The Table 6.10 shows mapping of original treebank POS tags to Interset's feature `pos`. Each column header indicates a particular `pos` value and each cell has native POS tags that are mapped to that `pos` value. This is just an intermediate step. Interset features are then converted to Interset tags through drivers. The harmonized POS tags will be as detailed as the original tags, but for our purpose of projection experiments, we will be using only coarse-grained part of the harmonized POS tags. As can be seen from the table, there are some slight variations among treebanks (`{bn,hi,te,ur}`) on the mapping of original POS tagset to harmonized tags although they have the same annotation style. The conversion of Tamil treebank (`ta`) is fairly simple since the original treebank (TamilTB) already uses positional tags[14]. Note that, unlike other IL counterparts, `ur` does not explicitly mark the preposition/postposition at the POS level.

[14]However, only part of the TamilTB 1.0 tagset (only essential information such as POS and subPOS) has been ported to Interset. That's why the numbers in Table 6.9 differ a lot. Full driver version will be made available soon.

Table 6.10: Mapping from original POS tagset to Interset feature (pos) mapping.

Treebank	noun	adj	num	verb	adv	prep	conj	part	int	punc
Bengali (bn)	NN, NNC, NNP, NNPC, NST, NSTC, PRP, PRPC, WQ, ECH, XC	DEM, JJ, JJC, QF, QFC	QC, QCC, QO	VM, VMC, VAUX	RB, RBC, RDP, INTF, NEG	PSP	CC, UT	RP	INJ	SYM
Hindi (hi)	NN, NNC, NNP, NNPC, NST, NSTC, PRP, PRPC, RDP, WQ, XC	DEM, JJ, JJC, PRP, RDP, WQ, XC	QC, QCC, QO, QF, QFC, QO, RDP, UT, XC	RDP, VAUX, VM, VMC, XC	INTF, RB, RBC, RDP, XC	PSP, XC	CC	NEG, RP	INJ	SYM
Tamil (ta)	N, R	D, J, Q	U	V	A	P	C	T	I	Z

continued on next page

Table 6.10 – *continued from previous page*

Treebank	noun	adj	num	verb	adv	prep	conj	part	int	punc
Telugu (te)	NN, NNC, NNP, NNPC, NST, NSTC, PRP, PRPC, WQ, ECH, XC	DEM, JJ, JJC, QF, QFC	QC, QCC, QO	VM, VMC, VAUX	RB, RBC, RDP, INTF, NEG	PSP	CC, UT	RP	INJ	SYM
Urdu (ur)	NN, NNC, NN-Ez, NNP, NNPC, NNZ, NST, PRP	DEM, JJ, JJC, JJZ	QC, QF, QO	VAUX, VM, VC	RB, RBC	–	CC	NEG, RP	INJ	SYM

6.2.3 Dependency harmonization

The previous section outlined how POS tagset of IL treebanks are mapped to common Interset tagset. This section describes about the harmonization of dependency structure of IL treebanks. The goal of this approach is to minimize errors that occur due to annotation differences at structural during delexicalized transfer. For example, if one were to parse Hindi sentences using the delexicalized source parser trained on Czech (`cs`) treebank, the parser would always make postpositions the head of postpositional phrases. But the Hindi (`hi`) treebank makes the noun argument the head of postpositional phrases (Table 6.7). So, the evaluation would often underestimate this type stylistic variations due to annotation differences. This type of errors can be avoided if treebanks follow same convention regarding annotations.

The delexicalized transfer approach in this section requires both source and target IL treebanks to be harmonized at both POS and structural level. IL treebanks (Table 5.2) except Urdu (`ur`) have already been harmonized in addition to dozens of other treebanks in HamleDT. We will briefly describe the dependency harmonization of IL treebanks in HamleDT.

Lang.	No punc	With punc
bn	99.9	99.1
hi	58.0	56.3
ta	100.0	99.8
te	100.0	99.4
ur	-	-

Table 6.11: UAS scores between original and harmonized treebanks

Table 6.11 compares original IL treebanks and harmonized IL treebanks. Evaluation with and without punctuation are shown under columns 3 and 2 respectively. The table shows that Hindi (`hi`) treebank is the most affected by harmonization at structural level. A harmonized Hindi tree is shown in Figure 6.6 along with the original tree.

Tamil (`ta`) treebank did not require too much harmonization since the treebank already follows Prague dependency style annotation for most of the syntactic phenomena. Harmonized Bengali (`bn`) and Telugu (`te`) treebanks too didn't change from their original structures. At the moment, Urdu (`ur`) treebank has not been harmonized at the dependency level.

6.2.4 Experiments

The goal of this experiment is to parse ILs using other language parsers (delexicalized). Delexicalized parsing [Zeman and Resnik, 2008, McDonald et al., 2011] is

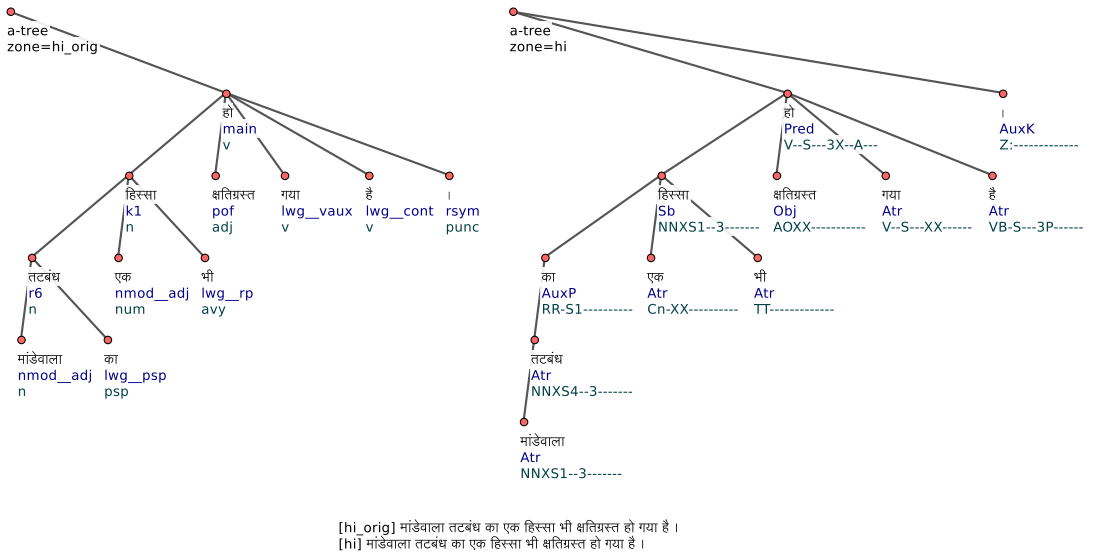


Figure 6.6: Hindi dependency tree: original vs. harmonized

another simple but effective transfer-based approach that requires source language POS tagger and dependency parser, and target language POS tagger. In general, parsing target sentences using delexicalized source parser can be described as follows,

1. Training delexicalized source parser
    - Convert or harmonize the training data POS (source) with some common POS tagset.
    - Take out the wordforms in the training data and replace them with POS tags or simply map all the wordforms to some character (such as '_').
    - Train the parser (for example: MST)
2. Parse target test sentences
    - Convert or harmonize the test data POS (target) with the common POS tagset.
    - Take out the wordforms in the test data and replace them with POS tags or simply map all the wordforms to some character (such as '_').
    - Parse the test data with the delexicalized source parser obtained from Step 1.
    - Replace the test data wordforms with original wordforms.

We perform delexicalization experiment in two settings:

1. with POS harmonization
2. with POS and dependency harmonization

POS harmonization

In this experiment, we use the source structure as it is from the native treebank annotation style, but we map only source treebank POS tags to PDT style tags (only the 1^{st} position). We then train a regular parser on the POS harmonized data, however, without source forms. For testing, we first tag the target test data and parse the target sentences with the trained model (again by removing target wordforms before the parsing step). It must also be noted that the target tagset should match the harmonized POS tagset – in our case, PDT tagset. Target tagging step can be done in a number of ways,

1. train harmonized POS tagger and tag target sentences directly with the harmonized POS tagger.

2. train a POS tagger with the native POS tagset, and perform harmonization after tagging the target sentences with the native POS tagger.

3. obtain harmonized POS tags for target sentences in an unsupervised manner, for example, by projecting the source tags onto the target—similar to [McDonald et al., 2011].

The first two methods require annotated data (from target language) for training the POS tagger. The third method requires only annotated data in source language. We do target tagging according to (2).

For this experiment, we show results for parsing five ILs (target) using delexicalized parsers trained on 30 language treebanks (source). We use POS harmonized treebanks to train delexicalized parsers. POS harmonized treebanks are obtained from HamleDT 2.0 [Zeman et al., 2012] which is both POS and dependency harmonized to PDT style annotation. This experiment requires only harmonization of POS tags of the original treebanks. So we replace original POS tags with harmonized POS tags in the original treebanks. We use coarse-grained harmonized POS tag instead of full length PDT style tag.

Results for this experiment is described in Section 6.2.5 (Tables 6.13, 6.14 and 6.15). Four (`bn, hi, ta, te`) out of five treebanks mentioned in Table 5.2 are harmonized in the HamleDT. For `ta` (as a source), we harmonize the current version of the Tamil dependency treebank (TamilTB 1.0) and train delexicalized parser on it (HamleDT 2.0 contains harmonized version of TamilTB.v0.1. This makes sense because `ta` treebank mentioned in Table 5.2 uses TamilTB 1.0 data). `ur` is not part of HamleDT yet, however, we do POS harmonization separately (because `ur` test data needs to be POS harmonized before parsing).

POS + dependency harmonization

In this experiment, we train delexicalized parsers that are both POS and dependency harmonized. This experiment is similar to the one given in Section 6.2.4, but

it uses fully harmonized HamleDT 2.0 treebanks. We train delexicalized parsers on HamleDT 2.0 treebanks, again by replacing forms by coarse-grained harmonized POS tags. We harmonize target test data (POS and dependency structure) before parsing with delexicalized parsers. We do not have dependency harmonization for `ur`. So, the results do not include `ur`. Results for parsing four ILs (POS + dependency harmonized test data) using delexicalized parsers (trained on HamleDT 2.0, POS + dependency harmonized) are given in Tables 6.13, 6.14 and 6.15, and the results are discussed in Section 6.2.5.

6.2.5 Results

This section contains results for the following configuration,

1. results for supervised parsers trained on POS harmonized and POS+dependency harmonized treebanks - Table 6.12

2. parsing ILs (target) using delexicalized parsers trained on HamleDT 2.0 (source) - Section 6.2.5

   - target test data is POS harmonized
   - POS harmonized original treebanks from HamleDT 2.0 are used to train delexicalized parsers

3. parsing ILs (target) using delexicalized parsers trained on HamleDT 2.0 (source) - Section 6.2.5

   - target test data is POS + dependency harmonized
   - Harmonized treebanks from HamleDT 2.0 are used to train delexicalized parsers

Lang.	POS harmonized				POS+dep harmonized			
	left	right	pred.	gold	left	right	pred.	gold
bn	53.6	04.6	72.1	77.7	53.6	04.6	73.0	77.8
hi	24.4	27.3	76.0	78.5	53.3	07.7	75.8	78.4
ta	50.9	09.5	57.6	67.2	50.9	09.5	58.7	68.6
te	65.8	02.4	82.6	86.2	65.8	02.4	83.1	86.2
ur	42.9	06.3	-	-	-	-	-	-

Table 6.12: Supervised parser results: POS harmonized vs. POS+dep harmonized

Delexicalization: POS harmonization

We provide two kinds of results:

1. results for test data of all lengths
2. results for test data of length 10 or less

All the delexicalized parsers are trained with non-projective setting. We removed punctuations during evaluation. The Table 6.13 shows results for test sentences of all lengths. The last row is an average unlabeled attachment score (UAS) for each IL over 30 delexicalized parsers. Based on the average[15], only `hi` crosses the left baseline results (shown in Table 6.12, left part). Source languages come from Indo-European (21), Uralic (3), Dravidian (2), Altaic (1), Japonic (1), language isolate (1) and Semitic (1) families. Target languages come from Indo-European (3) and Dravidian (2) families. The usual expectation is that if a source and a target language belong to same sub-branch or family, parsing the target language using a closely related source language parser can output better parse trees (since languages are syntactically closer to each other within the same family).

When we look at only closely related source languages for target ILs (i.e., `bn-hi, bn-ta, bn-te, bn-ur, hi-bn` and so on) in Table 6.14, there's a big gain in the average accuracy (44.1% to 59.6% for `bn`, 26.5% to 33.0% for `ta` and so on)for all ILs except `hi` for which the average accuracy drops from 30.2% to 28.0%. `hi` does not seem to benefit from closely related source languages, in fact, language isolate `eu` as source gives the best result (49.2%). In Table 6.14, parsers trained from `{eu, hu, la, tr}` are overall best source to parse ILs (each source crosses the left baseline for at least 3 target ILs). `{ro, de}` source parsers did not cross the left baseline for any of the target ILs. Most of the source languages (mostly Indo-European) did not cross the left baseline for most ILs.

The average accuracy in **bold** for ILs in Table 6.14 indicate that the numbers are higher than the left baseline in Table 6.12. Within ILs, `{hi, ta}` seem to act as best source for each other, and similarly `{bn, te}` too act as best source for each other. `ur` benefits from `{bn, te}`. This is surprising given the fact that `ur` is much closer to `hi` in terms of language relatedness than `bn` or `te`.

Measuring accuracy on limited sentence lengths (10 or less) brought slight improvements over full length sentences, with `{hi, ur}` gaining more than other ILs. The results are shown in Table 6.15.

POS tags play a central role in delexicalized dependency transfer. We measured accuracies for parsing ILs using both gold tags (harmonized) as well as tags obtained from target taggers. Remember that the target language POS taggers (Section 6.1.3) are trained with their native tagset. Harmonized or Interset tags for predicted native tags are obtained by applying harmonization (through Interset drivers). Harmonized gold POS tags for target sentences are already available

[15]The average calculation does not include $bn \rightarrow bn$, $hi \rightarrow hi$ and so on.

Lang.	Family	← bn →		← hi →		← ta →		← te →		← ur →	
		D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}
ar	Semitic	23.1	22.4	26.5	14.4	11.7	11.9	35.4	35.9	23.4	-
bg	IE	32.1	39.5	36.9	18.1	17.7	19	50.4	59.4	21.2	-
bn	IE	70.8	71.6	27.8	33.1	34.8	36.1	78.6	78.2	58.6	-
ca	IE	49.3	53.7	26.4	14.5	14.2	15	70.3	73.9	35.1	-
cs	IE	38.1	39.3	32.9	20.2	18	18.1	53.1	53.5	27.6	-
da	IE	35	40.8	30.1	28.4	23.2	33.9	40.6	45.9	35.8	-
de	IE	41.2	45.6	22.1	27.3	30.2	31.9	56.3	58.3	32.4	-
el	IE	39.8	41.4	28.5	17.2	27.3	27.1	57.2	58.2	33.5	-
en	IE	40.9	44.7	44.4	22.8	35.5	29.5	55.1	57.2	32.4	-
es	IE	48.7	50.7	26.6	15.3	15.5	16.4	65.4	74.7	35.2	-
et	Uralic	51.1	52	27	47.6	37.2	36.3	66.4	63.6	43	-
eu	L. Isolate	54.4	56.4	49.2	39.3	47.4	47.8	66.3	67.5	48.2	-
fa	IE	43.7	43.6	27.8	23.4	15.1	16.2	63.7	65.4	28.6	-
fi	Uralic	49.6	53.6	38.8	46.8	38.6	42	59.9	65.8	37.9	-
grc	IE	54.4	55.9	21.9	29	32.5	32.3	64.1	71.5	41.3	-
hi	IE	57.7	55.1	79.6	80.2	41.6	46.4	67.6	68.3	48.7	-
hu	Uralic	53.2	58.9	26.8	50	34.7	38.1	65.9	66.3	47	-
it	IE	39.8	44.7	25.8	11.7	16.1	16.4	57.2	59.9	24.6	-
ja	Japonic	53.6	55	24.1	50.7	43.5	44.2	68.3	70.7	37.1	-
la	IE	55.4	54.6	24.7	24.7	29.1	28.1	67.5	67.5	51.3	-
nl	IE	33.7	38.9	32.7	22.5	21.3	22.9	48.7	54.3	38.1	-
pt	IE	20.9	23.7	34.2	20.7	17.9	19.5	27.2	29	13.3	-
ro	IE	31.2	31.8	22.7	8.8	7.1	7.4	50.6	50.9	17.4	-
ru	IE	36.9	37.9	31.8	25.9	17.6	18.9	56.7	58.9	29.6	-
sk	IE	35.4	38.4	34.3	21	22.5	22.4	54.5	55.5	25.4	-
sl	IE	39.5	41.4	29.5	15.5	12.9	12.5	60.4	58.5	22.9	-
sv	IE	38.9	43.6	44.4	27.4	37.6	38.4	49.4	52.6	30.2	-
ta	Dravidian	57.3	55.9	34.2	61.7	58.4	58.5	69.1	69.6	42.6	-
te	Dravidian	63.9	62.4	21.9	24.1	22.5	26.1	82.6	82.6	53.9	-
tr	Altaic	59.6	59.2	22.2	51.1	45.5	46.4	71.7	71.2	45	-
avg		44.1	46.2	30.2	28	26.5	27.6	58.5	60.8	35.4	-

Table 6.13: Delexicalized parser results. D_P - POS harmonization; D_{PD} - POS+dependency harmonization; IE - Indo-European; L. isolate - language isolate;

in the HamleDT distribution. Figure 6.8 compares average accuracy for ILs when parsed with predicted and gold POS tags. Bengali (`bn`) and Tamil (`ta`) show visible differences in the average accuracy, where, for the `bn` treebank, it is from 45% (pred POS) to 50.7% (gold POS), and for the `ta` treebank, it is from 27.6% (pred POS) to 32.1% (gold POS). Figure 6.7 shows individual parsing accuracies for Tamil (`ta`) when parsed with delexicalized source parsers in both the settings. For some languages, the gap between some source parsers when parsed with gold POS gives better accuracy than parsing with predicted POS.

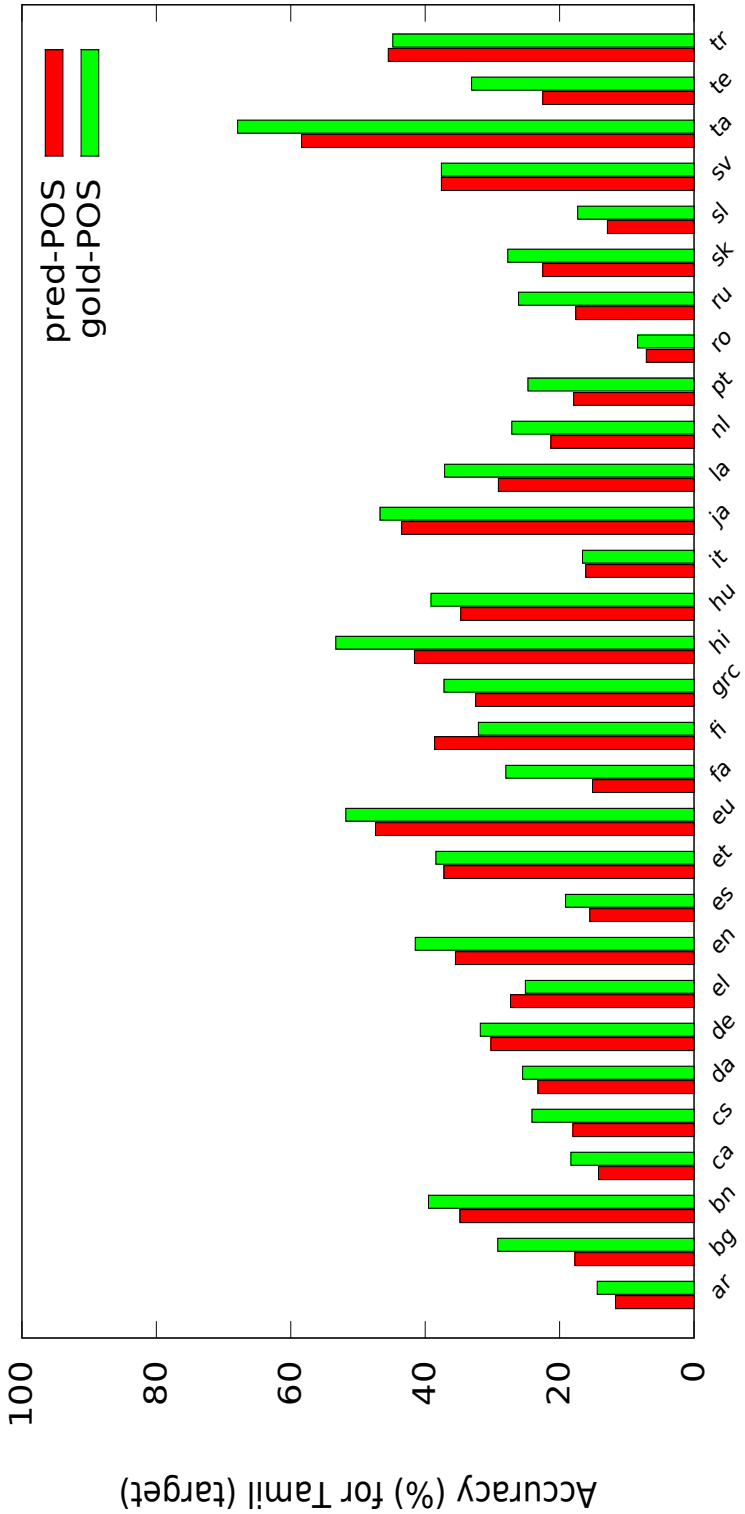


Figure 6.7: Parsing Tamil with POS harmonized delexicalized parsers: predicted POS vs. gold POS

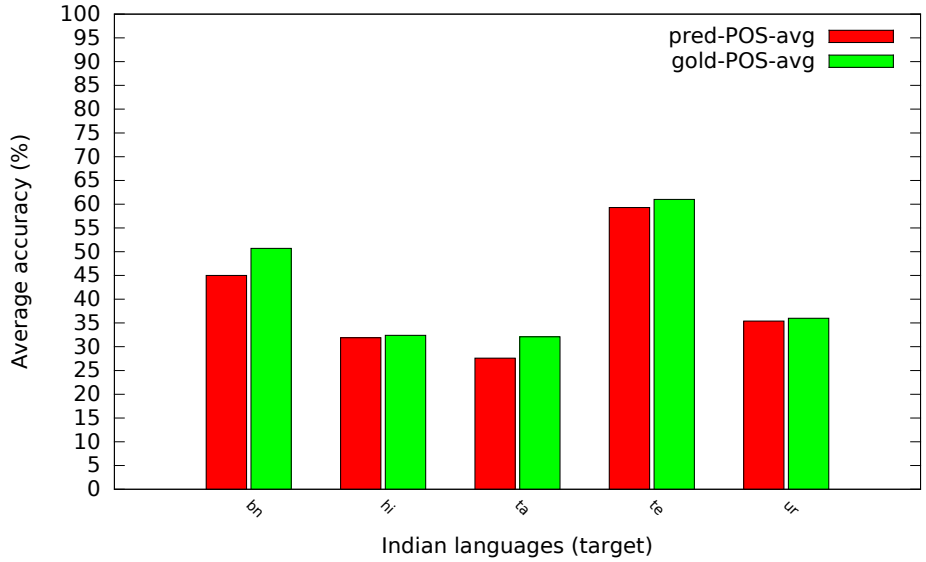


Figure 6.8: Parsing ILs with POS harmonized delexicalized parsers (average accuracy over 30 parsers): predicted POS vs. gold POS

Delexicalization: POS + dependency harmonization

This section contains results for transferring delexicalized parsers from HamleDT 2.0 treebanks that are both POS+dependency harmonized to target ILs. The Tables 6.13 and 6.15 show the final results for full length test sentences and sentences of length 10 or less. Supervised parser results for POS+dependency harmonized treebanks are given in Table 6.12 (last four columns). We do not have fully harmonized `ur` treebank, so we are not able to provide results for `ur` in this experiment.

One notable difference as a result of dependency harmonization in addition to POS harmonization is the increase in left baseline score for `hi` (from 24.4% to 53.3% - Table 6.12). From the baseline results shown in Table 6.12, one can see, `hi` is the most affected treebank after harmonization in terms of changes in edges (includes making postpositions to be the head of postpositional phrases). Other treebanks such as `{bn, ta, te}` are not heavily harmonized at the structural level. Tables 6.9 and 6.11 quantify these changes at POS and dependency structural level.

There's an upward trend in average accuracy (for `{bn, ta, te}`) from POS only harmonization to POS+dependency harmonization. However, in comparison with supervised results (Table 6.12), only two languages `{bn, te}` cross the left baseline. Unlike the original treebank, POS+dependency harmonized `hi` treebank is more left-branching.

There's also a clear distinction between non-IL source languages (Table 6.13 excluding ILs) and IL source languages (Table 6.14) in terms of average accuracy. Taken individually within ILs as source languages, `{hi, ta}` seem to act as best source for each other, and similarly `{bn, te}` act as best source for each other.

Lang.	← bn →		← hi →		← ta →		← te →		← ur →	
	D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}
bn	70.8	71.6	27.8	33.1	34.8	36.1	78.6	78.2	58.6	-
hi	57.7	55.1	79.6	80.2	41.6	46.4	67.6	68.3	48.7	-
ta	57.3	55.9	34.2	61.7	58.4	58.5	69.1	69.6	42.6	-
te	63.9	62.4	21.9	24.1	22.5	26.1	82.6	82.6	53.9	-
avg	**59.6**	**57.8**	**28.0**	39.6	33	36.2	**71.8**	**72.0**	**51.0**	-

Table 6.14: Delexicalized parser results in the case of ILs as source. D_P - POS harmonization; D_{PD} - POS+dependency harmonization;

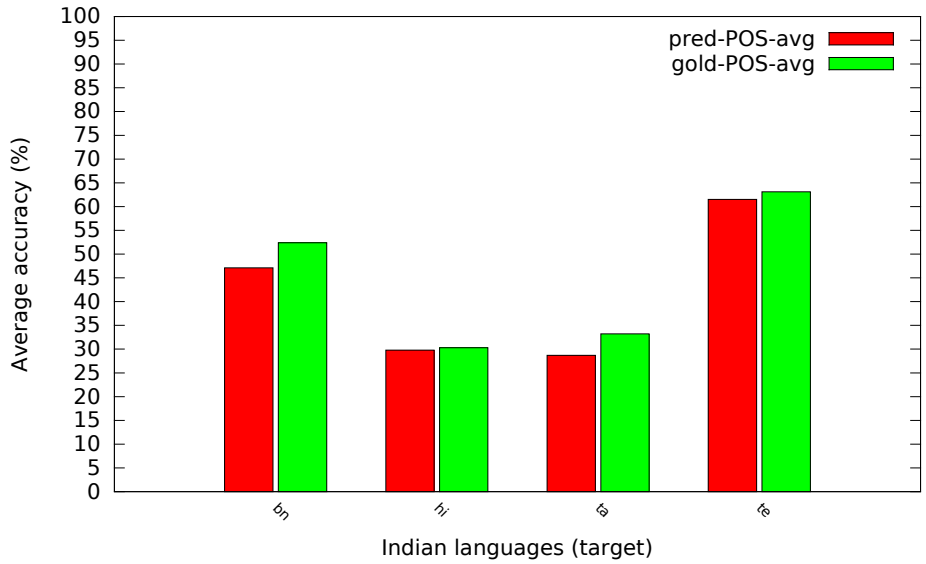


Figure 6.9: Parsing ILs with POS+dependency harmonized delexicalized parsers (average accuracy over 30 parsers): predicted POS vs. gold POS

Lang.	Family	← bn →		← hi →		← ta →		← te →		← ur →	
		D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}	D_P	D_{PD}
ar	Semitic	23.2	22.8	34.5	26.6	16	16	35.9	36.4	28.1	-
bg	IE	33.2	40.6	46.2	30.4	26.6	26.3	51.7	60.6	25.7	-
bn	IE	71.6	72.2	35.1	36.7	35.7	38.3	79.7	79.5	68.7	-
ca	IE	50.4	54.9	31.6	21	16.6	18	71.3	74.8	45.1	-
cs	IE	38.3	39.4	41	30.9	24.3	24	54.1	54.5	37.8	-
da	IE	36.4	41.1	33.5	34	27.7	36.6	41.7	46.8	43.2	-
de	IE	42	45.7	36.2	40.6	32.9	33.7	57.6	59	41.7	-
el	IE	39.9	41.8	31.5	23.2	30.3	28	58	59	38.5	-
en	IE	41.4	45.1	48.2	21.1	41.4	33.1	55.9	57.3	43.3	-
es	IE	50	51.9	31.7	20.6	18.9	18	66.2	75.3	46.1	-
et	Uralic	51.8	52.2	37	60.2	41.7	40.6	66.5	63.9	50	-
eu	L. Isolate	55	56.6	53.1	45.7	48	48.6	66.7	68.1	52.2	-
fa	IE	44.7	44.6	38.9	38.2	22.9	23.4	65	66.7	36.9	-
fi	Uralic	50.7	54.4	42.3	54.3	37.7	38.3	60.9	66	45.9	-
grc	IE	54.4	56.5	33.5	43.4	36.6	38	65	72.5	53.5	-
hi	IE	58.1	55.7	82.6	83.8	45.1	50.3	68.7	69.4	55.5	-
hu	Uralic	54.6	59	35	58	30.9	35.1	67.8	66.7	56.1	-
it	IE	40.3	45.1	26	12.2	19.7	19.7	58.5	61.1	35.7	-
ja	Japonic	54.4	55.5	30.7	50.2	42.9	43.4	69.9	71.8	44.9	-
la	IE	55.8	55	35.2	38.7	30.3	30.3	68.5	68.3	63.7	-
nl	IE	33.8	38.7	37.6	31	22.9	24.9	49.2	54.6	38.2	-
pt	IE	21.3	23.5	43.7	28.8	23.4	23.4	27.5	29.2	17.9	-
ro	IE	30.9	31.5	26.2	15	9.4	10	51.5	51.8	22.7	-
ru	IE	37.9	38.5	39.3	38	22.6	23.7	57.8	59.9	38.7	-
sk	IE	36.3	39.1	42.9	33.2	28.6	28.9	54.8	55.9	31.6	-
sl	IE	40.4	42.3	35.3	23	16.9	14	61.3	59	30.9	-
sv	IE	39.9	43.9	51.8	38.7	44	45.1	50.3	53.4	40.5	-
ta	Dravidian	58.1	57.1	45.7	70.2	58	57.4	70.6	71.1	49.3	-
te	Dravidian	63.9	62.7	32.9	35	28.3	31.7	83.4	83.4	64.5	-
tr	Altaic	59.8	59.6	28.1	50.4	45.7	44.6	72.7	71.6	52.9	-
avg		44.7	46.7	37.4	36.2	29.9	30.6	59.5	61.5	43.3	-

Table 6.15: Delexicalized parser results (evaluated on target sentences of length 10 or less). D_P - POS harmonization; D_{PD} - POS+dependency harmonization; IE - Indo-European; L. isolate - language isolate;

6.3 Transfer Based on Machine-translated Bitexts

It has been well established from previous works that the availability of bitexts or target POS taggers (or both) are very crucial for transferring dependency parsers from one or multiple source languages. The transfer approaches presented in Sections 6.1 and 6.2 assume the availability of parallel corpora (involving target languages) or target POS taggers.

6.3.1 Approach

Imagine a situation where we don't have direct access to parallel corpora or a target POS tagger but only to a translation system from an under-resourced language to a resource-rich language. For example, Google translation system is available for 80 languages, and for many languages, obtaining parallel corpora is the most complicated task[16]. This presents an interesting scenario for the existing transfer-based approaches. In a situation like this, we propose[17] to combine bitexts obtained from machine translation and target POS tags obtained from unsupervised clusters to transfer dependencies to target languages. Machine translated texts are obtained by translating from a target language to a resource-rich source language (for which parsers and taggers are available).

Our overall approach is similar to [Hwa et al., 2005] and [McDonald et al., 2011]. The heart of our approach lies in how we obtain bitexts and target POS taggers which are crucial for transfer parsers. Unlabeled target texts are available in plenty even for under-resourced languages. We obtain both the resources from unlabeled target texts only. Once we obtain bitexts through MT system, we transfer dependencies to target languages using projection approach (Section 6.1) and delexicalization approach (Section 6.2).

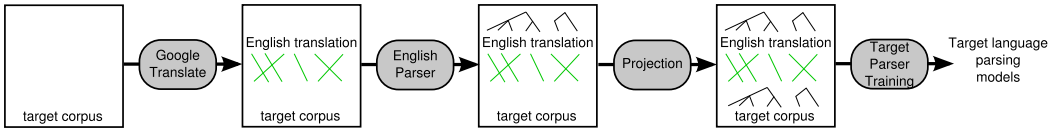


Figure 6.10: Schematic depiction of how the target parse trees are obtained (Credit: David Mareček)

Projection based on machine translated bitexts

We apply the projection approach described in Section 6.1 to bitexts obtained through MT system. We train parsers on target trees obtained through projection. Previous works have mostly relied on using target POS taggers for training

[16]Mainly due to inaccessibility through web

[17]This transfer approach on the whole is a joint work with other colleagues David Mareček and Zdeněk Žabokrtský

the parsers. [McDonald et al., 2011] demonstrated that POS tags alone carries much of the syntactic information and showed that target sentences parsed by other language parsers (through delexicalization) that share only common POS tagset can beat unsupervised accuracy.

To make this approach more broader and applicable in realistic scenarios, we induce target POS information using unsupervised techniques. Unsupervised POS tagging is arguably less complex than inducing tree structures, and previous works on unsupervised POS techniques [Blunsom and Cohn, 2011, Clark, 2000] have proven to be effective even in practical applications. So we use unsupervised target POS tags instead of supervised or universal POS tags, but for the sake of comparison, we also provide results with supervised and universal POS approaches. We experiment with various tagset size and show results for the one that gives the best average accuracy on target languages. This approach can be used within the transfer framework for languages that lack even POS taggers, thus making the approach very suitable for languages that do not have any labeled target language resources. The entire pipeline for the projection approach is shown in Figure 6.10

Delexicalization

We perform delexicalized transfer [McDonald et al., 2011] to machine translated bitexts. We train only English delexicalized parser (source) and parse target languages directly using the source parser. The delexicalized experiments in Section 6.2.4 use Interset based common tagset, but here we use universal POS tagset [Petrov et al., 2012] for the sake of comparison with existing results.

6.3.2 Obtaining machine-translated bitexts

The choice of resource-rich language for our dependency transfer is English. So we translate our target language texts to English. Our system is single source, i.e., all our experiments are carried out with English as a source language against a variety of target languages. Machine translations in English are obtained for two kinds of target language data: (i) for target sentences from treebanks in HamleDT and (ii) for target sentences (belonging to ILs) that already make up a parallel corpus with English (human translated).

English translation for treebank data (HamleDT)

We obtain translations for 18 target language texts (from HamleDT treebanks - Table 5.3). The translations[18] are obtained through Google Translate API[19](GTAPI).

[18]Thanks to Rudolf Rosa for translating the treebank data.

[19]Google Translate API - `https://developers.google.com/translate/`

#	ar	bg	ca	cs	da	de
Sentences	03.0K	13.2K	14.9K	25.6K	05.5K	38.0K
Tokens	116.8K	196.2K	443.3K	437.0K	100.2K	680.7K
Train/Test(%)	96/4	97/3	88/12	99/1	94/6	95/5
#	el	es	et	fi	hi	hu
Sentences	02.9K	16.0K	01.3K	04.3K	13.3K	06.4K
Tokens	70.2K	477.8K	09.5K	58.6K	294.5K	139.1K
Train/Test(%)	93/7	90/10	90/10	90/10	91/9	94/6
#	it	nl	pt	sl	sv	tr
Sentences	03.4K	13.7K	09.4K	01.9K	11.4K	05.9K
Tokens	76.3K	200.7K	212.5K	35.1K	197.1K	69.7K
Train/Test(%)	93/7	97/3	97/3	79/21	97/3	95/5

Table 6.16: Target language treebank texts that are translated to English using Google Translate API. *Tokens* size is shown for the original target language data not for English.

The translations provided by GTAPI[20] also came up with word alignment links between original texts and its translations. Table 6.16 shows the data for which we obtained English translations.

English translation for Indian languages (ILs)

We translate IL texts shown in Table 5.1 (only datasets belonging to IL-EN corpus direction) to English. Due to time constraints, we translated only first 2K sentences for each IL (`bn, hi, ta, te, ur`). Remember the IL texts already have English translations (human translators) for the entire data. We translate IL texts to English via Google MT system[21] through web interface.

6.3.3 Experiments

We transfer dependencies to target languages using projection and delexicalized transfer. English is the source for all the experiments. We train source side tools on translated English texts obtained from MT system.

[20]Only Google Translate API v1 provided word alignment outputs in addition to translated outputs. The current Google Translate API v2 does not provide word alignment outputs.

[21]`https://translate.google.com/`

Projection

The schema of the projection procedure is depicted in Figure 6.10. It consists of four steps:

1. The target language corpus is translated to English using Google Translate API v1. The API provides also alignment links, which will be used for projection. The translated English sentences are then tokenized and the original alignment links are adjusted.[22]

2. English sentences are tagged by Morce tagger [Spoustová et al., 2007] and parsed by the MST parser [McDonald et al., 2005b]. For parsing English, we used the parser model trained on the version of Penn treebank supplied during the CoNLL 2007 shared task [Nivre et al., 2007].

3. English dependency trees are projected to the target language sentences (only the training part of the treebank) using the alignment.

4. Three target parser models are trained (using MST parser with *non-projective, second-order setting*) on the projected target corpus with different POS annotations (next subsection).

The above projection procedure is used in the case of machine translated English-HamleDT data pairs (results in Table 6.17).

For machine translated English-IL data pairs, the projection procedure is slightly different: (i) we use Berkeley aligner for word alignment (since the MT system did not provide word alignment links) (ii) we use target POS taggers (regular supervised, not universal or unsupervised taggers) for testing and (iii) we did not perform delexicalization experiment for this dataset. Projection results for machine translated English-IL pairs are given in Table 6.19.

Training with different POS tags

This procedure is applicable only for English-HamleDT parallel datasets. To tag the target test data, we train two supervised taggers and one unsupervised tagger on the training section of the target treebanks.

- **Supervised POS:** We train Stanford tagger [Toutanova et al., 2003] on the training section of the treebanks.

- **Universal POS:** We first convert the annotated training data to universal POS tags [Petrov et al., 2012] and train the tagger on it.

[22]For instance, when a punctuation is separated from a form on the English side, we link the separated punctuation to the corresponding punctuation on the treebank data.

- **Unsupervised POS tagger:** We use unsupervised HMM POS tagger by Blunsom and Cohn [Blunsom and Cohn, 2011]. Not knowing which tagset size is suitable for the projected treebanks, we obtain POS tags for different tagset size: 20, 40, 80 and 160. Besides the training and testing parts of the treebanks, we used additional monolingual texts from W2C Wikipedia corpus [Majliš and Žabokrtský, 2012a] to enlarge the size of the data to one million words.

Transfer using delexicalized parser

We train delexicalized English parsers under two settings. In the first setting, we convert the POS tags of CoNLL 2007 English data into universal POS tags using the mapping provided by [Petrov et al., 2012], strip all word forms and train the parser. In the second setting, we first tag the English translations obtained by Google Translate using Morce tagger [Spoustová et al., 2007], convert them to universal POS tags and after stripping all word forms we trained the delexicalized parser on it. We obtained target universal POS tags using the universal POS tagger trained from the training part of the target treebanks. So, the main difference between [McDonald et al., 2011] and our approach is that they obtained target POS tags by POS projection from English whereas we used POS information from target language treebanks and trained universal POS taggers on them. At the moment, our experiments on unsupervised POS taggers deal with varying tagset size, and it would also be interesting in the future to make a comparison with different unsupervised approaches such as unsupervised POS projection [Das and Petrov, 2011b]. We apply the procedure described above only to English-HamleDT parallel datasets. Results for this experiment are given in Table 6.18.

Unsupervised parsing

To compare the projection results with a completely unsupervised approach, we use the software for dependency grammar induction of Mareček and Straka [Mareček and Straka, 2013].[23] We run[24] the experiments in the same manner as they described for all our testing languages. We perform this experiment only to English-HamleDT parallel datasets. Results for this experiment are given in Table 6.17.

6.3.4 Results

Our major results for experiments that use English-HamleDT parallel datasets are presented in Table 6.17 and 6.18. Left and right baselines in Table 6.17 indicate that some languages have a strong preference for either left or right-branching.

[23] `http://ufal.mff.cuni.cz/udp/`

[24] Thanks to David Mareček for running unsupervised experiments.

- *sup* presents UAS scores from a supervised parser trained on the training portion of target treebanks.
- *UDP* presents UAS scores achieved in unsupervised dependency parsing [Mareček and Straka, 2013] on the test portion of target treebank data.
- Projected parsers
  - *dir proj* shows UAS scores from directly projecting translated English test data onto target test sentences.
  - *gold*: test data is tagged with gold POS tags before parsing.
  - *sup*: test data is tagged by supervised POS tagger before parsing.
  - *univ*: test data is tagged by universal POS tagger before parsing.
- *unsup40* presents UAS scores for parsing the test data with unsupervised POS tags (tagset size 40). We also experimented with different tagset size: 20, 40, 80 and 160. We chose representative results for different tagset size based on best average accuracy.

Lang.	Baseline		Sup.	UDP	Projected parsers				
	left	right			dir proj	gold	sup	univ	unsup40
ar	05.2	58.8	74.2	34.1	40.3	60.0	**58.3**	56.1	51.8
bg	17.9	38.8	81.5	56.7	41.4	56.3	53.6	**55.3**	46.4
ca	24.7	28.8	87.6	24.6	46.0	60.2	**59.0**	-	56.9
cs	24.1	28.9	74.9	55.0	51.4	62.3	**58.4**	54.7	45.4
da	13.2	47.9	79.8	42.0	36.9	43.1	**42.1**	41.6	41.6
de	24.2	18.9	84.2	43.5	43.0	50.1	**48.7**	47.8	46.4
el	32.0	18.5	78.5	30.9	52.0	65.2	**65.4**	59.2	49.2
es	24.7	29.0	88.1	36.3	47.0	59.0	**58.2**	-	56.9
et	34.1	17.4	72.9	**63.8**	58.0	66.0	58.3	-	54.8
fi	39.3	13.6	60.7	36.7	43.1	46.2	**39.5**	-	37.1
hi	24.4	27.3	75.1	15.6	24.6	28.3	**28.0**	-	24.9
hu	42.8	05.3	75.2	34.7	41.1	53.2	**51.2**	48.2	43.4
it	23.0	37.4	80.5	49.7	53.1	62.0	**59.3**	53.7	55.6
nl	27.9	24.7	76.7	27.6	53.2	61.4	59.3	-	**60.9**
pt	25.8	31.1	84.2	39.8	56.4	66.5	**62.9**	62.2	61.8
sl	24.4	26.6	76.7	45.9	50.4	56.6	**53.1**	45.7	44.4
sv	25.9	27.8	82.7	49.1	48.2	55.8	54.3	**56.8**	53.9
tr	65.1	02.0	75.5	42.3	51.4	57.0	**57.2**	-	46.0
avg	27.7	26.8	78.3	40.5	46.5	56.1	**53.7**	*52.8*	48.7

Table 6.17: UAS for baselines, supervised/unsupervised parsers and various projected parsing models

Lang.	Delex CoNLL	Delex GT	*McD 2011*
ar	29.1	**31.9**	-
bg	**52.8**	51.4	-
cs	**35.6**	35.4	-
da	**44.6**	39.7	*45.5*
de	47.5	**48.1**	*47.5*
el	59.0	**60.5**	*65.2*
es	-	-	*52.4*
hu	45.0	**46.2**	-
it	52.9	**57.2**	*56.3*
nl	-	-	*66.5*
pt	**65.4**	63.5	*67.7*
sl	36.5	**44.2**	-
sv	**57.7**	54.5	*59.7*
avg	47.8	**48.4**	*57.6*

Table 6.18: Delexicalized transfer parsers comparison (unlabeled attachment scores)

For the sake of comparison, we reproduce delexicalized parser results on our data with two settings. (i) delexicalized parser trained on the POS tags from the CoNLL 2007 [Nivre et al., 2007] English data and (ii) delexicalized parser trained on the English translations we obtained from the MT system. The results are shown in Table 6.18. For both settings, we obtain target POS tags in a supervised manner. We also add results from [McDonald et al., 2011] for comparison. One intriguing aspect of this result is that delexicalized parsers obtained from machine translated texts perform better than the delexicalized parser obtained from CoNLL 2007 data (48.4 vs. 47.8). *'McD 2011'* has better overall results compared to our delexicalized parsers. We attribute this to difference in parser training parameters as well as the usage of POS tags projection instead of supervised POS tagger. When we make an overall comparison (Tables 6.17 & 6.18), the advantages of training the supervised taggers (both original/universal POS) are clearly visible. However, *'unsup40'* outperforms *'UDP'* by 8.2% absolute UAS score, and also gives slightly better results with respect to *'delex CoNLL'* and *'delex GT'*. Remember, both *'delex CoNLL'* and *'delex GT'* use supervised target tagger, that means, *'unsup40'* does not use any labeled target resources, but still performs better than other strategies. This clearly shows that, syntactic projection which crucially relies on bitext can still be beneficial even in the absence of high quality bitexts.

	Predicted POS		Gold POS	
Lang.	mt	human	mt	human
bn	47.8	51.8	49.2	50.8
hi	5.4	25.2	5.4	25.3
ta	42.9	45.9	41.4	42.9
te	58.2	65.9	60.5	68.1
ur	41.8	43.6	41.9	43.4
avg	39.2	46.5	39.7	46.1

Table 6.19: Projection results ILs: machine translated vs. human translated; Parallel corpus size: 2K.

In the case of projection of English dependencies to ILs, UAS scores are consistently lower for projection performed from machine translated bitexts. Only for Hindi (`hi`), the gap between projection from machine translated and normal bitexts is high. For other languages, the results are quite comparable between human translated and machine translated texts, however, most results did not even cross left baselines (see Table 6.3 for baseline results). It must be emphasized that the parallel corpus size used for this dependency transfer is only 2K sentence pairs (for both human translated and machine translated experiments in Table 6.19).

Chapter 7

Error Analysis: Projection

This chapter presents general error patterns in the projection based approach. We also analyze alignment error and its impact on the projection. We make this analysis for English to Tamil projection.

7.1 Introduction

In Section 6.1, we projected source trees to target sentences via parallel corpus and their word alignments. Then we trained target parsers on the projected trees. In our experiments, we carried out dependency projection for 8 parallel datasets involving 5 ILs, and for all of them English is the source language.

Errors that occur in projection based approach is multifaceted:

- There are many components in the projection pipeline such as source POS tagging, source parsing, word alignment task in the parallel corpus and so on. Thus, errors can occur in many places and even influence the rest of the tasks in the projection pipeline in an unexpected manner.
- For each component in the projection pipeline, there are number of settings need to be decided and we can even employ different tools for each component.

The error analysis for the projection based approach has to take into account the above points in consideration. In our case, we restrict our error analysis to only target language components, i.e., we do not try to analyze how POS tagging errors in source is causing the drop in accuracy for source parsing and in turn the projection accuracy. We perform error analysis on trained parsing models rather than the raw projected trees. Figure 7.1 shows the parsed output of a target sentence (bottom right) and its gold counterparts. Tamil treebank test data also contains its English translations and manual word alignments[1].

[1]This manual word alignment links are available only for Tamil test data.

7.2 Projection Errors: By POS and Wordforms

Since the projection is the last of the operation in obtaining the target trees, the projection errors may have been caused by any of the preprojection operations such as wrong source parse or wrong alignment links. Apart from that, the errors could also have been resulted simply due to the differing annotation scheme of the source and the target language. This section provides some insights into the patterns of attachment errors in the projection based on POS and form information. The Tables 7.1 and 7.2 show the attachment errors in the target parse trees with respect to POS and form. We parsed target test sentences with the parsing model trained on projected trees.

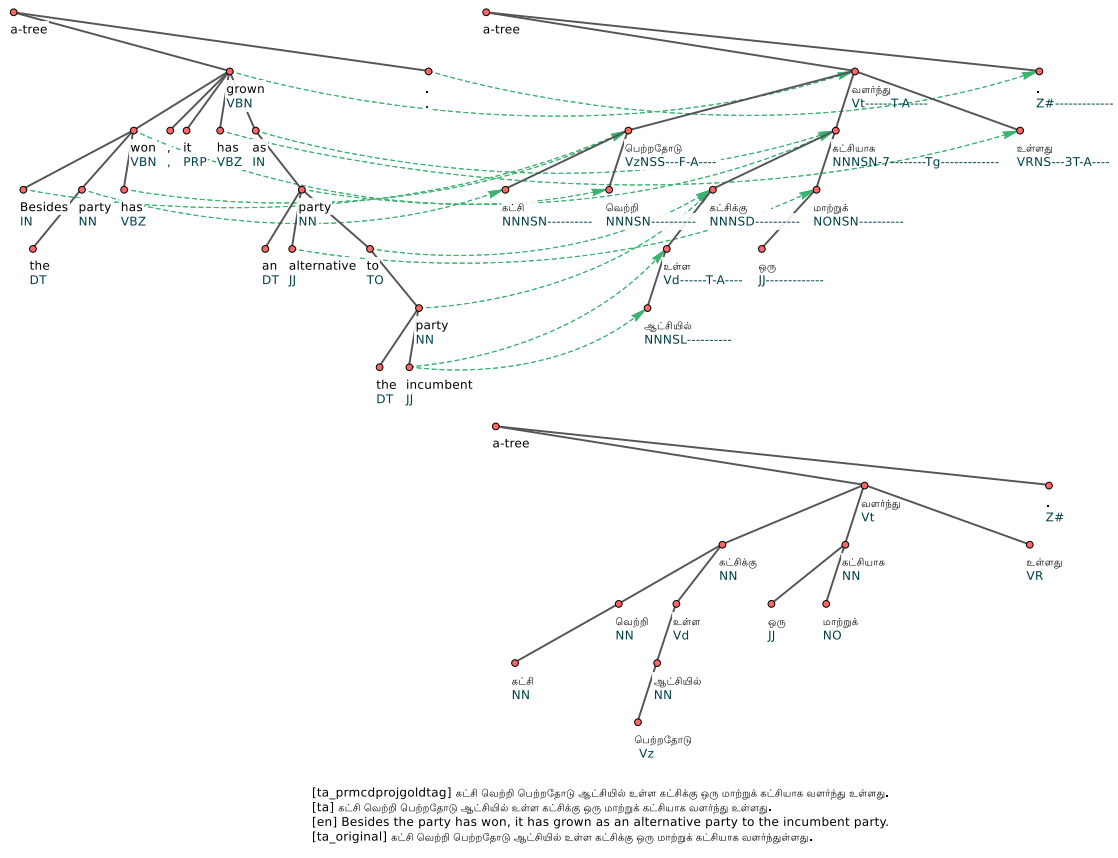


Figure 7.1: Tamil (`ta`): [T, L] - English tree; [T, R] - gold Tamil tree and gold alignment from English tree; [B, R] - parser output from the model trained on projected trees from English.

Note that we use gold POS tags for parsing the target sentences. The reason for this decision is that the parsing models trained on projected trees use supervised POS tags. Please recall that we run target tagger on the target side of the parallel corpus before training the parsers. Using gold POS information in our testing will indirectly

point if there's any systematic POS errors caused during target tagging operation. Also, using gold POS information during target parsing will hardly improve the situation because the target parsers are trained with supervised POS tags.

POS	Occurrences	Edge errors	%
NN	563	335	60.0
NE	134	75	56.0
Vt	134	96	72.0
Z#	124	4	03.0
VR	100	65	65.0
Vu	80	58	72.0
Rp	68	44	65.0
NO	68	17	25.0
Z:	65	53	82.0
Vr	65	31	48.0
PP	59	47	80.0
AA	50	30	60.0
Vd	48	18	38.0
Un	48	24	50.0
JJ	46	16	35.0
Vz	37	23	62.0
Nm	25	20	80.0
Uc	25	10	40.0
VT	24	24	100.0
DD	22	13	59.0
Na	19	6	32.0
Vk	17	17	100.0
QQ	15	1	07.0
VZ	13	12	92.0
CC	8	6	75.0
Summary	1892	1072	57.3

Table 7.1: Attachment errors in the projection by POS

The Table 7.1 shows top 25 POS tags by their frequency in the test data and count of the attachment errors. Target parsing models are trained with POS tags that are of length to minimize further tagging errors. So in the gold sentences too,

we use only first 2 positions of gold tags. Table 7.1 shows the top 2 POS tags have at least 50% attachment errors. Closed class POS tags are most informative in a sense that errors in those categories can be easily identified and rectified by applying special transformation rules for those POS tags.

- punctuations (POS: Z:) are attached often incorrectly 82%
- postpositions (POS: PP) attachment error rate is 80%.
- auxiliary verbs (POS: VT and VR) are often attached incorrectly (with attachment error rates 100% for VT and 65% for VR)

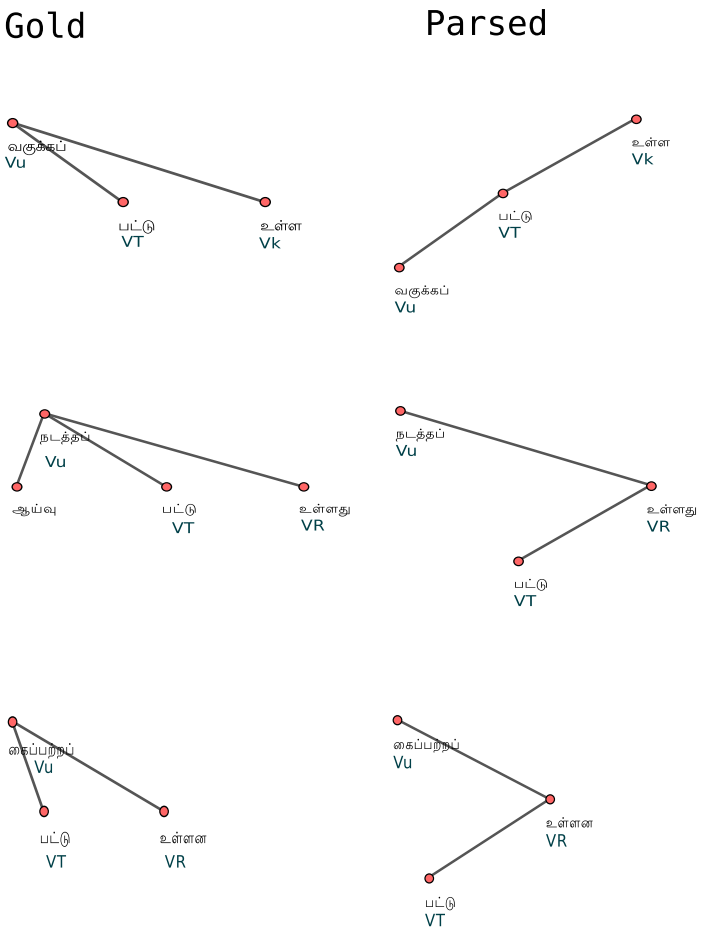


Figure 7.2: Attachment of nodes with POS category VT in gold and parse trees

When we looked at some of the most frequently occurring attachment errors that have closed class tags (for example: VT), the attachment errors are due to annotation differences. Figure 7.2 shows how a node with POS tag VT is attached in gold trees and parse outputs. This particular tag has a fixed distribution in

the data, this particular tag is part of verb clusters and this kind of discrepancies in attachments can be identified easily. Figure 7.2 shows which wordforms are attached incorrectly most often. Most of the wordforms in the top are function words and auxiliary verbs.

Wordform	Occurrences	Edge errors	%
.	124	4	03.0
,	56	49	88.0
உள்ளது	25	12	48.0
வேண்டும்	19	17	89.0
பட்டு	17	17	100.0
என்று	17	10	59.0
அவர்	16	10	62.0
இந்த	14	7	50.0
என்றும்	13	10	77.0
செய்து	13	11	85.0
உள்ள	11	8	73.0
வருகின்றனர்	10	6	60.0
அணு	9	2	22.0
புலிகள்	9	6	67.0
இந்தியா	9	7	78.0
போது	9	9	100.0
உள்ளார்	9	6	67.0
இல்லை	9	4	44.0
இது	8	4	50.0
கொண்டு	8	8	100.0
ஐ.நா.	8	4	50.0
அரசு	8	5	62.0
மற்றும்	8	6	75.0
இரு	7	2	29.0
இருந்து	7	7	100.0
Summary	1892	1072	57.3

Table 7.2: Attachment errors in the projection by wordforms

7.3 Alignment Errors

In realistic scenarios, resource-poor languages may or may not have huge parallel corpora. The alignment accuracy in the training of word alignment models directly depends on the size of the parallel corpus. Since accurate alignments is one of the essential requirement to obtain better predictions in projection based experiments, it is crucial to measure the impact the bitext size (or alignment error) will have on the projection performance. As expected, the Figure 7.3 shows the decline in Alignment Error Rate (AER) over the increase in parallel corpus size. AER [Och and Ney, 2003] measures the accuracy of word alignments by measuring the difference between alignments produced by a word aligner and gold alignments. We carried out this experiment on English-Tamil parallel corpus (Table 5.1, `en-ta` from EN-IL part of the data, however, we used more than 50K for AER calculation). For gold data, we manually created word alignment links between Tamil treebank test sentences[2] and their English translation (we also translated Tamil sentences to English since the treebank test data did not have any English translation).

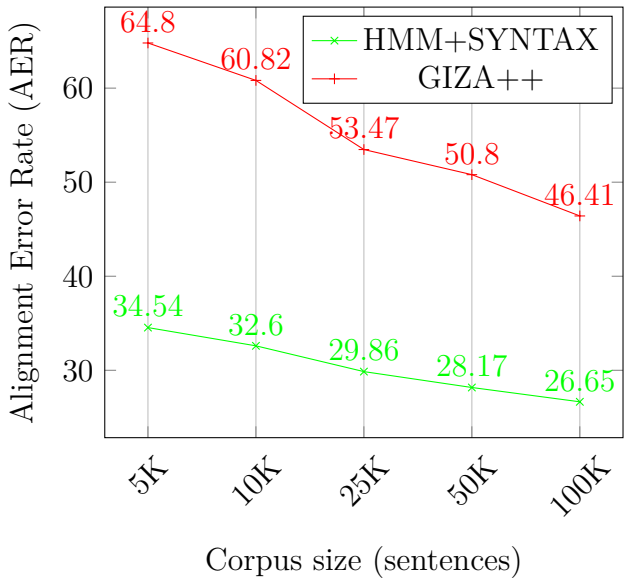


Figure 7.3: Alignment Error Rate with respect to parallel corpus size

There's a certain drop in AER if we train our word aligner on more corpus. There's also huge difference between GIZ++ and HMM (syntax) based alignment.

[2]We used Tamil dependency treebank (TamilTB) test data because of our intention to make parallel treebank for Tamil in future.

Impact on the Projection

AER is a generic measure of whether the overall word alignment quality is good or bad. But how can we make assessments about alignments within sentences? We can answer this question by further looking at individual alignments. The word aligned sentences usually have *one-to-one (1-1)*, *one-to-many (1-M)*, *many-to-one (M-1)* and *many-to-many (M-M)* alignments. Wrong alignments can certainly impact the projection accuracy. But it is also true that not all the wrong alignments will have the same impact in the projection. It is possible that some of the wrong alignments can have from very little impact to greater impact when we project the source language tree to the target language.

Improving the Word Alignment

The impact the alignment error has on the projection depends on the nature (head or modifier) of the source word on the source tree.

- **heads/internal nodes:** when a head/internal node on the source tree is aligned to a wrong word on the target sentence, then the target is more likely to have larger attachment errors due to the wrong identification of the head/internal node on the target side. Barring annotation differences, if a node is a head (of a phrase or clause or sentence) on the source, then the corresponding aligning word too likely to be a head (of that particular phrase or clause or sentence) on target side too. In the same way, modifiers align to modifiers.

  For example, coordination structures are larger structures and they include many conjuncts as their arguments. If the head of the coordination structure is aligned to a word on the target side, then all the conjuncts on the target side will have a wrong coordination head. This analogy can be extended to different types of substructures such as phrases (PP, NP, VP and so on) and clauses.
- **modifiers/leafs:** If the source word is a modifier in the source tree that is aligned to a wrong word on the target sentence, then the impact would be less severe than that of the heads.

Chapter 8

Conclusion

In this thesis, we presented our first-hand results for cross-lingual dependency transfer involving five Indian languages (ILs). The five ILs are: Hindi, Urdu, Telugu, Bengali and Tamil. We applied three dependency transfer methodologies that differ based on the target (IL) resources that they demand.

In *bitext projection based transfer* (Section 6.1), we transferred dependencies from English to ILs. Given the fact that English is typologically different than ILs, the projection yielded slightly better results than left/right baseline but only for 3 out of 8 parallel datasets (see Table 6.4) . What is most interesting in this experiment in terms of projection accuracy is that there's hardly any difference between parallel data that are collected from web (mostly human translated contents to be used in professional websites) and parallel data obtained through crowdsourcing in a short span of time (assumed to be more noisier than normal translations). This idea is further exploited in one of our other transfer approach.

In *delexicalization based transfer* (Section 6.2), we used 30 existing treebanks available via HamleDT as source and directly parsed ILs using 30 delexicalized parsers. We showed parsing results for ILs when source and target (IL) treebank annotations were harmonized at part-of-speech (POS) and dependency level. The results showed that using IL parsers as source over non-IL parsers either brought a big improvement over average parsing accuracy (for Bengali, Telugu and Urdu), negligible improvement (for Tamil) or even drop in average accuracy (for Hindi).

In *dependency transfer based on machine translated bitexts* (Section 6.3), we addressed a new under-resourced scenario in which we have access only to machine translation (MT) systems but not to the parallel corpora itself (which is essential for projection based transfer method). We initially applied this technique by translating 18 language texts (from HamleDT treebanks) to English and we transferred dependencies (through projection) from parsed English data back to those 18 languages. When we simulated real time scenario (by not using any target resources), we achieved 8.2 higher percentage points (absolute) than unsupervised results (Table 6.17). We also extended this approach to ILs by translating IL texts to English and transferring dependencies back to ILs from parsed English data. We compared projection based on human translated (HT) bitexts (only 2K sentence pairs for each)

against machine translated bitexts. Though projection based on MT bitexts did not outperform HT counterparts, for at least 4 out of 5 languages, the gap in projection accuracies between MT and HT bitexts were only 4.1 percentage points (absolute).

In all the transfer methodologies we make use of target language POS taggers at least in the testing phase. We conducted all the transfer experiments in a uniform way so that no particular language expertise would be required for applying these methodologies to new languages. To obtain parser for any language, we further make following generalizations based on the results we obtained for ILs :

1. If there's a treebank available for a target language, then train a *supervised parser*.
2. If there's no treebank available for a target language but only a parallel corpus,
   - If the source language has a parser (implicitly POS tagger too), then try *projection* or *delexicalized parsing* or *both*.
   - If the source language does not have a parser, then choose a different source language that has parser and use *delexicalized parsing*.
3. If there's no treebank or parallel corpus available for a target language but only an MT system, then create parallel corpus by translating target texts to a resource-rich source language and apply step (2).
4. If there's no treebank, parallel corpus or MT system available for a target language but only a POS tagger is available, then try *delexicalized parsing*. Choosing source languages as closer to target as possible can enhance the accuracy.
5. If for any of the above steps the target POS tagger is not available, then obtain target POS tags using unsupervised approach.
6. If none of the above steps is viable, then use *unsupervised parsing* to obtain target structures.

One of the drawbacks of the above approaches is that we did not make use of any language specific rules or tweak the transfer methodologies so that we can further shift the accuracies collectively toward better results. Also, unlike other language treebanks, ILs are unusually biased toward left-branching trees ($> 50\% UAS$ for Bengali, Tamil and Telugu). So our transfer based results are often compared against left/right baseline instead of supervised parser results.

8.1 Thesis Contributions

In this thesis, we made the following novel contributions:

- **Creation of dependency treebank and parallel corpus for a new under-resourced language:** In this thesis, we created a dependency treebank for Tamil language. This resource made it possible to show transfer results for Tamil. Also, we collected parallel corpus for English-Tamil pair that we made use of in the projection based transfer. Both the resources are available for free[1][2]. The resources we created are described in [Ramasamy and Žabokrtský, 2011] and [Ramasamy et al., 2012].

- **Cross-lingual dependency transfer results for Indian languages (ILs):** We applied transfer-based approaches to major Indian languages for which treebank data were available at least for evaluation. In the case of Tamil, we created our own resource. We are not aware of any other work related to cross-lingual dependency transfer mainly targeting ILs.

- **Cross-lingual dependency transfer using machine translated parallel texts:** We presented results for a new under-resourced scenario, where we have access to machine translation (MT) systems but not to parallel corpora. To our best knowledge, this is the first work that made use of machine translated bitexts for cross-lingual dependency transfer.

[1]Tamil dependency treebank: `http://ufal.mff.cuni.cz/~ramasamy/tamiltb/1.0/html`
[2]English-Tamil parallel corpus: `http://ufal.mff.cuni.cz/~ramasamy/parallel/html/`

Bibliography

[Abeillé et al., 2003] Abeillé, A., Clément, L., and Toussenel, F. (2003). Building a Treebank for French. In *Treebanks : Building and Using Parsed Corpora*, pages 165–188. Springer. 4

[Afonso et al., 2002] Afonso, S., Bick, E., Haber, R., and Santos, D. (2002). "Floresta sintá(c)tica": a treebank for Portuguese. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, pages 1968–1703. 4

[Atalay et al., 2003] Atalay, N. B., Oflazer, K., Say, B., and Inst, I. (2003). The annotation process in the Turkish treebank. In *In Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC*. 4

[Baker et al., 2002] Baker, P., Hardie, A., McEnery, T., Cunningham, H., and Gaizauskas, R. J. (2002). Emille, a 67-million word corpus of indic languages: Data collection, mark-up and harmonisation. In *LREC*. xi, 24, 26

[Begum et al., 2008a] Begum, R., Husain, S., Dhwaj, A., Sharma, D. M., Bai, L., and Sangal, R. (2008a). Dependency annotation scheme for indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*. 25

[Begum et al., 2008b] Begum, R., Husain, S., Dhwaj, A., Sharma, D. M., Bai, L., and Sangal, R. (2008b). Dependency Annotation Scheme for Indian Languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, IJCNLP 2008, pages 721–726. Asian Federation of Natural Language Processing (AFNLP). 30

[Bentivogli and Pianta, 2005] Bentivogli, L. and Pianta, E. (2005). Exploiting parallel texts in the creation of multilingual semantically annotated resources: the multisemcor corpus. *Nat. Lang. Eng.*, 11(3):247–261. 10

[Berg-Kirkpatrick et al., 2010] Berg-Kirkpatrick, T., Bouchard-Côté, A., DeNero, J., and Klein, D. (2010). Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California. Association for Computational Linguistics. 10

[Bharati et al., 1996] Bharati, A., Chaitanya, V., and Sangal, R. (1996). *Natural language processing : a Paninian perspective*. Prentice-Hall of India. 22

[Bharati et al., 2009] Bharati, A., Gupta, M., Yadav, V., Gali, K., and Sharma, D. M. (2009). Simple parser for Indian languages in a dependency framework. In *Proceedings of the Third Linguistic Annotation Workshop*, ACL-IJCNLP '09, pages 162–165, Stroudsburg, PA, USA. Association for Computational Linguistics. 30

[Bhat and Sharma, 2012] Bhat, R. A. and Sharma, D. D. M. (2012). Dependency treebank of urdu and its evaluation. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 157–165, Jeju, Republic of Korea. Association for Computational Linguistics. xi, 4, 24, 25, 76, 78

[Bhatt et al., 2009] Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D. M., and Xia, F. (2009). A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, ACL-IJCNLP '09, pages 186–189, Stroudsburg, PA, USA. Association for Computational Linguistics. 4, 78

[Blunsom and Cohn, 2011] Blunsom, P. and Cohn, T. (2011). A hierarchical Pitman-Yor process HMM for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, Portland, Oregon, USA. Association for Computational Linguistics. 122, 125

[Boguslavsky et al., 2000] Boguslavsky, I., Grigorieva, S., Grigoriev, N., Kreidlin, L., and Frid, N. (2000). Dependency treebank for Russian: Concept, tools, types of information. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 987–991. Association for Computational Linguistics Morristown, NJ, USA. 4

[Bojar et al., 2014] Bojar, O., Diatka, V., Rychlý, P., Straňák, P., Tamchyna, A., and Zeman, D. (2014). Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of the Ninth International Language Resources and Evaluation Conference (LREC'14)*, Reykjavik, Iceland. ELRA, European Language Resources Association. in prep. xi, 24, 74, 76

[Brants et al., 2002] Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol. 4

[Clark, 2000] Clark, A. (2000). Inducing syntactic categories by context distribution clustering. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*,

CoNLL '00, pages 91–94, Stroudsburg, PA, USA. Association for Computational Linguistics. 122

[Das and Petrov, 2011a] Das, D. and Petrov, S. (2011a). 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 - Volume 1*, HLT '11, pages 600–609, Stroudsburg, PA, USA. Association for Computational Linguistics. 10, 12, 25

[Das and Petrov, 2011b] Das, D. and Petrov, S. (2011b). 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 - Volume 1*, HLT '11, pages 600–609, Stroudsburg, PA, USA. Association for Computational Linguistics. 125

[de Marneffe and Manning, 2008] de Marneffe, M.-C. and Manning, C. D. (2008). The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, Cross-Parser '08, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics. 20

[DeNero and Klein, 2007] DeNero, J. and Klein, D. (2007). Tailoring word alignments to syntactic machine translation. In *ACL*. 95

[Dhanalakshmi et al., 2010] Dhanalakshmi, V., Anand Kumar, M., Rekha, R., Soman, K., and Rajendran, S. (2010). Grammar Teaching Tools for Tamil Language. In *Technology for Education Conference (T4E 2010)*. 30

[Durrett et al., 2012] Durrett, G., Pauls, A., and Klein, D. (2012). Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11, Jeju Island, Korea. Association for Computational Linguistics. 5, 13

[Erjavec, 2010] Erjavec, T. (2010). Multext-east version 4: Multilingual morphosyntactic specifications, lexicons and corpora. In Chair), N. C. C., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA). 20

[Ganchev et al., 2009] Ganchev, K., Gillenwater, J., and Taskar, B. (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: Volume*

1 - Volume 1, ACL '09, pages 369–377, Stroudsburg, PA, USA. Association for Computational Linguistics. 11, 25

[Garrette and Baldridge, 2013] Garrette, D. and Baldridge, J. (2013). Learning a part-of-speech tagger from two hours of annotation. pages 138–147. 6

[Green et al., 2012] Green, N., Ramasamy, L., and Žabokrtský, Z. (2012). Using an svm ensemble system for improved tamil dependency parsing. In *Proceedings of the ACL 2012 Joint Workshop on Statistical Parsing and Semantic Processing of Morphologically Rich Languages*, pages 72–77, Jeju, Republic of Korea. Association for Computational Linguistics. 14

[Hajič et al., 2006] Hajič, J., Hajičová, E., Panevová, J., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., and Mikulová, M. (2006). Prague Dependency Treebank 2.0. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia. 2, 34, 53

[Han et al., 2006] Han, N.-R., Ryu, S., Chae, S.-H., Yang, S.-y., Lee, S., and Palmer, M. (2006). Korean Treebank Annotations Version 2.0. Linguistic Data Consortium, LDC Catalog No.: LDC2006T09, Philadelphia. 4

[Husain et al., 2010] Husain, S., Mannem, P., Ambati, B., and Gadde, P. (2010). The ICON-2010 tools contest on Indian language dependency parsing. In *Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing*, Kharagpur, India. xi, 4, 24, 25, 76

[Hwa et al., 2002] Hwa, R., Resnik, P., and Weinberg, A. (2002). Breaking the Resource Bottleneck for Multilingual Parsing. Technical Report LAMP-TR-086,CS-TR-4355,UMIACS-TR-2002-35, University of Maryland, College Park. 83

[Hwa et al., 2005] Hwa, R., Resnik, P., Weinberg, A., Cabezas, C., and Kolak, O. (2005). Bootstrapping parsers via syntactic projection across parallel texts. *Nat. Lang. Eng.*, 11(3):311–325. 5, 9, 10, 25, 83, 84, 85, 91, 102, 121

[Janarthanam et al., 2007] Janarthanam, S., Nallasamy, U., Ramasamy, L., and Santhoshkumar, C. (2007). Robust Dependency Parser for Natural Language Dialog Systems in Tamil. In *Proceedings of the 5th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, IJCAI KRPDS-2007, pages 1–6. 30

[Jawaid and Zeman, 2011] Jawaid, B. and Zeman, D. (2011). Word-order issues in English-to-Urdu statistical machine translation. 74, 76

[Kawata and Bartels, 2000] Kawata, Y. and Bartels, J. (2000). Stylebook for the Japanese treebank in Verbmobil. In *Report 240*, Tübingen, Germany. 4

[Klein and Manning, 2003] Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics. 96

[Klein and Manning, 2004] Klein, D. and Manning, C. D. (2004). Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics. 10, 15

[Koo et al., 2008] Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio. Association for Computational Linguistics. 13, 14

[Krauwer, 2003] Krauwer, S. (2003). The Basic Language Resource Kit (BLARK) as the First Milestone for the Language Resources Roadmap. In *Proceedings of the 2003 International Workshop Speech and Computer SPECOM-2003*, pages 8–15. 3, 22

[Lehmann, 1989] Lehmann, T. (1989). *A Grammar of Modern Tamil*. Pondicherry Institute of Linguistics and Culture. 38, 45, 57

[Liang et al., 2006] Liang, P., Taskar, B., and Klein, D. (2006). Alignment by agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111, Stroudsburg, PA, USA. Association for Computational Linguistics. 15, 95

[Maamouri et al., 2012] Maamouri, M., Bies, A., Bies, S., Krouna, S., Tabassi, D., and Ciul, M. (2012). Arabic Treebank - Broadcast News v1.0. Linguistic Data Consortium, LDC Catalog No.: LDC2012T07, Philadelphia. 4

[Majliš and Žabokrtský, 2012a] Majliš, M. and Žabokrtský, Z. (2012a). Language richness of the web. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey. European Language Resources Association (ELRA). xi, 14, 24, 125

[Majliš and Žabokrtský, 2012b] Majliš, M. and Žabokrtský, Z. (2012b). Language richness of the web. In *Proceedings of LREC2012*, Istanbul, Turkey. ELRA, European Language Resources Association. In print. 26

[Mannem and Dara, 2011] Mannem, P. and Dara, A. (2011). Partial parsing from bitext projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT

'11, pages 1597–1606, Stroudsburg, PA, USA. Association for Computational Linguistics. 11

[Mareček, 2011] Mareček, D. (2011). Combining diverse word-alignment symmetrizations improves dependency tree projection. In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part I*, CICLing'11, pages 144–154, Berlin, Heidelberg. Springer-Verlag. 11, 12

[Mareček and Straka, 2013] Mareček, D. and Straka, M. (2013). Stop-probability estimates computed on a large corpus improve Unsupervised Dependency Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria. Association for Computational Linguistics. 125, 126

[Mareček and Žabokrtský, 2012] Mareček, D. and Žabokrtský, Z. (2012). Exploiting reducibility in unsupervised dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 297–307, Jeju Island, Korea. Association for Computational Linguistics. 15

[Marimon et al., 2012] Marimon, M., Fisas, B., Bel, N., Villegas, M., Vivaldi, J., Torner, S., Lorente, M., Vázquez, S., and Villegas, M. (2012). The iula treebank. In *LREC*, pages 1920–1926. 4

[McDonald et al., 2005a] McDonald, R., Crammer, K., and Pereira, F. (2005a). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 91–98, Stroudsburg, PA, USA. Association for Computational Linguistics. 93, 96

[McDonald et al., 2013] McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Täckström, O., Bedini, C., Bertomeu Castelló, N., and Lee, J. (2013). Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics. 6, 18, 82, 102

[McDonald et al., 2005b] McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005b). Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Langauge Technology Conference and Conference on Empirical Methods in Natural Language Processing (HTL/EMNLP)*, pages 523–530, Vancouver, BC, Canada. 124

[McDonald et al., 2011] McDonald, R., Petrov, S., and Hall, K. (2011). Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Confer-*

ence on Empirical Methods in Natural Language Processing, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics. 5, 6, 9, 12, 13, 110, 112, 121, 122, 125, 128

[Montemagni et al., 2003] Montemagni, S., Barsotti, F., Battista, M., Calzolari, N., Corazzari, O., Lenci, A., Zampolli, A., Fanciulli, F., Massetani, M., Raffaelli, R., Basili, R., Pazienza, M. T., Saracino, D., Zanzotto, F., Mana, N., Pianesi, F., and Delmonte, R. (2003). Building the Italian syntactic-semantic treebank. In Abeillé, A., editor, *Building and using Parsed Corpora*, Language and Speech series, pages 189–210, Dordrecht. Kluwer. 4

[Mukund et al., 2010] Mukund, S., Ghosh, D., and Srihari, R. K. (2010). Using cross-lingual projections to generate semantic role labeled corpus for urdu: a resource poor language. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 797–805, Stroudsburg, PA, USA. Association for Computational Linguistics. 10

[Naseem et al., 2010] Naseem, T., Chen, H., Barzilay, R., and Johnson, M. (2010). Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1234–1244, Stroudsburg, PA, USA. Association for Computational Linguistics. 16

[Nguyen et al., 2009] Nguyen, P. T., Vu, X. L., Nguyen, T. M. H., Nguyen, V. H., and Le, H. P. (2009). Building a large syntactically-annotated corpus of vietnamese. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 182–185, Suntec, Singapore. Association for Computational Linguistics. 4

[Nivre, 2005] Nivre, J. (2005). Dependency grammar and dependency parsing. Technical report, Växjö University. 2

[Nivre, 2009] Nivre, J. (2009). Parsing Indian Languages with MaltParser. In *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, ICON 2009, pages 12–18. 30

[Nivre et al., 2007] Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007). The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic. Association for Computational Linguistics. 96, 124, 128

[Och and Ney, 2003] Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51. 15, 95, 136

[Padó and Lapata, 2009] Padó, S. and Lapata, M. (2009). Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340. 10, 25

[Petrov et al., 2012] Petrov, S., Das, D., and McDonald, R. (2012). A universal part-of-speech tagset. In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA). xi, 16, 17, 18, 102, 122, 124, 125

[Popel et al., 2013] Popel, M., Mareček, D., ŠtÄ›pánek, J., Zeman, D., and Žabokrtský, Z. (2013). Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 517–527, Sofia, Bulgaria. Association for Computational Linguistics. 103, 104

[Post et al., 2012] Post, M., Callison-Burch, C., and Osborne, M. (2012). Constructing parallel corpora for six indian languages via crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 401–409, Montréal, Canada. Association for Computational Linguistics. xi, 6, 24, 26, 74, 75, 76, 95

[Ramasamy et al., 2012] Ramasamy, L., Bojar, O., and Žabokrtský, Z. (2012). Morphological processing
for english-tamil statistical machine translation. In *Proceedings of the Workshop on MT and Parsing in Indian Languages*. xi, 24, 26, 74, 76, 141

[Ramasamy and Žabokrtský, 2011] Ramasamy, L. and Žabokrtský, Z. (2011). Tamil Dependency Parsing: Results Using Rule Based and Corpus Based Approaches. In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part I*, CICLing'11, pages 82–95, Berlin, Heidelberg. Springer-Verlag. xi, 24, 76, 77, 141

[Ramasamy and Žabokrtský, 2012] Ramasamy, L. and Žabokrtský, Z. (2012). Prague dependency style treebank for Tamil. In *Proceedings of LREC 2012*, İstanbul, Turkey. 4, 25

[Rasooli et al., 2011] Rasooli, M. S., Moloodi, A., Kouhestani, M., and Minaei-Bidgoli, B. (2011). A syntactic valency lexicon for Persian verbs: The first steps towards Persian dependency treebank. In *5th Language & Technology Conference (LTC): Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 227–231, Poznań, Poland. 4

[Rosa et al., 2014] Rosa, R., Mašek, J., Mareček, D., Popel, M., Zeman, D., and Žabokrtský, Z. (2014). Hamledt 2.0: Thirty dependency treebanks stanfordized. In Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA). 78

[Selvam et al., 2009] Selvam, M., Natarajan, A. M., and Thangarajan, R. (2009). Structural Parsing of Natural Language Text in Tamil Language Using Dependency Model. *Int. J. Comput. Proc. Oriental Lang.*, 22(2-3):237–256. 30

[Smith and Eisner, 2009] Smith, D. A. and Eisner, J. (2009). Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 822–831, Stroudsburg, PA, USA. Association for Computational Linguistics. 11

[Snyder et al., 2009] Snyder, B., Naseem, T., and Barzilay, R. (2009). Unsupervised multilingual grammar induction. 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: Volume 1 - Volume 1*, ACL '09, pages 73–81, Stroudsburg, PA, USA. Association for Computational Linguistics. 15

[Søgaard, 2011] Søgaard, A. (2011). Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 682–686, Stroudsburg, PA, USA. Association for Computational Linguistics. 12, 13

[Spitkovsky et al., 2011] Spitkovsky, V. I., Alshawi, H., Chang, A. X., and Jurafsky, D. (2011). Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1281–1290, Stroudsburg, PA, USA. Association for Computational Linguistics. 15

[Spoustová et al., 2007] Spoustová, D., Hajič, J., Votrubec, J., Krbec, P., and Květoň, P. (2007). The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In *ACL '07: Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*, pages 67–74, Morristown, NJ, USA. Association for Computational Linguistics. 96, 124, 125

[Spreyer, 2010] Spreyer, K. (2010). Notes on the evaluation of dependency parsers obtained through cross-lingual projection. In *Coling 2010: Posters*, pages 1176–1184, Beijing, China. Coling 2010 Organizing Committee. 11

[Steedman et al., 2003] Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. (2003). Bootstrapping statistical parsers from small datasets. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 331–338, Stroudsburg, PA, USA. Association for Computational Linguistics. 9, 14

[Täckström et al., 2012] Täckström, O., McDonald, R., and Uszkoreit, J. (2012). Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487, Montréal, Canada. Association for Computational Linguistics. 5, 13

[Taylor et al., 2003] Taylor, A., Marcus, M., and Santorini, B. (2003). The penn treebank: An overview. In Abeillé, A., editor, *Treebanks*, volume 20 of *Text, Speech and Language Technology*, pages 5–22. Springer Netherlands. 2, 4

[Toutanova et al., 2003] Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics. 77, 93, 124

[Varga et al., 2005] Varga, D., Halácsy, P., Kornai, A., Nagy, V., Németh, L., and Trón, V. (2005). Parallel corpora for medium density languages. In *Proceedings of the RANLP 2005*, pages 590–596. 15, 26, 74

[Vempaty et al., 2010] Vempaty, C., Naidu, V., Husain, S., Kiran, R., Bai, L., Sharma, D. M., and Sangal, R. (2010). Issues in Analyzing Telugu Sentences towards Building a Telugu Treebank. In Gelbukh, A. F., editor, *CICLing*, volume 6008 of *Lecture Notes in Computer Science*, pages 50–59. Springer. 30

[Xue et al., 2010] Xue, N., Jiang, Z., Zhong, X., Palmer, M., Xia, F., Chiou, F.-D., and Chang, M. (2010). Chinese Treebank 7.0. Linguistic Data Consortium, LDC Catalog No.: LDC2010T07, Philadelphia. 4

[Yarowsky et al., 2001] Yarowsky, D., Ngai, G., and Wicentowski, R. (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, HLT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics. 10

[Zeman, 2008] Zeman, D. (2008). Reusable tagset conversion using tagset drivers. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., and

Tapias, D., editors, *Proceedings of the Sixth International Language Resources and Evaluation Conference, LREC 2008*, pages 28–30, Marrakech, Morocco. European Language Resources Association (ELRA). xi, 17, 18, 19

[Zeman et al., 2012] Zeman, D., Mareček, D., Popel, M., Ramasamy, L., Štěpánek, J., Žabokrtský, Z., and Hajič, J. (2012). Hamledt: To parse or not to parse? In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA). 2, 6, 17, 78, 82, 83, 93, 102, 112

[Zeman and Resnik, 2008] Zeman, D. and Resnik, P. (2008). Cross-language parser adaptation between related languages. In *IJCNLP 2008 Workshop on NLP for Less Privileged Languages*, pages 35–42, Hyderabad, India. Asian Federation of Natural Language Processing, International Institute of Information Technology. 12, 102, 110

[Zhu, 2005] Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison. 13

Appendix A

Projected Trees

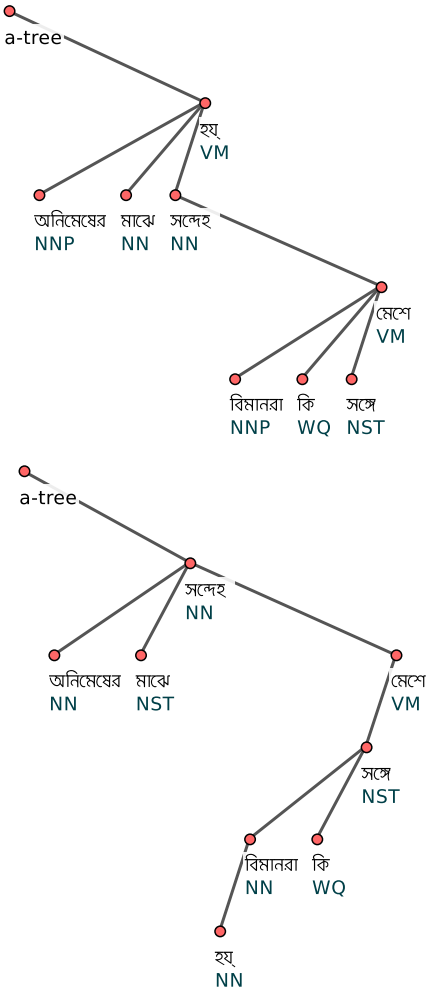
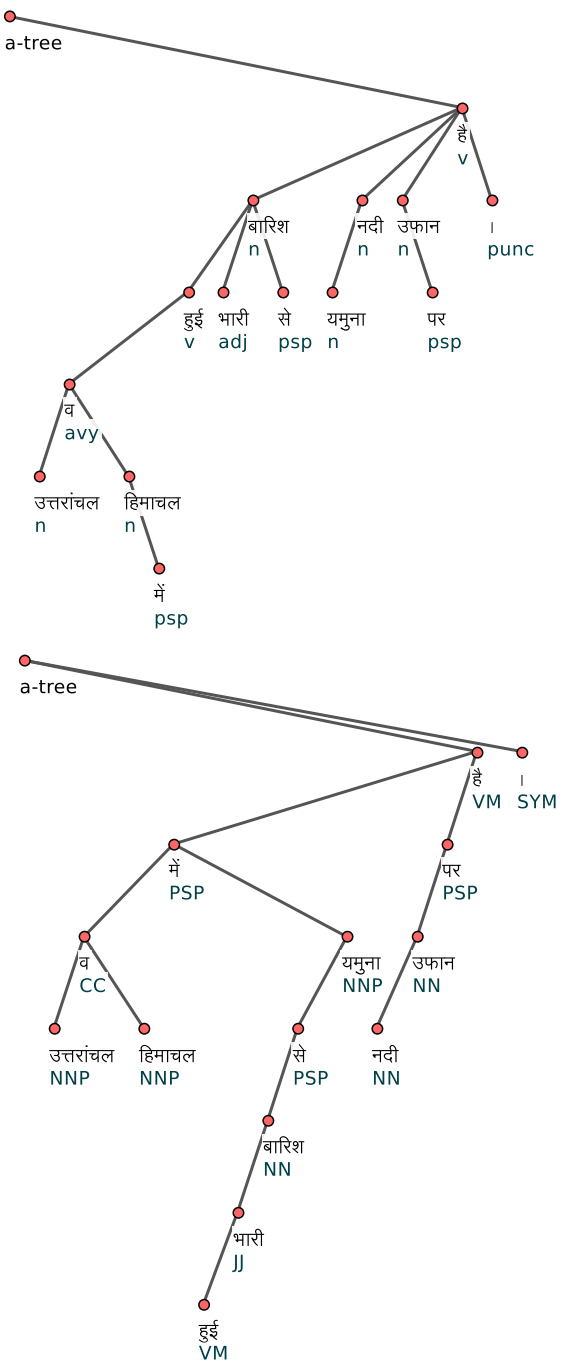
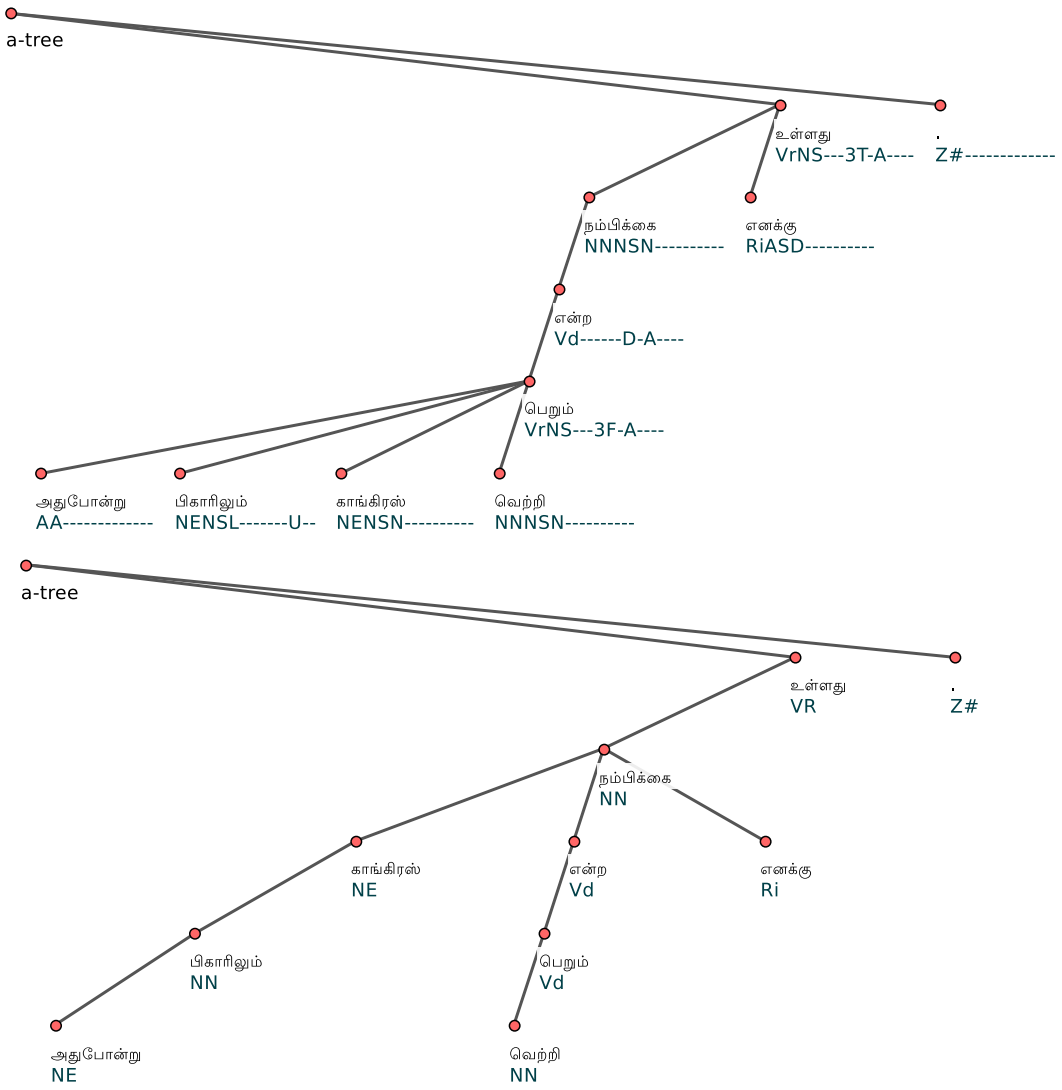


Figure A.1: Bengali (`bn`): [TOP] - gold tree, [BOTTOM] - parsed output from the model (MST, projective) trained on projected trees from English



[hi_prmcdproj] उत्तरांचल व हिमाचल में हुई भारी बारिश से यमुना नदी उफान पर है ।
[hi] उत्तरांचल व हिमाचल में हुई भारी बारिश से यमुना नदी उफान पर है ।

Figure A.2: Hindi (`hi`): [TOP] - gold tree, [BOTTOM] - parsed output from the model (MST, projective) trained on projected trees from English



[ta_prmcdproj] அதுபோன்று பிகாரிலும் காங்கிரஸ் வெற்றி பெறும் என்ற நம்பிக்கை எனக்கு உள்ளது.
[ta] அதுபோன்று பிகாரிலும் காங்கிரஸ் வெற்றி பெறும் என்ற நம்பிக்கை எனக்கு உள்ளது.

Figure A.3: Tamil (`ta`): [TOP] - gold tree, [BOTTOM] - parsed output from the model (MST, projective) trained on projected trees from English

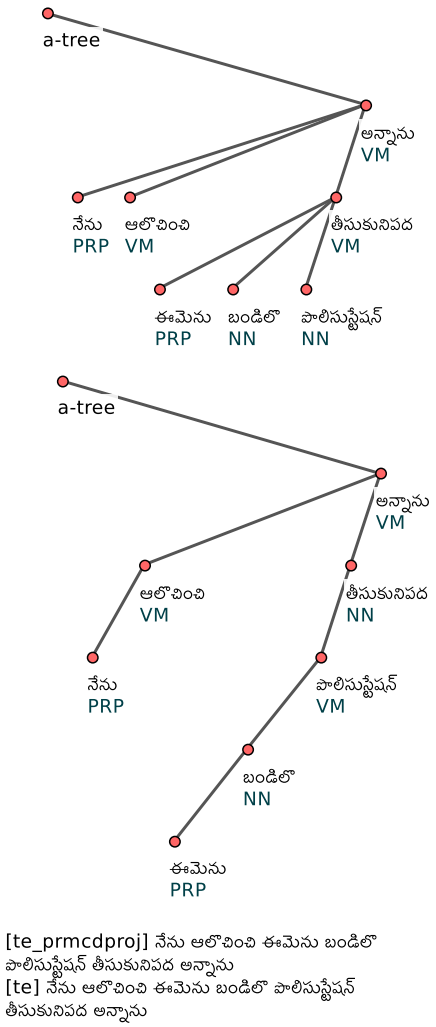


Figure A.4: Telugu (`te`): [TOP] - gold tree, [BOTTOM] - parsed output from the model (MST, projective) trained on projected trees from English

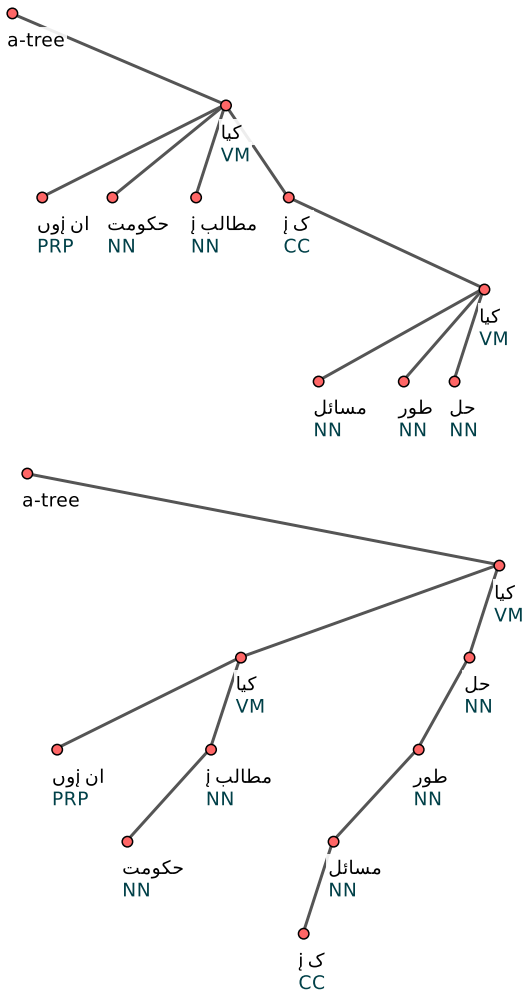


Figure A.5: Urdu (ur): [TOP] - gold tree, [BOTTOM] - parsed output from the model (MST, projective) trained on projected trees from English

Appendix B

Tamil Dependency Treebank

B.1 TamilTB 1.0

TamilTB 1.0 is accessible at this location: `http://ufal.mff.cuni.cz/~ramasamy/tamiltb/1.0/html`

B.1.1 Treex platform

TamilTB 1.0 is being developed under Treex, a successor to TectoMT. Programming scripts related to annotation or Tamil in general are available as Treex blocks. Those scripts will be available once the Treex distribution is installed. Treex can be accessible at this location: `http://ufal.mff.cuni.cz/treex`

B.2 TamilTB.v0.1

The data is available in three formats,

1. TMT format - XML-based format used in the TectoMT system
2. CoNLL format - tabular separated format in the CoNLL shared task style
3. TnT style POS tagged format - tabular separated columns with word forms, POS tags, and lemmas.

The single package containing the data and the documentation can be downloaded from the following link,

http://ufal.mff.cuni.cz/~ramasamy/tamiltb/0.1/download.html