Prague Dependency Style Treebank for Tamil

Loganathan Ramasamy and Zdeněk Žabokrtský

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics, Charles University in Prague
{ramasamy, zabokrtsky}@ufal.mff.cuni.cz

Abstract

Annotated corpora such as treebanks are important for the development of parsers, language applications as well as understanding of the language itself. Only very few languages possess these scarce resources. In this paper, we describe our efforts in syntactically annotating a small corpora (600 sentences) of Tamil language. Our annotation is similar to Prague Dependency Treebank (PDT) and consists of annotation at 2 levels or layers: (i) morphological layer *(m-layer)* and (ii) analytical layer *(a-layer)*. For both the layers, we introduce annotation schemes i.e. positional tagging for *m-layer* and dependency relations for *a-layers*. Finally, we discuss some of the issues in treebank development for Tamil.

Keywords: Syntax, Treebanking, Annotation, Grammar

1. Introduction and Previous work

The most important thing in Natural Language Processing (NLP) research is data, importantly the data annotated with linguistic descriptions. Much of the success in NLP in the present decade can be attributed to data driven approaches to linguistic challenges, which discover rules from data as opposed to traditional rule based paradigms. The data driven approaches require labeled or annotated data such as treebank (Marcus et al., 1993) (Hajič et al., 2006) or parallel corpora (Koehn, 2005) to train their systems. Unfortunately, only English and very few other languages have the privilege of having such rich annotated data due to various factors.

In this paper, we take up the case of building a dependency treebank for Tamil language for which no annotated data is available. The broad objectives for the design of the Tamil dependency treebank (TamilTB) include: (i) annotating data at morphological level and syntactic level (ii) in each level of annotation, trying for maximum level of linguistic representation and (iii) building large annotated corpora using automatic tools. We have chosen dependency annotation over constituency representation for one obvious reason: that the dependency annotation works well for free word order languages and the annotation is quite intuitive and easy to represent. One other reason is that, since treebanking for other Indian languages such as Hindi and Telugu (Begum et al., 2008) too focuses on dependency annotation scheme, it would be easier in the future to compare or adopt features from those efforts.

There is an active research on dependency parsing ((Bharati et al., 2009), (Nivre, 2009) and (Zeman, 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., 2008) (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 utilised 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 spoken language utterences using dependency framework. Those works did not make use of treebank to the parser development, rather they were based on linguistic rules. A somewhat detailed description of an effort to develop a Tamil dependency treebank appeared in (Ramasamy and Žabokrtský, 2011). This work will be the continuation of the work mentioned in (Ramasamy and Žabokrtský, 2011). To our knowledge, this is the first attempt to develop a dependency treebank for Tamil with respect to the objectives defined earlier.

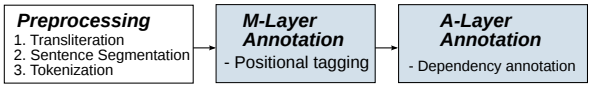
Before describing the annotation process and the annotation schemes, we briefly introduce the main features of the Tamil language. Tamil belongs to Dravidian family of languages and mainly spoken in southern India and also in parts of Sri Lanka, Malaysia and Singapore. Tamil is agglutinative and has a rich set of morphological suffixes. Tamil has nouns and verbs as two major word classes, and hundreds of wordforms can be produced by the application of concatenative and derivational morphology. It is a free word order language and follows Subject Object Verb (SOV) pattern. Tamil is strictly a *head final* language, meaning the head of a phrase or a clause or a sentence always occurs at the final position. In other words, all the arguments of a consituent occurs to the left of the head. In most cases, the *syntactic subject* in Tamil agrees with the *verb* in person-number-gender. The following sections describe the annotation process and annotation schemes in detail.

2. Annotation process and the Data

Our annotation scheme is based on Prague Dependency Treebank (PDT) (Hajič et al., 2006) (Hajič, 1998). PDT annotates the data in 3 levels or layers: (i) morphological layer *(m-layer)* (ii) surface syntax layer *(a-layer)* and (iii) tectogrammatical layer *(t-layer)*. In *m-layer* annotation, tokens in a sentence are annotated with their morphological tags and lemmas. In *a-layer* annotation, a sentence is annotated with its dependency structure. Tectogrammatical

(*t-layer*) annotation captures the deep syntax of a sentence. All 3 layers in PDT are interlinked, meaning the information about lower layer (for ex: *m-layer*) is available to upper layers (for ex: *a-layer*).

Our annotation process includes only the first 2 layers i.e. *m-layer* and *a-layer*. The Figure 1(a) shows various tasks involved in the annotation process and the Figure 1(b) shows general information about the data used for annotation. The data for the annotation comes from news domain, and even within news domain the text is chosen from different topics.



(a) The annotation process

Description	value
Source	`www.dinamani.com`
Format	UTF-8
Transliterated	yes
Number of sentences	600
Number of words	9581

(b) The data for annotation

Figure 1: The annotation process and data

Clitics	*um, E, EyE, AvaTu*
Postpositions	*kUta, utan, pati, kuRiTTu, iliruwTu, anRu, uL, ARu, Tavira, pOTu, pOla, pinnar, pin, arukE, aRRa, inRi, illATa, mITu, kIz, mEl, munpE, otti, paRRi, paRRiya, pOnRa, mUlam, vaziyAka* etc.
Auxiliary Verbs	*patta, pattu, uLLa, pata, mAttATu, patuvArkaL, uLLAr, uLLanar, illai, iruwTAr, iruwTaTu, pattaTu, pattana, mutiyum, kUtATu, vENtum, kUtum, iruppin, uLLana, mutiyATu, patATu, koNtu, ceyTu* etc.
Particles	*Aka, Ana* and their spelling variants *Akac, AkaT, Akap, Akak*
Demonstrative pronouns	*ap, ac, ic, aw, iw* etc. as prefixes

Table 1: List of words and affixes for tokenization

The data is preprocessed prior to the annotation process. Initially, the raw corpus in UTF-8 is transliterated[1] into Latin for the ease of representation inside the programming components. This step also helps to avoid problems processing Tamil script when splitting the tokens. After the transliteration, sentence segmentation is performed on the data to split the raw corpus into one sentence per line. We used simple heuristics such as *fullstop, name initials, attribution* etc. to split the data into sentences. Tokenization is one of the important steps in preprocessing. Tamil tokens are usually separated by spaces. Apart from that, Tamil is known to combine tokens with certain closed class words and affixes, which can be represented as separate words in languages such as English. For ex: Tamil, in certain situations combine *postpositions* with nouns, *clitics* with almost any tokens and *auxiliary verbs* with lexical verbs. The tokenization has to be applied for those combinations as well. We splitted those combination of words with the list of closed class words and affixes (refer Table 1). Initial splitting was done automatically using a few well known words from the list we constructed, and the remaining words or suffixes are found later when manually analyzing the data. The Table 1 shows the partial list of affixes and closed class words used for tokenization. Some of the tokens have been repeated (For ex: *AkaT, Akac, Akap* have the same root *Aka*), their multiple appearance is due to the presence of external *sandhi* characters. This tokenization process aids the *m-layer* annotation by reducing the tagging complexity as well as data sparsity to some extent. To reconstruct the sentences to original form, we set the `'no_space_after'` attribute to 1 whenever a token is splitted into multiple tokens. If the `'no_space_after'` attribute for token A is 1, then the following token B is part of the token A at the surface representation. This way we will be able to reproduce the original surface representation of the data.

[1]UTF-8 to Latin transliteration map is available here: http://ufal.mff.cuni.cz/~ramasamy/downloads/map.txt. Change the browser encoding to UTF-8 to view the file properly.

3. Morphological annotation (*m-layer*)

The *m-layer* annotation simply corresponds to morphological tagging of the data. The *m-layer* annotation consists of two steps: (i) assigning morphological tags to tokens and (ii) identifying lemmas for the tokens. These two steps correspond to assigning *m-layer* attributes `'tag'` and `'lemma'` in PDT. For step (i), we use *positional tagging* scheme (Hajič, 2004) to tag the tokens. The main advantage of the positional tagging is that it can accommodate morphological features. The main difference when compared to ordinary POS tagging is that, the *positional tag* is a *fixed length* string. Each character in the tag signifies a particular feature of a token. For our purpose, we have defined the length of the positional tag to be *9* positions. Figure 2 shows the structure of the positional tag with an example annotation of a Tamil word.

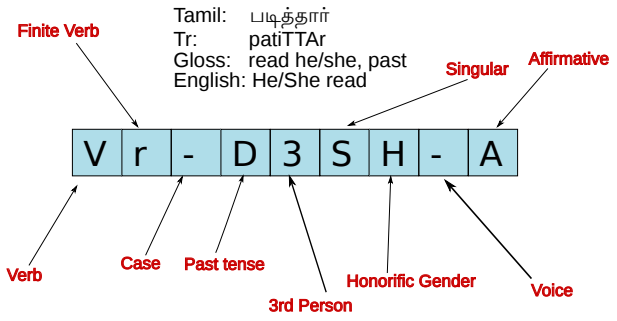


Figure 2: Positional tag

The Figure 3 (a) & (b) illustrate the positional tagging system and a list of possible values for the first position i.e. major POS. We have defined 14 major POS categories. This tagset includes the eight major POS defined in (Lehmann, 1989) as well as some overlapping categories (such as particles and numerals) as separate POS categories. The second

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) POS values

Description	Value
Corpus size	9581 words
Vocabulary size	3583 words
# of tags for this corpus	217
# words received unique tags	3464
# words received 2 tags	109
# words received 3 tags	9
# words received 4 tags	1

(c) m-layer annotation statistics

Figure 3: Positional tag system

position stands for *sub POS* which captures subtle difference in the major POS. The first 2 positions together represent POS tag as in the traditional sense. The remaining seven positions compactly code other morphological features of a token. Some of the *m-layer* annotation is performed manually and the remaining tags are found automatically by training the TnT tagger (Brants, 2000) on the hand annotated data. Wrong tag assignments were edited manually. Figure 3 (c) shows the basic statistics of the *m-layer* annotation. From the Figure, we observe that the entire corpus was tagged by 217 distinct tags (including all 9 positions). Most of the distinct tags belong to verbs, nouns and pronouns due to their morphological productivity. The Figure also shows how many distinct tags each word in the vocabulary can take. Over 96% of the tokens are *unambiguously* represented using a single tag. Only little over 3% of the tokens are ambiguous by having 2 possible tags. Tokens with 3 tags and 4 tags are almost negligible. This statistics implies that tokens can be assigned to distinct tags if we take into account the morphological features of the tokens.

Tamil nouns and verbs take variety of morphological suffixes and they are the two major classes of word types that participate in morphological processes. We used set of suffix based heuristics to identify lemmas of tokens. As mentioned earlier, the lemmas are stored as an *m-layer* attribute `'lemma'`. At present, lemmas are identified partially through automation by using fixed suffix list (mostly to handle verbs and nouns). After a complete pass over the data, wrong lemma guesses were corrected manually.

The Table 2 shows how the words can be tagged for positions from 3 to 9 in the positional tag. The positions 1 and 2 together occupy around 50 distint tags and they are detailed in (Ramasamy and Žabokrtský, 2011b). The remaining 3-9 positions encode various morphological aspects of tokens. Thus it is possible to use different tagsets (by selecting certain positions for ex: only first 2 positions) to train different POS taggers without involving much cost. In the Table 2, position values such as *I* and *X* in *Gender* and *X* in *Person* and *Number* are unused at present, and they are reserved for future purposes. The position value '-' indicates that the particular position is not relevant for tagging a particular token. For ex: when tagging a verb, the 3rd position (case) in the tag is not relevant.

As an another example for morphological tagging, the Table 3 shows how personal pronouns can be tagged. Other classes of pronouns such as interrogative and general referential pronouns can be derived by adding appropriate suffixes to personal pronouns. Tagging of other derived pronouns are detailed in (Ramasamy and Žabokrtský, 2011b).

4. Syntactic annotation (*a-layer*)

The *a-layer* annotation corresponds to dependency annotation. The sentence annotated at the *m-layer* is annotated for dependency relations. This step consists of two stages: (i) identifying the structure by attaching the *dependent* word as child to the *governing* word and (ii) labeling the relation with which the dependent and governing nodes (words) are related. Thus each sentence corresponds to a tree structure rooted at the predicate of the sentence or at the technical root (as in PDT). In the case of technical root, the predicate node is attached to the technical root. The purpose of the technical root is to store some meta information about the sentence such sentence id, language etc. Each edge has a label and it signifies the relation between the parent and child nodes. In PDT, each edge label or relation is stored as an *a-layer* attribute `'afun'` in the dependent node. Other than `'afun'`, attributes such as `'is_member'` will be set for conjuncts in coordination conjunction.

So far we have defined 21 dependency relations or analytical functions (*afun*) for labeling the edges. Most of the relations are similar to dependency relations of PDT. The Figure 4 shows our dependency annotation scheme with some examples. After the *m-layer* annotation is performed, the structure and dependency relations for the edges were produced automatically by the rule based parser (Ramasamy and Žabokrtský, 2011) and then corrected through a manual editing. The true dependency relation is then stored in the `'afun'` attribute of the dependent node. The *m-layer* and *a-layer* annotation is done for the dataset as mentioned in Figure 1(b). The annotated data (TamilTB) has been released and is available for download.[2].

(Ramasamy and Žabokrtský, 2011b) describes each dependency relation with examples. Here we try to explain three relations: (i) *AdjAtr* (ii) *Coord* and (iii) *AuxC*.

[2]TamilTB is available for download at: http://ufal.mff.cuni.cz/~ramasamy/tamiltb/0.1/download.html

Pos	Name	#	Value	Description	Example	Tag
3	**Case**	1	A	Accusative	*katciyai* (‘party’)	NNA - 3SN - -
		2	D	Dative	*vIttukku* (‘to/for the house’)	NND - 3SN - -
		3	I	Instrumental	*muyaRciyAl* (‘by the efforts’)	NNI - 3SN - -
		4	G	Genitive	*aracin* (‘government’s’)	NNG - 3SN - -
		5	L	Locative	*pOril* (‘in the war’)	NNL - 3SN - -
		6	N	Nominative	*ANtu* (‘year’)	NNN - 3SN - -
		7	S	Sociative	*TuNaiyOtu* (‘with the help’)	NNS - 3SN - -
4	**Tense**	1	D	past	*kattinAr* (‘built he’)	Vr - D3SHAA
		2	F	future	*uTavum* (‘it will help’)	Vr - F3SNAA
		3	P	present	*celkiRAr* (‘he is going’)	Vr - P3SHAA
		4	T	tenseless	*illai* (‘exist not’)	Vr - T3PNAA
5	**Person**	1	1	1^{st} person	*mERkoNtEn* (‘I undertook’)	Vr - D1SAAA
		2	2	2^{nd} person	*anjcukiRIrkaL* (‘you fear’)	Vr - P2PAAA
		3	3	3^{rd} person	*vivATikkum* (‘it will discuss’)	Vr - F3SNAA
		4	X	*unused*	*unused*	*unused*
6	**Number**	1	P	plural	*vivarangkaL* (‘details’)	NNN - 3PN - -
		2	S	singular	*nyUSilAwTu* (‘New Zealand’)	NEN - 3SN - -
		3	X	*unused*	*unused*	*unused*
7	**Gender**	1	F	feminine	*varuvAL* (‘she will come’)	Vr - F3SFAA
		2	M	masculine	*Atavanin* (‘man’s’)	NNG - 3SM - -
		3	N	neuter	*etuTTaTu* (‘it took’)	Vr- D3SNAA
		4	H	honorific (both masc. and fem.)	*avar* (‘he/she [polite]’)	RpN - 3SH - -
		5	A	animate (humans)	*yAr* (‘who?’)	RiN - 3SA - -
		6	I	inanimate (non humans)	*unused*	*unused*
		7	X	*unused*	*unused*	*unused*
8	**Voice**	1	A	active	*etuTTaTu* (‘it took’)	Vr- D3SNAA
		2	P	passive	*patukiRaTu* (‘being [*verb*]...’)	VR - P3SNPA
9	**Negation**	1	A	affirmative	*pinpaRRa* (‘to follow’)	Vu - T - - - AA
		2	N	negation	*mutiyATu* (‘cannot’)	VR - T3SN -A

Table 2: Examples for *m-layer* annotation

#	Person/Number	Pronoun	Tag
1	1^{st}/singular	*wAn* (‘I’)	RpN - 1SA - -
2	1^{st}/plural	*wAm* (‘we, exclusive’)	RpN - 1PA - -
3	”	*wAngkaL* (‘we, inclusive’)	RpN - 1PA - -
4	2^{nd}/singular	*wI* (‘you’)	RpN - 2SA - -
5	”	*wIngkaL* (‘you, honorific, singular’)	RpN - 2SH - -
6	2^{nd}/plural	*wIngkaL* (‘you, plural’)	RpN - 2PA - -
7	3^{rd}/singular	*avan* (‘that one - he’)	RpN - 3SM - -
8	”	*ivan* (‘this one - he’)	RpN - 3SM - -
9	”	*avaL* (‘that one - she’)	RpN - 3SF - -
10	”	*ivaL* (‘this one - she’)	RpN - 3SF - -
11	”	*aTu* (‘that one - it’)	RpN - 3SN - -
12	”	*iTu* (‘this one - it’)	RpN - 3SN - -
13	”	*avar* (‘that one - he/she hon.’)	RpN - 3SH - -
14	”	*ivar* (‘this one - he/she hon.’)	RpN - 3SH - -
15	3^{rd}/plural	*avai/avaikaL* (‘those ones’)	RpN - 3PN - -
16	”	*ivai/ivaikaL* (‘these ones’)	RpN - 3PN - -
17	”	*avarkaL* (‘those people’)	RpN - 3PA - -
18	”	*ivarkaL* (‘these people’)	RpN - 3PA - -

Table 3: Personal pronouns

4.1. AdjAtr

AdjAtr is a modifier relation. It is similar to *Atr* relation in functionality except that *AdjAtr* are marked on adjectivalized verbs. In English, *AdjAtr* is equivalent to gerunds, past participle and predicate of the relative clauses. In Tamil, all adjectival participles (for the 3 tenses) are marked with *AdjAtr* relation. The Figure 5 shows an example annotation for *AdjAtr* relation, in which the word *cETamataiwTa* (‘rained’) is marked with *AdjAtr* afun. The difference between *AdjAtr* and *Atr* can be easily understood by looking at the same example, where *ciment* (‘ciment’) is modifier as in noun-noun combinations.

4.2. Coord

Coord is used to mark the head of the coordination conjunction. There are at least 2 ways by which coordination conjunction can be done in Tamil. In the first method,

No	Afun	Afun	Examples
1	AAdjn	Adverbial Adjunct	Optional adverbs, optional PP phrases attaching to verb
2	AComp	Adverbial Complement	Obligatory adverbs, obligatory PP phrases attaching to verb
3	AdjAtr	Adjectival Attribute	Adjectivalized verbs, or relative clauses
4	Apos	Apposition	Heads of the apposition clauses - clauses attaching to 'enRa'
5	Atr	Attribute	Noun modifiers
6	AuxA	Determiners	Demonstrative pronouns (iwTa-'this', awTa-'that')
7	AuxC	Subordinating Conjunctions	Subordinating Conjunctions (enRu, ena, Aka)
8	AuxG	Punctuations	-, ", ', $, rU., (,), [,]
9	AuxK	Terminal Punctuation	:, . , ?
10	AuxP	Postpositional head	mlTu-'on', paRRi-'about', klz-'under'
11	AuxS	Technical Root	Technical Root
12	AuxV	Auxiliary Verb	uL, koNtu, iru
13	AuxX	Comma (not coordination)	,
14	AuxZ	Emphatic particles (clitics)	TAn(emphasis), um-'also, even', E-'even'
15	CC	Part of a word	kiLarwTu ezuwTu - 'rise' as in *rising against*, written as 2 words
16	Comp	Complement (not adverbial)	Obligatory attachments to non verbs, "belongs to the batch **of 1977**"
17	Coord	Coordination node	maRRum - 'and', um
18	Obj	Object	Object
19	Pnom	Nominal Predicate	Nominal Predicate , nouns as predicates
20	Pred	Main Predicate	Main Predicate
21	Sb	Subject	Subject

Figure 4: Dependency relations (Analytical functions)

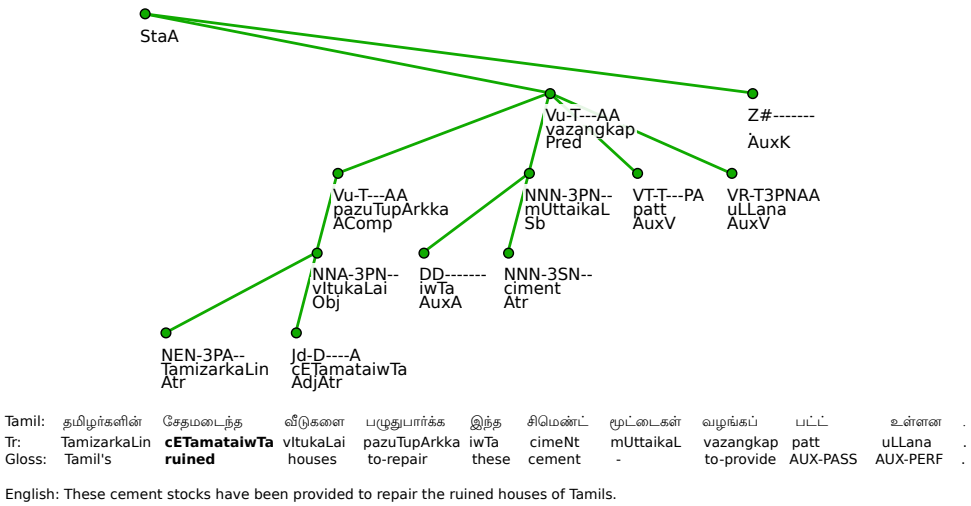


Figure 5: *AdjAtr:* Adjectival attribute

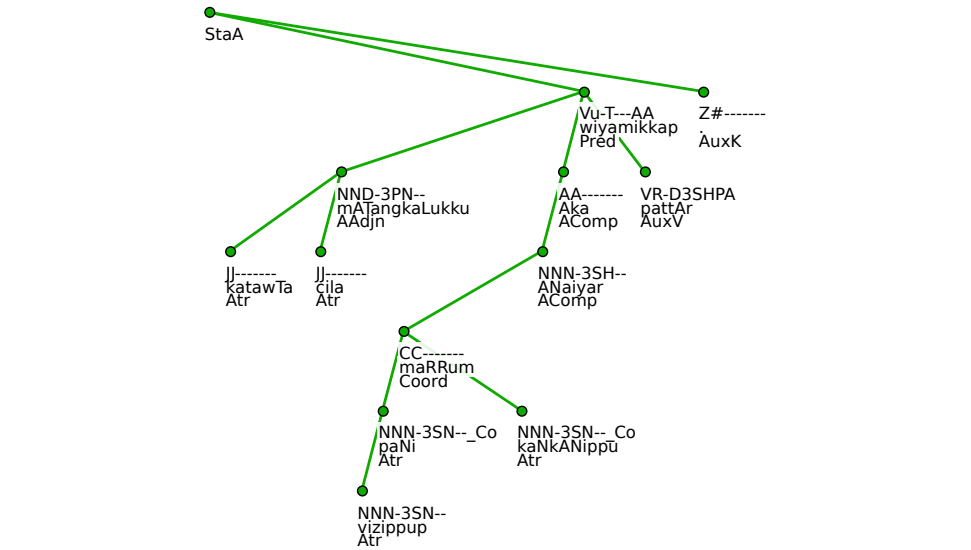


Figure 6: *Coord:* Coordination head

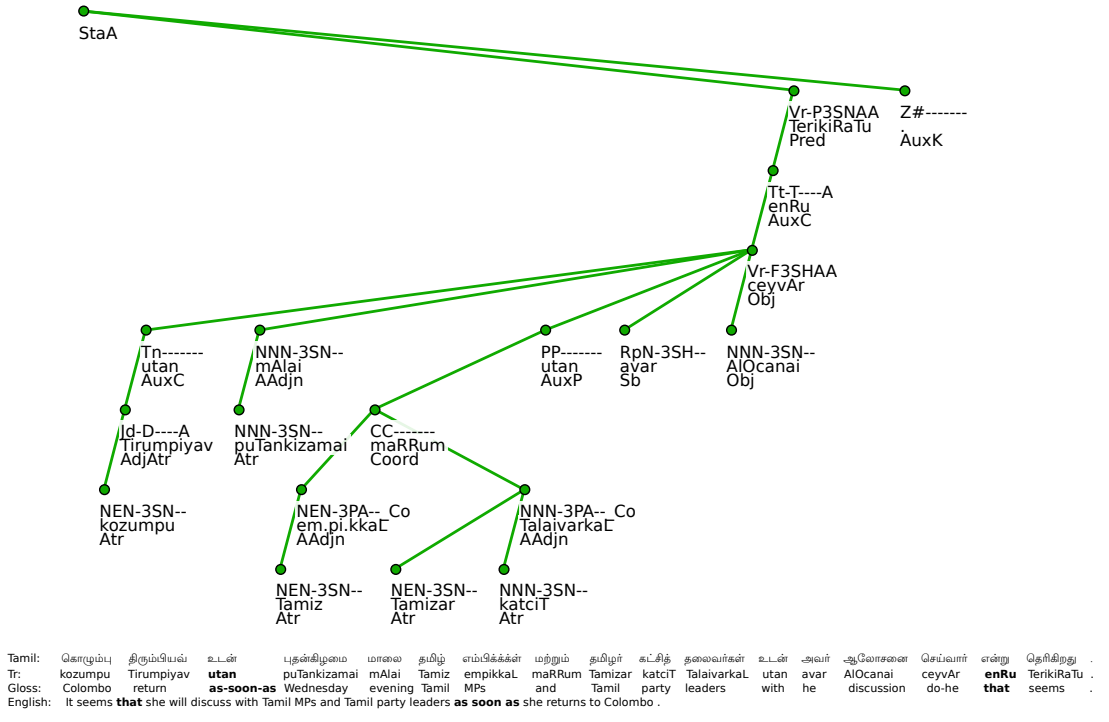


Figure 7: *AuxC:* Subordinating conjunctions

the coordination is done by adding *morphological suffix* to every conjoining elements, and in the other method, it is done by adding the word *maRRum* ('and') between the last 2 conjoining elements. The Figure 6 shows an example annotation of coordination conjunction ('and' style). The `'is_member'` attribute of each conjoining elements will be set to 1 to denote that they are coordination members. This style of conjunction also takes care of shared modifiers.

4.3. AuxC

AuxC is used to mark the subordinate conjunctions. 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 (Lehmann, 1989) can be verbs, nouns or postpositions after nominalized clauses. There are three complementizing verbs - *en* ('say'), *pOl* ('seem') and *Aku* ('become'). They have grammatical function during embedding of clauses, otherwise they retain their lexical meaings. The following list provides some of the noun complementizers - *pOTu* ('time, during'), *mun* ('before'), *piRaku* ('after'), *utan* ('immediacy, as soon as'), *varai* ('as long as') and etc. The postpositions can also be interpreted as subordinating conjunction words when they are preceded by nominalized clauses. The Figure 7 shows how the *AuxC* relation is marked. The example has *utan* ('as soon as') and *enRu* ('that') as subordinating conjunctions.

5. Issues in treebank development

We list below two main issues we faced during the annotation of the treebank.

5.1. Tokenization

There is a little confusion over whether the suffixes *Aka* (adverbial suffix) and *Ana* (adjectival suffix) should be separated from the wordforms. For ex: *Aka* can occur as a pure adverbial suffix as in *viraiv*Aka ('quickly') or as a particle as in *vAzTTuvaTaRk*Aka ('for the sake of greeting'). Splitting the *Aka* as in the latter case might be useful than the former case. At present, the tokenizer separates all the instances of *Aka* (adverbial suffix) and *Ana* (adjectival suffix) from the end of the tokens.

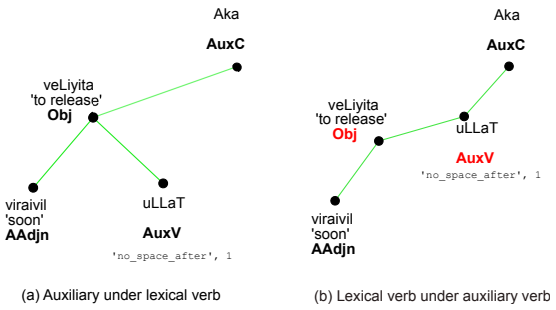


Figure 8: Auxiliary attachment dilemma

5.2. Aux dilemma

In *a-layer* annotation, issues such as, handling of auxiliary verbs whether the auxiliaries should be hanged under the lexical verbs or the lexical verbs should be hanged under the auxiliary verbs, still remain. One reason for this dilemma is, that in Tamil, lexical verbs always precede auxiliary verbs but it is the auxiliary verb which codes the agreement and establishes morphological clues when there is an embedding of a clause into another clause. On the one hand,

it is the lexical verb which is the head of a clause, so the lexical verb can be marked as a head. On the other hand, it is the auxiliary verb which makes a connection between the embedded clause and the clause being embedded into, so the auxiliary too can be qualified as a head, in which case the lexical verb becomes the child of the auxiliary verb. We chose to go by the first solution, i.e. attaching auxiliary verb under the lexical verb. Figure 8 shows an example for the annotation of a complex predicate *veLiyita uLLaTAka* ('to be released'), in which *veLiyita* ('to release') is the lexical verb and *uLLaT* ('be, exist') is an auxiliary verb. The Figure 8 shows both the choices, in which our style of annotation corresponds to the first part of the Figure. The `'no_space_after',1` indicates that the suffix *'Aka'* is part of the auxiliary verb *uLLaT* ('be, exist').

6. Conclusion and Future work

In this paper, we presented our efforts to develop a PDT style dependency treebank for Tamil. Apart from some of the issues we mentioned in the previous section, there are two things we would like to do in the future. First, the size of TamilTB is still very small compared to other popular treebanks. We obviously want to increase the size of the data, so that various language experiments can be performed. Second, the present annotation does not include tectogrammatical layer of annotation. We would like to add tectogrammactical annotation too to our data.

7. Acknowledgement

The research leading to these results has received funding from the European Commission's 7th Framework Program (FP7) under grant agreement n° 238405 (CLARA), and from grant MSM 0021620838. We also would like to thank anonymous reviewers for their useful comments.

8. References

R. Begum, S. Husain, A. Dhwaj, D. Sharma, L. Bai, and R. Sangal. 2008. 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).

A. Bharati, M. Gupta, V. Yadav, K. Gali, and D.M. Sharma. 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.

T. Brants. 2000. TnT – A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, Seattle, WA.

V. Dhanalakshmi, M. Anand Kumar, RU. Rekha, KP. Soman, and S. Rajendran. 2010. Grammar Teaching Tools for Tamil Language. In *Technology for Education Conference (T4E 2010)*.

J. Hajič, J. Panevová, E. Hajičová, P. Sgall, P. Pajas, J. Štěpánek, J. Havelka, M. Mikulová, Z. Žabokrtský, and M. Ševčíková-Razímová. 2006. Prague Dependency Treebank 2.0. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia.

J. Hajič. 2004. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Karolinum, Charles University Press, Prague, Czech Republic.

J. Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press.

S. Janarthanam, U. Nallasamy, L. Ramasamy, and C. Santhoshkumar. 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.

P. Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

T. Lehmann. 1989. *A Grammar of Modern Tamil*. Pondicherry Institute of Linguistics and Culture.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Comput. Linguist.*, 19:313–330, June.

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

L. Ramasamy and Z. Žabokrtský. 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.

L. Ramasamy and Z. Žabokrtský. 2011b. Tamil dependency treebank (TamilTB) - 0.1 annotation manual. Technical Report TR-2011-42, Univerzita Karlova v Praze, Praha, Czech Republic.

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

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

D. Zeman. 2009. Maximum spanning malt: Hiring world's leading dependency parsers to plant indian trees. In *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing*. NLP Association of India.