Automatic Transformation of the Thai Categorial Grammar Treebank to Dependency Trees

Christian Rishøj
CST
University of Copenhagen
crjensen@hum.ku.dk

Taneth Ruangrajitpakorn
HLT Lab
NECTEC
taneth.rua@nectec.or.th

Prachya Boonkwan
HLT Lab
NECTEC
prachya.boo@nectec.or.th

Thepchai Supnithi
HLT Lab
NECTEC
thepchai.sup@nectec.or.th

Abstract

A method for deriving an approximately labeled dependency treebank from the Thai Categorial Grammar Treebank has been implemented. The method involves a lexical dictionary for assigning dependency directions to the CG types associated with the grammatical entities in the CG bank, falling back on a generic mapping of CG types in case of unknown words. Currently, all but a handful of the trees in the Thai CG bank can unambiguously be transformed into directed dependency trees. Dependency labels can optionally be assigned with a learned classfier, which in a preliminary evaluation with a very small training set achieves 76.5% label accuracy. In the process, a number of annotation errors in the CG bank were identified and corrected. Although rather limited in its coverage, excluding e.g. long-distance dependencies, topicalisations and longer sentences, the resulting treebank is believed to be sound in terms structural annotational consistence and a valuable complement to the scarce Thai language resources in existence.

1 Introduction

Syntactic resources play an essential role for the majority of NLP applications, but for Thai language, openly available syntactic resources are few in number: So far, the only reported resources are the CG treebank [Ruangrajitpakorn et al., 2009] and the NAiST dependency bank [Wacharamanotham et al., 2007, Sudprasert, 2008]. Others are either unpublished or minuscule in size. Rather than relying exclusively on labor-intensive manual annotation for further expanding the resources, it would be economically sound to leverage existing efforts and transform an existing treebank in one formalism into another.

1.1 Categorial grammar

Categorial grammar (CG) is a lexicalised theory in natural language syntax motivated by the principle of constitutionality and organised according to the syntactic elements [Steedman, 2000, Ajdukiewicz, 1935], and forms the theoretical basis for the Thai CG treebank. The resource building effort has been very fruitful, but there remains phenomena of Thai languge, including long-distance dependencies and topicalisation [Warotamasikkhadit, 1997], which are unhandled by the instantiation of CG currently in use.

Additionally, although Thai language belongs to a fixed word order typology, Thai spoken language exhibits some flexibility in word order, due to the occasional preference of Thai language users for correspondence in rhyme. As an example, consider the following sentence[1]:

> อังกฤษ คิดค้น อย่าง หนัก วัคซีน
> ป้องกันเชื้อ ไวรัส ไข้หวัดนก
>
> (**Lit**: England/NE invent/V "-ly"/ADVPFX heavy/ADJ vaccine/N protect/V virus/N avian_flu/NP)
>
> "The British are strenuously developing an Avian Flu vaccine."

The adverbial compound formed by อย่าง หนัก ("strenuously") conventionally occurs after the direct object, but is in this sentence, it is realised in the pre-direct object position[2] in order for the last syllable [nak^L] of อย่าง หนัก to rhyme with the first syllable [wak^H] of วัคซีน ("vaccine"). To some degree, this phenomenon from spoken language shines through in written language, especially in the domains of news and recent politics, and is causing a challenge for the employment of CG grammar.

---

[1] The Thai adverbialising prefix อย่าง can be likened to the English "-ly" suffix, which produces an adverbial form from an adjective. Artificial word boundaries are inserted for clarity.

[2] When language users exploit this flexibility in word order to produce aesthetically pleasing sound patterns, it results in a marked form, but the phenomenon is nonetheless productive, and encountered frequently enough to necessitate handling in NLP applications.
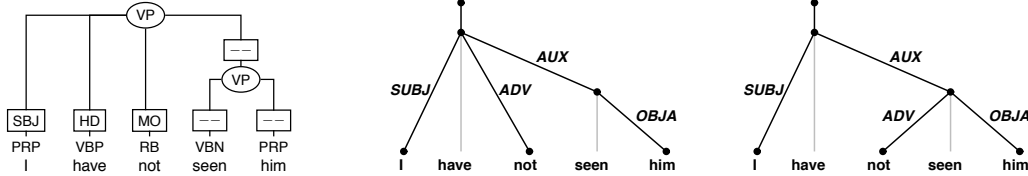
Figure 1: Example of ambiguity arising only in dependency representation [Daum et al., 2004]

1.2 Dependency representation

In recent years, dependency grammar has seen a dramatic increase in interest, likely due to a number of appealing properties of the representation. In comparison to phrase structure grammar, dependency structures provide a relatively direct encoding of predicate-argument structure, which is relevant to subsequent analyses [Nivre, 2005]. Dependency representation is arguably better suited for languages with flexible word order. Additionally, having no non-terminal nodes, dependency structures are often perceived as leaving room for less ambiguity as well as being more computationally managable.

Certainly, dependency representation has drawbacks of its own in terms of ambiguity, some of which are specific to dependency representation. In particular, Figure 1 shows a construction with an unambiguous constituent structure, which in dependency space is ambiguous with respect to the attachment of the adverb [Daum et al., 2004].

Furthermore, dependency structure allows for a number of ways to represent coordinated phrases, some having the *coordinating conjunction as head* (CCH) of the coordinate structure, and a special dependency label *CJT* that does not describe the grammatical function of the conjuncts (Figure 2a). Another option is having the *coordinating conjunction as dependent* (CCD) of one of the conjuncts, thus allowing one conjunct to occur with a dependency label expliciting its grammatical function (Figure 2b). McDonald and Nivre [2007] offer a thorough review of these and other candidate analyses in use. Unfortunately, none of the conceivable representations are unproblematic from a linguistic perspective [Daum et al., 2004], or offer the same transparency as the coordination rules of CCG do [Boonkwan, 2009].

Nonetheless, given the availability of generally applicable, trainable dependency parsers, and reports of beneficial applications of dependency analysis in tasks such as word-alignmeent [Ma et al., 2008] and reordering [Chang et al., 2009] for statistical machine translation, a dependency treebank of good quality is a highly desirable resource.
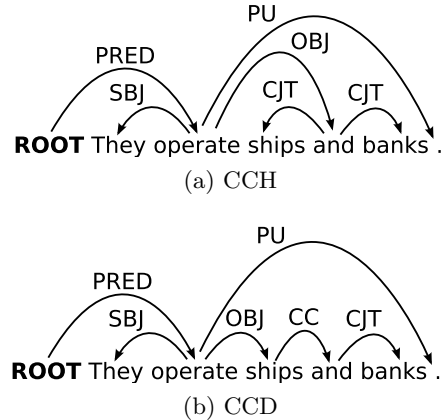

(a) CCH

(b) CCD

Figure 2: Two possible analyses of a coordination [Kübler et al., 2009]

1.3 Outline

The rest of the paper is organised as follows. In Section 2 we briefly review other works dealing with similar transformations, before presenting the approach taken in this work in Section 3. Section 4 describes the experimental setting and results. A discussion follows in Section 5, and conclusions in Section 6 .

2 Related work

In preperation for the CoNLL-X shared task on dependency parsing [Buchholz and Marsi, 2006], a number dependency trees were derived from a number of constituency-based phrase structure treebanks, most of which have grammatical function (e.g. "subject" and "object") as part of the annotation. The conversion process for such treebanks would involve a *head table* with rules of the form

- "the head child of a VP/clause is the child with the HD/predicator/hd/Head function" and

- "[the dependency label] for a token is the function of the biggest constituent of which this token is the lexical head".

The case of the Thai CG bank is different, as it does not directly contain any grammatical functions. On the other hand, identifying head tokens

is relatively straight-forward when augmenting the CG annotation with dependency directions.

Chanev et al. [2006] faced a similar situation in their transformation of the BulTreeBank to dependency representation. Heads were first identified from explicitly stated rules in a head table. Lacking explicit grammatical functions in the source treebank, they explored a heuristic rule-based approach for the labeling with a *dependency table,* containing rules based on parent constituents. Although good results are achieved, they report of errors like mistaken subjects and objects.

3 Methodology

The situation with the Thai CG bank is a little different. Together with a set of combinatory grammer rules, the CG type tags and bracketing present in the treebank unambigously specify the constituent structure of the treebank sentences. When the CG type tags are augmented with dependency directions, a dependency tree can be derived with relative ease from the CG-based constituents. Grammatical functions, however, are not immediately evident from the CG trees.

We first describe a relatively straight-forward method for deriving the dependency trees, and next consider the more daunting task of assigning functional labels to the dependency arcs. Figure 3 shows a schematical overview of the proprosed method.

3.1 Terminology

For any given CG type t, we use the *arity* (admittedly a bit sloppily) to denote the ordered list of arguments expected by a type. The arity of the type $s\backslash np/ws/np$ is thus $/np$, $/ws$ and $\backslash np$ — that is,

- a noun phrase from the right, followed by

- a subordinate clause beginning with the Thai word ว่า ("that", subordinate clause marker), and

- another noun phrase from the left.

Complementary, we define the *yield* of t as the set of possible CG types which may result from functional application of a CG rule. The transitive verb type $s\backslash np/np$, for example, yields

- $s\backslash np$ and

- s

after receiving a np to the right, and another np to the left, respectively. This is simply the basic combinatory CCG rules:

$$X/Y \quad Y \quad \Rightarrow \quad X$$
$$Y \quad X\backslash Y \quad \Rightarrow \quad X$$

Algorithm 1 Pseudo-code for propagating dependency directions to non-terminal nodes.

```
def propagateDirections(node):
  for child in node.children:
    propagateDirections(child)
  if node.hasDependencyDirection:
    return
  for child in node.children:
    if child.yields(node.type):
      node.depDirs = child.depDirs
      break
```

3.2 Dependency directions

The first transformation step, which can be regarded as a preprocessing step before the actual transformation, involves a lexical dictionary for assigning dependency directions to the CG types associated with the grammatical entities in the CG bank, falling back on a generic CG to CDG mapping in case of unknown words.

Note that only terminal nodes will have assigned dependency directions assigned by this procedure. Dependency directions are propagated to non-terminal nodes in a bottom-up fashion by the procedure `propagateDirections` (Algorithm 1). In identifying which child to adopt dependency directions from, the parent node type is checked against the yield of each child node.

3.3 Head finding

Dependency arcs are assigned by a procedure that implements the CDG dependency derivation rules introduced by Boonkwan et al. [2011] (motivated by Collins, 1999). The idea is to trace the derivation implied by the CG rules, registering the dependency relations specified by the CDG-augmentation along the way.

The derivation rules handled are as follows. Let $c : d$ signify a CDG type c and a dependency structure d, and the notion $h(d_1) \to h(d_2)$ represent a dependency arc between the head of d_1 and the head of d_2 (with $h(d_1)$ governing $h(d_2)$) . Then the derivation rules are:

$$
\begin{aligned}
X/{<}Y{:}d_1 \ \ Y{:}d_2 &\ \Rightarrow\ h(d_1) \leftarrow h(d_2) \\
X/{>}Y{:}d_1 \ \ Y{:}d_2 &\ \Rightarrow\ h(d_1) \to h(d_2) \\
Y{:}d_1 \ \ X\backslash{<}Y{:}d_2 &\ \Rightarrow\ h(d_1) \leftarrow h(d_2) \\
Y{:}d_1 \ \ X\backslash{>}Y{:}d_2 &\ \Rightarrow\ h(d_1) \to h(d_2)
\end{aligned}
$$

In a simple example, *Mary*[np] *drinks*[$s\backslash < np/ > np$] *fresh*[$np/ < np$] *milk*[np], *fresh* would combine with *milk*, yielding an np and a dependency arc specifying *milk* as head of *fresh*.

In addition to the standard combinatory rules for forward and backward functional application above, the Thai CG bank makes use of a CCG-style *serialisation* rule to handle e.g. serial verb
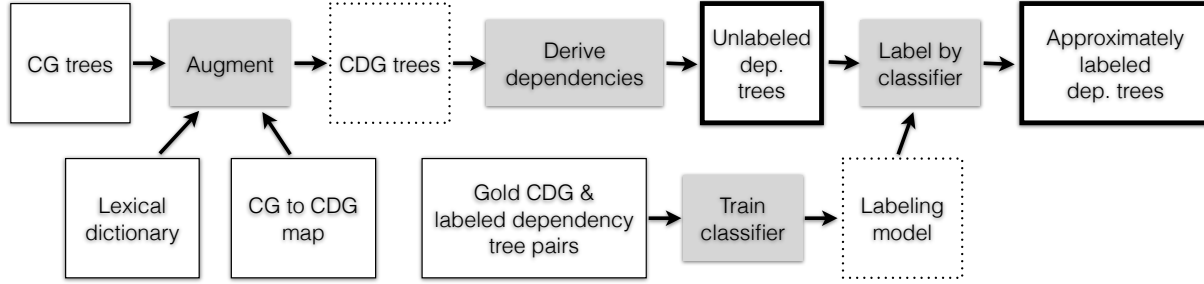
Figure 3: Transformation overview

constructions, which are used in Thai (and Chinese) to express serial or consecutive events. As Boonkwan [2009], we take the notion of a *serial verb construction* to mean a series of verbs or verb phrases without explicit connectives marked with (or understood to have) the same grammatical categories, and sharing at least one common argument, typically a subject.

As an example, the verbs ตรวจ ("examine") and พบ ("find") occur serially in the following sentence[3] from the CG bank, indicating a resultative course of events [Thepkanjana and Uehara, 2009]:

นักวิชาการ ตรวจ พบ ไวรัส โคโรน่า ใน ชะมด

(**Lit**: scientist/N examine/V find/V virus/N corona/N in/PP civet/N)

"The scientist examined the civet and found coronavirus."

We introduce a generalised derivation rule for serial constructions which simply designates the head of the first dependency structure as governing the head of the following dependency structure:

$$X{:}d_1 \ \ X{:}d_2 \ \ \Rightarrow \ \ h(d_1) \to h(d_2)$$

The rule is generalised in the sense that it handles *serial noun constructions* as well as serial verb constructions.

Further CCG-style combinatory rules, such as functional composition and type raising, are not currently in use in the Thai CG bank, and therefore not handled by the transformation.

An outline of the head finding procedure is given as Algorithm 2. Intuitively, the algorithm proceeds by, for each node in turn, beginning at the terminal nodes, identifying sibling nodes which satisfy the arity of the of the node CG type. For each sibling node satisfying an argument, the dependency derivation rule is applied and the sibling is removed.

---

[3]Articifial word boundaries inserted for clarity.

Algorithm 2 Pseudo-code for the head-finding procedure.

```
def assignHeads(node):
  for c in node.nonterminalChildren:
    assignHeads(c)
  for c in node.terminalChildren:
    assignHeads(c)
  for arg in node.type.arguments:
    if arg.side == 'right':
      sibl = node.rightSiblings.first
    else
      sibl = node.leftSiblings.last
    break unless arg.matches(sibl)
    if arg.side == 'right':
      if arg.dependencyDir == '>':
        registerHead(sibl, node)
      else: # <
        registerHead(node, sibl)
      node.rightSiblings.shift
    else: # left
      if arg.dependencyDir == '>':
        registerHead(node, sibl)
      else: # <
        registerHead(sibl, node)
      node.leftSiblings.pop
```

It is worth noting that while both terminal and non-terminal nodes are involved in this process, we are only interested in assigning dependency arcs to terminal nodes, as non-terminals are absent in the all-terminal dependency structure. This is ensured by an implementation detail of the procedure `registerHead` (omitted from Algorithm 2), in which non-terminal nodes act as proxies for their terminal heads.

3.4 Dependency labeling

Although the CDG-augmentation of the CG treebank implies a dependency structure for each sentence, there are no immediate clues available about the specific grammatical functions of dependency arcs. Obviously, there are some clear-cut cases: When a token with CDG type $((\text{s}\backslash < \text{np})\backslash > (\text{s}\backslash < \text{np}))\backslash < \text{num}$ modifies a token with CDG type num, it must be an application of a *quantifier* (with dependency type "quan"), as exemplified in Figure 4.

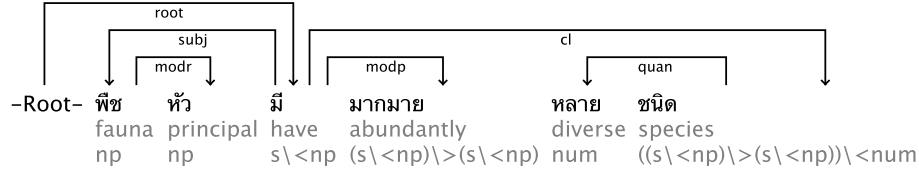Other cases are less obvious. Even when taking

Figure 4: An unmistakeable labeling case: Given the CDG types of หลาย ("diverse") and ชนิด ("species"), the only dependency label supported by the examined data is "quan" (*quantifier*).

dependency direction and argument position into consideration, there are still cases with several possible dependency types, as shown in Figure 5.

While for many practical purposes an unlabeled dependency structure is sufficient, having proper dependency labels is nonetheless desirable. In lack of an exact transformation, the approach explored in this work relies instead on training a classifier to predict the correct dependency label given local features of the tokens involved, as they occur in the dependency structure derived from the CDG tree.

From a related work (manuscript in preparation), we have obtained a number of CDG trees from labeled dependency trees. Such pairs of *labeled* dependency trees and CDG trees can serve as training material for the label classifier.

We evaluate different feature sets for classification task. The basic feature set contains:

- CDG type of token and its head

- # of left siblings

- # of right siblings

- Dependency direction (L/R)

- Concatenations of token/head CDG and # of left siblings

Other feature sets extend the basic set including additional information:

+forms Token and head word form.

+POS(L) Part-of-speech tag for the token, as assigned by a learned part-of-speech tagger[4].

+POS(G) "Gold" part-of-speech tag as found in the labeled dependency tree.

4 Experiments

The Thai CG bank made available for this research contains 1,428 phrases with corresponding CG trees. Trees are mostly medium to low in token count, with 49 single-token trees, and the longest phrase having 9 tokens. Median phrase length is 3

---

[4]SVMTool [Giménez and Marquez, 2004], trained on the Orchid corpus [Sornlertlamvanich et al., 1997].
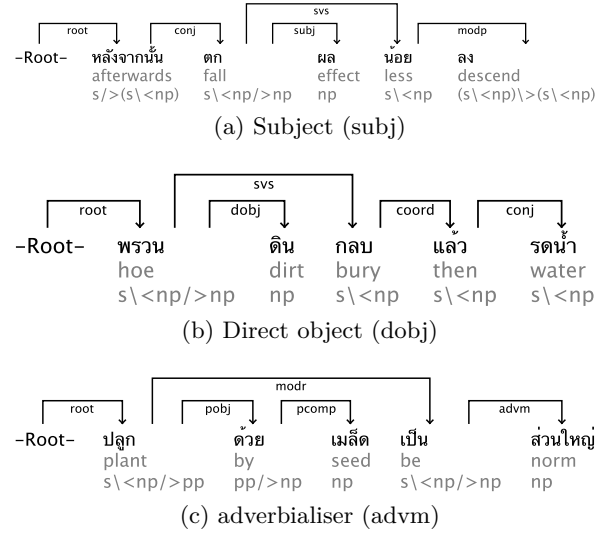

(a) Subject (subj)

(b) Direct object (dobj)

(c) adverbialiser (advm)

Figure 5: Different labeling of left-pointing dependency arcs with $\mathtt{s\backslash < np/ > np}$ as head CDG type and $\mathtt{np}$ as dependent CDG type. See Figure 6 for depency type abbreviations.

tokens. With a total token count of 5,143 tokens, the arithmetic mean length is 3.60.

As there is no explicit sentence boundary marker in Thai [Satayamas and Kawtrakul, 2004], it is often unclear what constitutes a sentence. Thus, rather than ordinary sentences, the treebank contains phrases of different types, reflecting the partitioning of token sequences made by the treebank annotators for the purpose of treebanking:

- 539 verb phrases or subject-omitted phrases ($\mathtt{s\backslash np}$),

- 363 sentences ($\mathtt{s}$),

- 372 noun phrases ($\mathtt{np}$) and

- 4 prepositional phrases ($\mathtt{pp}$).

The lexical dictionary used contains possible CDG types for 38.250 word forms, with an average of 2 types listed per word form. For six of the word forms, the dictionary lists several possible dependency directions for a single CG type. These confusable CDG types and the dictionary entries they occur for are listed in Table 1.

An example sentence from the treebank affected by the ambiguous mapping of the adverb ตอนนี้

Word form(s)	CG type	Possible CDG equivalents	Interpretation
น่า	$(s\backslash np)/(s\backslash np)$	$(s\backslash <np)/>(s\backslash <np)$ $(s\backslash <np)/<(s\backslash <np)$	Adverb "please" Auxiallary verb "should"
ตอนนั้น ("at that time") ตอนนี้ ("at present") ขณะนั้น ("at that moment") ขณะนี้ ("this moment")	s/s	$s/>s$ $s/<s$	Conjunction Adverb of time
กับ	$np\backslash np/np$	$np\backslash >np/>np$ $np\backslash <np/>np$	Preposition "with" Conjunction "and"

Table 1: Entries in the lexical dictionary which are ambiguous with respect to dependency direction

("at present") is [5]:

ตอนนี้ เธอ กำลัง ยุ่ง มาก

(**Lit**: this-moment/ADV she/PRON "-ing"/AUX busy/ADJ very/ADV)

"At this moment she is being very busy."

Examples of the latter two ambiguity classes of the lexical dictionary (Table 1) were not encountered in the treebank — i.e. the affected word forms do not occur with an ambiguous CG type. In dealing with these ambiguous entries in the lexical dictionary, we simply (and naively) choose the first mapping option.

The generic CG to CDG mapping, used in addition to the lexical dictionary as fallback for word forms not found in the lexical dictionary, also exhibits some degree of ambiguity. The seven CG types with multiple possible CDG equivalents are listed in Table 2.

In evaluating the classification-based approach to assigning dependency labels, a sample of 678 labeled dependency edges from the NAiST dependency treebank [Wacharamanotham et al., 2007] was used, along with corresponding CDG trees. The feature sets suggested in section 3.4 on page 4 were evaluated with four different classifiers[6] (see Table 3).

5 Discussion

Transforming CDG-augmented CG trees to unlabeled dependency trees was successful. In this work, the issue of ambiguous CDG types affects only a very small number of trees, but remains an issue to be aware of.

CG type	Possible CDG equivalents
$np\backslash np/np$	$np\backslash >np/>np$ $np\backslash <np/>np$
s/s	$s/>s$ $s/<s$
$s\backslash s$	$s\backslash >s$ $s\backslash <s$
$(s\backslash np)/(s\backslash np)$	$(s\backslash <np)/>(s\backslash <np)$ $(s\backslash <np)/<(s\backslash <np)$
$s/(s\backslash np)$	$s/>(s\backslash <np)$ $s/<(s\backslash <np)$
$s\backslash(s\backslash np)$	$s\backslash <(s\backslash <np)$ $s\backslash >(s\backslash <np)$
$s/(s\backslash np)/np$	$s/>(s\backslash np)/>np$ $s/<(s\backslash np)/>np$

Table 2: Cases of CD to CDG mappings which are ambigous with respect to dependency direction

For dependency labeling, only a small amount of training data (in the form of sentences with both CDG and labeled dependency analyses) was available for this prelimenary experiment. Using this, dependency labels were not reliably recoverable (76.5% label accuracy). However, it seems hopeful that better recovery of dependency labels can be obtained with this approach once more training material become available.

Head and dependent word forms, as well as part-of-speech tags, were beneficial as added features for the label classifier, resulting in a substantial reduction in error rate — from 0.357 to 0.235 ($\approx$ 34%).

On the other hand, rather than the approximated classifier-based approach to labeling, one could consider settling for an exact but partial labeling by only assigning those dependency labels which unambigously arise from tuples of head and dependent CDG types. However, the dependency labels obtainable with absolute certainty in this way are often of the less interesting kind — e.g. "conj" for a $s\backslash <np$ governed by a $s/>(s\backslash <np)$ — while more useful labels remain ambiguous.

---

[5]The Thai auxillary verb กำลัง indicates the present participle, meaning "in the act of", similar to the English suffix, "-ing". Articifial word boundaries are inserted for clarity.

[6]Experiments with the learners were done using leave-one-out cross-validation, with the exception of LibSVM, which was run using the standard K-fold cross-validation of the easy.py script [Chang and Lin, 2001].

Classifier	Basic	+forms	+POS(L)	+POS(G)	+POS(L) & +forms	+POS(G) & +forms
Random Forest	61.9%	69.6%	66.1%	68.9%	72.9%	71.8%
LibSVM	64.3%	73.9%	66.1%	70.2%	74.5%	**76.5%**
Nearest Neighbors	60.5%	64.8%	62.4%	64.5%	70.5%	70.4%
Naive Bayes	60.6%	63.7%	61.7%	65.8%	63.7%	66.2%

Table 3: Accuracy of different classifiers and feature sets in recovering the correct dependency labels

Complements *subject* (subj) • *clausal subject* (csubj) • *direct object* (dobj) • *indirect object* (iobj) • *prepositional object* (pobj) • *prepositional complement* (pcomp) • *subject or object predicative* (pred) • *clausal predicative* (cpred) • *conjunction* (conj) • *subordinating conjunction* (sconj) • *nominaliser* (nom) • *adverbialiser* (advm)

Adjuncts *parenthetical modifier* (modp) • *restrictive modifier* (modr) • *tense modifier* (modt) • *mood modifier* (modm) • *aspect modifier* (moda) • *locative modifier* (modl) • *parenthetical apposition* (appa) • *restrictive apposition* (appr) • *relative clause modification* (rel) • *determiner* (det) • *quantifier* (quan) • *classifier* (cl) • *coordination* (coord) • *negation* (neg) • *punctuation* (punc) • *double preposition* (dprep) • *parallel serial verb* (svp) • *sequence serial verb* (svs)

Figure 6: Dependency types from the annotation guidelines [Sudprasert, 2008] for the NAiST dependency treebank.

6   Conclusion and future work

In the process, a considerable amount of syntactical and annotational errors in the Thai CG bank were identified and corrected. The authors are of the belief that this work has not only provided a means for continual expansion of resources for Thai natural language processing, but also helped improve the quality the existing CG resource.

As a related work (manuscript in preparation) progresses, more CDG trees for labeled NAiST dependency trees will become available. With this extra training material, the learned classifier used for labeling in this work should become more reliable. Further improvement might also be achievable from an expanded feature set, including for example the argument position (in addition to side) and one or two generations of grandparent nodes.

Further development of the Thai CDG formalism is also expected, in particular for the analysis of sentence-like noun phrases. This will likely need special handling in the dependency representation as well. Currently, the NAiST annotation guidelines do not specify a label for this phenomenon.

References

K. Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1(1):27, 1935.

P. Boonkwan. A memory-based approach to the treatment of serial verb construction in combinatory categorial grammar. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, page 10–18, 2009.

P. Boonkwan, M. Steedman, and T. Supnithi. Grammar induction from text using small syntactic prototypes. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (to appear)*, Chiang Mai, 2011.

S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Tenth Conference on Computational Natural Language Learning*, page 149, 2006.

A. Chanev, K. Simov, P. Osenova, and S. Marinov. Dependency conversion and parsing of the BulTreeBank. In *Proc. of the LREC-Workshop Merging and Layering Linguistic Information*, 2006.

C. Chang and C. Lin. LIBSVM: a library for support vector machines, 2001. URL http://www.csie.ntu.edu.tw/cjlin/libsvm.

P. C Chang, H. Tseng, D. Jurafsky, and C. D Manning. Discriminative reordering with chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, page 51–59, 2009.

M. Collins. *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania, 1999.

M. Daum, K. Foth, and W. Menzel. Automatic transformation of phrase treebanks to dependency trees. In *Proceedings of LREC*, 2004.

J. Giménez and L. Marquez. SVMTool: a general POS tagger generator based on support vector machines. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004.

S. Kübler, R. McDonald, and J. Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–127, 2009.

Y. Ma, S. Ozdowska, Y. Sun, and A. Way. Improving word alignment using syntactic dependencies. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)*, page 69–77, 2008.

R. McDonald and J. Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

J. Nivre. Dependency grammar and dependency parsing. *MSI report*, 5133, 2005.

T. Ruangrajitpakorn, K. Trakultaweekoon, and T. Supnithi. A syntactic resource for thai: CG treebank. In *Proceedings of the 7th Workshop on Asian Language Resources*, page 96–101, 2009.

V. Satayamas and A. Kawtrakul. Wide-Coverage grammar extraction from thai treebank. In *Proceedings of Papillon 2004 Workshops on Multilingual Lexical Databases*, 2004.

V. Sornlertlamvanich, T. Charoenporn, and H. Isahara. ORCHID: thai part-of-speech tagged corpus. Technical Report TR-NECTEC-1997-001, 1997.

M. Steedman. *The syntactic process*, volume 131. MIT Press, 2000.

S. Sudprasert. Dependency annotation guideline for thai, version 1.4 (in thai). Technical report, 2008. URL http://naist.cpe.ku.ac.th/tred/.

K. Thepkanjana and S. Uehara. Resultative constructions with "implied-result" and "entailed-result" verbs in thai and english: a contrastive study. *Linguistics*, 47(3):589–618, 2009. ISSN 0024-3949. URL http://www.thefreelibrary.com/-a0204693751.

C. Wacharamanotham, M. Suktarachan, and A. Kawtrakul. The development of web-based annotation system for thai treebank. page 305, Thailand, 2007. URL http://naist.cpe.ku.ac.th/tred/.

U. Warotamasikkhadit. Fronting and backing topicalization in thai. *Mon-Khmer Studies*, 27: 303–6, 1997.