The Thai Universal Dependency Treebank

Panyut Sriwirote[1] **Wei Qi Leong**[2] **Charin Polpanumas**[3]
Santhawat Thanyawong[4] **William Chandra Tjhi**[2]
Wirote Aroonmanakun[1] **Attapol T. Rutherford**[1*]
[1]Department of Linguistics, Chulalongkorn University, Thailand
panyutsriwirote@gmail.com, {awirote,attapol.t}@chula.ac.th
[2]AI Singapore, Singapore
{weiqi,wtjhi}@aisingapore.org
[3]Amazon, Japan
cebril@gmail.com
[4]Faculty of Humanities and Social Sciences, Prince of Songkla University, Thailand
santhawat.t@psu.ac.th

Abstract

Automatic dependency parsing of Thai sentences has been underexplored, as evidenced by the lack of large Thai dependency treebanks with complete dependency structures and the lack of a published evaluation of state-of-the-art models, especially transformer-based parsers. In this work, we addressed these gaps by introducing the Thai Universal Dependency Treebank (TUD), a new Thai treebank consisting of 3,627 trees annotated according to the Universal Dependencies (UD) framework. We then benchmarked 92 dependency parsing models that incorporate pretrained transformers on Thai-PUD and our TUD, achieving state-of-the-art results and shedding light on the optimal model components for Thai dependency parsing. Our error analysis of the models also reveals that polyfunctional words, serial verb construction, and lack of rich morphosyntactic features present main challenges for Thai dependency parsing.

1 Introduction

Dependency parsing is the task of identifying the dependencies between words in a sentence according to the syntax of its language. Many downstream NLP tasks rely on the support of automatic grammatical analysis of sentences, such as event extraction (McClosky et al., 2011; Zhang et al., 2021) and relation extraction (Tian et al., 2021). To allow multilingual approaches to automatic syntactic analysis despite cross-lingual variation, Universal Dependencies (UD) (de Marneffe et al., 2021) provides a consistent framework for annotating parts-of-speech (POS), morphological features, and syntactic relations across all human languages. However, training an accurate dependency parser still requires a large annotated dataset in the target language, and dependency treebanking often requires highly trained annotators. For these reasons, accurate dependency parsing is currently only possible in resource-rich languages.

Dependency parsing for the Thai language is currently not viably accurate due to the lack of large annotated datasets. We found only two publicly available dependency treebanks for Thai: Thai-PUD and Blackboard Treebank (Arreerard et al., 2022). Thai-PUD contains 1,000 sentences of complete dependency annotation, which might not be enough for training dependency parsers. Blackboard Treebank contains 38,588 sentences, but it is only annotated with arcs and does not comply with the standard of UD. The lack of large and reliable training data presents a major obstacle in the field of Thai dependency parsing. In addition, we found no previous studies on the application and evaluation of state-of-the-art transformer-based models for Thai dependency parsing, while transformer-based parsers have achieved state-of-the-art performance in many languages (Straka et al., 2019; Martin et al., 2020; Mrini et al., 2020; Eggleston and O'Connor, 2022).

In this work, we create the Thai Universal Dependency Treebank (TUD), a large corpus annotated with dependency relations according to the UD framework. Our first challenge of this annotation project is that Thai has no word or sentence

*Corresponding author.

Transactions of the Association for Computational Linguistics, vol. 13, pp. 376–391, 2025. https://doi.org/10.1162/tacl_a_00745
Action Editor: Hai Zhao. Submission batch: 7/2024; Revision batch: 9/2024; Published 4/2025.

boundaries. We first had to segment raw text into words and then group them into sentences using the boundaries implied by the annotated dependencies. Our second challenge is the adaptation of the UD framework to fit Thai syntactic idiosyncrasies. We provide guidelines for annotating Thai serial verb constructions and grammatical particles, which are prevalent in the language but have not been explicitly covered in previous works or in the UD guideline itself. TUD is publicly released on GitHub[1] along with the source code and raw data.

Our main contributions in this work can be summarized as follows:

1. We create the largest UD-style Thai dependency treebank, comprising 3,627 trees.

2. We found that the Stanza graph-based parser (Qi et al., 2020) paired with a state-of-the-art encoder-only language model performs best on our dataset, achieving a unlabeled attachment score (UAS) score of 90.90 and LAS score of 84.54.

3. Our error analysis reveals that polyfunctional words, serial verb construction, and lack of rich morphosyntactic features present main challenges for Thai dependency parsing.

2 Related Work

Previously, two Thai dependency treebanks were released, namely, Thai-PUD[2] and Blackboard Treebank.[3] Thai-PUD is a small treebank consisting of 1,000 sentences translated from various European languages and annotated according to the UD framework. The size of this dataset is small compared to other UD treebanks of high-resource languages. It was created as part of the CoNLL 2017 and 2018 Shared Tasks (Zeman et al., 2017, 2018) but the organizer decided to exclude it from the former due to ''consistency issues,'' the details of which were not clarified. Blackboard Treebank is a large non-UD treebank consisting of 130,561 Thai clauses. This dataset does not provide a complete annotation. Subordinate and relative clauses are detached and analyzed as their own trees separate from the main clauses. Moreover, Blackboard Treebank only provides dependency arcs and not dependency types.

Traditional dependency parsers are either transition-based or graph-based. Transition-based models attach dependencies incrementally based on a sequence of actions called ''transitions'' while graph-based models assign scores to all possible dependencies and then extract a valid tree with the best score (McDonald and Nivre, 2007). Recent works have proposed novel transition systems (Ma et al., 2018; Fernández-González and Gómez-Rodríguez, 2019), improved postprocessing of traditionally parsed results (Zmigrod et al., 2020; Mohammadshahi and Henderson, 2021), incorporated higher-order information (Ji et al., 2019; Zhang et al., 2020), and employed joint training with related tasks (Zhou and Zhao, 2019) to boost performance.

So far, no Thai dependency parsers have achieved performance competitive with state-of-the-art models in high-resource languages such as English, for which a result as high as 97.42 UAS and 96.26 labeled attachment score (LAS) has been reported (Mrini et al., 2020). Singkul and Woraratpanya (2019) utilize convolutional neural networks and long short-term memory models as encoders for transition-based and graph-based parsers, achieving 78.48 UAS (LAS not reported) on Thai-PUD. Yasuoka (2023) formulates the task as sequence labeling for Thai dependency parsing and utilizes a pretrained transformer, achieving 77.53 LAS (UAS not reported) on Thai-PUD. Language-agnostic models, capable of parsing any structures in any languages, have also been tested on Thai but tend to perform poorly. Straka (2018) relies only on static embeddings and achieves only 0.88 UAS and 0.65 LAS on Thai-PUD. Kondratyuk and Straka (2019) leverage multilingual BERT (Devlin et al., 2019) but still achieve only 49.05 UAS and 26.06 LAS on Thai-PUD.

3 Annotation Guidelines for Thai Universal Dependency Treebank

We follow the UD annotation guidelines for the most part. However, we need to clarify parts of the guidelines to accommodate the idiosyncrasies of the Thai language.

[1] https://github.com/nlp-chula/TUD.

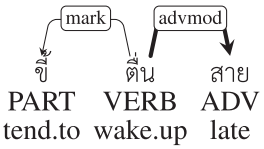
[2] https://universaldependencies.org/treebanks/th_pud/index.html.

[3] https://aiforthai.in.th/corpus.php.

3.1 How to Distinguish Morphological Units from Syntactic Units

Unlike many writing systems, such as English, in which word boundaries are standardized in the orthography by having spaces appear between each word, the Thai script does not separate words by spaces or any other punctuation. This creates ambiguities regarding where exactly each ''word'' starts and ends even among native speakers, obscuring the boundary between morphology and syntax. This is not ideal since the separation between syntax and morphology is especially important in dependency treebanks as it determines if an expression will become one or multiple tokens in the tree. When in doubt, we apply the lexical integrity principle, which states that subparts of a word cannot be modified by syntactic operations, such as negation or adverbial modification.

Some common ambiguities involve the adjective-forming particles น่า 'worthy of VERB,' ใจ 'having a mind that VERB,' ขี้ 'tend to VERB,' ช่าง 'good at VERB,' and หัว 'having a head that VERB.' These words are highly productive and are sometimes discussed as ''prefixes'' (Smyth, 2002), a term that might imply morphological units. However, by applying the lexical integrity principle, we can see that words coming after these so-called prefixes can either be negated, adverbially modified, or both, meaning they should be separated as their own tokens. Tree (1) shows an example where the word after ขี้ is modified by an adverb.



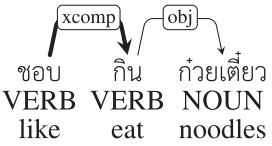
(1) 'tend to wake up late'

3.2 Serial Verb Constructions in TUD

Thai syntax allows two or more verb phrases (VPs) to be strung together in the same clause without explicit linking words or inflectional morphology. According to the UD framework, this corresponds to either the *advcl*, *xcomp*, *ccomp*, or *compound* relations, based on 1) whether the following verbs are core arguments, and 2) the ''same subject'' criterion.

3.2.1 VP Complement

If the subject of the second verb cannot be explicitly stated and must be the same as the subject of the first verb, then *xcomp* is used when the second verb is a core argument. For example,

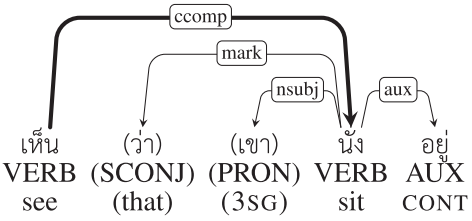


(2) '(Someone) likes to eat noodles.'

In (2), 'eat' is a verbal complement of 'like,' and the two verbs share the same subject. Therefore, *xcomp* is the dependency relation.

3.2.2 Clausal Complement

If the subject of the second verb could be different from the subject of the first verb and the second verb is a core argument of the first verb, then *ccomp* is used. A good test for this relation is to insert the complementizer ว่า and an appropriate subject to see if the meaning remains the same. For example,



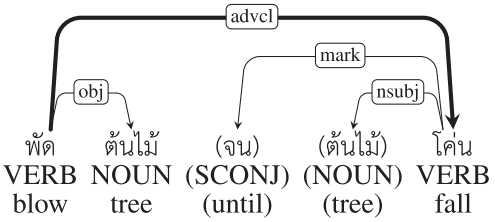
(3) '(Someone) sees that s/he is sitting.'

In (3), we could add the complementizer ว่า 'that' and a subject เขา 's/he' between the two VPs and the meaning remains unchanged.

3.2.3 Resultative Serial Verb Construction

Resultative serial verb constructions (RSVCs) are a type of serial verb construction in which the first VP describes an action, and the second VP indicates the result or outcome of that action. We use *advcl* for this construction because the meaning of the sentence can be explicitly expressed by inserting an appropriate (non-complementizer)

subordinating conjunction and a subject. For example,

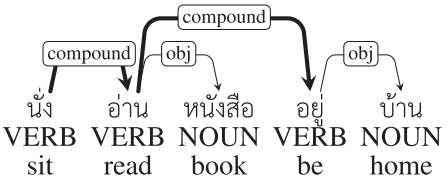


(4) '(Something) blows a tree until the tree falls.'

In (4), the meaning of the sentence would be the same as when we insert the subordinating conjunction จน 'until' and a subject ต้นไม้ 'tree'. Hence, *advcl* is the appropriate relation.

3.2.4 Other Serial Verb Constructions

In contrast to RSVCs, dependent VPs in other Thai serial verb constructions must share their subjects with the head VP. They differ from VP complements (Section 3.2.1) in that they are non-core dependents. These VPs give additional detail that is not necessary to complete the expression, providing directional, perceptional, purposive, and stative information or describing concurrent events (Takahashi, 2009). We use the *compound* relation[4] for this construction. For example,



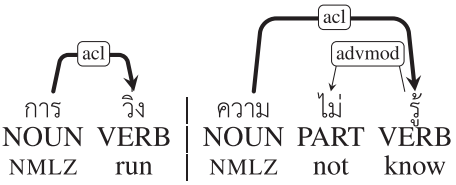
(5) '(Someone) sits and reads books at home.'

In (5), 'sit' and 'read books' happen simultaneously and 'be home' is the state in which the reading of books happens. These are not required complements so the *compound* relation is used.

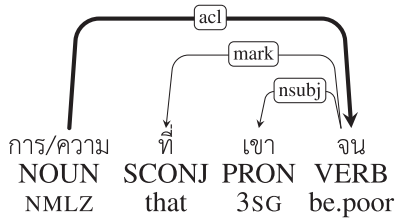
3.3 Nominalizers

The nominalizers การ and ความ are analyzed as head nouns to whom subsequent verbs are attached using the *acl* relation. They can be followed by VP as in (6) or by clauses as in (7).

[4]In UD, *compound:svc* can be used to distinguish serial verbs from compound words. However, the current version of TUD does not contain relation subtypes (see Section 4).



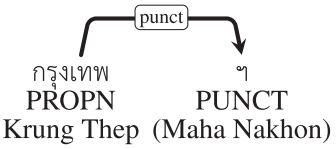
(6) 'running | ignorance'



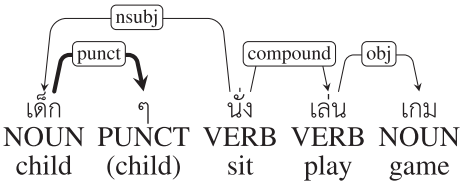
(7) 'the fact that s/he is poor'

3.4 Thai Punctuation Marks

In our treebank, ฯ (abbreviation mark) and ๆ (iteration mark) are labeled as having the PUNCT part-of-speech and connect to their heads using the *punct* relation. The head of ฯ is the same as the head of the whole abbreviated expression, as in (8), and the head of ๆ is the same as the head of the repeated expression, as in (9).



(8) 'Bangkok'



(9) 'Children sit and play games.'

3.5 Thai ''Adjectives''

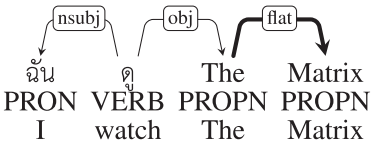
In Thai linguistics, ''adjectives'' are often analyzed as ''adjectival verbs'' rather than being its own category. This is due to the fact that so-called adjectives in Thai have almost identical syntactic distribution to normal verbs (Prasithrathsint, 2000).

However, in our treebank, we decided to label adjectival verbs as ADJ rather than VERB because it is the most versatile—that is, end users can collapse ADJ and VERB into VERB if a different

analysis is desired whereas the reverse cannot easily be done.

3.6 Foreign Multi-word Names

For foreign multi-word names that have been borrowed into Thai, we follow the ''borrowed analysis'' as described in the UD guidelines.[5] This choice is motivated by the perception of native Thai speakers and the low prevalence of genuine code-switching in Thai society. All tokens in a foreign multi-word name are labeled using identical POS, which is the same as the POS of the whole expression, and all tokens are connected to the first token in the name using the *flat* relation as in Tree (10).



(10) 'I watch The Matrix.'

3.7 Parataxis

In UD, the parataxis relation is used when elements are ''placed side by side without any explicit coordination, subordination, or argument relation with the head word.''[6] However, because the Thai writing system does not mark sentence boundaries, every syntactically complete element in the same paragraph is always ''side by side,'' making most parataxis textually indistinguishable from the act of saying the elements separately. Rather than labeling them all as parataxis, which is not very useful, we only use the parataxis relation in the following two cases:

1. When an element clearly intervenes inside another sentence. For example, เขา *จำไม่ได้-แล้วว่าใคร* เรียกผม 'He - *can't remember who he was* - called me.'

2. When there is a pair of parentheses surrounding an element, which marks the element as being related to another element. For example, เขาเรียกผม (จำไม่ได้แล้วว่าเมื่อไหร่) 'He called me (can't remember when that was).'

Note that these two cases are not mutually exclusive. An intervening element itself being surrounded by a pair of parentheses is also common.

3.8 Differences from Thai-PUD

Despite both being based on the UD framework, some differences between the annotation of Thai-PUD and our TUD are worth noting. Some appear to be due to mistakes in Thai-PUD, namely, labeling relative pronouns, such as ที่, ซึ่ง, อัน, all meaning 'which,' as DET instead of PRON, labeling subordinating conjunctions, such as ว่า 'that', ถ้า 'if', หาก 'if', เพื่อให้ 'so that', as ADP instead of SCONJ, and labeling serial verbs using the *acl* relation instead of *compound*. Others are due to difference in analysis. Thai-PUD analyzes ๆ as a symbol (SYM) and the words การ and ความ in (6) as gerund-forming prefixes. Therefore, การ and ความ are combined with subsequent verbs into single tokens and labeled as verbs. We decided not to follow this analysis since การ and ความ can be separated from the verbs as shown in (6) and (7).

4 Thai Universal Dependency Treebank Creation Process

Our text sources cover a wider range of document types and more diverse domains than Thai-PUD. The raw text for TUD was taken from two sources: the Thai National Corpus (Aroonmanakun et al., 2009) and the November 2020 dump of Thai Wikipedia. We randomly sampled 5,000 paragraphs from selected documents while making sure the documents were from a wide range of document types, including news articles, Wikipedia articles, essays, advertisement, interviews, and stories. The selected documents covered diverse topics, such as politics, crime, entertainment, sport, history, religion, culture, and science. Then each paragraph was automatically tokenized using the dictionary-based (newmm) tokenizer from the library PyThaiNLP (Phatthiyaphaibun et al., 2016).

Annotation was done on Datasaur's labeling platform.[7] We initially recruited a total of 10 annotators, all of whom either completed at least a bachelor's degree education in linguistics or were in the process of getting one, and one annotation manager/instructor specialized in Thai

[5] https://universaldependencies.org/foreign.html.

[6] https://universaldependencies.org/u/dep/parataxis.html.

[7] https://datasaur.ai/.

linguistics. Each annotator had four tasks: 1) correct tokenization errors resulting from the preprocessing step, 2) label each token's Universal POS (UPOS), 3) identify dependency arcs, and 4) label each arc's Universal Dependency relation (DEPREL) without a subtype. The LEMMA field of CoNLL-U format was not annotated because the Thai language does not have inflectional morphology.

To train the annotators, they were first instructed to study the manual and review the concepts of head-dependent relationship, syntactic distribution, and polyfunctional words. The instructor also provided some parsed examples from the real data. After that, the annotators were assigned to annotate a set of pilot sentences specifically designed to test their understanding. Once pilot sentences were completely annotated, the instructor would identify common errors in the pilot annotation and re-emphasize what the correct annotation should be. However, after a manual quality review of each annotator's work, only two were identified as producing the highest-quality annotations. Consequently, we chose these two annotators to complete the remainder of the dataset. To assess annotation consistency after training, we randomly sampled 20 sentences, containing 399 tokens, from the treebank and had them fully annotated separately by the two annotators. For UPOS and dependency relations, the Cohen's Kappa scores (Cohen, 1960) were 0.92 and 0.84, respectively. For dependency arcs, we could not use Cohen's Kappa since the arcs can be arbitrarily long. Instead, we calculated the ''UAS'' and ''LAS'' scores by treating one annotator as a ''gold-standard'' annotation. Our UAS was 0.85 and our LAS was 0.78. We considered these four scores to be reasonably high given the complex nature of dependency annotation. Finally, the annotators went on to annotate real sentences prepared in the previous step. During annotation, any perceived errors or ambiguities not covered by the annotation manual were recorded and adjudicated on a case-by-case basis.

After all paragraphs had been annotated, the data was exported into CoNLL-U format and then automatically split into separate trees based on the annotated dependency links, i.e., all connected tokens were grouped as individual trees. Any resulting trees that either 1) contained only a single token, 2) were not completely labeled, or 3) were not valid dependency trees such as trees that contained loops or more than a single root were corrected. A final layer of quality control was done by randomly sampling 50 trees from the treebank and having their annotation errors identified by the instructor. The identified errors were then targeted for manual correction throughout the treebank by searching for the involved tokens and/or labels. The final treebank consists of 3,627 valid and completely labeled trees.

UPOS	Train	Dev	Test	UPOS	Train	Dev	Test
NOUN	18777	2270	2310	CCONJ	2063	239	270
VERB	14881	1802	1867	ADJ	1575	223	197
ADP	4517	530	560	PART	1366	156	169
ADV	4498	557	521	NUM	1161	165	118
AUX	3424	401	421	DET	1140	137	144
PRON	2796	322	350	PUNCT	871	104	125
SCONJ	2438	321	335	SYM	16	1	1
PROPN	2488	293	295				

Table 1: UPOS distribution in each split of TUD.

DEPREL	Train	Dev	Test	DEPREL	Train	Dev	Test
nmod	6268	781	810	punct	865	104	122
obj	5474	655	663	cop	834	83	94
advmod	5366	692	644	flat	709	72	88
compound	5272	656	666	clf	539	79	74
nsubj	4529	548	568	fixed	442	69	60
acl	4539	485	563	xcomp	349	43	46
case	4442	522	548	list	348	29	21
root	2902	362	363	dep	74	5	7
obl	2811	328	322	discourse	67	10	8
mark	2720	326	360	dislocated	64	11	8
aux	2548	311	319	orphan	71	3	6
conj	1992	208	249	csubj	59	11	9
cc	1898	221	254	appos	63	5	9
advcl	1784	225	211	iobj	50	5	6
amod	1449	215	168	parataxis	27	2	3
ccomp	1304	168	174	expl	14	2	2
det	1117	135	145	vocative	1	1	1
nummod	1020	149	92				

Table 2: DEPREL distribution in each split of TUD.

We randomly split TUD into three splits, train, dev, and test, using an 8:1:1 ratio. The most common UPOS tags by far are NOUN and VERB, representing main content words, while the least common tag is SYM (Table 1). All SYM tokens are mathematical symbols, such as the percent (%) and addition (+) signs. DEPREL distribution is generally smooth with a sharp drop in frequency for the last 10 relations (Table 2). Vocatives are the most rare, with only one instance present in each split. Most of the dependencies are local. The average distance between a head and its dependent is 2.48 (Figure 1). Most heads are on the left of the dependent, consistent with the fact that the Thai language is generally head-initial. The dependency arcs are overwhelmingly projective, with only 0.07% being non-projective. Consequently, only 1.5% of the trees contain at least one non-projective arc.

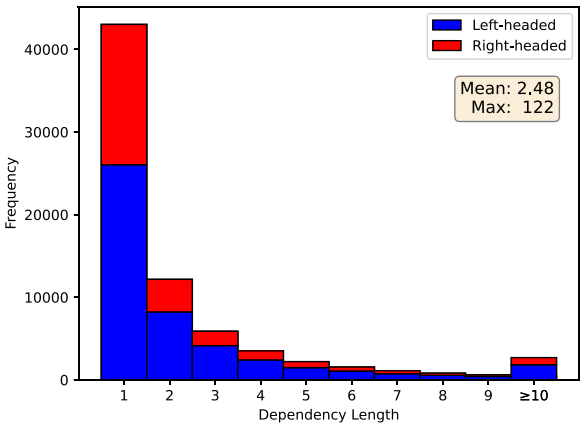


Figure 1: Dependency length distribution in TUD.

5 Thai Dependency Parsing Models

The models to be evaluated in our experiment can be categorized into two types: 1) baseline models and 2) strong open-source models. We used the baseline models to identify the effects of individual components and methods on Thai dependency parsing performance. We tested the strong open-source models to test the quality of our dataset and find out the accuracy rates that could be expected. All models used the gold-standard tokenization that comes with the treebank.

5.1 Baseline Models

Feature extractors convert natural language inputs into numeric representations. Each token in an input sentence is mapped to a set of embeddings that are then concatenated to create the final representation. The possible embeddings are:

1. *Word embedding.* It is obtained from a state-of-the-art pretrained Thai transformer, which is either WangchanBERTa (Lowphansirikul et al., 2021) or PhayaThaiBERT (Sriwirote et al., 2023). WangchanBERTa is based on the RoBERTa architecture (Liu et al., 2019) and is pretrained on a large Thai corpus. PhayaThaiBERT improves upon WangchanBERTa by adding new vocabulary and further pretrained it on a larger corpus.

2. *UPOS embedding.* An embedding is assigned for each possible UPOS tag. To test the importance of UPOS tag quality, the model received either gold-standard tag, automatic tag, or no tag at all (and hence no UPOS embedding).

3. *Sentence embedding.* Each token's embedding is optionally augmented with a sentence embedding shared between all tokens of the same sentence, following Altıntaş and Tantuğ (2023). The embedding for the special token <s> is taken as the sentence embedding.

4. *Super-token embedding.* Also following Altıntaş and Tantuğ (2023), each token's embedding is optionally augmented with ''super-token embeddings'' that represent groups of 2–5 surrounding tokens. These embeddings are obtained by passing convolutional neural network (CNN) filters of varying sizes through the sequence of embeddings obtained prior.

Parsers predict a sentence's dependency structure given a sequence of its token embeddings. They can be classified into two parsing methods:

1. *Graph-based parsers.* We implemented the deep biaffine attention architecture proposed by Dozat and Manning (2017). Our architecture is identical to the one proposed in the original paper, but we replace the BiLSTM encoders with the feature vectors described above. A root-constrained algorithm proposed by Zmigrod et al. (2020) is used to extract maximum arborescence from the adjacency matrices produced by the parsers.

2. *Transition-based parsers.* They can be further classified based on the transition system they use. We tested arc-standard and arc-eager systems (Nivre, 2008). The oracles in both cases are simple feedforward neural networks that greedily predict an appropriate transition based on the features of three tokens. The first two are determined by the transition system while the third is the one that comes immediately after the first two tokens.

The hyperparameters for baseline models (Figure 2) are shown in the Appendix (Table 8).

5.2 Strong Open-source Models

We trained UDPipe 2.1.0 (Straka, 2018),[8] Stanza 1.8.2 (Qi et al., 2020),[9] and Trankit 1.1.1 (Nguyen et al., 2021).[10] All of these models are graph-based

[8] https://ufal.mff.cuni.cz/udpipe/2.
[9] https://stanfordnlp.github.io/stanza/.
[10] https://trankit.readthedocs.io/en/latest/index.html.

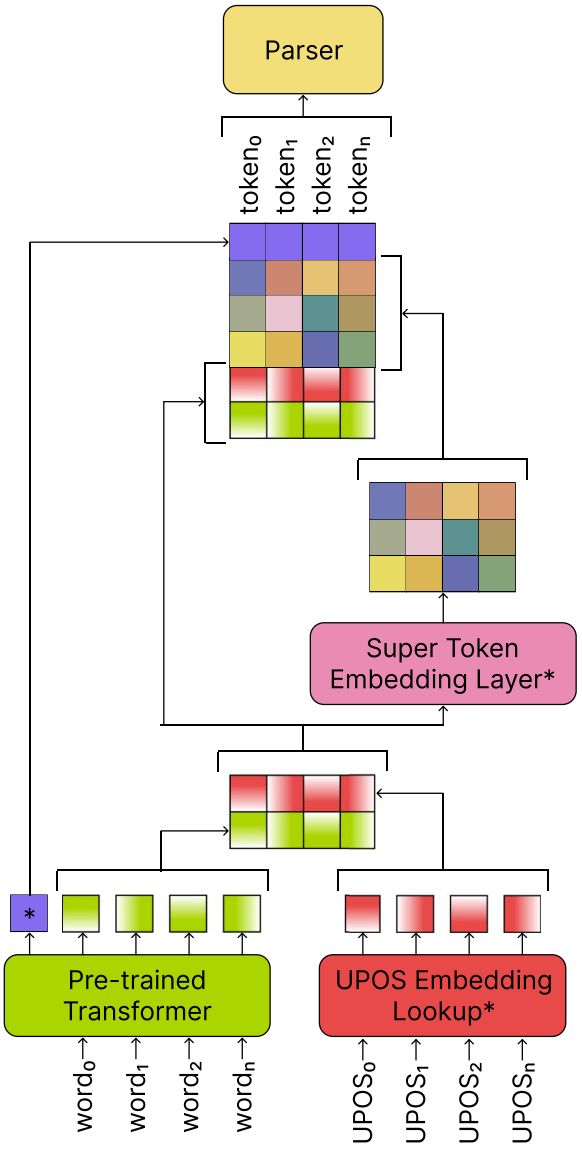


Figure 2: General architecture of our baseline models. Components marked with asterisks (*) were optional, i.e., some models might not incorporate them depending on their design choices.

parsers. These toolkits currently do not support Thai parsing out of the box. Minimal modification to their training scripts was done so that the models took advantage of the state-of-the-art pretrained Thai language model. The default hyperparameters were used. UDPipe and Trankit models use the same intermediate representations to *jointly* predict both UPOS tags and dependency structure. They do not use POS tag at inference time, so they were only trained using gold-standard POS tags and not tested using automatic POS tags. They are also allowed to use XPOS and FEATS tags, which exist only in Thai-PUD.

	Thai-PUD		TUD	
UPOS	**W**	**P**	**W**	**P**
ADJ	0.7978	**0.8508**	0.6486	**0.6852**
ADP	0.9578	**0.9677**	0.9206	**0.9272**
ADV	0.8528	**0.8705**	0.7665	**0.7792**
AUX	0.9565	**0.9710**	0.8483	**0.8508**
CCONJ	0.9434	**0.9636**	0.8675	**0.8813**
DET	0.9469	**0.9596**	0.9007	**0.9122**
NOUN	0.9597	**0.9711**	0.9640	**0.9672**
NUM	**1.0000**	**1.0000**	**0.9391**	0.9264
PART	0.9556	**0.9663**	0.8395	**0.8402**
PRON	0.9552	**0.9925**	0.9330	**0.9418**
PROPN	0.9341	**0.9375**	0.9037	**0.9223**
PUNCT	**1.0000**	**1.0000**	**0.9881**	0.9843
SCONJ	–	–	0.8205	**0.8479**
SYM	**1.0000**	**1.0000**	**1.0000**	**1.0000**
VERB	0.9502	**0.9610**	0.9240	**0.9292**
Macro Average	0.9458	**0.9580**	0.8843	**0.8930**

Table 3: F1 scores of our UPOS taggers on each label and treebank. W stands for WangchanBERTa. P stands for PhayathaiBERT. Note that the SCONJ tag does not exist in Thai-PUD because the ADP tag is used instead for subordinating conjunctions.

5.3 UPOS Tagger

We trained our own POS tagger. We formulated the task as a token classification (sequence tagging) and fine-tuned a pretrained language model to predict POS tags. We experimented with two competitive Thai pretrained language models: WangchanBERTa and PhayaThaiBERT. The hyperparameters are shown in the Appendix (Table 9). Our experiment showed that PhayaThaiBERT yields better UPOS accuracy than WangchanBERTa (Table 3). Therefore, only the UPOS tags predicted by the PhayathaiBERT-based taggers were used to train the parsers that relied on automatically tagged UPOS.

5.4 Benchmarking Treebanks

We experimented with Thai-PUD and our TUD. We did not experiment with Blackboard Treebank due to its lack of dependency relation types. Since Thai-PUD does not have official train-dev-test splits, we randomly split it into three subsets using an 8:1:1 ratio.

6 Results and Discussion

In running these series of experiments, we want to know 1) Which parsing method is better for Thai? 2) Is PhayaThaiBERT a better Thai encoder

Model				Thai-PUD						TUD					
				Gold POS		Auto POS		No POS		Gold POS		Auto POS		No POS	
T vs G	S vs E	∅ vs A	W vs P	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
T	S	∅	W	88.14	80.39	85.28	76.65	85.60	75.45	89.47	82.60	86.27	76.22	86.59	76.81
T	S	A	W	88.83	82.23	88.14	80.20	86.25	76.60	89.82	83.18	86.59	76.52	86.80	76.87
T	E	∅	W	87.40	80.53	88.00	79.60	84.54	75.03	89.20	82.27	86.33	76.53	86.02	76.02
T	E	A	W	88.42	81.91	87.77	80.39	86.39	78.08	89.41	82.62	86.24	76.70	86.37	76.55
T	S	∅	P	89.57	82.33	87.91	79.51	84.73	75.27	90.15	83.57	87.05	77.60	87.19	77.64
T	S	A	P	89.43	83.48	88.28	80.94	85.65	76.70	90.04	83.74	**87.26**	77.55	87.09	77.68
T	E	∅	P	89.11	82.60	88.92	80.48	86.48	78.17	89.93	83.42	86.82	77.09	86.54	77.07
T	E	A	P	89.39	83.76	88.37	81.17	87.45	79.51	89.77	83.42	87.00	**77.68**	86.76	77.61
G	–	∅	W	85.97	80.43	83.43	76.60	84.36	77.34	86.33	79.64	84.25	74.59	84.77	74.41
G	–	A	W	87.82	82.69	86.29	79.79	83.80	76.14	87.99	81.01	81.44	71.50	85.62	75.53
G	–	∅	P	89.29	84.82	88.42	82.19	87.91	81.68	88.75	82.25	85.73	76.12	86.40	76.56
G	–	A	P	89.80	84.91	88.65	82.60	88.74	82.05	89.48	82.98	86.03	76.40	85.84	76.14
	UDPipe*		W	88.92	83.06	–	–	–	–	86.06	77.01	–	–	–	–
	UDPipe*		P	89.89	83.53	–	–	–	–	86.67	77.78	–	–	–	–
	Stanza*		W	91.37	86.16	89.85	83.34	89.29	83.06	90.12	83.30	86.31	76.60	87.01	77.39
	Stanza*		P	**92.02**	**87.22**	**90.54**	**84.54**	**90.72**	**84.73**	**90.90**	**84.54**	86.93	77.51	**87.39**	**78.09**
	Trankit*		W	89.62	84.08	–	–	–	–	86.22	76.19	–	–	–	–
	Trankit*		P	91.28	86.11	–	–	–	–	86.71	77.01	–	–	–	–

Table 4: Evaluation results of each model on each treebank's test split. T = Transition-based, G = Graph-based, S = Arc-standard, E = Arc-eager, A = Augmented with sentence and super token embeddings, W = WangchanBERTa, P = PhayaThaiBERT. *Open-source models are all graph-based.

	UAS		LAS	
Factor	Coefficient	*p*-value	Coefficient	*p*-value
(Intercept)	84.7953	<0.001***	76.0953	<0.001***
Model Category: Open-source Models	2.0221	<0.001***	1.7424	0.035*
Architecture: Transition-Standard	1.0420	0.011*	0.0563	0.937
Architecture: Transition-Eager	1.0622	0.010*	0.4103	0.566
Encoder: PhayaThaiBERT	1.2665	<0.001***	1.6927	0.001**
Augmented: Yes	0.4487	0.174	0.7599	0.195
UPOS Quality: Gold	2.2607	<0.001***	4.5311	<0.001***
UPOS Quality: Auto	0.4217	0.259	0.8011	0.227

Table 5: Linear regression results for UAS ($R^2 = 0.54$) and LAS ($R^2 = 0.507$). The reference categories are baseline models, graph-based architecture, WangchanBERTa as encoder, non-augmented, and agnostic UPOS.

than WangchanBERTa? 3) Do sentence embeddings and super-token embeddings help? and 4) Do gold-standard POS tags play an important role? From the experimental results shown in Table 4, we fit a linear regression model where the experimental conditions are predictors and the target variable is UAS. Then we ran statistical tests on the regression coefficients to obtain *p*-values. We repeated this analysis with LAS as a target variable (Table 5).

Which Parsing Architecture Is Better for Thai? Our results suggest that transition-based models perform significantly better than graph-based models in UAS ($p < 0.05$) but perform similarly in LAS ($p = 0.937$). The overall *best* model is Stanza, although it is a graph-based model. This is because Stanza also employs additional techniques not present in other models, namely, 1) augmenting each token's representation with fastText's static pretrained word embeddings (Grave et al., 2018), and 2) including terms that explicitly model the probability of each link between a head and a dependent based on their distance and linear order (Qi et al., 2018). Nevertheless, note that many of our transition-based models achieve very competitive results with Stanza. In future work, it would be interesting to see if transition-based models could outperform it once additional techniques are implemented.

	Thai-PUD				TUD			
Rank	**UPOS Confusion**	**Tokens**	**DEPREL Confusion**	**Tokens**	**UPOS Confusion**	**Tokens**	**DEPREL Confusion**	**Tokens**
1	NOUN-PROPN	ดิสแพตช์, ไลน์	compound-flat:name	ที่, เซนต์	ADV-VERB	มา, ไป	compound-nmod	ประเทศ, สาว
2	ADJ-VERB	เฉลี่ย, ใกล้ชิด	acl-xcomp	ดู, ใช้	ADV-AUX	ได้, อยู่	nmod-obl	การ, ประเทศ
3	ADJ-ADV	ใหม่, น้อย	nmod-obl	ปี, ทะเล	AUX-VERB	เป็น, ได้	advmod-compound	มา, ไป
4	ADP-ADV	กว่า, จึง	nsubj-obj	ที่, ซึ่ง	ADJ-VERB	ดี, ร้าย	advmod-aux	ได้, แล้ว
5	AUX-VERB	เป็น, ได้	obj-obl	กัน, จรรยาบรรณ	NOUN-PROPN	เมทริกซ์, มะกัน	acl-compound	พนัน, เสพ
6	ADJ-NOUN	ปัจจุบัน, หนุ่ม	compound-obj	ประภาคาร, พื้นฐาน	NOUN-VERB	คมนาคม, พนัน	nsubj-obj	ที่
7	ADV-VERB	พร้อม, สมบูรณ์	nsubj-obl:tmod	ที่, อัน	CCONJ-SCONJ	โดย, ซึ่ง	clf-nmod	คน, แบบ
8	ADP-NOUN	เชิง	advcl-root	ก่อ, แหล่ง	SCONJ-VERB	ให้	compound-obj	การ, ชีวิต
9	NOUN-VERB	ประดิษฐ์, โชว์	appos-flat:name	ไมเคิล, ปีเตอร์	ADJ-ADV	มาก, น้อย	obj-obl	ที่, ความ
10	ADP-VERB	ตั้ง, ต่อ	clf-compound	กลุ่ม, เฮกตาร์	ADP-VERB	ถึง, ให้	ccomp-compound	เรียน, เชื่อม

Table 6: The 10 most common confusions made by the taggers for UPOS and the parsers for DEPREL along with some of their most frequently associated tokens.

Is PhayaThaiBERT a Better Thai Encoder than WangchanBERTa? Our results show that PhayaThaiBERT significantly leads to better performance than WangchanBERTa in dependency parsing ($p < 0.05$), supporting the claim in its paper (Sriwirote et al., 2023) that PhayaThaiBERT is an improved version of WangchanBERTa. Our models outperformed Thai dependency parsers in previous works, which do not use these pretrained language models (Singkul and Woraratpanya, 2019; Yasuoka, 2023; Straka, 2018; Kondratyuk and Straka, 2019), highlighting the importance of large, contextualized, language-specific, pretrained encoders.

Do Sentence Embeddings and Super-token Embeddings Help? Contrary to Altıntaş and Tantuğ (2023), we found that augmenting token embeddings with sentence embeddings and super-token embeddings does not result in significant improvement (UAS $p = 0.174$; LAS $p = 0.195$). Nevertheless, many instances of improvement over non-augmented models are observed even in transition-based models, which are not experimented with in the original work. This suggests that the method itself has some potential, although it does not improve all the models consistently enough to be statistically significant. More experimentation is therefore needed to find the conditions in which the method works best.

Do Gold-standard UPOS Tags Play an Important Role? Gold-standard UPOS tags clearly lead to significantly superior performance ($p < 0.05$). The same cannot be said for automatically tagged UPOS, however, as using them does not result in significant improvement over not using them at all (UAS $p = 0.259$; LAS $p = 0.227$), and in some instances, it even leads to slight degradation (Table 4). This suggests that while our UPOS taggers were reasonably accurate (Table 3), the tags they correctly predicted were not actually crucial to dependency parsing. Therefore, we need to improve the POS tagger to improve the Thai parsing results.

7 Error Analysis

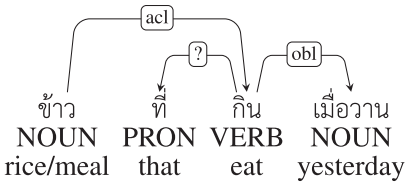
In this section, we will identify some challenges unique to Thai dependency parsing, which should give us valuable insights that could help close the gap in performance between Thai and high-resource languages. The full test split prediction of every model is available on TUD's GitHub repository for later, more extensive investigation.

The Vast Majority of UPOS Confusions Were Caused by Polyfunctional Words. Polyfunctional words are words that can function in more than one role. The most common mistakes in this category were caused by the fact that Thai has a large number of deverbal grammaticalizations. Among the most versatile, and thus the most confused (Table 6), are the verbs เป็น, ได้, อยู่, ไป, มา, ถึง, ให้, which can also function as auxiliary verbs (เป็น, ได้, อยู่), adverbs (ไป, มา, ได้, อยู่, ให้), subordinating conjunctions (ให้), and adpositions (ถึง, ให้) depending on context, especially word order.[11] What makes it particularly hard in Thai is the fact that no morphological clues are available since Thai words are never inflected. Another noteworthy UPOS confusion was the confusion between adjectives and verbs, which reflects a well known feature of Thai syntax, in which the two classes have almost identical syntactic distribution (Prasithrathsint, 2000).

[11] Due to space constraints, please refer to Smyth (2002) for a more complete discussion of the many functions of ให้, ได้ and เป็น.

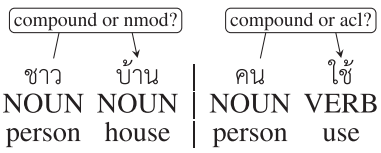
Distinction Between Common Nouns and Proper Nouns in Thai is Unclear. In many languages, proper nouns are capitalized and cannot be accompanied by articles. These features do not exist in Thai so proper nouns have to be identified either from context or from world knowledge, causing confusion between the NOUN and PROPN tags. This tends to happen with foreign proper names, such as ดิสแพตช์ 'Dispatch' and เมทริกซ์ 'Matrix' (Table 6), likely because the encoders are pretrained on mainly Thai text.

The Roles of Thai Relative Pronouns are Difficult to Determine Without World Knowledge. In Thai, like in many languages, the relative pronouns ที่, ซึ่ง, อัน ('that/which') are fronted in relative clauses. This fronting eliminates word order as a clue for identifying their syntactic roles. Combining with the pro-drop nature of Thai and the absence of subject-verb agreement, even less information is available. Tree (11) illustrates this ambiguity. Only by using the world knowledge that rice/meal is inanimate and thus cannot eat anything can we determine that the correct annotation for ? is *obj* and not *nsubj*.



(11) 'rice/meal that (someone) ate yesterday'

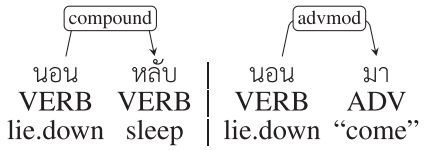
Parsers Cannot Easily Distinguish Between Compounds and Syntactic Phrases. Determining whether a sequence of words constitutes a compound and not a phrase in Thai relies on multiple factors and the two classes can be described to exist on a spectrum (Hongthong et al., 2019). This distinction is notoriously difficult because the difference is quite subtle (Kriengket et al., 2007). Tree (12) shows some compounds that were mistaken as syntactic phrases by the parsers.



(12) 'villager | servant'

ชาวบ้าน means a villager (compound noun) and not a person who stays in a house (noun phrase). คนใช้ means a servant (compound noun) and not a user (noun phrase). The correct dependencies depend on the meaning of these phrases in a larger context of a sentence.

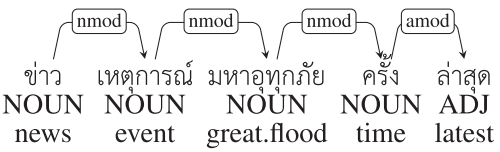
Verbal and Verbal-like Dependents Were Often Confused with One Another. Without explicit linking words, structures involving verbal dependents (*xcomp*, *compound*, *ccomp*, *advcl*) and verbal-like dependents (*advmod*, *aux*), which frequently share word forms with verbs, can all appear like sequences of two verb phrases, which leads to ambiguity. This confusion was also exhibited by our annotators during pilot annotation since distinguishing between them often requires performing non-trivial linguistic tests. Tree (13) shows some examples.



(13) 'lie down and sleep | have laid down'

Despite looking superficially similar, the sentence on the left describes simultaneous actions (lying down and sleeping as opposed to lying down and doing something else) while the sentence on the right usually does not because the word มา cannot be interpreted as a verb despite having the same form as the word meaning 'to come.' It is rather an adverb indicating that the action has already taken place and has some implication affecting the present. Although in uncommon context, it could still be interpreted as a verb, e.g., when arriving at a place while lying down.

Many Mistakes Were Correlated with ''Chain Dependencies.'' Some relations, most notably *nmod* and *compound* (and *acl* in Thai-PUD), allow two or more tokens of the same UPOS to be chained together into one structure. This creates ambiguity when subsequent dependents need to be attached to one of the tokens in the chain. To illustrate, Tree (14) shows part of a real tree from TUD.



(14) 'news about the latest great flood event'

Thai-PUD		TUD	
DEPREL	**Count**	**DEPREL**	**Count**
N/A	1586	N/A	4533
compound	1280	compound	4386
acl	963	nmod	3925
obj	820	obj	2878
obl	600	acl	2490
xcomp	562	ccomp	2035
advmod	404	advcl	1995
conj	403	advmod	1944
nmod:poss	261	conj	1731
nmod	259	acl, obj	1260

Table 7: The 10 most common DEPRELs with which the true heads and the incorrectly predicted heads are linked in gold annotation. N/A means that the predicted and true heads are not on the same dependency chain. The numbers shown are aggregated from all sets of prediction.

The text that follows the above structure is a relative clause ที่คร่าชีวิตผู้คน 'which takes people's lives,' which can potentially be attached to any of the four nouns in the chain. Lacking world knowledge that flood kills people, many parsers attached the relative clause to ข่าว 'news' or เหตุการณ์ 'event' instead of มหาอุทกภัย 'great flood.'

We perform further analysis by investigating the relative position of the true heads and the heads incorrectly predicted by the parsers and discover that the two tokens often form structures involving the aforementioned relations (Table 7). Note that the vast majority of the mistakes involve only one DEPREL between the true heads and the predicted heads, indicating that the parsers frequently missed the correct heads by one arc when these relations are around the true heads, analogous to the ambiguity in Tree (14).

8 Conclusion

In this work, we introduce the Thai Universal Dependency Treebank (TUD), a new Thai treebank consisting of 3,627 trees annotated according to the UD framework. We found that strong open-source graph-based parsing system with the state-of-the-art PLM achieves the best results on our dataset. We also found that an accurate POS tagger is crucial to good parsing results, but the current automatic POS tagger is not accurate enough to improve parsing performance. Our error analysis shows that polyfunctional words and ambiguities created by the lack of inflectional morphology present the main challenges for dependency parsing in Thai.

Acknowledgments

The authors thank the reviewers and the action editor for their helpful comments, which contribute to a significant improvement of this work. This work is supported by the National Research Foundation, Singapore under its AI Singapore Programme. This research has also received funding support from the NSRF via the Program Management Unit for Human Resources & Institutional Development, Research, and Innovation [grant number B0SF640234].

References

Mücahit Altıntaş and A. Cüneyd Tantuğ. 2023. Improving the performance of graph based dependency parsing by guiding bi-affine layer with augmented global and local features. *Intelligent Systems with Applications*, 18:200190. https://doi.org/10.1016/j.iswa.2023.200190

Wirote Aroonmanakun, Kachen Tansiri, and Pairit Nittayanuparp. 2009. Thai National Corpus: A progress report. In *Proceedings of the 7th Workshop on Asian Language Resources*, ALR7, pages 153–158, USA. Association for Computational Linguistics. https://doi.org/10.3115/1690299.1690321

Ratchakrit Arreerard, Stephen Mander, and Scott Piao. 2022. Survey on Thai NLP language resources and tools. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6495–6505, Marseille, France. European Language Resources Association.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46. https://doi.org/10.1177/001316446002000104

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308. https://doi.org/10.1162/coli_a_00402

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training

of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. https://doi.org/10.48550/arXiv.1611.01734

Chloe Eggleston and Brendan O'Connor. 2022. Cross-dialect social media dependency parsing for social scientific entity attribute analysis. In *Proceedings of the Eighth Workshop on Noisy User-generated Text (W-NUT 2022)*, pages 38–50, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 710–716, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1076

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. https://doi.org/10.48550/arXiv.1802.06893

Kamolchanok Hongthong, Kingkarn Thepkanjana, and Wirote Aroonmanakun. 2019. Is there a dichotomy between synthetic compounds and phrases in Thai? *Taiwan Journal of Linguistics*, 17(1):49–77. https://doi.org/10.6519/TJL.201901_17(1).0002

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1237

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1279

Kanyanut Kriengket, Krit Kosawat, and Sunant Anchaleenukul. 2007. A computational linguistics study of compound nouns in Thai. In *Proceedings of the Seventh International Symposium on Natural Language Processing (SNLP 2007)*, pages 31–36.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. https://doi.org/10.48550/arXiv.1907.11692

Lalita Lowphansirikul, Charin Polpanumas, Nawat Jantrakulchai, and Sarana Nutanong. 2021. WangchanBERTa: Pretraining transformer-based Thai language models. https://doi.org/10.48550/arXiv.2101.09635

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1130

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: A tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.645

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for*

Computational Linguistics: Human Language Technologies, pages 1626–1635, Portland, Oregon, USA. Association for Computational Linguistics.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131, Prague, Czech Republic. Association for Computational Linguistics.

Alireza Mohammadshahi and James Henderson. 2021. Recursive non-autoregressive graph-to-graph transformer for dependency parsing with iterative refinement. *Transactions of the Association for Computational Linguistics*, 9:120–138. https://doi.org/10.1162/tacl_a_00358

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.65

Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. 2021. Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 80–90, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-demos.10

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553. https://doi.org/10.1162/coli.07-056-R1-07-027

Wannaphong Phatthiyaphaibun, Korakot Chaovavanich, Charin Polpanumas, Arthit Suriyawongkul, Lalita Lowphansirikul, and Pattarawat Chormai. 2016. PyThaiNLP: Thai natural language processing in Python. https://doi.org/10.5281/zenodo.3519354

Amara Prasithrathsint. 2000. Adjectives as verbs in Thai. *Linguistic Typology*, 4(2):251–272. https://doi.org/10.1515/lity.2000.4.2.251

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal Dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/K18-2016

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. https://doi.org/10.48550/arXiv.2003.07082

Sattaya Singkul and Kuntpong Woraratpanya. 2019. Thai dependency parsing with character embedding. In *2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–5. https://doi.org/10.1109/ICITEED.2019.8930002

David Smyth. 2002. *Thai: An Essential Grammar*, 1st edition. Routledge.

Panyut Sriwirote, Jalinee Thapiang, Vasan Timtong, and Attapol T. Rutherford. 2023. PhayaThaiBERT: Enhancing a pretrained Thai language model with unassimilated loanwords. https://doi.org/10.48550/arXiv.2311.12475

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Milan Straka, Jana Straková, and Jan Hajič. 2019. Evaluating contextualized embeddings on 54 languages in pos tagging, lemmatization and dependency parsing. https://doi.org/10.48550/arXiv.1908.07448

Kiyoko Takahashi. 2009. Basic serial verb constructions in Thai. *Journal of the Southeast Asian Linguistics Society*, 1:215–229.

Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. 2021. Dependency-driven relation extraction with attentive graph convolutional networks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4458–4471. https://doi.org/10.18653/v1/2021.acl-long.344

Koichi Yasuoka. 2023. Sequence-labeling RoBERTa model for dependency-parsing in Classical Chinese and its application to Vietnamese and Thai. In *2023 8th International Conference on Business and Industrial Research (ICBIR)*, pages 169–173. https://doi.org/10.1109/ICBIR57571.2023.10147628

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/K18-2001

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/K17-3001

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.302

Yue Zhang, Bo Zhang, Rui Wang, Junjie Cao, Chen Li, and Zuyi Bao. 2021. Entity relation extraction as dependency parsing in visually rich documents. *arXiv preprint arXiv:2110.09915*. https://doi.org/10.18653/v1/2021.emnlp-main.218

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1230

Ran Zmigrod, Tim Vieira, and Ryan Cotterell. 2020. Please mind the root: Decoding arborescences for dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4809–4819, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.390

A Appendix

Hyperparameter	Value
No. of parser's hidden layers*	1
Parser's hidden dim.*	768
UPOS embedding dim.	768
Super-token filter sizes	2, 3, 4, 5
Super-token embedding dim.	192 each
Dropout	0.1
Learning rate scheduler	Linear
Warmup ratio	0.1
Peak learning rate	3e-5
Weight decay	0.01
Adam ϵ	1e-8
Adam β_1	0.9
Adam β_2	0.999
Batch size	8
No. of training epochs	10

Table 8: Hyperparameters used in the training of our models. *For transition-based parsers, this refers to the layers inside the oracles. For graph-based parsers, it refers to the layers inside the feedforward neural networks that create separate head and dependent representations for each token.

Hyperparameter	Value
Dropout	0.1
Learning rate scheduler	Linear
Warmup ratio	0.1
Peak learning rate	3e-5
Weight decay	0.01
Adam ϵ	1e-8
Adam β_1	0.9
Adam β_2	0.999
Batch size	32
No. of training epochs	20

Table 9: Hyperparameters used in the training of our UPOS taggers.