Building a Treebank for Vietnamese Syntactic Parsing

NGUYEN Thi Quy

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

Building a Treebank for Vietnamese Syntactic Parsing

Author:
NGUYEN Thi Quy

Supervisor:
Assoc. Prof. Yusuke MIYAO

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

June 2017

**A dissertation submitted to Department of Informatics,
School of Multidisciplinary Sciences,
SOKENDAI (The Graduate University for Advanced Studies),
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy**

Advisory Committee

1. Assoc.Prof. Yusuke MIYAO	National Institute of Informatics, SOKENDAI
2. Prof. Ken SATOH	National Institute of Informatics, SOKENDAI
3. Assoc.Prof. Jun'ichi YAMAGISHI	National Institute of Informatics, SOKENDAI
4. Prof. Akiko AIZAWA	National Institute of Informatics
5. Assoc.Prof. Masayuki ASAHARA	National Institute for Japanese Language and Linguistics

Acknowledgements

Foremost, I want to express my gratitude and thanks to my supervisor, Prof. Yusuke Miyao. He always gives me valuable advice and encouragement when I have trouble in research, writing paper, preparing slides, and so on. This thesis would not been completed without his help. It is my honor to be his student.

I would like to thank my advisory committee members: Prof. Ken Satoh, Prof. Akiko Aizawa, Prof. Junichi Yamagishi, and Prof. Masayuki Asahara for spending time to help and give detailed comments on my presentations. Their comments are valuable sources so that I can improve my research and complete my thesis.

I also want to thank my friends and colleagues at National Institute of Informatics (NII), Japan for their support and advice.

SOKENDAI (THE GRADUATE UNIVERSITY FOR ADVANCED STUDIES)

Abstract

School of Multidisciplinary Sciences
Department of Informatics

Doctor of Philosophy

Building a Treebank for Vietnamese Syntactic Parsing

by NGUYEN Thi Quy

Treebanks, corpora annotated with syntactic structures, are important resources for researchers in natural language processing, linguistic theory, as well as speech processing. They provide training and testing materials so that different algorithms can be compared. However, it is not a trivial task to construct high-quality treebanks. We have not yet had a proper treebank for such a low-resource language as Vietnamese, which has probably lowered the performance of Vietnamese language processing. In order to alleviate such a situation, this thesis has tackled with two objectives, viz., (1) developing a consistent and accurate Vietnamese treebank and (2) applying our treebank to parsing, a crucial problem in improving the quality of speech processing and natural language processing applications. This study is not only beneficial for the development of computational processing technologies for Vietnamese, a language spoken by over 90 million people, but also for similar languages such as Thai, Laos, and so on.

For the first objective, we propose an annotation scheme for the Vietnamese treebank. In comparison with the previous one (VLSP treebank's scheme), our scheme is better because it can cover and distinguish among various constructions and linguistic phenomena in Vietnamese. We also develop three sets of guidelines corresponding to three annotation layers of our treebank, including: word segmentation guidelines (44 pages), part-of-speech (POS) tagging guidelines (73 pages), and bracketing guidelines (182 pages). Our guidelines contain rules to address the challenges of Vietnamese language. Specifically,we hand-crafted 9 rules for segmenting ambiguous expressions, 34 rules for tagging ambiguous words, and 39 rules for bracketing ambiguous expressions. These guidelines, which are used to train the annotators, are valuable resources that serve the use of the treebank.

In addition to developing the annotation guidelines, we describe other issues of ensuring the annotation quality including an appropriate annotation process, a well-designed process of training annotators, and software tools to support the annotation as well as to control the quality. Inter-annotator agreement, intra-annotator agreement, and accuracy of the developed treebank are higher than 90%, which shows that the annotated treebank is reliable and satisfactory. In comparison with the VLSP treebank, our annotation scheme is more fine-grained than the one of the VLSP treebank. For example, our POS tag set includes 33 tags, while there are 17 tags in the VLSP treebank. However, our treebank gives the higher performance in comparison with the VLSP treebank on all of the tasks, namely, automatic word segmentation, POS tagging, and parsing. This indicates that our treebank is more consistent than the VLSP treebank.

For the second objective, we first evaluate representative parsing models on the Vietnamese treebank. We then investigate the errors produced by the parsers and find the reasons for them. Our analysis focuses on four possible sources of the parsing errors, viz., limited training data, word segmentation errors, confusing POS tags, and ambiguous constructions. We use an analysis method that combines the advantages of automatic tools and a manual analysis. While automatic analysis can be applied to a large amount of parsing output, manual investigation can capture the reasons of the parsing errors precisely. As a result, we find that parsing models based on conditional random field (CRF) and neural network are good for Vietnamese. On the other hand, the quality of Vietnamese parsing can also be improved through enriching contextual information, such as using the hierarchical state-splitting for unlexicalized parsing or exploiting the rich input features of the surface spans for CRF parsing. However, these performances (about 72% in F-score) of Vietnamese parsing are still far lower than the performances reported for English (about 90% in F-score) and Chinese (about 86% in F-score). This indicates that existing models cannot capture contextual information like words working as prefixes and suffixes.

The investigation of parsing errors has revealed the frequent errors in Vietnamese parsing that are VP attachment, NP attachment, PP attachment, and clause attachment. In addition, we found that we could not obtain significant improvement of the Vietnamese parsing by simply enlarging the training data. Among the three factors of word segmentation errors, POS tagging errors, and ambiguous constructions, although the first and second ones have significantly contributed to many parsing errors, the third one is the major problem that causes the low performance of Vietnamese parsing. Ambiguous constructions in Vietnamese appear in many forms, such as ambiguous POS sequences or ambiguous symbol sequences. They are caused by the characteristics of Vietnamese such as the lack of inflectional morphemes, post-head modifying lexical words, and dropping words. This research has also shown that although Vietnamese has many confusing constructions, these ambiguities can be tackled based on contextual information such

as the words playing the roles as prefixes and suffixes, function words, fine-grained categorizations, head words of the phrases, main verbs of the clauses, etc.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Treebanks are parsed text corpora that include syntactic analysis of natural language text. They are important resources for researchers in natural language processing (NLP), speech processing, and linguistic research. For example, in NLP, treebanks provide training and testing materials for developing word segmentation, part-of-speech (POS) tagging tools and syntactic parsers. These tools then are applied to improving the quality of NLP applications, such as machine translation [4–6] and question answering [7–9].

Because of its importance, treebanks have been developed for many languages. Most typically, the Penn English Treebank [10] has played a crucial role in the success of English part-of-speech taggers [11, 12] and parsers [13–15]. The methodology and annotation scheme of the Penn English Treebank have been adapted for the Penn Korean Treebank [16], Penn Chinese Treebank [17], French Treebank [18], etc.

To strengthen the automatic processing of the Vietnamese language, a Vietnamese Treebank (VLSP Treebank) was built as a part of a national project, entitled "Vietnamese language and speech processing (VLSP)" [3]. This corpus contains 10,374 sentences in social and political topics, collected from an online daily newspaper, the Youth (Tuổi Trẻ) [3]. The VLSP Treebank was annotated with three layers: word segmentation, part-of-speech (POS) tagging and bracketing[1]. However, Nguyen et al. [19, 20] showed

[1]Bracketing is the annotation of constituent structure.

that the quality of VLSP Treebank, including the quality of the annotation scheme, the annotation guidelines and the annotation process, is not satisfactory and is a possible source for the low performance of Vietnamese language processing [21–23].

To alleviate these issues, in this work, we carried out two main tasks. Firstly, we build a new treebank for Vietnamese. Secondly, we train several state-of-the-art parsers on the new treebank to find the actual problems that cause the low performance of Vietnamese parsing. Regarding the first task, we study the engineering issues for keeping annotation consistency and accuracy while ensuring a reasonable annotation speed, including annotation guidelines, annotation process, annotator, and supporting tools. Besides evaluating our treebank through several measurements such as inter-annotator agreement, intra-annotator agreement, and annotation accuracy, we also compare our treebank with the VLSP treebank on the automatic word segmentation, part of speech tagging, and parsing. For the second task, after evaluating our treebank on different state-of-the-art parsing methods, we investigate all possible problems that can affect the parsing quality including a small size of treebank, ambiguous POS tags, and confusing constructions.

This study is not only beneficial for the development of computational processing technologies for Vietnamese, a language spoken by over 90 million people, but also for similar languages such as Thai, Laos, and so on. This study also promotes the computational linguistics studies on how to transfer methods developed for a popular language, like English, to a language that has not yet intensively studied.

1.2 Treebank annotation scheme

Treebanks are typically annotated with four consecutive layers ordered as follows (1) word segmentation, (2) morphological analysis, (3) POS, and (4) syntactic structure. The upper layer is annotated on top of the previous one, which makes it more complicated. It is clear that the most complicated one is the top layer—syntactic structure. There are two primary types of syntactic annotation, constituency annotation and functional annotation. According to Chomsky [24], grammatical functions can be derived from the constituent structure. Following this viewpoint, several treebanks such as the Lancaster Parsed Corpus [25] and the original Penn Treebank [10] were annotated with constituent structures (phrase structures).

However, since Melčuk [26]—the father of dependency syntax, assumed that the functional structure is more fundamental than the constituent structure, the functional annotation has become increasingly important. Basically, several dependency treebanks have been built for the languages with free word order, such as the Japanese Treebank [27], the Turkish Treebank [28], and the Prague dependency treebank for Czech [29]. Meanwhile, for the languages with fixed word order, the constituent structure has been used but with extended for functional annotation. An example for this extension is the Penn Treebank II [30] (PTB) that originated from the original Penn Treebank [10]. Following the success of the PTB, constituent treebanks for a wide range of languages such as Chinese [17], Korean [16], Arabic [31], and Spanish [32], have been built in which the annotation scheme are the adapted versions of the PTB's scheme. Vietnamese word order is also quite fixed. Therefore, our annotation scheme is also adapted from the scheme of the PTB.

1.3 Challenges of annotating Vietnamese

Difficulties of Vietnamese language have been recognized by many researchers [21, 33–35]. Until now, there are still little consensus in annotating Vietnamese syntax. Several main reasons for such situation are as follows. In comparison with English, Vietnamese does not have word delimiters and inflectional morphemes. While similar problems also occur in Chinese [36], annotating Vietnamese is probably more difficult because the modern Vietnamese writing system is based on Latin characters, which represent the pronunciation but not the meanings of words. As a result, there are many polysemous expressions, i.e., expressions having the same surface form but different interpretations, in Vietnamese. Difficulties in annotating Vietnamese are also caused by word orders. Although Vietnamese is a subject-verb-object (SVO) language like English and Chinese, its word orders are different from the others. For example, in Vietnamese, the word order in noun phrases is exactly the same as those in simple sentences, which leads to ambiguities in labelling these two types of expressions. In addition, other problems such as dropping words, conflicted definitions among linguists, etc. have also caused many challenges for annotating Vietnamese texts.

Figure 1.1 presents several ambiguous expressions in Vietnamese. We can see that expressions in Figures 1.1a, 1.1b and 1.1c have the same POS sequence *Vv Nn Vv* (a verb stands in front of a noun and another verb). However, they should be annotated in different ways. In Figure 1.1a, the noun and the later verb should separately modify the head verb. Meanwhile, the later verb in Figure 1.1b is a modifier of the noun. In Figure 1.1c, the noun and the later verb are the subject and the predicate of a simple sentence that modifies the head verb $mong_{to\ hope}$. These ambiguous expressions do not occur in English because the modifying lexical word stands in front of the head word in English noun phrases (see English translation below Figure 1.1b). In addition, we

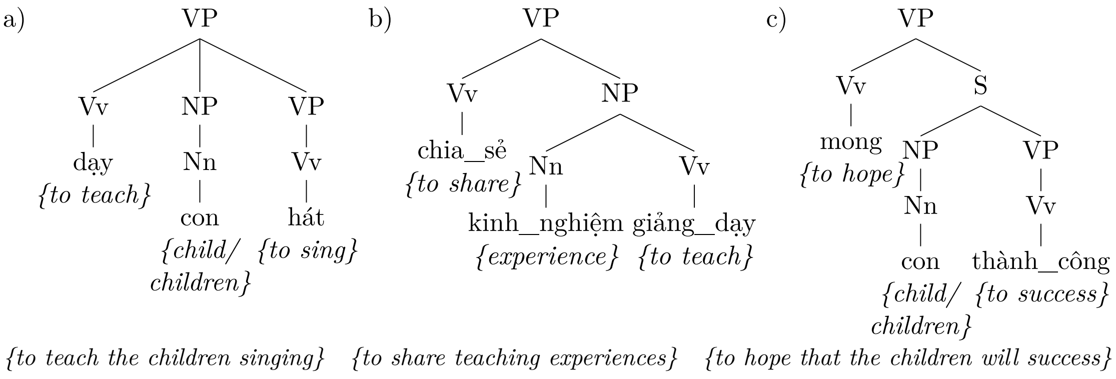


FIGURE 1.1: An example shows that one POS sequence *Vv Nn Vv* can have different annotations. Meanings of the present symbols are: VP–verb phrase; NP–noun phrase; S–sentence; Vv–verb; and Nn–noun.

can recognize subordinating clauses in English based on the conjunctions introducing the clauses such as *that*, *which*, etc. (Figure 1.1c). Unfortunately, such clues are not presented in Vietnamese text, leading to confusions of annotating Vietnamese in both manual and automatic annotations.

Because of the above-mentioned challenges, building a consistent and accurate treebank for Vietnamese is not a trivial problem.

1.4 Contributions

Our research includes two tasks: building a treebank for Vietnamese, and training a syntactic parser on the treebank. For the first task, our contributions consist of:

- Proposing annotation rules to address the challenges of Vietnamese language,
- Designing better POS and bracket tags for Vietnamese treebank,
- Developing three sets of annotation guidelines for Vietnamese treebank including word segmentation guidelines, POS tagging guidelines and bracketing guidelines,
- Proposing an appropriate annotation procedure to develop a treebank
- Building tools to support the annotation as well as quality control.

To date, we have completed the annotation of about 20,000 sentences of our treebank. The treebank has archived the inter-annotator agreement, intra-annotator agreement, and annotation accuracy of more than 90%. This indicates that the treebank is reliable. In comparison with the previous Vietnamese treebank, the new developed treebank produces better performances on three fundamental NLP tasks of word segmentation, POS tagging, and parsing.

For the second task, we have trained different parsing methods, including probabilistic context free grammars (PCFGs), conditional random fields (CRFs), and neural networks, on our treebank. By comparing the parsing outputs produced by the parsers, we have found the parsing techniques appropriate for Vietnamese language. We have also carried out a comprehensive analysis based on the parsing outputs. The results of this task have revealed the reasons for the low performance of Vietnamese parsing. Our analysis results have also showed the most frequent errors caused by different parsing methods. These results are valuable clues for improving the quality of Vietnamese parsing in the future.

1.5 Thesis overview

The remain of this thesis is organized as follows.

In Chapter 2, we present the background by introducing efforts on treebank development, how treebanks are used, and the engineering issues that need to be considered in developing a treebank. It then provides an overview of parsing methods. Finally, we present the characteristics of the Vietnamese language and how the treebank and parsing were developed in Vietnamese.

In Chapter 3, building a syntactic treebank for Vietnamese, we firstly present our method to construct a high-quality treebank and our data preparation. Next, we describe challenges and solutions in building the annotation guidelines for Vietnamese in Section 3.3. We also discuss our method about how to ensure the annotation quality while still remaining a reasonable annotation speed in Section 3.4.

Chapter 4 presents an empirical investigation of error types in Vietnamese parsers. In this chapter, we firstly evaluate the Vietnamese treebank on different parsing methods. We then investigate four possible sources for the errors produced by the parsers: the small size of training data, the word segmentation errors, the confusing POS tags, and the ambiguous constructions. Our analysis method combines the automatic tool and manual analysis. By comparing the analysis results, we can understand which error types can be tackled by the parsing methods as well as which error types are difficult for the parsers. In addition, manual analysis on the parsing errors can help us capture reasons for each error type.

Finally, in Chapter 5, we summarize the results obtained in this research and give directions for future studies.

Chapter 2

Background

This chapter gives the background that is used throughout this thesis. It contains the issues as follows. Firstly, we introduce the efforts on treebank development and how a treebank is used. We then present the engineering issues that need to be considered in developing a treebank. In addition, an overview of parsing methods is also provides in this chapter. Finally, we present the characteristics of Vietnamese and how the treebank and parsing were developed in Vietnamese language.

2.1 Efforts on treebank construction

The treebank is annotated with the syntactic structure beyond the part-of-speech tags. However, depending on the intended use, it can be enhanced with various type of information such as semantic information. There are two main groups of the treebank classified by the enhanced information: constituency treebank–treebank annotated with syntactic information and dependency treebank–treebank annotated with dependency information.

2.1.1 Constituency treebank

In the original constituency treebank such as the original Penn Treebank [10], the syntactic information added beyond the POS tags is phrase categories such as NP (noun phrase) and VP (verb phrase). In addition, constituency treebanks can be annotated

```
( (S
          (NP-SBJ-1 (PRP They) )
          (VP (VBP are) (DT all)
                    (VP (VBN priced)
                              (NP (-NONE- *-1) )
                              (PP-CLR (IN at)
                                        (NP (JJ par) ))))
          (. .) ))
```

Figure 2.1: A sentence annotated with phrase structure extracted from the Penn English Treebank. The original sentence is *"They are all priced at par."*

with grammar functions, e.g., Penn Treebank II [30]. Figure 2.1 is an example extracted from the PTB corpus that shows how a simple sentence is annotated. This example can be presented as a tree as shown in Figure 2.2. In this tree, the leaf nodes (terminals) are words in the sentence. Pre-terminals (preceding terminals) are POS tags of the words. Higher levels are the phrase categories. We can also see that the noun phrase (NP) *They* and the prepositional phrase (PP) *at par* were annotated with the function tags *SBJ* (subject) and *CLR* (closely related) respectively to indicate their syntactic functions. In addition, the null argument[1] of the verb *priced* was also marked with an asterisk coindexed (1) with the noun phrase *They*. Specifically, the verb *priced* requires a direct object to complete its meaning. However, this object is used the subject of the sentence, which is the noun phase *They*. Therefore, the null object of the verb *priced* is marked with the asterisk. The subject noun phrase (NP-SBJ) and asterisk are also marked with the same index (1) to indicate that they are similar.

The PTB was annotated with two layers of part-of-speech tagging and bracketing. The POS tag set used in this treebank includes 36 tags as shown in Table 2.1. The bracketing layer is a combination of the constituent structure and grammatical functions that includes constituency tags, function tags, null elements, and reference indexes. Tables 2.2, 2.3, and 2.4 present the constituent tags, function tags, and the null elements used in the PTB respectively. These tables include 26 constituent tags, 10 function tags, and 6 null elements. The PTB is the first large-scale annotated corpus. It consists of about 50,000 sentences of American English collected from the Wall Street Journal (WSJ).

Following the success of the PTB, the Penn Chinese Treebank [17] has been built. This treebank has the annotation scheme adapted from the PTB. However, as words in Chinese sentences are not distinguished by the blank spaces, the Penn Chinese Treebank was provided word boundaries before annotating POS and brackets. This treebank has used 33 POS tags, 17 constituent tags, 26 function tags, and 7 null elements. The Penn

[1]An argument is a mandatory syntactic unit that completes meaning of the sentence. When an argument is omitted, it is marked by a null element in the Penn Treebank II.

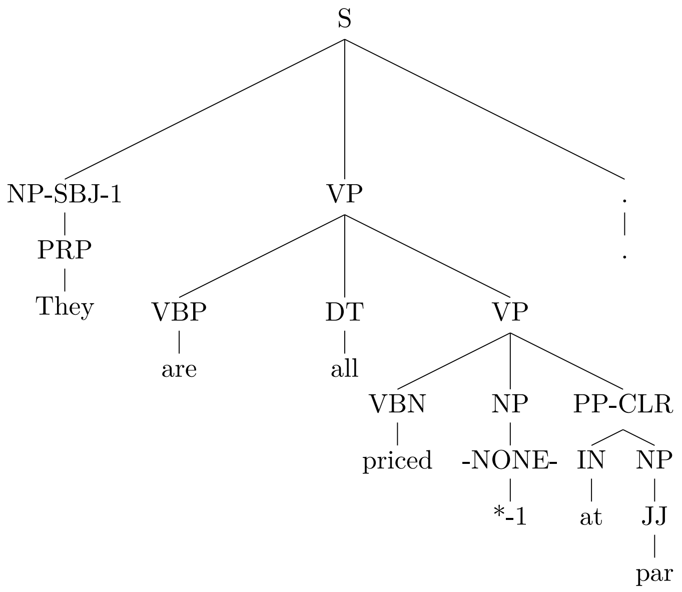


FIGURE 2.2: The constituent tree of the sentence *"They are all priced at par."* annotated according to the Penn Treebank II's scheme.

TABLE 2.1: The Penn Treebank POS tag set.

No.	**Tag**	**Meaning**	**No.**	**Tag**	**Meaning**
1	CC	Coordinating conj.	19	PP$	Possessive pronoun
2	CD	Cardinal number	20	RB	Adverb
3	DT	Determiner	21	RBR	Adverb, comparative
4	EX	Existential there	22	RBS	Adverb, superlative
5	FW	Foreign word	23	RP	Particle
6	IN	Preposition	24	SYM	Symbol
7	JJ	Adjective	25	TO	Infinitive to
8	JJR	Adjective, comparative	26	UH	Interjection
9	JJS	Adjective, superlative	27	VB	Verb, base form
10	LS	List item marker	28	VBD	Verb, past tense
11	MD	Modal	29	VBG	Verb, gerund/present pple
12	NN	Noun, singular or mass	30	VBN	Verb, past participle
13	NNS	Noun, plural	31	VBP	Verb, non-3rd ps. sg. present
14	NNP	Proper noun, singular	32	VBZ	Verb, 3rd ps. sg. present
15	NNPS	Proper noun, plural	33	WDT	Wh-determiner
16	PDT	Predeterminer	34	WP	Wh-pronoun
17	POS	Possessive ending	35	WP$	Possessive wh-pronoun
18	PRP	Personal pronoun	36	WRB	Wh-adverb

Table 2.2: The Penn Treebank constituent tag set.

No.	**Tag**	**Meaning**
1	ADJP	Adjective phrase
2	ADVP	Adverb phrase
3	CONJP	Conjunction phrase
4	FRAG	Fragment
5	INTJ	Interjection
6	LST	List marker
7	NAC	Used to show the scope of certain prenominal modifiers within a NP
8	NP	Noun phrase
9	NX	Used within certain complex NPs to mark the head of the NP
10	PP	Prepositional phrase
11	PRN	Parenthetical
12	PRT	Particle
13	QP	Quantifier phrase
14	RRC	Reduced relative clause
15	UCP	Unlike coordinated phrase
16	VP	Verb phrase
17	WHADJP	Wh-adjective phrase
18	WHAVP	Wh-adverb phrase
19	WHNP	Wh-noun phrase
20	WHPP	Wh-prepositional phrase
21	X	Unknown, uncertain, or unbracketable
22	S	Simple declarative clause
23	SBAR	Clause introduced by a (possibly empty) subordinating conjunction
24	SBARQ	Direct question introduced by a wh-word or a wh-phrase
25	SINV	Inverted declarative sentence
26	SQ	Inverted yes/no question, or main clause of a wh-question

Table 2.3: The Penn Treebank function tags.

No.	**Tag**	**Meaning**	**No.**	**Tag**	**Meaning**
1	ADV	Adverbial	11	DIR	Direction
2	NOM	Nominal	12	EXT	Extent
3	DTV	Dative	13	LOC	Locative
4	LGS	Logical subject	14	MNR	Manner
5	PRD	Predicate	15	PRP	Purpose or reason
6	PUT	Locative complement of put.	16	TMP	Temporal
7	SBJ	Surface subject	17	CLR	Closely related
8	TPC	Topicalized	18	CLF	Cleft
9	VOC	Vocative	19	HLN	Headline
10	BNF	Benefactive	20	TTL	Title

TABLE 2.4: The Penn Treebank null elements.

No.	Label	Meaning
1	*T*	Trace of A-movement
2	(NP *)	Arbitrary PRO, controlled PRO, and trace of A-movement
3	0	Null complementizer
4	*U*	Unit
5	*?*	Placeholder for ellipsed material
6	*NOT*	Anti-placehoder in template gapping

Chinese Treebank is also a large scale corpus consisting of 40,000 sentences. The material was collected from Xinhua newswise, Hong Kong news, and the Sinorama magazine.

The Penn Korean Treebank [16] also follows the PTB styles. It was annotated with two layers of POS tagging and bracketing. However, different from PTB and the Penn Chinese Treebank where the material was collected from the newspapers, the Penn Korean Treebank is constituted by texts from military language training manuals. In addition, each word in the Penn Korean Treebank was annotated not only with POS tags but also with morphological tags because Korean is a morphologically rich language. This treebank includes about 54 thousand words in 5,000 sentences.

Furthermore, adapted versions of the PTB's scheme are also found in the Penn Arabic Treebank [31], the Spanish Treebank [32], etc.

2.1.2 Dependency treebank

The dependency structure, different from the phrase structure, does not use phrasal nodes. In the dependency structure, the main verb is considered as the center of a clause. The dependency relations between the main verb and the other words are presented by directed links. For example, Figure 2.3 shows the Stanford Dependencies representation for the sentence *They are all priced at par.* The graph representation for this sentence is given in Figure 2.4. In general, each representation *X(g-i, d-j)* is a binary relation between the head word g at the position i of the sentence and the dependent word d at the position j of the sentence. Six grammatical relations in the sentence are expressed as follows:

- nsubjpass(priced-4, They-1): The word *They* at position 1 of the sentence is the head word of the passive nominal subject of the passive clause. In this passive clause, the word *priced* at position 4 is the main verb.

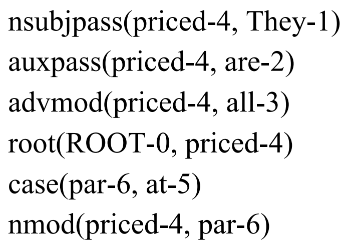

nsubjpass(priced-4, They-1)
auxpass(priced-4, are-2)
advmod(priced-4, all-3)
root(ROOT-0, priced-4)
case(par-6, at-5)
nmod(priced-4, par-6)

Figure 2.3: Stanford Dependencies representation for the sentence *"They are all priced at par"*.

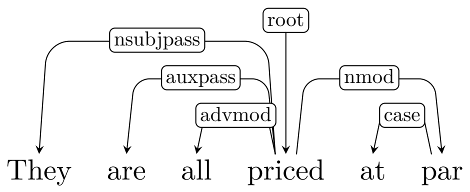


Figure 2.4: Graph representation for the sentence *"They are all priced at par"*.

- auxpass(priced-4, are-2): *are* is a non-main verb playing a role as the passive auxiliary of the passive clause.
- advmod(priced-4, all-3): *all* is an adverbial modifier of the main verb *priced*.
- root(ROOT-0, priced-4): The root of the sentence is the word *priced*. ROOT is a fake node used as the governor. ROOT is indexed with 0, since the indexing of real words in the sentence starts at 1.
- case(par-6, at-5): The preposition *at* does not directly modify the main verb. It introduces the noun phrase *par*.
- nmod(priced-4, par-6): *par* is a nominal modifier of the main verb *placed*.

You can see that the word *at* in this example does not have a direct relation with the main verb. However, its head word directly connects to the head verb of the sentence.

Similarly to the constituent structure, the dependency structure is also a primary syntax. While the constituent structure is usually selected for languages having fixed word order, the dependency structure is suitable with languages having freer word order such as Japanese [27], Russian [37], Turkish [28], and Czech [29]. This is because the phrase structure is not rigid in languages with free word order. These languages allow syntactically discontinuous expressions. The dependency structure is selected because it

Table 2.5: Grammatical relations used in UD version 2.

No.	Relation	Meaning	No.	Relation	Meaning
1	acl	Clausal modifier of noun (adjectival clause)	19	expl	Expletive
			20	fixed	Fixed multiword expression
2	advcl	Adverbial clause modifier	21	flat	Flat multiword expression
3	advmod	Adverbial modifier	22	goeswith	Goes with
4	amod	Adjectival modifier	23	iobj	Indirect object
5	appos	Appositional modifier	24	list	List
6	aux	Auxiliary	25	mark	Marker
7	case	Case marking	26	nmod	Nominal modifier
8	cc	Coordinating conjunction	27	nsubj	Nominal subject
9	ccomp	Clausal complement	28	nummod	Numeric modifier
10	clf	Classifier	29	obj	Object
11	compound	Compound	30	obl	Oblique nominal
12	conj	Conjunct	31	orphan	Orphan
13	cop	Copula	32	parataxis	Parataxis
14	csubj	Clausal subject	33	punct	Punctuation
15	dep	Unspecified dependency	34	reparandum	Overridden disfluency
16	det	Determiner	35	root	Root
17	discourse	Discourse element	36	vocative	Vocative
18	dislocated	Dislocated elements	37	xcomp	Open clausal complement

is flatter than the constituency structure, such as the finite verb phrase constituents[2] are not admitted in the dependency structure. It should be noted that despite of the structure a treebank is annotated under, we can convert the treebank from one structure to another [38–40].

Recently, Universal Dependencies [41] (UD), a project that develops cross-linguistically consistent treebank annotation for many languages, has received a wide interest from many researchers. The universal annotation scheme is a combination of Stanford dependencies [42], Google universal POS tags [43], and Interset interlingua for morphosyntactic tag sets [44]. UD version 2 contains 37 grammatical relations between words that are listed in Table 2.5. To date, 70 treebanks of 50 languages have been released[3]. Other 10 treebanks will be released next time.

2.1.3 Other structures

Some treebanks have a combined annotation scheme between constituent annotations and dependency annotations. For instance, in the TIGER Treebank [45] for German, each sentence is presented as a syntactic tree where non-terminal nodes present phrasal categories and edges indicate syntactic functions. A variation of this treebank type is

[2]Verb phrases generally divided into two types, finite and non-finite. The finite verb phrases have head words that are the finite verbs. While, heads of the non-finite verb phrases can be an infinitive, a participle, or a gerund (non-finite verbs). A finite verb has a subject. However, a non-finite verb appears below the finite verb in the hierarchy of syntactic structure.

[3]http://universaldependencies.org

found in Italian Syntactic-Semantic Treebank [46]. In this treebank, constituent structure and dependency relations are distributed over two different layers.

Both dependency and constituency treebanks can be enhanced with semantic annotation. For example, in the Sinica Treebank [47] for Chinese, a semantic role[4] is assigned to each constituent. Semantic roles are also assigned to the dependency-based annotation for the Turin University Treebank [48].

While dependency and constituency treebanks are usually intended to be theory-neutral, in which they use uncontroversial categories that can be recognized in most syntactic theories [49], some treebanks follow a specific linguistic theory. For examples, head-driven phrase structure grammar (HPSG) [50] has been used for the English Treebank [51], the Bulgarian TreeBank (BulTreeBank) [52], etc.; The CCGBank that uses the combinatory categorial grammar (CCG) [53] were also created for English [54], German [55], etc.

2.2 The use of treebank

Treebanks are used to train word segmentation tools, part-of-speech tagging tools and parsers. These tools then are applied to improve the performance of natural language processing applications.

2.2.1 Word segmentation tool

Word segmentation is the task of splitting a sentence into words. For languages like English, it is not necessary to do word segmentation because words are separated by blank spaces. However, such delimiters do not appear in Chinese, Japanese, Vietnamese, Thai, Myanmar, etc. Various automatic word segmentation tools have been developed for these languages by using treebanks for training and testing. For examples, Sun and Xu [56], Chen et al. [57] and Chen et al. [58] used the Penn Chinese Treebank [17] to train their Chinese word segmentation. Kaji and Kitsuregawa [59] produced Japanese word segmentation by using the Japanese Treebank [27].

Meanwhile, there is also research on word segmentation that used corpora annotated with only word boundaries. For example, Dinh et al. [60] and Huyen et al. [61] used a Vietnamese word segmentation corpus including 1,264 articles (507,358 words) of the "Politics – Society" section downloaded from the newspaper Tuổi Trẻ. Noyunsan et al.

[4]A semantic role expresses a relationship that a dependent has with the main verb of a sentence. A relationship can be accompaniment, agent, beneficiary, causer, dative, result, etc.

[62] used a Thai word segmentation corpus, BEST[5], containing about 5 million words belonging to four different genres, including academic articles, encyclopaedic text, novel(s), and news. Ding et al. [63] used a Myanmar word segmentation corpus including 60,000 sentences. One reason for the use of such corpora is that there is no treebanks available for these languages. Another possible reason is that the size of the word segmentation corpus is much larger than that of the treebank, which is better for statistical methods.

2.2.2 POS tagger

Many POS taggers were developed by using treebank to train and evaluate their models. According to a report at ACL Wiki[6], there are 18 state-of-the-art POS taggers developed for English by using the Penn Treebank Wall Street Journal (WSJ) and 3 state-of-the-art French POS taggers used the French Treebank [18]. The best English POS tagger was created by Choi [64] which achieved an accuracy of 97.64%. For French, the best tagger [65] has an accuracy of 97.80%.

For other languages such as Chinese, Japanese, Korean, etc., many POS taggers have been developed. For example, Sun and Wan [66] and Huang et al. [67] used the Penn Chinese Treebank [17] to develop the Chinese POS taggers. The tagger by Sun and Wan [66] obtained an accuracy of 95.34%. For Japanese language, many POS taggers [59, 68, 69] were trained on the Japanese Treebank [27].

2.2.3 Parser

The development of treebanks has promoted research in parsing. We report several English and Chinese parsers and their accuracy in Table 2.6. In this table, column F_1 presents the evaluation results for constituency parsers[7]. The last two columns, *UAS* (unlabeled attachment score) and *LAS* (labeled attachment score), present the evaluation results for dependency parsers. For English, the parsing accuracy has been about 90% for both constituency and dependency parsers. Meanwhile, Chinese parsers have reached about 86% in F-score. It is also necessary to note that these parsers used the same portion of the treebanks in their training and evaluating stages. For the Penn Treebank, sections 2-21 (39,832 sentences) were used for training, section 22 (1,700 sentences) was used as the dev set and 23 (2,416 sentences) was used as the test set. The Penn Chinese Treebank was divided as follows: articles 26-270 (16,091 sentences)

[5]http://thailang.nectec.or.th/best/
[6]https://www.aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)
[7]Parsing evaluation methods are presented in Section 2.4.6

TABLE 2.6: Parsing results on different treebanks. "F_1" presents the evaluation results for constituency parsers. UAS (unlabeled attachment score) and LAS (labeled attachment score) present the evaluation results for dependency parsers.

Parser	**Method**	Treebanks	$\mathbf{F}_1$	UAS	LAS
English					
Dyer et al. [74]	RecurrentNN - G	PTB [30]	92.4		
Dyer et al. [74]	RecurrentNN - D	PTB [30]	89.8		
Petrov and Klein [70]	Unlexicalized PCFG	PTB [30]	90.1		
Hall et al. [72]	CRF	PTB [30]	89.2		
Durrett and Klein [73]	Neural CRF	PTB [30]	91.1		
Klein and Manning [75]	Unlexicalized PCFG	PTB [30]	85.7		
Socher et al. [15]	RecursiveNN & PCFG	PTB [30]	90.44		
Dyer et al. [76]	Stack LSTM	Dep. [38]		93.1	90.9
Chen and Manning [77]	Neural network	Dep. [38]		91.8	89.6
Chinese					
Dyer et al. [74]	RecurrentNN - G	CTB [17]	82.7		
Dyer et al. [74]	RecurrentNN - D	CTB [17]	80.7		
Petrov and Klein [70]	Unlexicalized PCFG	CTB [17]	78.6		
Wang et al. [78]	FeedforwardNN	CTB [17]	86.6		
Dyer et al. [76]	Stack LSTM	CTB-Dep.		87.2	85.7
Chen and Manning [77]	Neural network	CTB-Dep.		83.9	82.4

as the training set, articles 1-25 (803 sentences) as the developing set, articles 271-300 (1,910 sentences) as the testing set.

In addition, the development of different treebanks that followed the same syntactic theory has promoted the development of multi-lingual parsers. For example, the constituency parser by Petrov and Klein [70] was trained on three treebanks, the Penn Treebank, the Penn Chinese Treebank, and the German Treebank [71]. The parser by Hall et al. [72] and Durrett and Klein [73] were trained on ten different treebanks for English, Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish, and Swedish.

2.2.4 Applications of syntactic analysis tools

Word segmentation and POS tagging are the fundamental pre-processing steps not only for syntactic parsers but also for machine translation [79, 80], text classification [81, 82] and so on. Specially, parsers were used in a wide variety of natural language processing applications. For example, in machine translation, parsers were applied for reordering [4–6]. In question answering, Jijkoun et al. [7] improved the number of correctly answered questions by using a dependency parser, Verberne et al. [8] used syntactic information to improve why-question answering, Galitsky et al. [9] applied a syntactic parser to find structural similarity between questions and answers in order to rank the candidate answers, etc. In information retrieval, parsers were used to parse text before retrieving. Chinkina et al. [83] and Barr et al. [84] showed that part-of-speech tags and other syntactic information can be used to improve the web search results significantly.

2.3 Technologies for treebank development

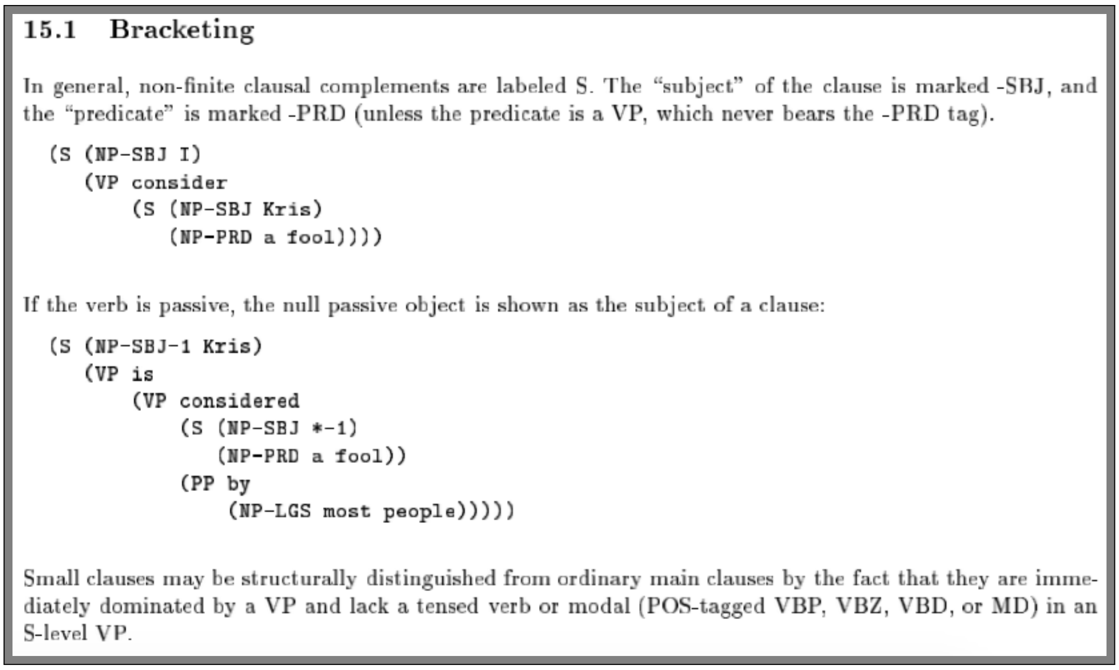
Building a high-quality treebank while ensuring a reasonable annotation speed and cost is one of the most important consideration in corpus development. This requires well-documented annotation guidelines, an appropriate division of labor, good supporting tools, annotators who understand the guidelines and familiar with the annotation, and a proper annotation process. In addition, the material selection is also an issue that needs to be considered in developing the treebank.

2.3.1 Annotation guidelines

In a consistent treebank, expressions having the same structure have to be annotated in the same way. Annotation guidelines are required for ensuring this. The guidelines are used to train the annotators as well as serving the use of the treebank in the future. Therefore, the guidelines have to present an annotation scheme, general principles of annotation. In addition, the linguistic phenomena and their annotation examples have to be described as detailed as possible in the guidelines. Figure 2.5 shows a guideline sample extracted from the bracketing guidelines of the PTB. This guideline illustrates how to bracket a small clause and its near relatives. We can see from the figure that beside a clear instruction, we need to give examples with completed annotations so that annotators can be easily understand the guideline.

Because of the diversity of the linguistic phenomena, the guidelines cannot be completed before the treebank is annotated. The guideline development often includes at least two stages [10, 16, 36]. The first version of the guidelines is created on the basis of the linguistic literature and group meetings. Then, the guidelines are improved with the newly found constructions during the annotation process.

For a treebank, guidelines usually include several sets corresponding to the annotated layers. For example, the guidelines for the Penn English Treebank include two sets, POS tagging guidelines involving 32 pages [85] and bracketing guidelines consisting of 317 pages [86]. For the case of Chinese, in which word boundaries are not apparent by blank spaces as those in English, the word segmentation guidelines (30 pages) [87]


15.1 Bracketing

In general, non-finite clausal complements are labeled S. The "subject" of the clause is marked -SBJ, and the "predicate" is marked -PRD (unless the predicate is a VP, which never bears the -PRD tag).

```
(S (NP-SBJ I)
   (VP consider
       (S (NP-SBJ Kris)
          (NP-PRD a fool))))
```

If the verb is passive, the null passive object is shown as the subject of a clause:

```
(S (NP-SBJ-1 Kris)
   (VP is
       (VP considered
           (S (NP-SBJ *-1)
              (NP-PRD a fool))
           (PP by
               (NP-LGS most people)))))
```

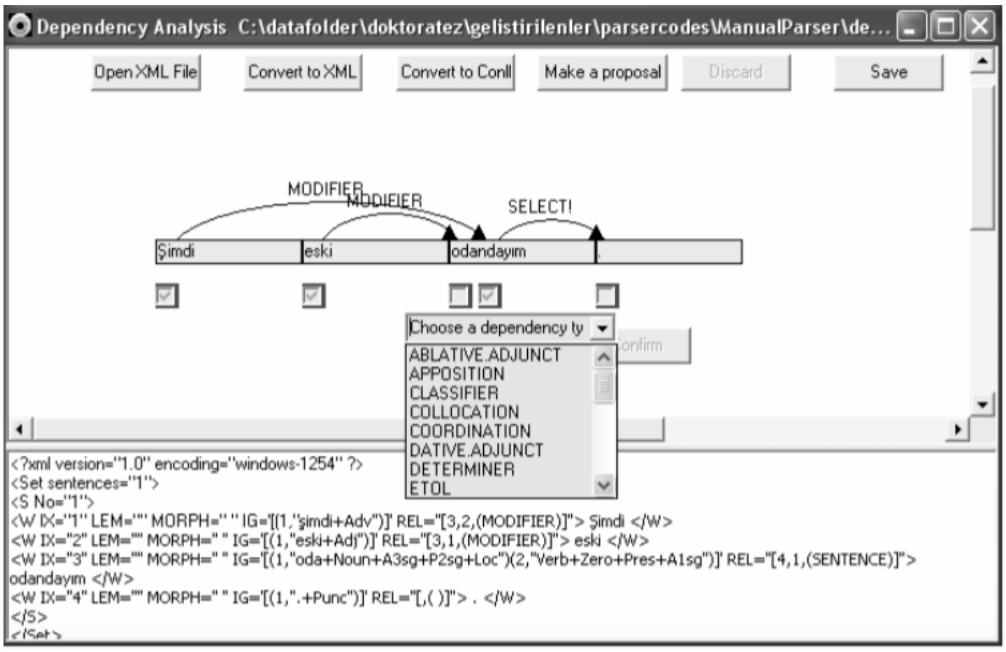
Small clauses may be structurally distinguished from ordinary main clauses by the fact that they are immediately dominated by a VP and lack a tensed verb or modal (POS-tagged VBP, VBZ, VBD, or MD) in an S-level VP.

FIGURE 2.5: A screen shot of the PTB bracketing guidelines.

were built together with the POS tagging guidelines (43 pages) [88] and the bracketing guidelines (191 pages) [89].

2.3.2 Division of labor

A treebank can be annotated manually, automatically or semi-automatically. Manual annotation means human annotators have to do their work from scratch, such as SynTag for Swedish [90], or in the first stage of the treebank development when a parser was not available and there is no data to train a parser either [3, 17]. Since manual annotation is very expensive and also takes time, it cannot be applied on a large corpus. We instead use automatic annotation in such cases. For example, the Google Books n-gram corpus has been automatically annotated with syntactic information [91]; similarly to 700 million words of Lassy Large Treebank for Dutch [92]. Although the automatic annotation can be applied on large corpora, it usually produces a high ratio of errors. One of the reasons is that automatic annotation is unable to annotate particular constructions as correctly as the manual annotation can. In order to utilize the advantages of manual and automatic annotations, high-quality treebanks such as Penn English Treebank [10] and Penn Chinese Treebank [17] applied semi-automatic annotation. In this annotation method, automatic tools were used to annotate the data before human editing. Marcus et al. [10] showed that semi-automated tagging is superior to the entirely manual tagging on three measures: speed, consistency, and accuracy.



```
<?xml version="1.0" encoding="windows-1254" ?>
<Set sentences="1">
<S No="1">
<W IX="1" LEM="" MORPH=" " IG='[(1,"şimdi+Adv")]' REL="[3,2,(MODIFIER)]"> Şimdi </W>
<W IX="2" LEM="" MORPH=" " IG='[(1,"eski+Adj")]' REL="[3,1,(MODIFIER)]"> eski </W>
<W IX="3" LEM="" MORPH=" " IG='[(1,"oda+Noun+A3sg+P2sg+Loc")(2,"Verb+Zero+Pres+A1sg")]' REL="[4,1,(SENTENCE)]">
odandayım </W>
<W IX="4" LEM="" MORPH=" " IG='[(1,".+Punc")]' REL="[,( )]"> . </W>
</S>
</Set>
```

FIGURE 2.6: A screen shot of the ITU treebank annotation tool.

2.3.3 Tools

Supporting tools are used to ensure the annotation quality as well as improving the annotation speed. Various tools are required. In semi-automatic annotation, we have to prepare tools for preliminary annotation before manual editing, including POS tagger, parser, etc. An annotation tool is required to support the manual annotation or editing. By using the annotation tool, the annotation speed can be improved significantly as errors caused by typing can be eliminated. Figure 2.6 presents a screen shot of the annotation tool of the ITU treebank [93]. This tool is designed only for Turkish with dependency structures. We can see from this figure that all annotations are done by selecting check boxes and combo boxes hence avoiding typing errors. After the annotating, a variety of tools are used to detect and correct the annotation errors in the treebanks [19, 20, 94, 95].

2.3.4 Annotator

Human annotators play a crucial role in building a high-quality treebank and improving the annotation speed. Therefore, annotators should have a good background (e.g. graduate training in linguistics), understand the guidelines and familiar with the annotation. To ensure this, besides finding good-background annotators, we need to have an appropriate training process. The quality of annotators are measured through the accuracy and inter-annotator agreement. In treebanks such as the Penn English Treebank [10] and the Penn Chinese Treebank [17], the inter-annotator agreements were higher than 90%.

2.3.5 Annotation process

A proper annotation process is also necessary in building a high-quality treebank. In general, an annotation process includes feasibility study, guideline development, treebank annotation, and quality control. However, the order of these steps can be different between treebanks. For example, in the Penn Chinese Treebank, Xue et al. [17] designed an annotation process of five steps. In the feasibility study, they identified major controversial topics and tested whether consistent annotation was possible. They then created the first draft of the annotation guidelines based on the literature. The annotation included two passes. In the first pass, they recognized difficult constructions and revised the guidelines. The second pass involved correcting the output of the first pass and measuring the inter-annotator agreement. Finally, they cleaned up the data with the support of automatic tools. Different from Xue et al. [17], Han et al. [16] completed the guidelines before the annotation stage began. Instead of applying the quality control while annotating the treebank as Xue et al. [17], Han et al. [16] conducted the quality control when the annotation of the treebank was finished.

For multi-layer treebanks, it is also necessary to consider the order to annotate different layers and whether the layers should be annotated separately or in parallel. In many cases, there are dependencies between layers. For example, we have to determine word boundaries in Chinese and Japanese sentences before assigning a POS tag to them; or determining POS can be based on the phrase category or vice versa. These dependencies stipulate the order in which the layers should be annotated. In practice, there are treebanks in which their layers are annotated together [96]. However, each annotator usually works with a single layer at a time [17, 27, 97].

2.3.6 Material

Material selection is also an essential issue that needs to be considered in the construction of a treebank. An important question is whether we should select a balanced sample of different text genres or focus on a specific text type or domain. Most available treebanks include texts collected from newspapers. An example is the Wall Street Journal section of the Penn Treebank [10], which has impacted as a common schema for many other treebanks such as the TIGER Treebank for German [45], the French Treebank [18], and the Penn Chinese Treebank [17]. However, Gildea [98] showed that parsing models trained on a treebank that focuses on news domain did not give a satisfied performance on other domains because the treeabank does not cover all syntactic as well as semantic aspects. This may promote the development of specific-domain treebanks such as the Penn Korean Treebank focusing on the military [16] and SCTB Treebank for Chinese [99] focusing on the scientific domain. For the case of the ITU Web Treebank [97], the selected material was different from the above mentioned treebanks. It is an unnatural language used in the web. Besides treebanks with a specific text type or domain, several treebanks with different text genres have been built. For example, the SUSANNE Treebank [100] was developed based on a subset of the Brown Corpus, which is a balanced treebank.

2.4 An overview of parsing methods

While the probabilistic context-free grammar (PCFG) is a typical generative parsing model, conditional random fields (CRF) represent the discriminative parsing method. These two algorithms have been successfully applied on many languages [70, 72, 75, 101–103]. Recently, neural networks have also been used for parsing [15, 73, 74]. In Chapter 4 of this thesis, we will evaluate the Vietnamese Treebank on these parsing models. In the following subsections, we will briefly describe these parsing methods (2.4.1, 2.4.2, 2.4.4, 2.4.5, and 2.4.3). Then, we introduce techniques to evaluate parsers (Subsection 2.4.6). Finally, we present parsing developments that based on these methods (Subsection 2.4.7).

2.4.1 Probabilistic context-free grammar

Probabilistic context-free grammar (PCFG) is the most fundamental model for constituency parsing. Before defining PCFG, we have to study CFG first.

A context-free grammar G includes four components:

- N is a finite set of non-terminal symbols.
- Σ is a finite set of terminal symbols.
- R is a finite set of rules. Each rule has the form as $\alpha \to \beta$ in which $\alpha \in N$ and $\beta \in \{N \cup \Sigma\}$.
- $S \in N$ is a start symbol.

A left-most derivation is a sequence of strings including $s_1 \ldots s_n$ where

- $s_1 = S$, the start symbol

- s_n is a string of words taken from Σ
- s_i (for i = *2 ...* n) is derived from s_{i-1} by replacing the the left-most non-terminal α in s_{i-1} by β where $\alpha \rightarrow \beta \in R$

A PCFG is a CFG in which each rule $\alpha \rightarrow \beta \in R$ is assigned with a probability. This probability is defined as $P(\alpha \rightarrow \beta \mid \alpha)$, which is the conditional probability of choosing the rule $\alpha \rightarrow \beta$, given that α is a non-terminal symbol being expanded in the derivation. For any non-terminal $\alpha \in N$, we have the following constraints:

$$\sum_{\alpha \rightarrow \beta \in R} P(\alpha \rightarrow \beta \mid \alpha) = 1 \tag{2.1}$$

$$P(\alpha \rightarrow \beta \mid \alpha) \geq 0 \tag{2.2}$$

Assume that t is a parse tree (derivation) of a sentence s, the probability of the tree t is the product of the probabilities of the rules $\alpha_1 \rightarrow \beta_1$, $\alpha_2 \rightarrow \beta_2$, ..., $\alpha_n \rightarrow \beta_n$ that form t. Formally, this probability can be defined as follows:

$$P(t) = \prod_{i=1}^{n} P(\alpha_i \rightarrow \beta_i \mid \alpha_i) \tag{2.3}$$

Assume $T(s)$ is a set of possible parse trees of a sentence s under the grammar G, the most likely tree for the sentence s is defined as follows:

$$t_{best}(s) = \arg\max_{t \in T(s)} P(t) \tag{2.4}$$

For example, Figure 2.7 presents two parse trees for the sentence *program trading has increased chances for market crashes* under the PCFG given in Table 2.7. The probability for each tree is computed as follows:

$P(t_1) = 1.0 * 0.2 * 0.2 * 0.7 * 0.4 * 0.1 * 0.8 * 0.2 * 0.2 * 0.5 * 0.7 * 1.0 * 0.4 * 1.0 * 0.3 * 0.6 = \mathbf{0.0000009032}$

$P(t_2) = 1.0 * 0.2 * 0.2 * 0.1 * 0.1 * 0.8 * 0.2 * 0.2 * 0.5 * 0.7 * 1.0 * 0.4 * 1.0 * 0.3 * 0.6 = 0.0000003226$

The probability of the tree t_1 is higher than that of the tree t_2. So, the PCFG parser will select the tree t_1.

TABLE 2.7: A simple PCFG. For each rule, we have to assign a probability, such as the rule NP $\longrightarrow$ NN NN has a probability of 0.2.

S $\longrightarrow$ NP VP	1.0	NN $\longrightarrow$ program	0.2
NP $\longrightarrow$ NN NN	0.2	NN $\longrightarrow$ trading	0.5
NP $\longrightarrow$ NN NNS	0.2	NN $\longrightarrow$ market	0.3
NP $\longrightarrow$ NP NP	0.1	NNS $\longrightarrow$ crashes	0.6
NP $\longrightarrow$ NP PP	0.4	NNS $\longrightarrow$ chances	0.4
NP $\longrightarrow$ NNS	0.1	IN $\longrightarrow$ for	1.0
VP $\longrightarrow$ VBZ VP	0.2	VBZ $\longrightarrow$ has	0.7
VP $\longrightarrow$ VBN NP	0.7	VBZ $\longrightarrow$ crashes	0.3
VP $\longrightarrow$ VBN NP PP	0.1	VBN $\longrightarrow$ increased	1.0
PP $\longrightarrow$ IN NP	0.8	VB $\longrightarrow$ program	0.7
PP $\longrightarrow$ IN CC IN NP	0.2	VB $\longrightarrow$ market	0.3

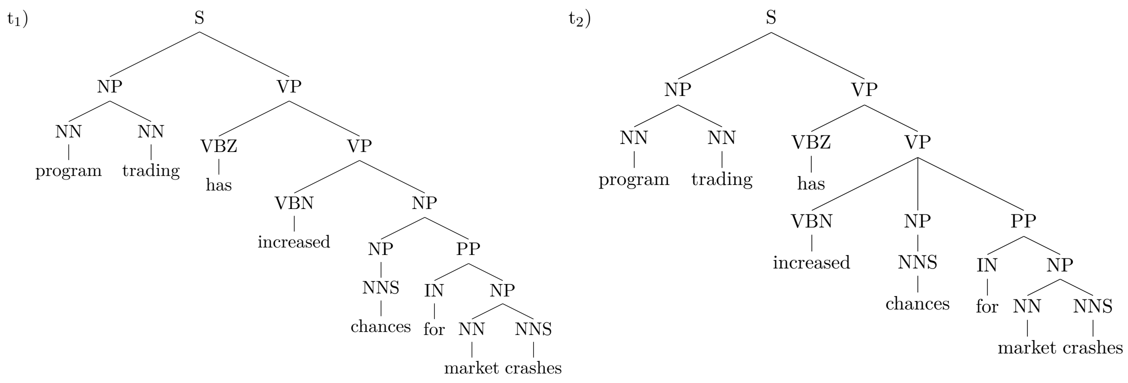


FIGURE 2.7: Two parse trees for the sentence *program trading has increased chances for market crashes* under the PCFG in Table 2.7.

2.4.2 Conditional random field

Similar to PCFG parsing, conditional random field (CRF) parsing also begins from a context-free grammar. However, instead of assigning the probability to each rule so that $\forall \alpha \in N, \sum_{\alpha \to \beta \in R} P(\alpha \to \beta \mid \alpha) = 1$, CRF defines a scoring function (clique potential function) $\phi(r \mid s; \theta)$ for each input sentence s and an anchored rule r. An anchored rule is the conjunction of the CFG rule and the span which refers to the start, stop, and split positions where the rule is anchored. A tree t is a collection of anchored rules.

CRF parsing typically achieves higher performance in comparison with PCFG parsing due to the ability to incorporate arbitrary features. For example, although span length affects the parsing probability, it is not easy to take span length into account in PCFG parsing. While features used in CRF parsing can be span length, the first and last words of the span, the words immediately preceding and immediately following the span, the words at and around the split point of the binary rule, the parent of the rule, the identity of the rule, and span shape features.

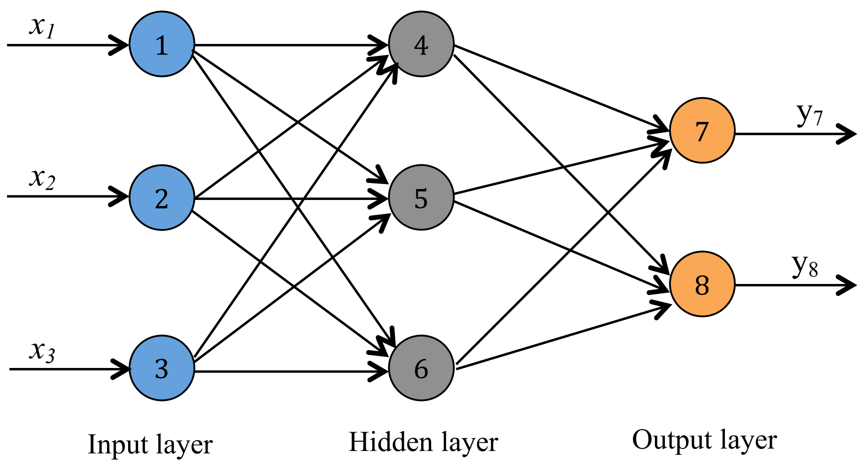


FIGURE 2.8: Feedforward neural network.

In CRF parsing, assuming that *T(s)* is a set of possible parse trees for a given sentence s, the probability of a tree $t \in$ *T(s)* conditioned on the sentence s is defined as follows:

$$P(t \mid s; \theta) = \frac{1}{Z_s} \prod_{r \in t} \phi(r \mid s; \theta) \tag{2.5}$$

where

$$Z_s = \sum_{t' \in T(s)} \prod_{r \in t'} \phi(r \mid s; \theta) \tag{2.6}$$

Assume that we have a vector function *f(r,s)* that computes the value for each feature, a feature f_i will have the value f_i*(r,s)*. The scoring function is presented as follows:

$$\phi(r \mid s; \theta) = exp\left(\sum_i \theta_i f_i(r, s)\right) \tag{2.7}$$

where θ_i is the parameter corresponding to feature f_i that can be learned through maximizing the log conditional likelihood of the training data [72, 73, 103].

2.4.3 Feedforward neural network

Figure 2.8 illustrates a feedforward neural network that includes no cycles. The information moves forward, from the input nodes, through the hidden nodes and to the output nodes. A neural network can have hidden layers or not. Each node in the network is associated with all nodes of the next layer, and each association has a weight w_{ij}.

In the feedforward neural network in figure 2.8, the input nodes (including nodes 1, 2, and 3) do not compute any thing. They are connected with the input data (x_i) and pass the values to the processing nodes. For nodes that belong to the first hidden layer, the input value at each node is computed as follows:

$$v_j = \sum_{i=1}^{m} w_{ij} x_i \tag{2.8}$$

where m is number of input nodes.

For other nodes from the second hidden layer to the output layer, the input value of each node is computed as follows:

$$v_j = \sum_{i=1}^{m} w_{ij} y_i \tag{2.9}$$

where m is the number of neurons of the previous layer, and y_i are the output of the previous neurons.

The output value of each neuron is computed as follows:

$$y_j = a + f(v_j) \tag{2.10}$$

where a is called as bias, f is the activation function that can be a linear function, a logistic function, etc. The weights w_{ij} are learned during the training process.

The feedforward neural network was used in Durrett and Klein [73] to learn dense features. Firstly, a sequence of words was extracted on the basis of anchor rules. These words were then embedded to form a vector representation. This vector was fed through a one-layer feedforward neural network in order to produce a dense feature representation.

2.4.4 Recurrent neural networks

Figure 2.9 presents a typical recurrent neural network (RNN) including input, hidden state, and output.

- The input includes two components, h_{t-1} is a real-valued vector representing the previous hidden state, and x_t is the embedding of the input word of the word sequence at the time step t.
- The hidden state at the time step t (h_t) is computed based on h_{t-1} and x_t as follows:

$$h_t = F(Wx_t + Uh_{t-1}) \tag{2.11}$$

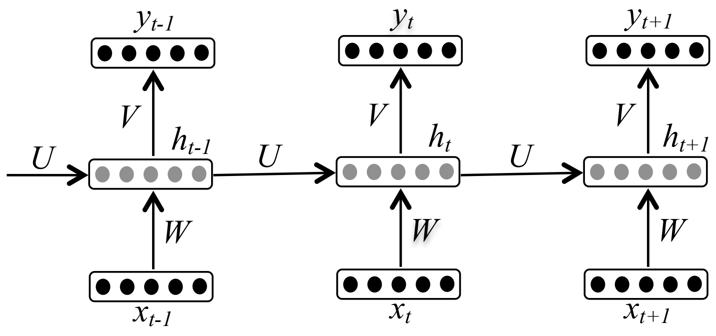


FIGURE 2.9: Recurrent neural network.

where F is usually a nonlinear function such as *tanh* or *ReLU*. h_t can be considered as memory of the network.

- y_t = *softmax(*Vh_t*)* is the output at the time step t. It is a vector of probabilities used to predict the next word in the sentence.

We can see from Figure 2.9 that RNN is actually a chain that repeats modules of neural network. The hidden state at each time step t is computed recurrently on the basis of the previous hidden state and the embedding of the current word (x_t). In addition, RNN uses the same parameters W, U, and V across the time steps. These parameters are learned using back-propagation through time. This means that the gradient at each output depends not only on the calculations of the current time step, but also on the previous time steps. Theoretically, this allows the network to learn long-term dependencies over the sequence. However, Pascanu et al. [104] showed that it is difficult to train RNN because of the vanishing gradient and exploding gradient problems. Fortunately, Long Short-Term Memory (LSTM) [105] was specifically designed to solve these problems. The LSTM remembers information for long periods of time.

Long Short-Term Memory is fundamentally similar to RNN in terms of architecture, i.e., it has a chain of repeating modules of neural network. However, each repeating module has a simple structure in RNN, such as a *tanh* layer, while in LSTM, the repeating module includes four neural network layers, viz three *sigmoid* layers and one *tanh* layer. Moreover, beside the output at each time step, LSTM also computes a cell state C_t at each time step.

The three *sigmoid* layers in each module represent three gate functions, namely forget gate, input gate, and output gate. They control how much the next cell state will be influenced by the previous cell state, how much it will be influenced by the new input, and how much of the cell state will be released as output respectively. These functions are computed on the basic of the current input and the hidden state of the previous time step.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.12}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.13}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.14}$$

A vector of new candidate cell state is created using a *tanh* layer as follows.

$$\tilde{C_t} = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.15}$$

The new cell state is then created by combining the previous cell state weighted by the forget gate and the candidate cell state weighted by the input gate.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t} \tag{2.16}$$

Finally, values of the output gate are multiplied by the new cell state through a *tanh* function. This help us to determine things that we want to output.

$$h_t = o_t * tanh(C_t) \tag{2.17}$$

LSTM has been successfully applied to machine translation [106]. The idea is that LSTM reads the input sequence in a left-to-right order and encodes it into a vector. Then, the LSTM extracts the output sequence using information in those vectors. The syntactic constituency parsing can be formulated as a translation problem. This was done by linearizing the parse tree [107] as the target language. So, we can apply the above translation model to parsing as well.

2.4.5 Recursive neural network

While RNN is usually used for sequential structures because they operate on the linear progress of time, recursive neural network generates a hierarchical structure by combining child representations to generate parent representations. For example, $s_{[l,m]}$ and $s_{[m,n]}$ in Figure 2.10 are vector representations of the child nodes. They are combined

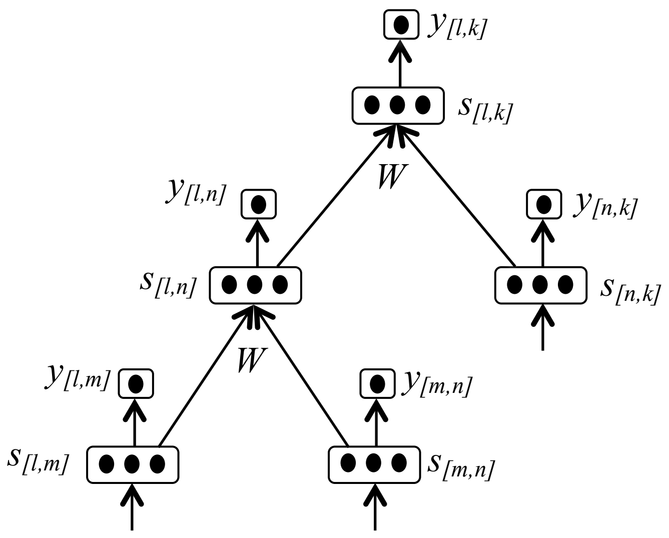


FIGURE 2.10: Recursive neural network.

into the parent $s_{[l,n]}$ by using a weight matrix W that is shared across the whole network, and a non-linearity function such as *tanh* as follows:

$$s_{[l,n]} = tanh(W[s_{[l,m]}; s_{[m,n]}] + b) \tag{2.18}$$

In parsing, l, m, and n are indexes of the word sequence. $s_{[l,n]}$ presents a non-terminal node covering from l to n. In the case $s_{[l,m]}$ *and* $s_{[m,n]}$ are leaf nodes, they are word vectors that can be computed by using Word2Vec [108] or GloVe [109]. Note that a parent node has to have the same dimensionality as its children. Suppose $s_{[l,m]}$ *and* $s_{[m,n]} \in \mathbb{R}^{n\times 1}$, *[*$s_{[l,m]}$*;* $s_{[m,n]}$*]* denotes the concatenation of the two nodes, which is a $\mathbb{R}^{2n\times 1}$ vector, and hence $W \in \mathbb{R}^{n\times 2n}$.

$y_{[l,n]}$ shows the score indicating how reasonable a parent should be created. It can be computed using an inner product as follows:

$$y_{[l,n]} = W^{score} s_{[l,n]} \tag{2.19}$$

where $W^{score} \in \mathbb{R}^{1\times n}$ is a row vector.

2.4.6 Evaluating parsers

In this section, we present four evaluation methods: PARSEVAL, Leaf-Ancestor (LA), dependency-based evaluation, and error analysis proposed by Kummerfeld et al. [110].

PARSEVAL [111] is considered as a standard measure to estimate the performance of constituency parsers [72, 75, 103, 112–114]. PARSEVAL computes recall, precision, and F$_1$-score over parse trees as follows.

$$\mathit{Recall} = \frac{\mathit{number\ of\ correct\ constituents}}{\mathit{number\ of\ constituents\ in\ the\ gold\ parse\ tree}} \tag{2.20}$$

$$\mathit{Precision} = \frac{\mathit{number\ of\ correct\ constituents}}{\mathit{number\ of\ constituents\ in\ the\ output\ parse\ tree}} \tag{2.21}$$

$$F_1 = \frac{\mathit{2\ *\ Recall\ *\ Precision}}{\mathit{Recall\ +\ Precision}} \tag{2.22}$$

It should be noted that PARSEVAL does not take POS tags into account when computing these scores.

TEDEVAL (tree-edit distance evaluation) [115] is different from PARSEVAL in terms that it allows different numbers of terminals between parse and gold trees, while PARSEVAL requires those numbers identical. TEDEVAL computes the TED accuracy on the basis of deleting (DEL) and adding (ADD) nonterminals and word entries (each word entry includes a word and its POS tag) from/to the parse tree to make it the same as the gold tree. Assuming a is an operation, i.e., DEL or ADD, if the function $C(a) = 1$ assigns a unit cost to each operation a, the cost of a sequence $\langle\ a_1,\ ...,\ a_m\ \rangle$ is defined:

$$C(\langle a_1, ..., a_m\rangle) = \sum_{i=1}^{m} C(a_i) \tag{2.23}$$

The tree-edit distance (TED) is defined as follows:

$$TED(y_1, y_2) = \min_{ES(y_1,y_2)} C(ES(y_1, y_2)) \tag{2.24}$$

Where $ES(y_1,y_2) = \langle\ a_1,\ ...,\ a_m\ \rangle$ is a sequence of operations that tunes tree y_1 to tree y_2.

The accuracy of a parse tree p in comparison with the gold tree g is defined as follows (TED accuracy):

$$TEDEVAL(p, g) = 1 - \frac{TED(p, g)}{|p| + |g| - 2} \tag{2.25}$$

Where |p| is the cost of deleting all word entries and nonterminals in the parse tree; |g| is the cost to add all word entries and nonterminals of the gold tree to the parse tree.

Leaf-Ancestor (LA) [116] is also different from the PARSEVAL. It does not check constituent labels and their word span. For each terminal node, LA computes the sequence of node labels from the terminal node to the root node S in the parser output and the gold tree. The similarity between the two trees is computed by using the Levenshtein distance [117]. The score of a tree is the average value of all terminals in the tree.

Dependency evaluation [118, 119] does not use the phrase structure, which means that the phrase categories as well as the function tags are not considered during the evaluation. In the dependency evaluation, constituent trees are converted into dependency structures with a form like "word pos head". Then recall and precision are computed as follows:

$$\begin{aligned} recall = \textit{percentage of the triples "word pos head" in the gold tree} \\ \textit{that are also found in the parser output} \end{aligned} \tag{2.26}$$

$$\begin{aligned} precision = \textit{percentage of the triples "word pos head" in the parser output} \\ \textit{that are also found in the gold tree} \end{aligned} \tag{2.27}$$

In comparison with other metrics, PARSEVAL is more accurate [120]. For example, dependency evaluation gives an unsatisfactory result on attachment errors because it did not consider propagating errors while PARSEVAL often produces a reasonable evaluation. Moreover, PARSEVAL penalizes every constituents while the dependency evaluation does not consider missing nodes or additional nodes during converting the phrase structure to the dependency structure. This means that these error types are ignored by the dependency evaluation as well.

Kummerfeld et al. [110] claimed that evaluation scores produced by PARSEVAL do not express the linguistic information corresponding to each type of parsing errors. Therefore, they proposed an error analysis tool that automatically compares the parser output with the gold trees and classifies the errors into different types such as PP attachment, NP attachment, and coordination. However, this tool was designed specific for English and then modified to make it compliant to Chinese [121]. For other languages that consist of constructions that are different from English and Chinese, this analysis tool is not applicable.

2.4.7 Development of parsing methods

In the last two decades, many approaches have been studied on parsing and largely focused on major languages such as English and Chinese. One dominant approach is the generative model for parsing with PCFG. However, the original PCFG trained on a corpus such as English Penn Treebank cannot attain the high-quality performance due to the lack of contextual information. Various strategies have been proposed to enrich the grammar. For example, Collins [102] associated each category with a lexical item; Klein and Manning [75] proposed a method to increase structural annotations. Matsuzaki et al. [114] splitted non-terminal symbols into a fixed number of subsymbols. Specially, Petrov and Klein [70] (Berkeley parser) used a hierarchical split-merge strategy to create an accurate grammar automatically. Their parser has attained high performance on the English Penn Treebank (90.7% of F-score) and many other languages.

Discriminative parsing with conditional random fields (CRF) is also a popular approach [72, 103, 113]. In these parsers, syntactic information can be enriched via features. There have been many methods of exploiting these features. While Petrov and Klein [113] produced a CFG parsing with latent annotations, Finkel et al. [103] focused on the implementation. Hall et al. [72] extracted rich input features of surface spans. Their parser outperformed the Berkeley parser on a wide range of languages.

Another popular approach to parsing is using neural networks. For example, Socher et al. [15] combined a standard PCFG with a recursive neural network in which the neural network is used to learn features and phrase representations. Different from Socher et al. [15], Durrett and Klein [73] extended a CRF parsing with a feedforward neural network, in which the feedforward neural network was used to learn dense features of the surface spans. This parser has attained an F-score of over 91% in English. Recently, Dyer et al. [74] have presented another neural-based parser that uses a recurrent neural network. This parser achieved the state-of-the-art performance in English.

2.5 Previous work on Vietnamese treebank and parsing

In this section, we first describe the Vietnamese characteristics in Section 2.5.1 to show that annotating Vietnamese is a challenging problem. Then we will present efforts in building the Vietnamese treebank and parser in Sections 2.5.2 and 2.5.3 respectively.

Table 2.8: Examples of polysemous expressions in Vietnamese.

No.	Vietnamese sentence	Translation
1	Cái bàn$_{table}$ là$_{to\ be}$ của$_{of}$ tôi$_{I/me}$	The table is mine
2	Cái bàn_là$_{iron}$ của$_{of}$ tôi$_{I/me}$	The iron is mine/my iron

2.5.1 Characteristics of Vietnamese language

Vietnamese belongs to Austroasiatic language family. It is written in Latin characters. This section describes four main characteristics of the Vietnamese language that lead to difficulties in building a treebank.

2.5.1.1 No word delimiters or inflectional morphemes

Unlike Western languages, in which blank spaces denote word delimiters, blank spaces in Vietnamese not only play the roles of word delimiters but also those of syllable delimiters. Furthermore, unlike English and Japanese, Vietnamese is not an inflectional language, in which morphological forms can provide useful clues for word segmentation, POS tagging, and bracketing. While similar problems also occur with Chinese [36], it may be more difficult to annotate Vietnamese words because the modern Vietnamese writing system is based on Latin characters, which represent its phonology but not its semantics. As a result, Vietnamese has many polysemous expressions, i.e., terminologies that have the same surface structure but different interpretations. For example, the two sentences[8] in rows 1 and 2 of Table 2.8 have the same surface structure. However, we treat the expression *bàn*$_{table}$ *là*$_{to\ be/to\ iron}$ in row 1 as two single-syllabic words while the expression *bàn là* in row 2 as a compound word. This treatment creates two sentences that have different interpretations.

2.5.1.2 Flexible word orders

Although people agree that Vietnamese (like English and Chinese) is a subject-verb-object (SVO) language, the Vietnamese word order is different from that in English and Chinese. Modifying lexical words in English and Chinese noun phrases precede the head noun. However, in Vietnamese, modifying lexical words come after the head word in all phrase types. Moreover, there are several exceptions, such as modifying lexical words

[8]In this example, *cái* is a classifier noun in Vietnamese. Classifier nouns indicate two types of entities: animate and inanimate things.

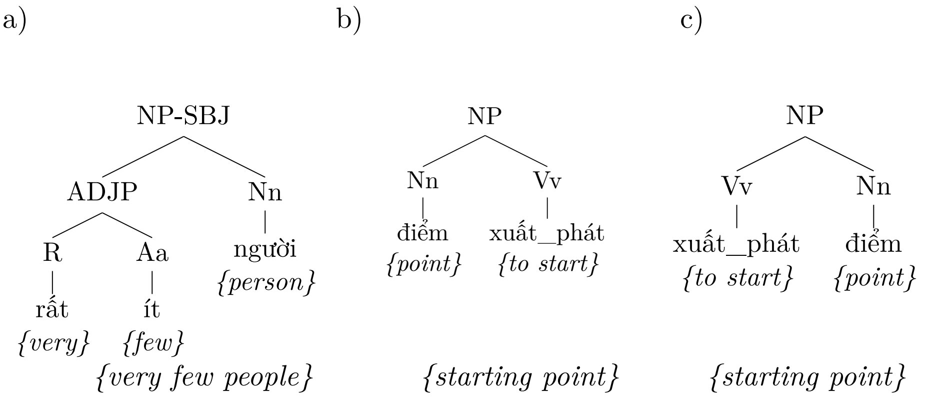


FIGURE 2.11: Examples of exceptions of word order in Vietnamese.

have to be placed before the head word (see Figure 2.11a), or different types of word order have the same interpretation (as can be seen in Figures 2.11b and 2.11c).

Nevertheless, there are cases where word orders in Vietnamese and English are similar. For instance, relative clauses come after the head noun or things that are possessed follow the owner, as shown in Figure 2.12a.

2.5.1.3 Word omission

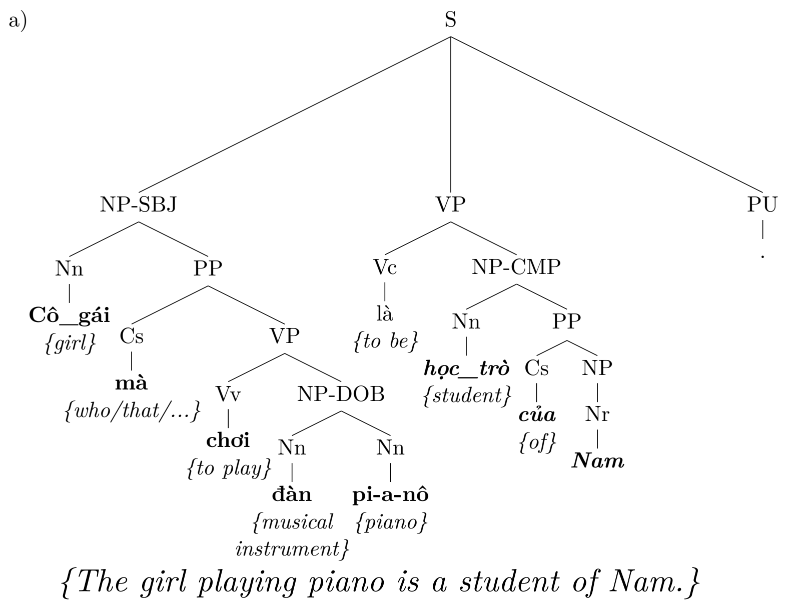
The problem of dropped words occurs frequently in Vietnamese for various types of words. We can see in Figure 2.12a that the subordinate conjunction $mà_{who/which/that/...}$ is a unique visual sign to recognize relative clauses. Similarly, the preposition $của_{of}$ can be used to identify possessive relations. However, these words can be omitted in real texts (Figure 2.12b).

2.5.1.4 Conflicting definitions among linguists

In Vietnamese, there is little consensus in community about how to define words, phrases and grammatical structures. For example, the two expressions of $cá_{fish}$ $rô_{anabas}$ *{anabas}* and $bệnh_{illness}$ $sởi_{measles}$ *{measles}* have the same construction: a combination of a categorization noun[9] and a specific noun. However, *cá_rô* is treated as a compound word while *bệnh sởi* is treated as two single words [1, 122].

As we mentioned above, people agree that Vietnamese is a subject-verb-object (SVO) language. However, Figure 2.13a shows a sentence that the head word of the predicate is not a verb. For sentences that do not have the main verb, we can use the coordinating conjunction *thì* to link the subject and the predicate as shown in Figure 2.13b.

[9]Categorization nouns indicate general entities, such as $cá_{fish}$ and $cây_{tree}$.



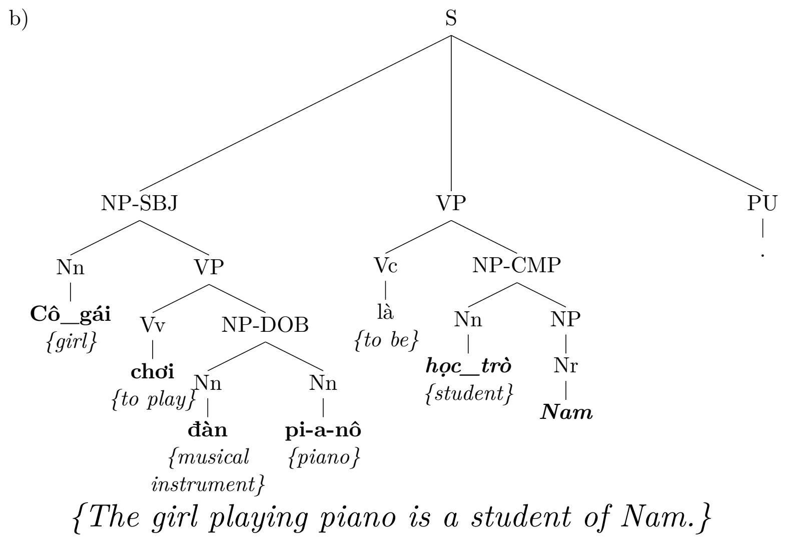


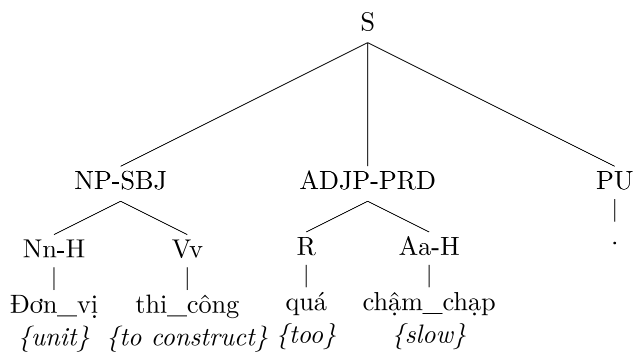
Figure 2.12: Examples illustrate flexible word orders in Vietnamese.

However, when the conjunction *thì* is used, linguists disagree about how to bracket this sentence. Diep [1] considered this sentence as a single sentence (Figure 2.13b), in which the conjunction *thì* is used to link the subject and the predicate. SCSSV [2], in contrast, considered this sentence as a subordinate compound sentence (Figure 2.13c) because the conjunction *thì* is used to link two clauses of a subordinate compound sentence.

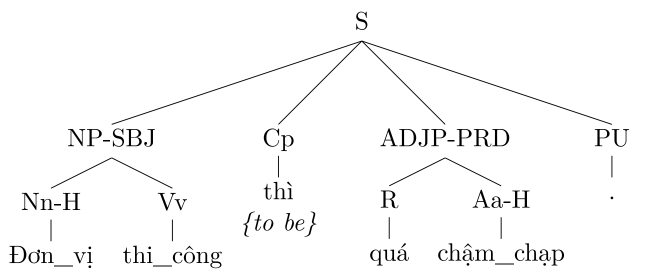
The above-mentioned characteristics of the Vietnamese language and their combinations cause many challenges for annotating Vietnamese texts. We will describe these challenges and how we solve them to build a Vietnamese treebank in Chapter 3.

Meaning: The construction unit is too slow.

a)



b)



c)

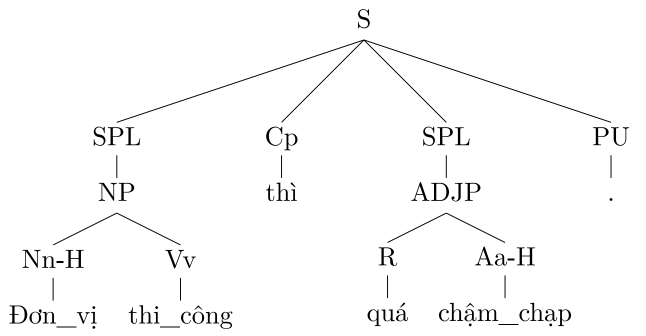


FIGURE 2.13: Examples showing different ways of bracketing a sentence in Vietnamese. Figure 2.13a shows a sentence in which the head word of the predicate is not a verb. Figure 2.13b and 2.13c present two bracketing ways by [1] and [2] respectively.

Table 2.9: VLSP treebank's POS tag set.

No.	**POS tag**	**Description**
1	N	Noun
2	Np	Proper noun
3	Nc	Classifier noun
4	Nu	Unit noun
5	V	Verb
6	A	Adjective
7	P	Pronoun
8	L	Determiner
9	M	Number
10	R	Adverb
11	E	Preposition (Subordinating conjunction)
12	C	Coordinating conjunction
13	I	Interjection
14	T	Particle
15	B	Borrowed/foreign word
16	Y	Abbreviation
17	X	Can-not-classified word

2.5.2 Vietnamese treebank

To strengthen the automatic processing of the Vietnamese language, a Vietnamese Treebank (VLSP Treebank) was built as a part of a national project, entitled "Vietnamese language and speech processing (VLSP)" [3]. This corpus contains 10,374 sentences in social and political topics, collected from an online daily newspaper, the Youth (Tuổi Trẻ) [3]. However, previous studies [19, 20] showed that the VLSP Treebank contains inconsistent and inaccurate annotations.

2.5.2.1 VLSP annotation scheme

The VLSP Treebank was annotated with three layers: word segmentation (WS), part-of-speech (POS) tagging and bracketing. The tag sets created for this treebank include 17 POS tags (except the tags for the punctuation), 16 constituency tags, 19 function tags and 3 null-element tags which are presented in Tables 2.9, 2.10, 2.11, and 2.12 respectively.

The VLSP annotation scheme includes inappropriate considerations for WS, POS and bracketing. For example,

Table 2.10: Constituency tags used in VLSP Treebank.

No.	Constituency tag	Meaning
1	NP	Noun phrase
2	VP	Verb phrase
3	AP	Adjective phrase
4	RP	Adverb phrase
5	PP	Prepositional phrase
6	QP	Quantitative phrase
7	MDP	Modal phrase
8	UCP	Unlike corordinated phrase
9	LST	List mark phrase
10	WHNP	Interrogative noun phrase
11	WHAP	Interrogative adjective phrase
12	WHRP	Interrogative adverb phrase
13	WHPP	Interrogative prepositional phrase
14	S	Statement sentence
15	SQ	Question sentence
16	SBAR	Subordinate clause

Table 2.11: Function tags used in VLSP Treebank.

No.	Function tag	Meaning
1	H	Head of phrase
2	SUB	Subject
3	DOB	Direct object
4	IOB	Indirect object
5	TPC	Topic
6	PRD	Predicate
7	LGS	Logical subject
8	EXT	Frequency or range complement
9	VOC	Vocative
10	TMP	Temporal adjunct
11	LOC	Location adjunct
12	DIR	Direction adjunct
13	MNR	Manner adjunct
14	PRP	Purpose adjunct
15	CND	Condition adjunct
16	CNC	Cnc adjunct
17	ADV	Adverbial adjunct
18	EXC	Exclamination sentence
19	CMD	Cammand sentence

Table 2.12: Null element tags used in VLSP Treebank.

No.	Null-element tag	Meaning
1	*T*	Null element (trace within sentence)
2	*E*	Null element in ellipsis phenomenon
3	*0*	Null element in complementizer

- Inappropriate word segmentation. Nguyen et al. [3] proposed a substitution transformation rule to recognize coordinate and subordinate compound words in Vietnamese as follows:
  "if A (or B) can be substituted by A', A", . . . (or B', B", . . .) of the same type, then AB is not likely to be a word. The more productive the transformation is, the less likely that AB is a word."
  According to this rule, expressions like *ăn*$_{to\ eat}$ *ở*$_{to\ live}$ *{to live/to behave/...}* and *ăn*$_{to\ eat}$ *uống*$_{to\ drink}$ *{to eat and drink/to give feasts/...}* are not compound words because the second syllable (B) can be substituted by B' having the same type (verb). Regarding the meaning of the expressions, Nguyen et al. [3] suggested a non-compositionality rule as follows
  "If the meaning of AB can not be inferred from the meanings of A and B, then AB is likely to be a word".
  According to this rule, cases like *ăn*$_{to\ eat}$ *ở*$_{to\ live}$ *{to live/to behave/...}*, *ăn*$_{to\ eat}$ *uống*$_{to\ drink}$ *{to give feasts/...}*, *quần*$_{trousers}$ *áo*$_{shirt}$ *{clothing/clothes/...}*, etc. are not compound words either because their meanings (AB) can be inferred from the meanings of A and B. However, linguists [122, 123] consider these expressions as compound words. We share the same point of view with Corp. [123] and Hoang [122] that these expressions can be treated as single words or compound words depending on their context. The reason is that although the meaning of a combination form (AB) can be inferred from the meanings of A and B, it is different from the meaning of A and B.

- Insufficient POS tag set. Nguyen et al. [3] classified words based on their combination abilities and syntactic functions. However, with 17 POS tags, their tag set cannot cover all combination abilities as well as syntactic functions of Vietnamese words. For example, they used the tag *P* to annotate all pronouns. However, personal pronouns, such as *tôi*$_{I}$ and *chúng_ta*$_{we}$, always play roles of the head words in noun phrases. Demonstrative pronouns, which are used to express space or time such as *này*$_{this}$ and *đó*$_{that}$, in contrast, can be modifiers of the head nouns in noun phrases. In addition, combination abilities of demonstrative pronouns and personal pronouns are also different. Demonstrative pronouns can be combined with nouns. For instance, in the expression *cây*$_{tree}$ *này*$_{this}$ *{this tree}*, the demonstrative

TABLE 2.13: Inconsistencies in labeling POS tags for words indicating "the whole" in the VLSP Treebank. For each pair of X_Y, X denotes POS tag and Y denotes the number of times that the word was tagged as X in VLSP treebank. Tags are separated by the comma.

No.	Word	POS tag and number of occurent
1	toàn_bộ	L_31, P_3, N_1
2	tất_cả	P_59, L_11
3	toàn_thể	L_2, P_1
4	toàn	L_5, R_24, P_9, A_2, N_2
5	cả	P_313, T_92, R_9, A_2, L_2

pronoun $này_{this}$ is a post-modifier of the noun $cây_{tree}$. However, personal pronouns cannot be combined with nouns. This example shows that personal pronouns and demonstrative pronouns should be annotated with different POS tags.

- Insufficient brackets. Nguyen et al. [3]'s bracketing guidelines did not explicitly mention any rules to label reduced relative clauses. However, when investigating their treebank, we found that reduced relative clauses like the bold expression in Figure 2.14a) are annotated as a predicate. Reduced relative clauses should be bracketed as post-modifiers of the head words that they modify (as an example in Figure 2.14b).

2.5.2.2 VLSP guildelines

VLSP has designed three sets of guidelines for their treebank. The word segmentation guidelines [124] include 11 pages, the POS tagging guidelines [125] include 7 pages, and the bracketing guidelines [126] consist of 56 pages. In comparison with the guidelines for the PTB and the Penn Chinese Treebank, these guidelines are too short to cover all complex and diversified syntactic constructions and linguistic phenomena in Vietnamese.

Another problem is that the guidelines contain overlapping and conflicting instructions through different annotation layers. For example, in the POS tagging guidelines (version 2) [125] (page 6), words, e.g., $hơn_{morethan}$, $trên_{over}$, $dưới_{lessthan}$, $gần_{almost/nearly}$, and $khoảng_{about}$, that modify a quantifier are tagged with the label *A* (adjective). However, these words are considered as an adjunct (label R) in the bracketing guidelines (version 2.2) [126] (page 17). Important constructions (e.g., reduced relative clauses and constructions including a pair of conjunctions) were not covered in the guidelines. Furthermore, the guidelines did not address several challenges of the Vietnamese language, which are the main clues of inconsistent annotations. As a result, the VLSP treebank contains many inconsistent annotations. For instance, the words $tất\_cả_{all}$ in Figure 2.15a) and 2.15b) have the same syntactic function, combination ability and

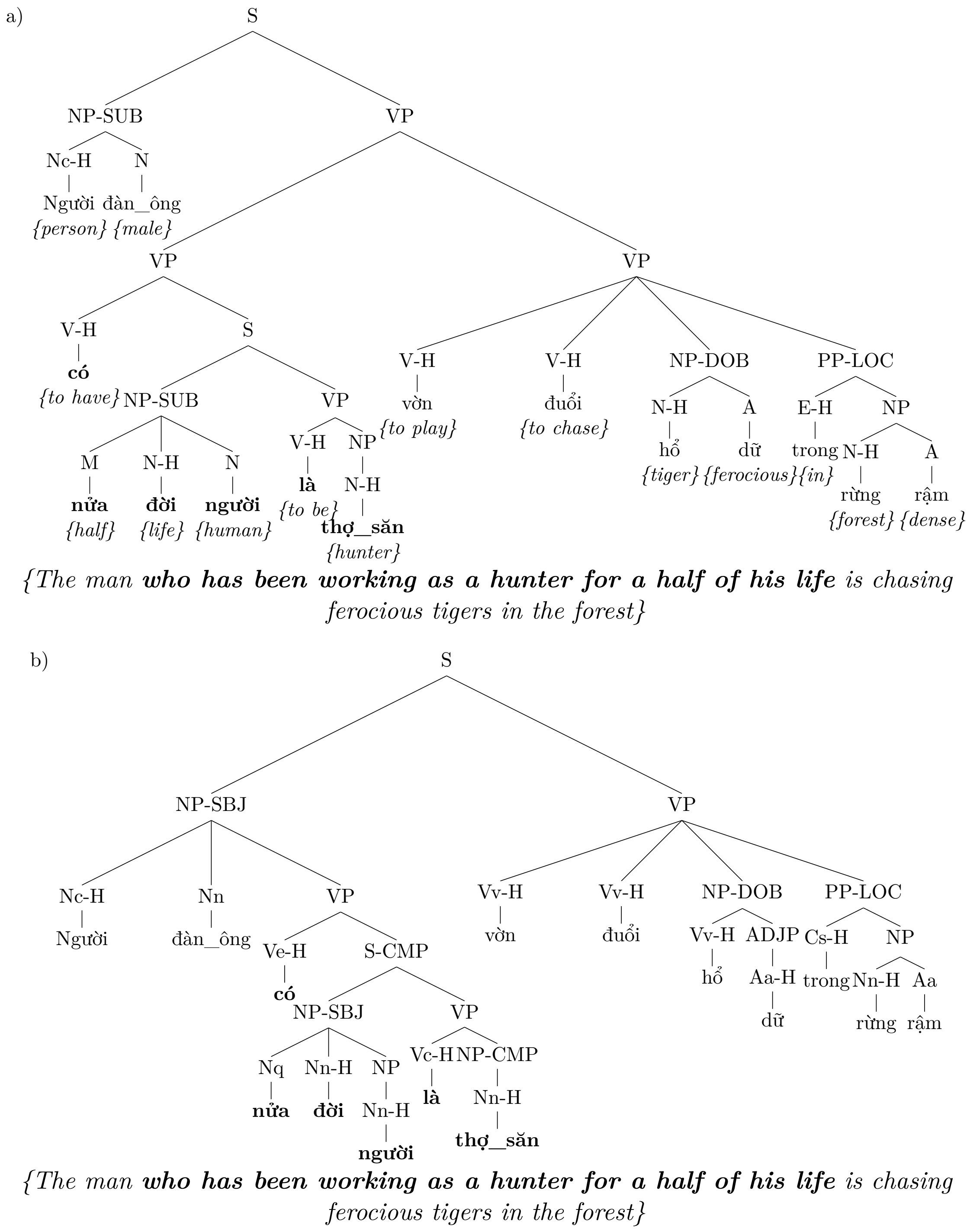


FIGURE 2.14: Examples show how the reduced relative clause was bracketed in the VLSP treebank (Figure a) and our treebank (Figure b).

meaning, but they are annotated with different POS tags, i.e., P and L in the VLSP treebank. Table 2.13 presents the inconsistency of POS labeling in the VLSP treebank for words that denote "the whole", which play a role as a pre-modifier in noun phrases. Such inconsistent annotations are possible sources for the low performance of Vietnamese language processing [21–23].

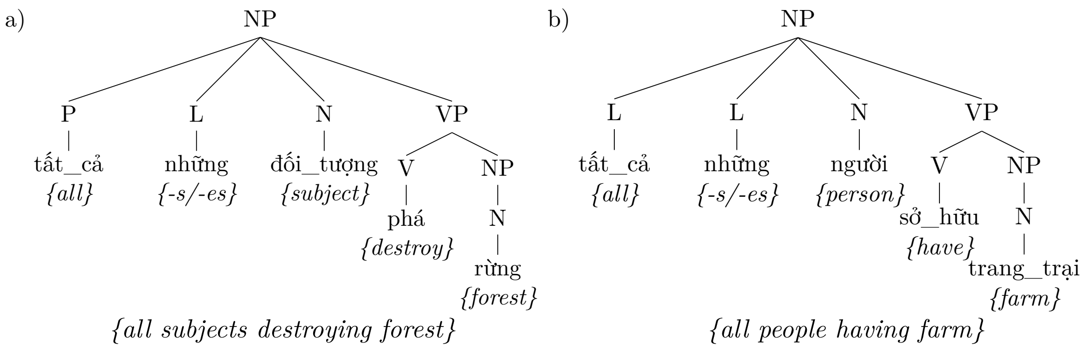


FIGURE 2.15: Examples showing that *tất_cả* was assigned to different tags in VLSP Treebank.

2.5.2.3 VLSP supporting tools

Nguyen et al. [3] implemented several tools for their treebank developemnt including preprocessing tools, an annotation tool and a tool for detecting annotation errors (detection tool). However, their tools did not completely support the annotation. For instance, their annotation tool did not fully support annotators. When annotating the VLSP treebank, annotators had to manually type all tags and brackets, which may cause typing errors. Another disadvantage comes from the detection tool as follows. Nguyen et al. [3]'s detection tool is applicable to only the word segmentation and POS tagging layers. For the bracketing errors, Nguyen et al. [3] did not study the detection method. However, there is also a problem when we applied Nguyen et al. [3]'s detection tool to the word segmentation and POS tagging layers. This tool calculated the error possibility of a word based on its surrounding words. For example, according to Nguyen et al. [3]'s method, instances of *tiền của* from rows 1 to 4 of Table 2.14 are candidates of word segmentation errors. The score for evaluating error possibility is computed based on the frequency of the surrounding words. Instances that have the maximum score are considered as incorrectly annotated. However, the surrounding words of the instances of *tiền của* are different. This results equal scores for the instances. Therefore, Nguyen et al. [3]'s method can not detect the annotation error in such cases.

2.5.3 Vietnamese parsing

Since the VLSP treebank has been built [127], a few research adapted available constituent parsers to construct a parser for Vietnamese. For example, AC. Le et al. [22] modified the Bikel's parser [128] and Le-Hong et al. [129] modified the LTAG parser developed by LORIA laboratory[10]. However, parsing performances are far lower than the

[10]http://www.loria.fr

TABLE 2.14: Examples containing instances of *tiền của*. Sentences with a star-marked number are incorrect annotated.

No.	Sentence
1	công_văn$_{document}$ xin$_{ask}$ vay$_{borrow}$ **tiền**$_{money}$ **của**$_{of}$ Công_ty$_{company}$ Tiếp_thị$_{marketing}$ *{document for borrowing money of the marketing company}*
2	mánh$_{trick}$ kiếm$_{find}$ **tiền**$_{money}$ **của**$_{of}$ đám$_{group}$ trẻ$_{child}$ *{trick of the children for finding the money}*
3	phong_trào$_{movement}$ góp$_{contribute}$ **tiền_của**$_{fortune}$ cho$_{for/to}$ công_cuộc$_{work}$ kháng_chiến$_{resistance\ war}$ *{movement of contributing fortune to the resistance war}*
4*	tập_trung$_{gather}$ **tiền**$_{money}$ **của**$_{fortune}$ mà$_{to}$ giúp$_{help}$ dân$_{people}$ *{gather the fortune to help people}*

TABLE 2.15: Results of evaluating Vietnamese treebank on MST and Malt parsers

	MST		Malt	
	UAS	LAS	UAS	LAS
With gold POS	79.08	71.66	77.37	70.49
Automatic POS	76.21	66.95	74.52	65.77

performances reported for English, Chinese, French, etc. To date, the F$_1$-scores reported for the English parser is higher than 90%. The performance of the Chinese constituency parser is also higher than 86% [78, 130]. For Vietnamese, the parsing accuracy reported by Nguyen et al. [127] is around 78% of F-score for sentences having less than 40 words. We also trained the Berkelay parser [70] on the VLSP Treebank, and the parser has achieved an F$_1$ of 71.71%.

Recently, Nguyen et al. [23] has proposed a method to convert the VLSP Treebank to the dependency structure. They also evaluated their converted treebank using MST [131] and Malt parsers [132]. The parsing performances are presented in Table 2.15. We can see that these performances are still far lower than the performances reported for English and Chinese dependency parsers (Table 2.6).

Chapter

3

Building a syntactic treebank for Vietnamese

In this research, we develop a new constituency treebank for Vietnamese. Our treebank is annotated with three layers: word segmentation, part-of-speech tagging, and bracketing. For each layer, we develop detailed annotation guidelines by presenting Vietnamese linguistic issues as well as methods to address them. We also describe approaches to control the annotation quality while ensuring a reasonable annotation speed. Specifically, we design an appropriate annotation process and an effective process to train annotators. In addition, we implement several supporting tools to improve the annotation speed and to control the consistency of the treebank. This chapter is organized as follows: we introduce our research in Section 3.1. Section 3.2 presents our methodology for creating a high-quality treebank and data preparation. Next, we discuss challenges of Vietnamese language and solutions to build annotation guidelines in Section 3.3. Then, in Section 3.4, we present our methods to keep annotation consistency and accuracy while ensure a reasonable annotation speed. Finally, we conclude our work in Section 3.5.

3.1 Introduction

Treebanks–corpora annotated with syntactic structures, are important resources for researchers in natural language processing. Treebanks provide training and testing materials for developing word segmentation tools, part-of-speech (POS) tagging tools and parsers. These tools have been applied to improve the quality of natural language processing applications. For examples, word segmentation and POS tagging are the fundamental pre-processing steps not only for syntactic parsers but also for machine translation [79, 80], text classification [81, 82] and so on. Specially, parsers were used in a wide variety of natural language processing applications. In machine translation, parsers were applied for reordering [4–6]. In question answering, Jijkoun et al. [7] improved the number of correctly answered questions by using a dependency parser; Verberne et al. [8] used syntactic information to improve why-question answering; Galitsky et al. [9] applied a syntactic parser to find structural similarity between questions and answers in order to rank candidate answers. In information retrieval, parsers were used to parse text before retrieving. Chinkina et al. [83] and Barr et al. [84] showed that part-of-speech tags and other syntactic information can be used to improve the web search results significantly.

Because of its importance, treebank has been developed for many languages. Most typically, the Penn English Treebank [10] has played a crucial role in the success of English part-of-speech taggers [11, 12] and parsers [13–15]. The methodology and annotation scheme of the Penn English Treebank have been adapted for the Penn Korean Treebank [16], Penn Chinese Treebank [17], French Treebank [18], etc.

To strengthen the automatic processing of the Vietnamese language, a Vietnamese Treebank (VLSP Treebank) was built as a part of a national project, entitled “Vietnamese language and speech processing (VLSP)” [3]. This corpus contains 10,374 sentences in social and political topics, collected from an online daily newspaper, the Youth (Tuổi Trẻ) [3]. The VLSP Treebank was annotated with three layers: word segmentation, part-of-speech (POS) tagging and bracketing. However, Nguyen et al. [19, 20] showed that the quality of VLSP Treebank, including the quality of the annotation scheme, the annotation guidelines and the annotation process, is not satisfactory and is a possible source for the low performance of Vietnamese language processing [21–23]. To alleviate these issues, we have been building a new Vietnamese Treebank with about 40,000 sentences covering 14 topics collected from the Vietnamese online newspapers. Our treebank is annotated with three layers: word segmentation, part-of-speech tagging, and bracketing, as shown in Figure 3.1[1].

[1]Underscore "_" is used to link syllables of Vietnamese multi-syllable words. English translations of Vietnamese words are given as subscripts. If a Vietnamese word does not have a translatable meaning,

Original sentence:

Nam kể về tai nạn hôm qua.
{Nam tells about the yesterday's accident.}

1. Word segmentation:

Nam kể$_{\textit{to tell}}$ *về*$_{\textit{about}}$ *tai_nạn*$_{\textit{accident}}$ *hôm_qua*$_{\textit{yesterday}}$ *.*

2. POS tagging:

Nam/Nr kể/Vv về/Cs tai_nạn/Nn hôm_qua/Nt ./PU

3. Bracketing:

```
(S
    (NP-SBJ (Nr-H Nam))
    (VP (Vv-H kể)
        (PP-DOB (Cs-H về)
             (NP (Nn-H tai_nạn)
                 (NP-TMP (Nt-H hôm_qua)))))
    (PU .))
```

FIGURE 3.1: An example to illustrate the process of treeing a Vietnamese sentence.

We have found that ensuring the annotation consistency and accuracy is one of the most important considerations in annotating treebank. However, it is not easy to keep the treebank accurate and consistent, especially for Vietnamese, due to the following reasons. In comparison with English, Vietnamese does not have word delimiters and inflectional morphemes. While similar problems also occur in Chinese [36], annotating Vietnamese may be more difficult because the modern Vietnamese writing system is based on Latin characters, which represent the pronunciation but not the meanings of words. As a result, there are many polysemous expressions, i.e., expressions having the same surface form but different interpretations, in Vietnamese. Difficulties in annotating Vietnamese are also caused by word orders. Although Vietnamese is a subject-verb-object (SVO) language like English and Chinese, its word orders are different from the others. For example, in Vietnamese, the word order in noun phrases is exactly the same as those in simple sentences, which leads to ambiguities in labelling these two types of expressions. In addition, other problems such as dropping words, conflicted definitions among linguists, etc. have also caused many challenges for annotating Vietnamese texts.

In order to keep the treebank accurate and consistent, we have carefully documented the annotation guidelines. An annotation scheme, principles to treat different linguistic phenomena, annotation examples and other relevant issues are presented as detailed as possible in the guidelines. Then, we have carefully trained annotators to ensure that they understand the guidelines and familiar with the annotation. During the annotation process, we used several supporting tools to control the annotation quality as well as keeping a reasonable annotation speed.

the subscript is blank. The translation for the Vietnamese sentence is given in curly brackets below the original text.

In this chapter, we described Vietnamese linguistic issues and our solutions to tackle them completely. We also discuss several methods to ensure that our treebank is as widely useful as possible. Beside representing our annotation scheme completely, we further compare it with that of the VLSP Treebank [3] in this chapter. Moreover, we present engineering issues to ensure annotation consistency and accuracy while still keeping a reasonable annotation speed, including an annotation process, a process used to train annotators and measure inter-annotator agreement and accuracy, and several tools to support annotation and quality control. This work also reveals several techniques to adapt methods of Chinese Penn Treebank [87–89] and English Penn Treebank [85, 86] to Vietnamese Treebank. Our research, therefore, is not only beneficial for the development of computational processing technologies for Vietnamese, a language spoken by over 90 million people, but also for similar languages such as Thai and Laos. This study also promotes the computational linguistic studies on how to transfer methods developed for a popular language, like English, to a language that has not yet intensively been studied.

The following sections are organized as follows. Section 3.2 presents our method to create a high-quality treebank and our data preparation. Next, we present challenges and solutions in building the annotation guidelines for Vietnamese in Section 3.3. Section 3.4 discusses our method to ensure the annotation quality while still remaining a reasonable annotation speed. Finally, Section 3.5 concludes this work and describes the future work.

3.2 Methods and material

3.2.1 Methodology for creating a high-quality treebank

Xue et al. [17] described four issues to ensure the annotation quality: annotation guidelines, tools to support the annotation, the quality of annotators and annotation process. Regarding the first issue, we prepared the guidelines for Vietnamese Treebank including three sets: word segmentation guidelines, POS tagging guidelines and bracketing guidelines. Our guidelines, the only document that annotators have to follow during the annotation process, cover all possible structures that can appear in the real texts. Furthermore, for each instruction, we provide as many illustrated examples as possible. By doing so, we can boost the generality of our instructions and make them easy to follow. To alleviate the conflicted word and phrases definitions in Vietnamese, we study annotation criteria that can be automatically transferred to each other. Hence, our treebank can be used under different points of view.

To build the guidelines that satisfy the above-mentioned properties, we adopt the following approaches:

- We refer to Vietnamese grammar books [1, 2] and discuss with our collaborators, who are linguistics experts, to solve the ambiguities and difficulties.
- We study the guidelines of Chinese Penn Treebank [87–89], English Penn Treebank [85, 86] and a previous Vietnamese Treebank [124–126], and adapt them to our guidelines if possible.
- During the annotation process, annotators are requested to discuss with us about constructions that they cannot annotate or feel unsure. These constructions are important clues to revise the guidelines.
- We conduct nine rounds of measurement of accuracy and inter-annotator agreement, for which both annotators annotate the same data. The inconsistencies and annotation errors found in each round are important clues to improve annotation guidelines and to train the annotators again.

Beside preparing consistent and complete annotation guidelines, using tools, such as an annotation tool, pre-processing tools and cleaning up tools, not only improves the annotation speed but also helps us to avoid annotation errors.

In this work, we recruited two Vietnamese annotators. One of them is a master student in linguistics (A1); she also joins in developing the annotation guidelines. The other (A2) is a senior student in linguistics. Both annotators are excellent in Vietnamese grammar. However, building a treebank is far more complicated than things mentioned in literature. Therefore, we have to carry out an appropriate process of training annotators to make them familiar with the annotation of the treebank.

Regarding the annotation process, we conducted the following steps:

1. We refer to the literature and discuss with linguistics experts to create a draft of the annotation guidelines.
2. After being trained with the drafted guidelines, the two annotators are required to annotate 600 texts (about 8,000 sentences). During this annotation, we receive feedbacks from the annotators and revise constructions that our guidelines do not cover or instructions that can not apply to new texts.
3. We re-train the annotators with the updated guidelines, and measure the accuracy and the inter-annotator agreement.

Table 3.1: Topics of texts in our treebank.

No.	**Topic**	**No.**	**Topic**
1	politics-society	8	health
2	information technology	9	world
3	life	10	sports
4	education	11	entertainment
5	science	12	travel
6	economic	13	law
7	military	14	life of youth

Table 3.2: Statistics of word types in our treebank.

Data	**Internal structures**	**Single words**	**Two-syllable words**	**Three-syllable words**	**Others**	**Total**
VLSP	4,210	180,634	41,773	1,504	423	228,544
Ours	1,806	95,524	34,492	1,180	255	133,257

4. The annotators edit the 600 texts and annotate the new texts. We update the guidelines for new constructions (if any).

5. Annotation results of each annotator are checked and edited by the other annotator.

6. Cleaning up the treebank: We run tools to detect annotation errors. These errors are manually edited by the annotators before we release the treebank.

3.2.2 Data preparation

Our treebank includes two sets of raw texts: VLSP and our own texts. The raw texts of VLSP, downloaded from Youth (Tuổi Trẻ)–an online daily newspaper[2], focuses on social and political topics [3]. Meanwhile, our texts, including about 30,000 sentences collected from a Vietnamese online newspaper–Thanhnien news[3], cover 14 topics as shown in Table 3.1. We selected an equal number of texts for each topic. So far, we have annotated 10,374 sentences of the VLSP texts and 5,168 sentences of our texts. Statistics of the data sets are shown in Tables 3.2[4], 3.3, and 3.4[5].

[2]http://tuoitre.com
[3]http://thanhnien.vn
[4]Types of words are presented in Section 3.3.1
[5]Internal structures are presented in Section 3.3.3.4

Table 3.3: Statistics of POS tagging in our treebank. The column "Words" presents the number of words that were annotated with one POS tag, two POS tags, or so on. The column "Instances" presents the number of instances of words in the corpus.

Types	**VLSP**		**Ours**	
	Words	**Instances**	**Words**	**Instances**
One-POS words	12,262	95,412	9,197	67,576
Two-POS words	1,119	38,127	846	29,927
Three-POS words	249	33,741	201	19,046
Four-POS words	71	24,787	55	6,747
5-POS words	24	24,109	20	6,109
6-POS words	8	6,836	6	2,093
7-POS words	2	1,321	2	368

Table 3.4: Statistics of internal structures in our corpus. The last two columns present frequency of each construction in the data sets.

Constructions	**VSLP**	**Ours**
Sv + Vietnamese original word (Nn/Vv/Aa/...)	665	608
Ncs + modifier (Vv/Aa/...)	772	597
Nc + modifier (Nn)	2,713	571
Categorization noun (Nn) + modifier	52	27
Repetition form	8	3
Total	4,210	1,806

3.3 Annotation guidelines

This section will present in detail the annotation guidelines for the three layers in our treebank including word segmentation, part-of-speech tagging, and bracketing. Specifically, in each subsection, we will describe challenges of each task, suggest solutions to alleviate the challenges, and compare our guidelines with those by VLSP.

3.3.1 Word segmentation guidelines

3.3.1.1 Word categories

Words are the most basic units of a treebank [133], and defining words is the first step in the annotation process. In this work, we follow the word definition by Di Sciullo and Williams [133]. Specifically, a word is defined as the smallest syntactic unit that conveys complete meanings. Moreover, there is no syntactic rules can be applied to a word to analyze its structure. On the basis of this definition, we classify Vietnamese words into seven categories.

- **Single words** are words that have only one free syllable[6]. A single word can be a lexical word such as *quần*$_{trousers}$ and *hát*$_{to\ sing}$ or a function word such as *sẽ*$_{will}$ and *mà*$_{that/which}$.

- **Coordinating compounds** are words that include two or more syllables, where the syllables can be single words. However, meanings of an coordinating compound are equally combined meanings of its components. For example, both *đất*$_{land}$ and *nước*$_{water}$ are single words. However, if we treat *đất nước* as an coordinating compound, it means *country.*

- **Subordinate compound words** are words that include two or more syllables, where the syllables are combined according to a main-subordinate relationship. The main syllable is a word. The other syllables are not necessarily words. For examples, *chân*$_{foot}$ and *vịt*$_{duck}$ are single words. The combination of *chân* and *vịt* will create a subordinate compound word which means *presser foot (of a sewing machine).* In the subordinate compound word *gầy guộc {skinny}*, *gầy*$_{thin/skinny}$ can stand alone as a single word. However, *guộc* is a bound syllable[7] that does not have meaning.

- **Reduplicative words** are constructed from the phonetic repetition phenomenon of syllables. Reduplicative words include two types, full word reduplication, such as *xa*$_{far}$ *xa*$_{far}$ *{in the distance}*, and partial reduplication, such as *long lanh {glistening}.*

- **Reiteration forms** look like full reduplicative words. However, the reiteration form is constructed contingently by reiterating a word many times when we want to emphasize a large amount, a high frequency, etc. For examples, *người người {everybody}* is a reiteration form created by reiterating the single word *người*$_{person}$; *tối tối {every night}* is created by reiterating the single word *tối*$_{night}$;

- **Other multi-syllable words** are constructed from syllables that do not have meaning in Vietnamese. The syllables in a word do not have phonetic or semantic association. The words can be originally Vietnamese such as *bồ nông {pelican}*, or transcribed from Chinese such as *bù nhìn {puppet}* or French such as *xà phòng {soap}*

- **Other types** that are considered in our word segmentation guidelines are proper names, names of law, resolution and agreement, phone numbers, fax, e-mail

[6]Free syllables are syllables having either lexical meaning or functional meaning. A free syllable can stand alone as a word.

[7]A bound syllable is a syllable that can not stand alone as a single word. A bound syllable does not necessarily have meaning. It always combines with other syllables or words to create a compound word.

TABLE 3.5: Examples shows how inflection in English is presented in Vietnamese.

No.	Expression	Meaning	WS	Number of words
1	nhà$_{\text{-}er}$ nghiên cứu$_{research}$	researcher	nhà nghiên_cứu	2 words
2	nghiên cứu$_{research}$ viên$_{\text{-}er}$	researcher	nghiên_cứu_viên	1 word

TABLE 3.6: Examples shows an ambiguity between reduplicative words and reiteration forms in Vietnamese.

No.	Category	Expression	Meaning	Number of words
1	Reduplicative word	tối$_{dark}$ tối$_{dark}$	slightly dark	1 word
2	Reiteration form	tối$_{night}$ tối$_{night}$	every night	1 word/2 words

addresses, websites, nick names, home addresses, idioms and locutions, numeral expressions, foreign words, abbreviations, and punctuations.

3.3.1.2 Challenges of word segmentation

Due to several characteristics of Vietnamese, including no explicit word delimiters and inflectional morphemes (Section 2.5.1.1) and conflicted word definitions (Section 2.5.1.4), the task of word segmentation becomes complicated. As our observation, there are three main ambiguities in Vietnamese word segmentation as follows.

Single words or coordinating compound words. Rows 1 and 2 in Table 3.7, for example, show that the expression *quần áo* can be segmented in two different ways: (1) two single words which mean *trousers and shirt*, and (2) a coordinating compound word which means *clothing* or *clothes*.

Single words or subordinate compound words. Table 3.5, for instance, shows that the word *nghiên cứu* means *to research*. To express *researcher*, we can add *nhà* before *nghiên cứu* or add *viên* after *nghiên cứu*. Although expressions in rows 1 and 2 have the same meaning, they are segmented in different ways by linguists [1, 2].

Reduplicative words or reiteration forms. Table 3.6 shows two examples in which a reduplicative word and a reiteration form are constructed in the same way, which is repeating a single word two times. However, the reduplicative word (row 1) is treated as a word by linguists. Meanwhile, there is little consensus in community about treating reiteration forms (row 2). For example, Hoang [122] did not consider reiteration forms as words. In contrast, SCSSV [2] treated them as words.

3.3.1.3 Policy for annotation of word segmentation

In our WS guidelines, we propose the following rules to address the difficulties in Vietnamese word segmentation:

1. If A and B[8] have different meanings and the meaning of the combination form (A_B) is different from the split form (A B), we select the form that has a more appropriate meaning for the context. Rows 1 and 2 in Table 3.7 show an expression have two different interpretations because of different word segmentations.

2. If A and B have different meanings and A_B has the same meaning as A or B, A_B is a compound word, as illustrated in row 3 of Table 3.7.

3. If A and B have the same meaning, A_B is treated as a compound word (row 4 in Table 3.7).

4. If another syllable can be inserted between A and B, A and B are words (rows 5 and 6 in Table 3.7).

5. If A or B (or both A and B) is bound syllable, A_B is treated as a compound word. For example, we can not consider *đúa* in row 7 of Table 3.7 as a single word because it is a bound syllable. *đúa* does not have any meaning in Vietnamese. Hence, it is treated as a component of the compound word.

6. For expressions composed by a categorization noun (A) and other words (B), if B indicates something different from what the expression indicates, A_B is treated as a compound word. In contrast, if B has a similar meaning to A B, A and B are treated as two words (rows 8 and 9 in Table 3.7).

7. An expression composed by one or more Sino-Vietnamese syllables and an original Vietnamese word is not treated as a word when the Sino-Vietnamese syllables are the elements used to create similar words, such as antonyms and hyponyms. For example, *nghiên_cứu*$_{to\ research}$ and *nghiên_cứu viên {researcher}* in row 10 of Table 3.7 are similar words, in which the Sino-Vietnamese syllable *viên* plays the same role as the morpheme *-er/-or* in English. Expressions like *nghiên_cứu viên* are not treated as words in our treebank. However, in cases as *sinh*$_{to\ bear}$ *viên* (row 11 of Table 3.7), we consider this expression as a word because *sinh*$_{to\ bear}$ and *sinh_viên*$_{student}$ are not similar to each other.

[8]Without loss of generalization, we assume the expression we want to segment is A B, where A and B can be syllables or words.

TABLE 3.7: Examples to illustrate the principles of word segmentation.

No.	Expression (A B)	Number of words	Meaning
1	quần$_{trousers}$ áo$_{shirt}$	1 word	clothes/clothing
2	quần$_{trousers}$ áo$_{shirt}$	2 words	trousers and shirt
3	ăn$_{to\ eat}$ nói$_{to\ speak}$	1 word	to speak
4	tìm$_{to\ search}$ kiếm$_{to\ search}$	1 word	to search
5	nồi$_{pot}$ đồng$_{copper}$	2 words	copper pot
6	nồi$_{pot}$ bằng$_{by}$ đồng$_{copper}$	3 words	copper pot
7	đen$_{black}$ đúa	1 word	black
8	cá$_{fish}$ heo$_{pig}$	1 word	dolphin
9	cá$_{fish}$ lia_thia$_{betta\ fish}$	2 words	betta fish
10	nghiên_cứu$_{to\ research}$ viên$_{\text{-}er}$	2 words	researcher
11	sinh$_{to\ bear}$ viên$_{\text{-}er}$	1 word	student
12	nhà$_{\text{-}er}$ nghiên_cứu$_{to\ research}$	2 words	researcher
13	ầm$_{boom}$ ầm$_{boom}$	1 word	rumble
14	quần$_{trousers}$ quần$_{trousers}$ áo$_{shirt}$ áo$_{shirt}$	4 words	clothes/clothing
15	quần$_{trousers}$ quần$_{trousers}$ với$_{with}$ áo$_{shirt}$ áo$_{shirt}$	5 words	clothes/clothing
16	xa$_{far}$ lơ xa$_{far}$ lắc	1 word	very far

8. Special classifier nouns[9] are treated as single words (row 12 in Table 3.7).

9. If reduplicative words or reiteration forms are constituted by two syllables, they are treated as single words (row 13 in Table 3.7). In cases they have more than two syllables, if we can insert a comma or a conjunction between the syllables, we treat each syllable as a single word (rows 14 and 15 in Table 3.7). Otherwise, the whole expression is considered as a word (row 16 in Table 3.7).

The above-mentioned rules do not necessarily conform to rules used by linguists. For example, Diep [1] treats the Sino-Vietnamese syllable *viên*$_{-er}$, as shown in row 10 in Table 3.7, as a component of compound words and treats the special classifier noun *nhà*$_{-er}$ in row 12 as a single word. We, in contrast, treat both *viên*$_{-er}$ and *nhà*$_{-er}$ as single words because we found that both words have the same grammatical function used to create similar words. Such rules make the word segmentation more consistent. However, it is worthy to note that word types that have received little consensus from linguists are labeled with internal structure tags. This makes them more flexible to be converted according to specific requirements. Details of using the internal structure tags are presented in Section 3.3.3.4

[9]Special classifier nouns, e.g., *sự*$_{-ing/-ion/-ity/...}$, *việc*$_{-ing/-ion/-ity/...}$, and *nhà*$_{-er/-or}$, are considered as classifier nouns by Nguyen et al. [3]. However, combination ability of special classifier nouns is different from that of classifier nouns such as *cái* and *con*. Classifier nouns are placed in front of categorization nouns to indicate animate things (*con cá*$_{fish}$) and inanimate things (*cái bàn*$_{table}$). While special classifier nouns play the same role as affixes in English, they can combine with a verb, such as việc$_{-ion}$ lựa_chọn$_{to\ select}$ {selection}, or an adjective such as sự$_{-ity}$ nhập_nhằng$_{ambiguous}$ {ambiguity}.

3.3.1.4 Comparison with the VLSP treebank

We and Nguyen et al. [3, 124] agree that Vietnamese has single words, coordinating compounds, and so on. However, our word definitions are different from those by Nguyen et al. [3] as follows.

In Vietnamese, reiteration forms can have two or more syllables as mentioned in Section 3.3.1.3. Vietnamese linguists consider them neither as words nor as phrases [1, 2]. However, reiteration forms are considered as words in the VLSP Treebank. In our guidelines, we consider only reiteration forms including two syllables as words because of the following reasons. Firstly, the two-syllable reiterations are stable expressions, which means that we can not insert any other words between the two syllables of the expressions. Secondly, there is little consistency for considering the reiteration forms constituted by two syllables. For example, SCSSV [2] considers *oe oe {cries of newborns}* as a full reduplicative word. Diep [1] and Corp. [123], in contrast, treat it as a reiteration form. Finally, meanings of two-syllable reiteration forms do not change when we split or combine the syllables. Regarding reiteration forms constituted by more than two syllables, we have observed that they are presented with many forms by Vietnamese people. For example, the reiteration form *quần*$_{trousers}$ *quần*$_{trousers}$ *áo*$_{shirt}$ *áo*$_{shirt}$ can be presented with various structures such as *quần*$_{trousers}$ *quần*$_{trousers}$ *với*$_{with}$ *áo*$_{shirt}$ *áo*$_{shirt}$ or *quần*$_{trousers}$ *quần*$_{trousers}$, *áo*$_{shirt}$ *áo*$_{shirt}$. We, therefore, do not treat such reiteration forms as words. We, instead, use internal structure tags to mark them so that they can be converted according to specific requirements.

For compound words, Nguyen et al. [3] suggested that if the meaning of the combination form A_B can not be inferred from the meaning of the split form A B, then A B likely to be a word (non-compositionality rule). However, as we mentioned above, there are many polysemous expressions. Hence, segmenting an expression in different ways will cause different interpretations, as illustrated in rows 1 and 2 of Table 3.7. Therefore, in our guidelines, if the meaning of the combination form A_B is different from the meaning of the split form A B, we select the form that has the most appropriate interpretation for the context (rule 1).

In addition, Nguyen et al. [3] also proposed that if A (or B) can be substituted by A', A", . . . (or B', B", . . .) of the same type, then AB is not likely to be a word. Based on this rule, expressions in Table 3.8 are not compound words because B can be replaced by B' having the same type (verb). If we apply the non-compositionality rule of Nguyen et al. [3] to these examples, they are also not compound words. The reason is that although the meanings of compound words (AB) are different from those of A and B, we can infer them from the meanings of A and B. Unlike Nguyen et al. [3],

Table 3.8: Examples show the compound words in Vietnamese that have the same previous syllable (A) but different following syllables (B).

No.	**Expression**	**Meaning**	**WS**
1	ăn$_{to\ eat}$ ở$_{to\ live}$	board and lodging, to behave, etc.	1 word/2 words
2	ăn$_{to\ eat}$ uống$_{to\ drink}$	food and drink, to give feasts, etc.	1 word/2 words

according to our first rule of word segmentation, expressions in Table 3.8 can be treated as compound words or splitted into single words based on their real context.

3.3.2 Part-of-speech tagging guidelines

In this section, we first present definitions of POS tags for Vietnamese. Then, we will discuss ambiguities in tagging POS. Next, we describe our methods to tag POS. Finally, we compare our POS tag set with that of the VLSP Treebank [3, 125].

3.3.2.1 Building a part-of-speech tag set

Designing a POS tag set can be based on meaning of words, morphological information, syntactic distribution, etc. However, to build a treebank, meaning of words has not been considered [10, 16, 17]. For inflectional languages like English or Korean, the POS tag set was designed on the basis of morphological information [10, 16]. The POS tag set of Penn Chinese Treebank was designed on the basis of syntactic distribution because Chinese has very little, if any, inflectional morphology [17].

For the Vietnamese language, we based on the combination abilities[10] and the syntactic functions[11] of words to classify them. We referred to the linguistics literature, carefully analyzed roles of words on the basis of the VLSP Treebank, and discussed with our linguistics colleagues to create a new POS tag set for Vietnamese consisting of 33 tags, as shown in Table 3.9. The detailed definition of each POS tag is available in our POS tagging guidelines.

In our POS tag set, the two-capital-character POS tags, e.g., VA, VN, and NA, are used to tag words of which there are conflicted definitions among linguists. For example, rows 1 and 2 of Table 3.10 show two different POS tags (noun and adjective) that the word

[10]Combination abilities of a word express the abilities of the word to combine with other words. For example, the demonstrative pronoun *này*$_{this/these}$ can combine with a noun (e.g., *cây*$_{tree}$ *này* $_{this}$ *{this tree}*). While personal pronouns, such as *chúng_tôi*$_{we/us}$, *họ*$_{they/them}$, etc. do not have that combination ability.

[11]Syntactic functions of a word denote the syntactic roles of the word in phrases (such as a head of a phrase or a modifier) and in sentences.

TABLE 3.9: The POS tag set designed for our treebank.

No.	POS tag	Meaning of tag	No.	POS tag	Meaning of tag
1	SV	Sino-Vietnamese syllable	17	NA	Noun-adjective
			18	Vcp	Comparative verb
2	Nc	Classifier noun	19	Vv	Other verb
3	Ncs	Special classifier noun	20	An	Ordinal number
4	Nu	Unit noun	21	Aa	Other adjective
5	Nun	Special unit noun	22	Pd	Demonstrative pronoun
6	Nw	Quantifier indicating the whole	23	Pp	Other pronoun
			24	R	Adjunct
7	Num	Number	25	Cs	Preposition or conjunction introducing a clause
8	Nq	Other quantifier			
9	Nr	Proper noun	26	Cp	Other conjunction
10	Nt	Noun of time	27	ON	Onomatopoeia
11	Nn	Other noun	28	ID	Idioms
12	Ve	Exitting verb	29	E	Exclamation word
13	Vc	Copula "là" verb	30	M	Modifier word
14	D	Directional verb	31	FW	Foreign word
15	VA	Verb-adjective	32	X	Unidentified word
16	VN	Verb-noun	33	PU	Punctuation

TABLE 3.10: Examples show different POS tags of a word in Vietnamese.

No.	Phrase	Word	POS tagging
1	tăng_cường$_{\textit{to increase}}$ an_ninh$_{\textit{security}}$ *{increase security}*	an_ninh	adjective
2	cơ_quan$_{\textit{service}}$ an_ninh$_{\textit{security}}$ *{security service}*	an_ninh	noun
3	tăng_cường$_{\textit{to increase}}$ an_ninh$_{\textit{security}}$ *{increase security}*	an_ninh	noun or adjective
4	cơ_quan$_{\textit{service}}$ an_ninh$_{\textit{security}}$ *{security service}*	an_ninh	noun or adjective

an_ninh was assigned by the Lac Viet dictionary [123]. Meanwhile, Hoang [122] claimed that *an_ninh* can be treated as a noun or an adjective in all contexts (illustrated in rows 3 and 4). In our treebank, for cases like *an_ninh*, we tagged them with double-POS tags so that they can be automatically changed to others.

3.3.2.2 Challenges of POS tagging

Polysemous expressions cause ambiguities not only in word segmentation but also in POS tagging. For example, we can interpret the word *mới* in two ways as shown in rows

1 and 2 in Table 3.11. If we treat *mới* as an adjective modifying the preceding noun *nghiên_cứu*$_{research}$, *mới* means *new*. However, if we treat it as an adjunct modifying the following verb *thực_hiện*$_{to\ conduct}$, *mới* means *recently* or *just*.

Another ambiguity is between verbs and nouns. For instance, the word *báo cáo* in Table 3.11 can be labelled as a verb, which means *to report* in row 3, and as a noun, which means *report* in row 4. We can see that such an ambiguity does not occur in inflectional languages, e.g. English, since a verbal noun will be used when a verb appears in the position of a noun. However, as mentioned above, Vietnamese does not have inflectional morphemes, which are useful for recognizing POS. Furthermore, a verb or an adjective can have the same position as a noun in Vietnamese. In fact, this ambiguity can be addressed by adding a special classifier noun or a classifier noun before the word *báo cáo*. Specifically, for the case of *báo cáo*$_{to\ report}$ (row 5), if there is a special classifier noun of *việc*[12] preceding it, it is easy for us to label *báo cáo* as a verb. Meanwhile, for the case of *báo cáo* (row 6), if we place a classifier noun of *cuốn* or *bản* before it, *báo cáo* becomes a noun whose meaning is *report*. Although those classifier nouns are useful clues for us to decide the POS tag of a word, they are usually omitted in Vietnamese sentences.

Confusion between adjectives and verbs is also common in Vietnamese POS tagging. Some linguists [1, 2] have claimed that such confusion can be tackled based on the adjuncts modifying the word, such as adjuncts indicating degree and tenses will modify adjectives and verbs, respectively. However, this approach does not necessarily work sufficiently with real texts. In practice, many verbs and adjectives in Vietnamese can be modified by the same adjunct. The adjunct indicating tense *sẽ*$_{will}$ in Table 3.11 illustrates this point. We can see that *sẽ* can modify both the adjective *đẹp*$_{beautiful}$ (row 7) and the verb *đi*$_{go}$ (row 8).

The task of POS tagging becomes more challenged because there is no consensus among linguists about methodologies to classify words. For instance, both Diep [1] and SCSSV [2] classified words based on their meanings, their combination abilities, and their syntactic functions. However, Diep [1] treated words expressing the whole, e.g., *cả*$_{all}$, *tất_cả*$_{all}$, and *toàn_bộ*$_{all}$, as pronouns, while SCSSV [2] treated them as nouns, and Hoang [122] treated *cả* as a pronoun and *tất_cả* as a noun in all contexts.

[12] *Việc* is a special classifier noun that is understood as *-ion, -ment, -ing, -ity, -ness*, etc. when it precedes verbs or adjectives. A combined expression of the special classifier noun *việc* and a verb or an adjective is understood as a noun in English. For example, *học_tập* means *to learn*, hence, we can say *việc học_tập* to express *learning*.

TABLE 3.11: Examples illustrating ambiguities in POS tagging.

No.	Word in context	Word	POS
1	Một nghiên cứu mới thực hiện tại Nhật. *{A new reseach conducted in Japan.}*	mới$_{new}$	Adjective
2	Một nghiên cứu mới thực hiện tại Nhật *{A research has just been conducted in Japan.}*	mới$_{just}$	Adjunct
3	Báo cáo rất tốt. *{Reporting is very good.}*	báo_cáo$_{to\ report}$	Verb
4	Báo cáo rất tốt. *{The report is very good.}*	báo_cáo$_{report}$	Noun
5	**Việc** báo cáo rất tốt. *{Reporting is very good.}*	báo_cáo$_{to\ report}$	Verb
6	**Cuốn/bản** báo cáo rất tốt. *{The report is very good.}*	báo_cáo$_{report}$	Noun
7	Bạn sẽ đẹp vào tối nay. *{You will be a beautiful girl tonight.}*	sẽ$_{will}$	Adjunct
8	Tôi sẽ đi vào tối nay. *{I will leave tonight.}*	sẽ$_{will}$	Adjunct

3.3.2.3 Policies for annotating part-of-speech

We assigned POS tags for words on the basis of the following criteria:

- **Combination abilities of a word**. For example, *khó_khăn* can be understood as *difficulty* or *difficult.* However, if it is a noun, it cannot combine with the adjunct *rất*$_{very}$. If it is an adjective, it cannot combine with the quantifier *những*$_{-s/-es}$.
- **Syntactic functions of a word**. For example, a quantifier indicating the whole will be tagged as Nw if it modifies a noun. It will be labeled as Pp if it is the head word of a noun phrase.
- **Meanings of a word in a sentence**. For example, considering the verb *đi*$_{go}$ and the adjective *đẹp*$_{beautiful}$ in Table 3.11, their combination abilities are identical since they are both modified by the adjunct *đã.* They also have the same syntactic function that is the head word of the predicate. However, if we take into account their meanings, they are totally different. The verb *đi* expresses an action, while the adjective *đẹp* expresses quality or property of something. This explains the reason why we should consider the meaning of words in specific sentences in addition to their combination ability and syntactic function.

There are words that give us no clues to determine their POS if we only refer to single sentences as the case of *báo_cáo* mentioned above. In such cases, annotators are required to observe all of texts surrounding the words and decide the most appropriate tag. When contexts are not helpful, we decide to tag words with all cases that they can occur. For example, given the sentence in row 1 of Table 3.11, there is no clue to determine if the word *mới* is an adjective or an adjunct. Therefore, in our corpus, we duplicate that sentence. The word *mới* in the first sentence will be annotated as an adjective. *mới* in the second sentence will be annotated as an adjunct.

It should be noted that the cases like *mới*$_{new/just}$ are ambiguous words, which are totally different from unidentified words. An ambiguous word has two or more POSs. The exact POS of its instances can be determined based on the context. Meanwhile, POS of an unidentified word can not be defined. In our treebank, unidentified words are labeled with the POS tag *X* (unknown tag). Duplicating sentences to label all possible POS tags is probably not good for training POS taggers. However, this is good for the bracketing layer because annotators can use these POS tags to annotate phrases. For example, if *mới* is an adjective (i.e., means *new*), it will be bracketed as a post-modifier of the noun *nghiên_cứu*$_{research}$. Meanwhile, if *mới* is an adjunct (i.e., means *recently* or *just*), it is annotated as a pre-modifier of the verb *thực_hiện*$_{to\ conduct}$. Although our treebank contains duplicated sentences, we have documented all duplicated cases (including file names, sentence no., and reasons of duplication). These information can be used to pre-process the treebank before using it.

3.3.2.4 Comparison with the VLSP Treebank

Our criteria to classify words are the same as those of the VLSP Treebank [3], i.e., both of us based on the combination abilities and the syntactic functions of words. The POS tag set designed by Nguyen et al. [3] cannot cover all combination abilities as well as syntactic functions of Vietnamese words. For example, they used the tag *P* to annotate all pronouns. However, pronouns should be fine-grained according to their specific functions. Specifically, demonstrative pronouns used to express space or time, such as *này*$_{this}$ and *đó*$_{that}$, should be distinguished from personal pronouns, e.g., *tôi*$_{I}$ and *chúng_ta*$_{we}$ because their syntactic functions are different. Demonstrative pronouns can be modifiers of head nouns in noun phrases. Meanwhile, personal pronouns always play roles of the head words in noun phrases. These two types of pronouns also have different combination abilities. Demonstrative pronouns can be combined with nouns, e.g., *cây*$_{tree}$ *này*$_{this}$ *{this tree}*, while personal pronouns cannot.

By creating 33 POS tags, we not only solve the above-mentioned controversial cases but also better label roles of words in phrases or sentences. For instance, demonstrative pronouns modifying the head word of a noun phase will be labeled as *Pd*, and other

TABLE 3.12: Examples of polysemous words in Vietnamese.

No.	**Sentence**	**Word**	**Combination ability and syntactic function**	**Our POS tag**	**VLSP Treebank's POS tag**
1	Tôi là quần áo rất nhanh *{I iron clothes very fast}*	là$_{to\ iron}$	*là$_{to\ iron}$* can be modified by a direct object and a modifier	Vv	V
2	Tôi là sinh viên giỏi *{I am a good student}*	là$_{to\ be}$	Copula *là$_{to\ be}$* is used to express the equivalent between two objects	Vc	V
3	Tôi với bạn có cùng họ *{You and I have the same last name}*	với$_{and}$	*với$_{and}$* is a coordinating conjunction	Cp	C
4	Tôi sẽ đi với bạn *{I will go with you}*	với$_{with}$	*với$_{with}$* is a subordinate conjunction	Cs	C
5	Nhà ấy rất giàu *{That family is very rich}*	ấy$_{that}$	*ấy$_{that}$* is a demonstrative pronoun modifying the head noun *nhà$_{family}$*	Pd	P
6	Ấy đang làm gì vậy? *{What are you doing?}*	ấy$_{you}$	*ấy$_{you}$* is a personal pronoun which is always a head word of a noun phrase	Pp	P

pronouns that are head words of noun phrases will be annotated with a *Pp* label. Such annotations are better than those by Nguyen et al. [3] since they do not produce any conflicts.

Nguyen et al. [3] claimed that tags can be splitted into more specific sub-tags easily based on lexical information. However, as presented above, polysemous expressions can cause difficulties in splitting or tagging POS for Vietnamese words. Table 3.12 shows several examples of polysemous words, to which we should assign different POS tags as their combination abilities and syntactic functions are different.

3.3.3 Bracketing guidelines

In this section, we first describe an annotation scheme for the Vietnamese Treebank. Then, we discuss challenges of bracketing the Vietnamese sentences. We next present our methods to tackle the challenges. After that, we describe how internal structures are annotated in our treebank. Finally, we compare our bracketing guidelines with those of the VLSP Treebank [3, 126].

3.3.3.1 Representation scheme

Following the English Penn Treebank [10], phrases in our treebank are bracketed on the basis of the syntactic relationships between constituents. These relationships are expressed by constituency tags and functional tags. Additionally, we also use null elements to mark ellipses and reference indices to mark syntactic movements. Similar to the VLSP Treebank [3], we also use the functional tag H to tag head words of phrases. Details of each tag type are explained below.

TABLE 3.13: Our constituency tags.

No.	Tag	Explanation
1	NP	Noun phrase
2	QP	Quantitative phrase
3	VP	Verb phrase
4	ADJP	Adjective phrase
5	PP	Prepositional phrase
6	RP	Adjunct phrase
7	CONJP	Conjunction phrase
8	UCP	Unlike coordinated phrase
9	QNP	Questioning noun phrase
10	QRP	Questioning adjunct phrase
11	QPP	Questioning prepositional phrase
12	QADJP	Questioning adjective phrase
13	MDP	Modal phrase
14	S	Simple/compound declarative sentence
15	SQ	Question
16	SPL	Special sentence
17	SBAR	Subordinate clause
18	XP	Unknown phrase

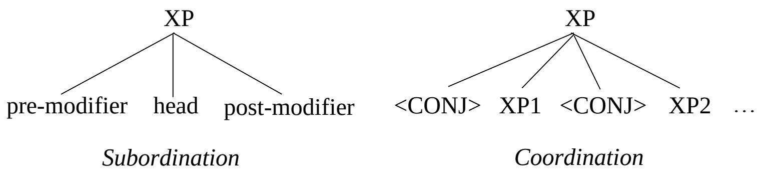
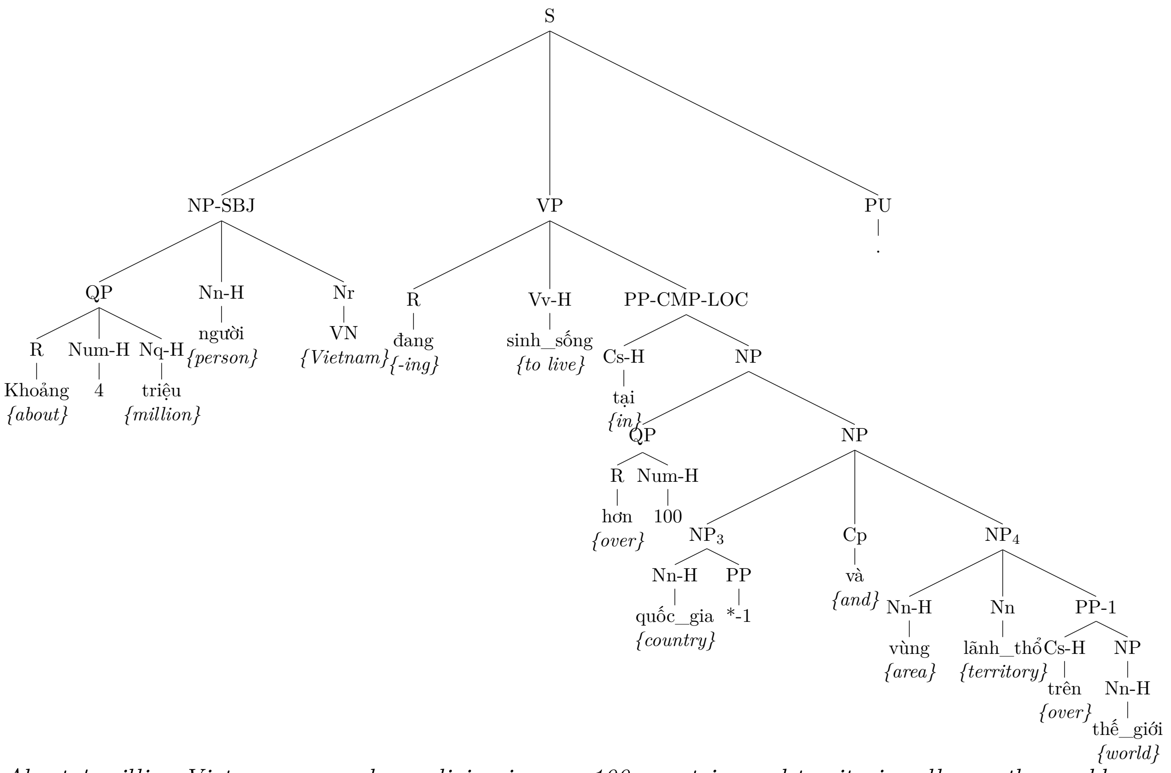


FIGURE 3.2: Grammatical relations used in our bracketing guidelines.

Constituency tags. The 18 constituency tags showed in Table 3.13 indicate syntactic categories of phrases. In our treebank, phrases are bracketed on the basis of two grammatical relations: subordination and coordination. Their structures are presented in Figure 3.2. In subordinate relationships, a head can include one or more terminal nodes. A pre-modifier and a post-modifier can include terminal and non-terminal nodes. In coordinate relationships, *CONJ* can be dropped. If *CONJ* is used, it can be a terminal or a non-terminal node. *XP1* and *XP2* are terminal or non-terminal nodes, and can have the same type or different types. Figure 3.3 shows a tree that includes both grammatical relations. The relationship between NP_3 and NP_4 is coordination. *VP* expresses a modification relation in which *sinh_sống*$_{to\ live}$ is the head word, the adjunct *đang*[13] is a pre-modifier, and the prepositional phrase *PP*, which is annotated with two functional tags *CMP* and *LOC*, is the post-modifier.

[13] *đang* is an adjunct used to express the continuation. For example, *sinh_sống* means *to live*. To express *to be living*, we use *đang sinh_sống*.



About 4 million Vietnamese people are living in over 100 countries and territories all over the world.

FIGURE 3.3: An example showing two relations used in our treebank.

TABLE 3.14: Our functional tags.

No.	Tag	Explanation	No.	Tag	Explanation
1	H	Head of phrase	12	TMP	Temporal
2	SBJ	Subject	13	LOC	Locative
3	LGS	Logical subject	14	MNR	Manner
4	PRD	Predicate that is not VP	15	PRP	Purpose or reason
5	DOB	Direct object	16	CND	Condition
6	IOB	Indirect object	17	CNC	Concessions
7	CMP	Complement	18	ADV	Adverbial
8	TPC	Topicalized	19	HLN	Headline
9	MDP	Modal phrase	20	TTL	Title
10	VOC	Vocative	21	EXC	Exclamative sentence
11	PRN	Parenthetical	22	CMD	Imperative sentence

Functional tags. Table 3.14 shows 22 functional tags designed for our treebank. These tags give additional information about syntactic function of phrases. For example, in Figure 3.3, *SBJ* following *NP* indicates that *NP* is the subject of the sentence. In addition, our functional tags also distinguish among different types of adverbial phrases. For example, in Figure 3.3, two functional tags, *CMP* and *LOC* following *PP* indicate that the *PP* not only plays a role of a complement of the verb *sinh_sống*$_{\textit{to live}}$, but it is also an adverbial phrase indicating location.

TABLE 3.15: Our null elements.

No.	**Tag**	**Explanation**
1	*T*	Trace of phrase movement
2	*E*	Ellipses without trace for phrase
3	*	Ellipses with trace for phrase
4	*0*	Null complementizer
5	*P*	Null passive verb
6	*H*	Ellipses with trace for head word
7	*D*	Ellipses with trace for direct object of reduced relative clause

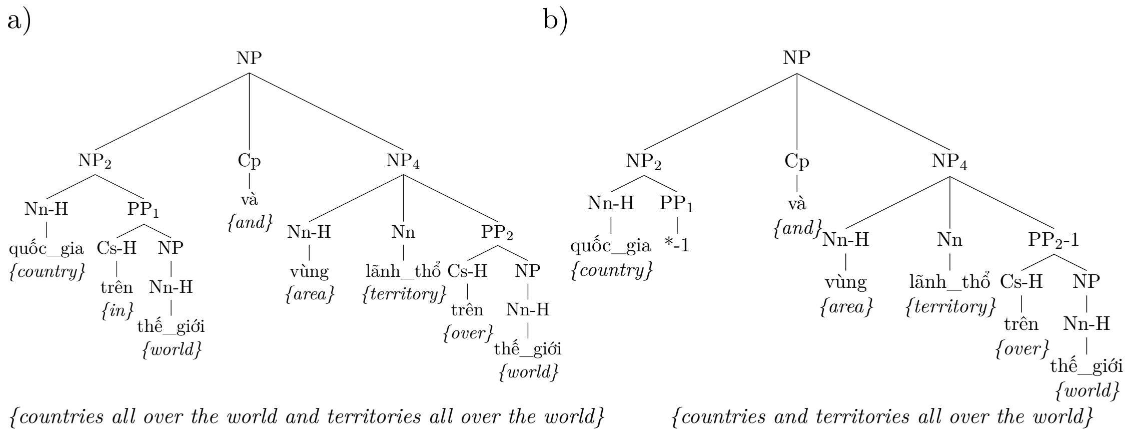


FIGURE 3.4: An example showing the use of null elements in our treebank.

Null elements. Our null elements include seven categories as showed in Table 3.15. They are used to mark ellipses. This marking is to understand the syntactic structure of a sentence so that we can understand the meaning of the sentence. For example, Figure 3.4a shows a complete structure of a noun phrase in Vietnamese, in which post-modifiers of NP_2 and NP_4 include the same prepositional phrases. In some cases, PP_1 is dropped. Therefore, we mark PP_1 with the label *, and mark the same reference indices for * and PP_2 (Figure 3.4b).

3.3.3.2 Challenges of bracketing

The four characteristics of the Vietnamese language cause many confusing expressions in bracketing Vietnamese texts. Specifically, expressions that have the same structure or the same surface form can be bracketed in different ways. Follows are some common ambiguous cases in Vietnamese bracketing.

Noun phrases or simple sentences. In Vietnamese, both noun phrases and simple sentences have the same word orders. For example, for the expression $chim_{bird}$ $hót_{to\ sing}$ in rows 1 and 2 of Table 3.16, the word order is the verb follows the noun. However,

Table 3.16: Examples illustrating ambiguities between noun phrase and simple sentences.

No.	**Bracketing**	**Word order**	**Example**
1	S	Verb follows noun	Tôi$_{I}$ đã$_{\textit{-ed}}$ tìm$_{\textit{to find}}$ ra nơi$_{\textit{place}}$ **chim**$_{\textbf{bird}}$ **hót**$_{\textbf{to sing}}$ *{I have found the place where the birds sing}*
2	NP		**Chim**$_{\textbf{bird}}$ **hót**$_{\textbf{to sing}}$ là$_{\textit{to be}}$ chim$_{\textit{bird}}$ sơn_ca$_{\textit{nightingale}}$ *{The singing birds are nightingale}/* *{The birds that are singing are nightingale}*
3	S	Adjective follows noun	Cây$_{\textit{tree}}$ này$_{\textit{this}}$ **lá**$_{\textbf{leaf}}$ **vàng**$_{\textbf{yellow}}$ *{This tree's leaves are yellow/This tree's leaves have been yellow}*
4	NP		Mùa$_{\textit{season}}$ thu$_{\textit{Autumn}}$ có$_{\textit{to have}}$ **lá**$_{\textbf{leaf}}$ **vàng**$_{\textbf{yellow}}$ *{There are yellow leaves in Autumn}*

the expression in row 1 is labelled as a simple sentence while it is a noun phrase in row 2. This ambiguity occurs due to the following reasons: (1) verbs are not marked with morphological information, which is an important clue to distinguish between modifying verbs and predicative verb phrases; (2) when a verb follows a noun, it can be a modifier, a relative clause, or a predicate. In Vietnamese, a relative clause is placed after a subordinate conjunction *mà*$_{which/that/...}$, *rằng*$_{which/that/...}$, etc., but such subordinate conjunctions are frequently dropped.

The same situation occurs with the expression *lá*$_{leaf}$ *vàng*$_{yellow}$ in rows 3 and 4, in which the adjective follows the noun. In this case, the expression in row 3 is labeled as a simple sentence while it is a noun phrase in row 4.

Separated constituents or a simple sentence. There are expressions that have the same structure, but should be bracketed in different ways. For instance, Figure 3.5 shows verb phrases having the same structure of *Vv NP VP*. However, the expressions *NP VP* modifying the head verbs can be bracketed as simple sentences (Figures a and c) or annotated separately (Figures b and d). We can see in the figure that such an ambiguity does not occur in English, despite the fact that the English and Vietnamese phrases convey the same content. One of the reasons for this is that in English, we have clues, e.g., inflexions, to distinguish between subjects and objects, and between predicative verb phrases and modifying verb phrases. In contrast, such clues are not presented in Vietnamese. Subordinate conjunctions introducing a relative clause, as mentioned above, is frequently dropped.

Ambiguities caused by flexible word orders. The flexible word orders in Vietnamese make the bracketing task even more difficult. Rows 1, 2, and 3 of Table 3.17 present common word orders, in which lexical words follow head words in phrases. On the other hand, in many phrases, the word order is reversed, i.e., modifying lexical words are placed in front of the head words (as presented in row 4 of Table 3.17). There are

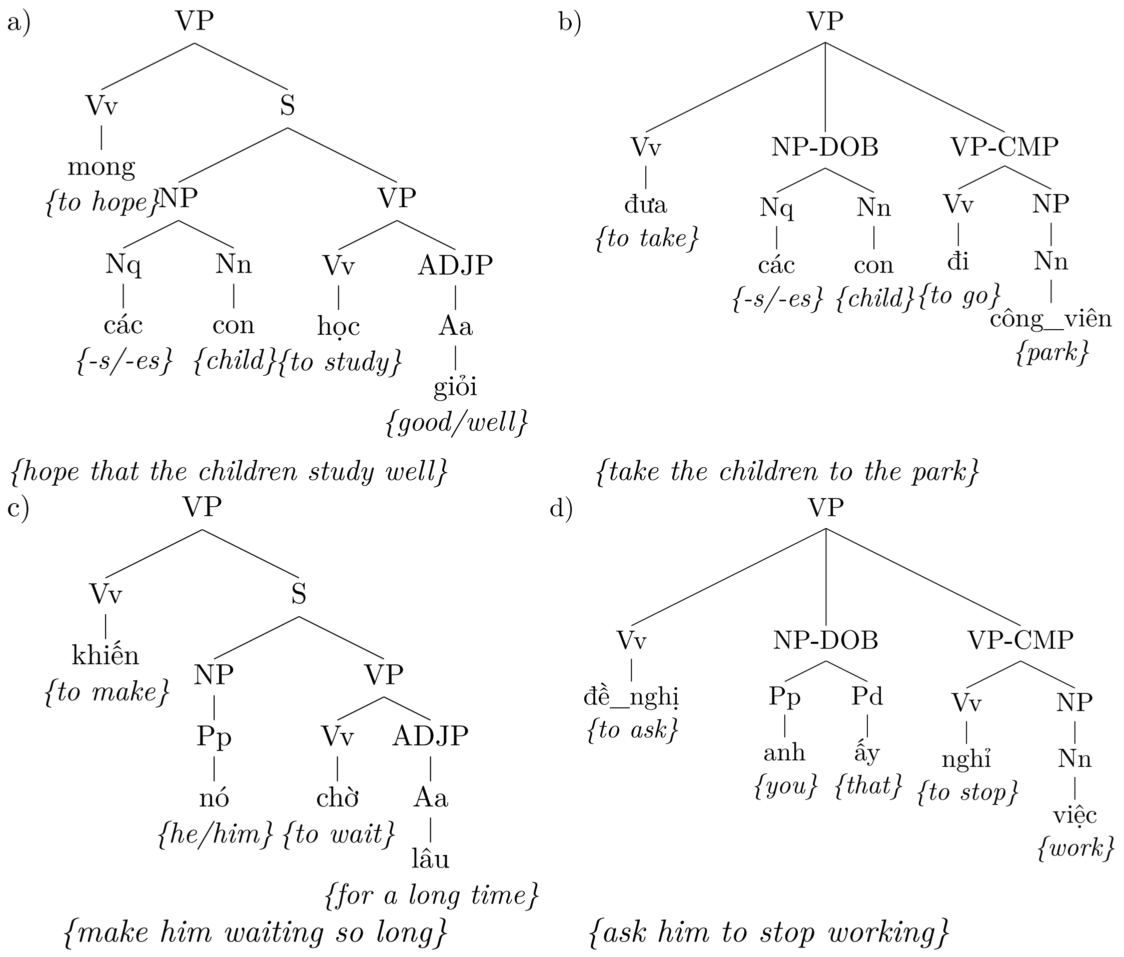


FIGURE 3.5: Expressions that have the same structure, but should be bracketed in different ways.

also cases of which the word orders are opposite to each other, but the interpretation is the same, as illustrated in Table 3.18. Such flexible word orders cause ambiguities in distinguishing phrases that have the same POS sequences. Table 3.19 presents examples of such ambiguities between a noun phrase and an adjective phrase (rows 1 and 2), a verb phrase and an adjective phrase (rows 3 and 4), and a noun phrase and a verb phrase (rows 5 and 6).

Flexible word orders can be generalised to flexible phrase orders, which also cause bracketing ambiguities. We can see from Figures 3.6a and 3.6b that although we invert the noun phrase and the prepositional phrase, interpretations of the verb phrases are the same. However, it is worth to note that when the prepositional phrase is placed at the end of the verb phrase, we can bracket the verb phrase in two ways: (1) the noun phrase and the prepositional phrase are two separated phrases (Figure 3.6a), or (2) the prepositional phrase is a post-modifier of the previous noun (Figure 3.6c).

Ellipses. Ellipses frequently occur in Vietnamese texts and cause confusion for not only

Table 3.17: Examples of noun phrases, verb phrases and adjective phrase illustrating fixed word order in Vietnamese.

No.	Phrase	POS sequence	Example
1	NP	Nn Aa	người$_{person}$ đẹp$_{beautiful}$ *{beautiful person}*
2	VP	Vv Aa	chạy$_{to\ run}$ nhanh$_{quick}$ *{run quickly}*
3	ADJP	Aa Nn	đẹp$_{beautiful}$ người$_{person}$ *{good looking}*
4	NP	Aa Nn	nhiều$_{a\ lot}$ kinh_nghiệm$_{experience}$ *{a lot of experiences}*

Table 3.18: Several examples illustrating flexible word order in Vietnamese.

No.	Phrase	POS sequence	Example
1	VP	Aa Vv	lầm_lũi$_{silent}$ sống$_{to\ live}$ *{live silently}*
2	VP	Vv Aa	sống$_{to\ live}$ lầm_lũi$_{silently}$ *{live silently}*
3	VP	Aa Vv	trực_tiếp$_{direct}$ báo_cáo$_{to\ report}$ *{report directly}*
4	VP	Vv Aa	báo_cáo$_{to\ report}$ trực_tiếp$_{direct}$ *{report directly}*

Table 3.19: Expressions illustrating ambiguities in differentiating phrases that have the same POS sequences.

No.	Pair of phrases	POS sequense	Example	Treatment
1	NP and ADJP	Aa Nn	nhiều$_{a\ lot}$ kinh_nghiệm$_{experience}$ *{a lot of experiences}*	NP
2		Aa Nn	nhiều$_{a\ lot}$ kinh_nghiệm$_{experience}$ *{experienced}*	ADJP
3	VP and ADJP	Aa Vv	lầm_lũi$_{silently}$ sống$_{to\ live}$ *{live silently}*	VP
4		Aa Vv	ít$_{little}$ học$_{learn}$ *{unlearned}*	ADJP
5	VP and NP	Vv Nn	xuất_phát$_{to\ start}$ điểm$_{point}$ *{starting point}*	NP
6		Vv Nn	học$_{to\ study}$ toán$_{math}$ *{study math}*	VP

annotators but also linguists. Diep [1] treated the sentence in Figure 3.7 as a single sentence (Figure 3.7b), in which the expression before the comma is a subordinate component expressing the manner. However, this sentence can be understood as a subordinate compound sentence [2], in which the subject of the first clause is dropped because it is identical to that of the second clause (Figure 3.7c).

3.3.3.3 Policies for annotating brackets

In general, we have addressed ambiguities of the bracketing layer by checking expressions' meanings in their contexts and their meanings after inserting, replacing, and reordering their words. For some cases, we have to remove words in expressions to determine how the words are important to the expressions. Details of applying these criteria to real texts are described below.

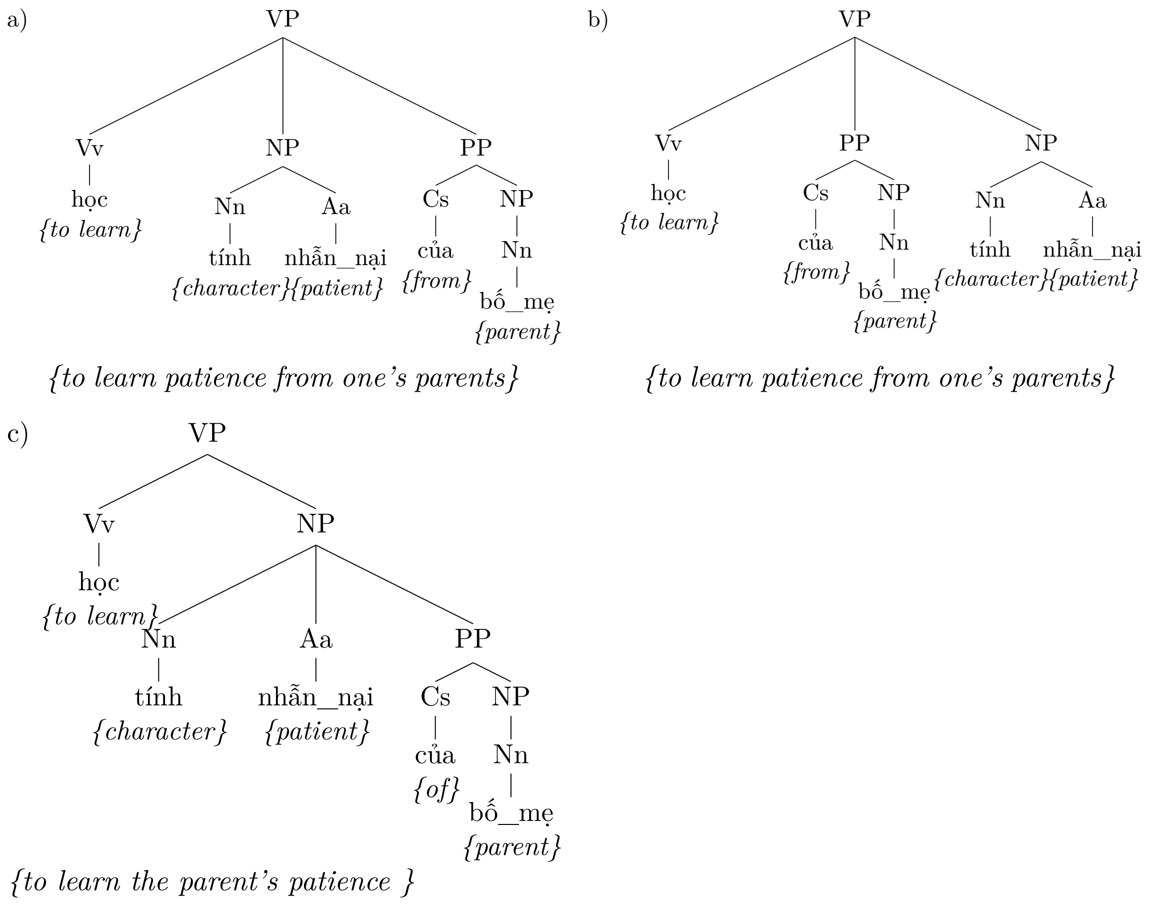


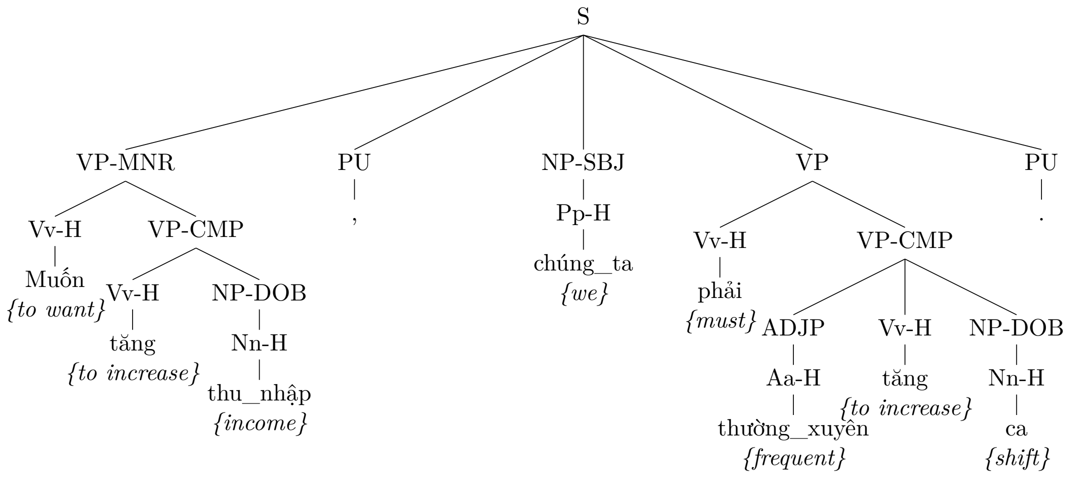
FIGURE 3.6: An example illustrating flexible word orders in Vietnamese.

Distinguish between noun phrases and simple sentences. For cases that an expression is constituted by a noun preceding a verb (as rows 1 and 2 of Table 3.16), we will insert a subordinate conjunction or a demonstrative pronoun into the expression and check its meaning to decide the appropriate label. For instance, we can insert the subordinate conjunction *mà*$_{that/which}$ before the bold expression in row 1 of Table 3.16 (as demonstrated in row 1 of Table 3.20) without changing the original interpretation. This indicates that the bold expression in this sentence is a relative clause that should be bracketed as a simple sentence (with the S label). Meanwhile, the bold expression in row 2 of Table 3.16 should be annotated as a noun phrase because we can insert the demonstrative pronoun *đó*$_{that/those}$[14] after the verb *hót*$_{to\ sing}$ (as shown in row 2 of Table 3.20). This insertion reveals that the verb *hót {singing}* (like the demonstrative pronoun *đó*$_{that/those}$) is a modifier of the head noun *chim*$_{bird}$. On the other hand, we can also insert the subordinate conjunction *mà*$_{that/which}$ before the verb *hót*$_{to\ sing}$ in row 2 of Table 3.16 (as demonstrated in row 3 of Table 3.20). As a result, *hót*$_{to\ sing}$ can be recognized as a reduced relative clause modifying the previous noun *chim*$_{bird}$. In this

[14]Demonstrative pronouns in Vietnamese, such as *này*$_{this/these}$, *đó*$_{that/those}$, *ấy*$_{that/those}$, and *kia*$_{that/those}$ play the same roles as demonstrative adjectives in English, but they are placed at the end of noun phrases.

a) Original text:
Muốn tăng thu nhập, chúng ta phải thường xuyên tăng ca.
{To increase the income, we must work overtime frequently.}

b) Bracketed as a single sentence



c) Bracketed as a compound sentence

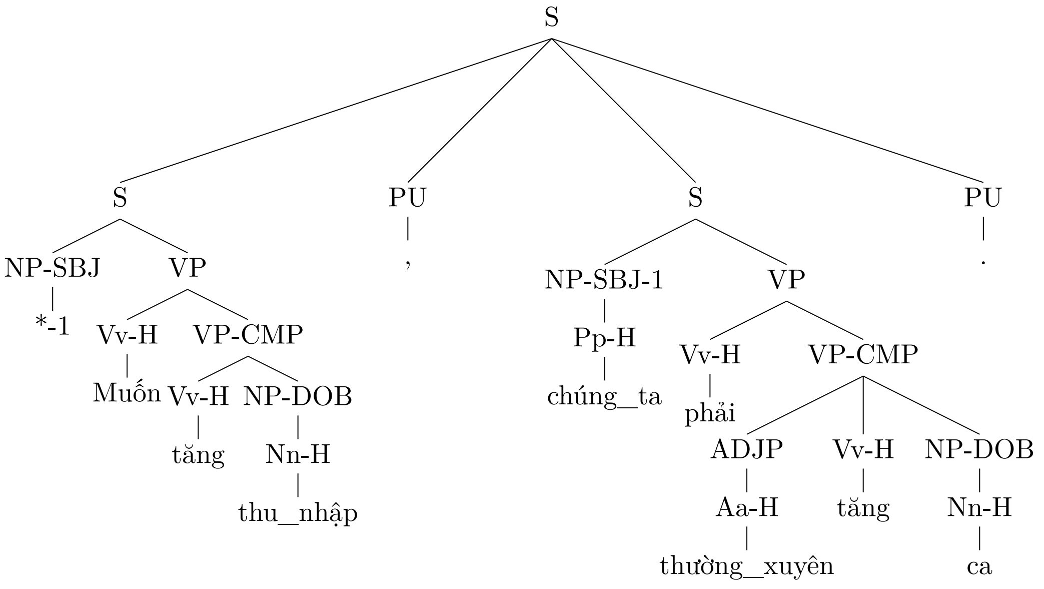


FIGURE 3.7: An example of confusion caused by ellipsis.

example, *hót*$_{to\ sing}$ can be considered as either a modifying verb phrase or a reduced relative clause. In our treebank, we decided to annotate such phrases as modifying phrases to the previous noun. There are two reasons for our selection. Firstly, meaning of sentences that contain such phrases does not change when we treat the phrases like that. Secondly, distinguishing between modifying verb phrases and reduced relative clauses is not an easy task (as illustrated in the above-mentioned example). Bracketing them in the same structure will ensure consistent annotations. For cases like the expression in row 4 of Table 3.20, we can label the verb *hoạt_động*$_{to\ operate}$ as a modifier of the

TABLE 3.20: Examples of bracketing expressions that have the same structure.

No.	**Ambiguity**	**Phrase in context**	**Considered expression**	**Treatment**	**Reason of treatment**
1	S or NP	Tôi đã tìm ra nơi **mà** chim hót *{I have found the place where the birds sing}*	chim$_{bird}$ hót$_{to\ sing}$ *{the birds sing}*	S	We can insert the subordinate conjunction *mà*$_{that}$ before the considered expression
2	S or NP	Chim hót **đó** là chim sơn ca *{Those singing birds are nightingale}*	chim$_{bird}$ hót$_{to\ sing}$ *{the singing birds}*	NP	We can insert the demonstrative pronoun *đó*$_{that/those}$ after the verb *hót*
3	S or NP	Chim **mà** hót là chim sơn ca *{The birds that are singing are nightingale}*	chim$_{bird}$ hót$_{to\ sing}$ *{the birds that are singing}*	NP	We can insert the subordinate conjunction *mà*$_{that}$ before the verb *hót*
4	S or NP	Nguyên tắc hoạt động **của máy này** tạo ra công suất rất lớn *{The operating principle of this machine creates a big capacity}*	Nguyên_tắc$_{principle}$ hoạt_động$_{to\ oprerate}$ *{operating principle}*	NP	A head noun can be modified by a possessive prepositional phrase that its head preposition is *của*$_{of}$
5	S or NP	Cây này lá **đã** vàng *{This tree's leaves have been yellow}*	lá$_{leaf}$ vàng$_{yellow}$ *{the leaf has been yellow}*	S	We can insert the adjunct indicating tense *đã*$_{-ed}$ before the adjective *vàng*$_{yellow}$
6	S or NP	Mùa thu có lá vàng *{There are yellow leaves in Autumn}*	lá$_{leaf}$ vàng$_{yellow}$ *{yellow leaves}*	NP	We can not add an adjunct indicating tense as a modifier of the adjective *vàng*$_{yellow}$
7	NP or ADJP	Tôi có nhiều kinh nghiệm. *{I have a lot of experiences.}*	nhiều$_{alot}$ kinh_nghiệm$_{experience}$ *{a lot of experiences}*	NP	We can replace *nhiều* by the other quantifiers such as *2* or *vài*$_{several}$
8	NP or ADJP	Tôi là người nhiều kinh nghiệm. *{I am an experienced person.}*	nhiều$_{alot}$ kinh_nghiệm$_{experience}$ *{exprerienced}*	ADJP	Phrase *nhiều kinh_nghiệm* modifies the noun *người* $_{person}$ about quality.
9	VP or ADJP	Anh ấy lầm lũi sống. *{He lives silently.}* Anh ấy sống lầm lũi. *{He lives silently.}*	lầm_lũi$_{silently}$ sống$_{live}$ *{lives silently}* sống$_{live}$ lầm_lũi$_{silently}$ *{lives silently}*	VP VP	Inverting the adjective *lầm_lũi* and the verb *sống* does not cause meaning change.
10	VP or ADJP	Tôi học ít. *{I learn little.}* Tôi ít học. *{I am unlearned}*	học$_{learn}$ ít$_{little}$ *{learn little}* ít$_{little}$ học$_{learn}$ *{unlearned}*	VP ADJP	Inverting the adjective *ít*$_{little}$ and the verb *học*$_{learn}$ causes meaning change.

noun *nguyên_tắc*$_{principle}$ by adding the possessive prepositional phrase *của*$_{of}$ *máy*$_{machine}$ *này*$_{this}$ *{of this machine}* after *hoạt_động*$_{to\ operate}$. Since the possessive prepositional phrase can be a modifier of the noun *nguyên_tắc*$_{principle}$, the verb *hoạt_động*$_{to\ operate}$ also plays the same role.

Regarding expressions that are constituted by an adjective preceding a noun (bold expressions in rows 3 and 4 of Table 3.16), if we can insert an adjunct indicating tense as a pre-modifier of the adjective, the expressions should be bracketed with an S label (example 5 in Table 3.20). In contrast, the expressions will be treated as noun phrases (example 6 in Table 3.20).

Distinguish between noun phrases and adjective phrases. For a noun phrase and an adjective phrase that have the same structure as shown in rows 1 and 2 of Table 3.19, to determine the phrase tags, we can base on the criterion of word replacement or meaning of the expression in the context. Rows 7 and 8 of Table 3.20 show the expressions *nhiều kinh nghiệm* presented in two contexts. In row 7, the word *nhiều*$_{many/a\ lot\ of}$ modifies the noun *kinh nghiệm* in terms of quantity. We can replace *nhiều* by other quantifiers, such as *2* or *vài*$_{several}$. Therefore, the expression *nhiều kinh nghiệm {a lot of experiences}* in this case should be bracketed as a noun phrase. In contrast, the expression *nhiều kinh nghiệm {experienced}* in row 8 of Table 3.20 should be bracketed

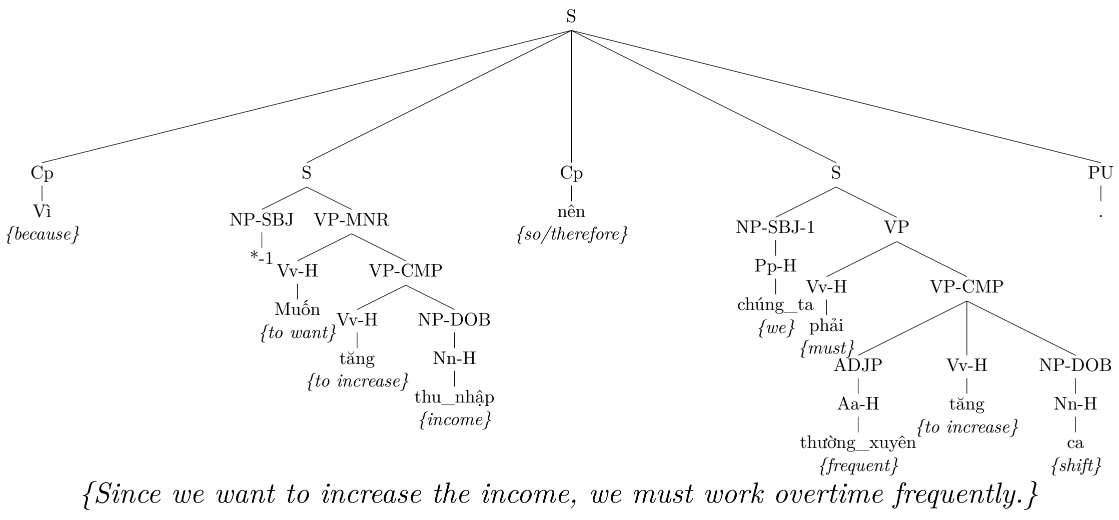


FIGURE 3.8: An example about the elliptical compound sentence in our treebank.

with an *ADJP* because it modifies the noun *người$_{person}$* in terms of quality and we can not replace *nhiều* in this context by a number or a quantifier, e.g., *vài$_{several}$*, *các$_{-s/-es}$* or *một ít$_{alittle}$*.

Distinguish between verb phrases and adjective phrases. For a verb phrase and an adjective phrase that have the same structure, if its words can be inverted without changing the meaning, the phrase is annotated with a *VP* label (see example 9 in Table 3.20). Otherwise, if the inversion causes another interpretation, the phrase are bracketed on the basic of the common word orders in Vietnamese, the modifying lexical word follows the head word (see example 10 in Table 3.20).

Ellipses. Since the meaning of a sentence does not change when we annotate it as a simple sentence or a compound sentence, we select a consistent structure for each type of sentences. For example, sentences like the one in Figure 3.7 is annotated as a simple sentence (Figure 3.7b). However, we bracket sentences like the one in Figure 3.8 as a compound sentence because this sentence is constructed with a conjunction pair, *vì$_{because}$ ... nên$_{so/therefore}$*. In our guidelines, conjunction pairs *vì$_{because}$ ... nên$_{so/therefore}$*, *không những ... mà còn {not only ... but also}*, *tuy$_{although}$ ... nhưng$_{but}$*, etc. are used to link not only two clauses of a compound sentence but also two coordinate phrases. In Figure 3.8, although the first expression–*Muốn tăng thu nhập* is a verb phrase, similar to the sentence in Figure 3.7, the expression is placed between two conjunctions *vì$_{because}$* and *nên$_{so/therefore}$*, and coordinated with a clause. Therefore, we treat the first expression as an elliptical clause, in which the subject is dropped because it is the same as that of the second clause.

TABLE 3.21: Our internal structure tags.

Group	Tag	Description
Word	Nn_w	A combination of one or more Sino-Vietnamese elements and a Vietnamese original word to create a noun
	Vv_w	A combination of a Sino-Vietnamese element and a Vietnamese original word to create a verb
	Aa_w	A combination of a Sino-Vietnamese element and a Vietnamese original word to create an adjective
	R_w	A combination of a Sino-Vietnamese element and a Vietnamese original word to create an adjunct
Supra-word sub-phrase	Nn_swsp	A sequence of a classifier noun (Nc) and its modifier that means as a noun
	Nn_swsp	A sequence of a special classifier noun (Ncs) and its modifier that means as a noun
	Nn_swsp	A sequence of a categorization noun (Nn) and its modifier that means as a noun
	Vv_swsp_Rt	Repetition form that means as a verb
	Aa_swsp_Rt	Repetition form that means as an adjective
	Nn_swsp_Rt	Repetition form that means as a noun
	ON_swsp_Rt	Repetition form is a sound

3.3.3.4 Internal structures

In this project, we aim to build a treebank that is widely useful for the research community. Therefore, for cases that have little consensus in the community as to methodology to treat them, we use structures that can be automatically converted into others. Discussions in the above sections already showed several structures. In this section, we will describe the use of internal structures to bracket the other cases.

In the bracketing guidelines, internal structure tags are classified into two types, *word* (w) and *supra-word sub-phrase* (swsp), as showed in Table 3.21. Tags that belong to the *word* group are used to bracket expressions of Sino-Vietnamese elements and Vietnamese original words. In these expressions, the Sino-Vietnamese elements play the same role as affixes in English.

As mentioned above, the Sino-Vietnamese elements are not treated as words by linguists [1, 2]. They consider expressions composed by Sino-Vietnamese elements and Vietnamese original words as subordinate compound words. However, in Vietnamese, special classifier nouns are treated as words by linguists, despite the fact that they have the same grammatical functions as Sino-Vietnamese elements. In our corpus, we consider both special classifier nouns and Sino-Vietnamese syllables as single words. We use internal structure tags to mark two types of expressions: (1) expressions composed

TABLE 3.22: Examples show expressions bracketed with internal structure tags.

No.	Types of expression	Examples
1	SV and Vv	(Nn_w (Vv nghiên_cứu) (SV viên)) *{researcher}*
2	Ncs and Vv	(Nn_swsp (Ncs nhà) (Vv nghiên_cứu)) *{researcher}*

TABLE 3.23: Examples show different types of treatment for an expression constituted by a classifier noun and a common noun.

No.	Author	Treatment
1	Corp. [123]	(NP (Num 2) (Nn cây) (Nn lúa)) *{two rice plants}*
2	SCSSV [2]	(NP (Num 2) (Nn cây_lúa)) *{two rice plants}*
3	Our	(NP (Num 2) (Nn_swsp (Nn cây) (Nn lúa))) *{two rice plants}*

by Sino-Vietnamese elements and Vietnamese original words, and (2) expressions composed by special classifier nouns and other words, in which Sino-Vietnamese syllables and special classifier nouns play the same role as affixes in English. Table 3.22 shows two examples of such expressions.

Other internal structure tags in the *swsp* group are used to bracket expressions that are considered as words or phrases by linguists. For example, rows 1 and 2 in Table 3.23 show two different ways that Corp. [123] and SCSSV [2] (respectively) treat *cây lúa*– an expression composed by a categorization noun and a specific noun in Vietnamese. Specifically, Corp. [123] considers *cây*$_{tree/plant}$ *lúa*$_{rice}$ *{rice plant}* as a phrase. SCSSV [2], in contrast, labels it as a compound word. In our guidelines, expressions composed by a categorization noun and a specific noun are segmented on the basis of the word segmentation rule 6. Splitted expressions, like *cây lúa* will be bracketed with the internal structure tag *Nn_swsp*, as illustrated in row 3.

Expressions bracketed with internal structure tags can be automatically converted to make them suitable for specific needs. For example, if we want to treat the example in row 2 of Table 3.22 as a word, we can combine the words *nhà* and *nghiên_cứu* automatically. The POS tag of the new word *nhà_nghiên_cứu*$_{researcher}$ is *Nn*, which are characters standing in front of the hyphen in the internal structure tag *Nn_swsp*.

3.3.3.5 Comparison with VLSP Treebank

Regarding bracketing tags, Nguyen et al. [3] did not use internal structure tags to bracket expressions having little consensus in the linguistic community. Our treebank is the first corpus that has addressed controversial expressions of the Vietnamese language. Additionally, on the basis of Nguyen et al. [3]'s tag sets, we have redefined several tags

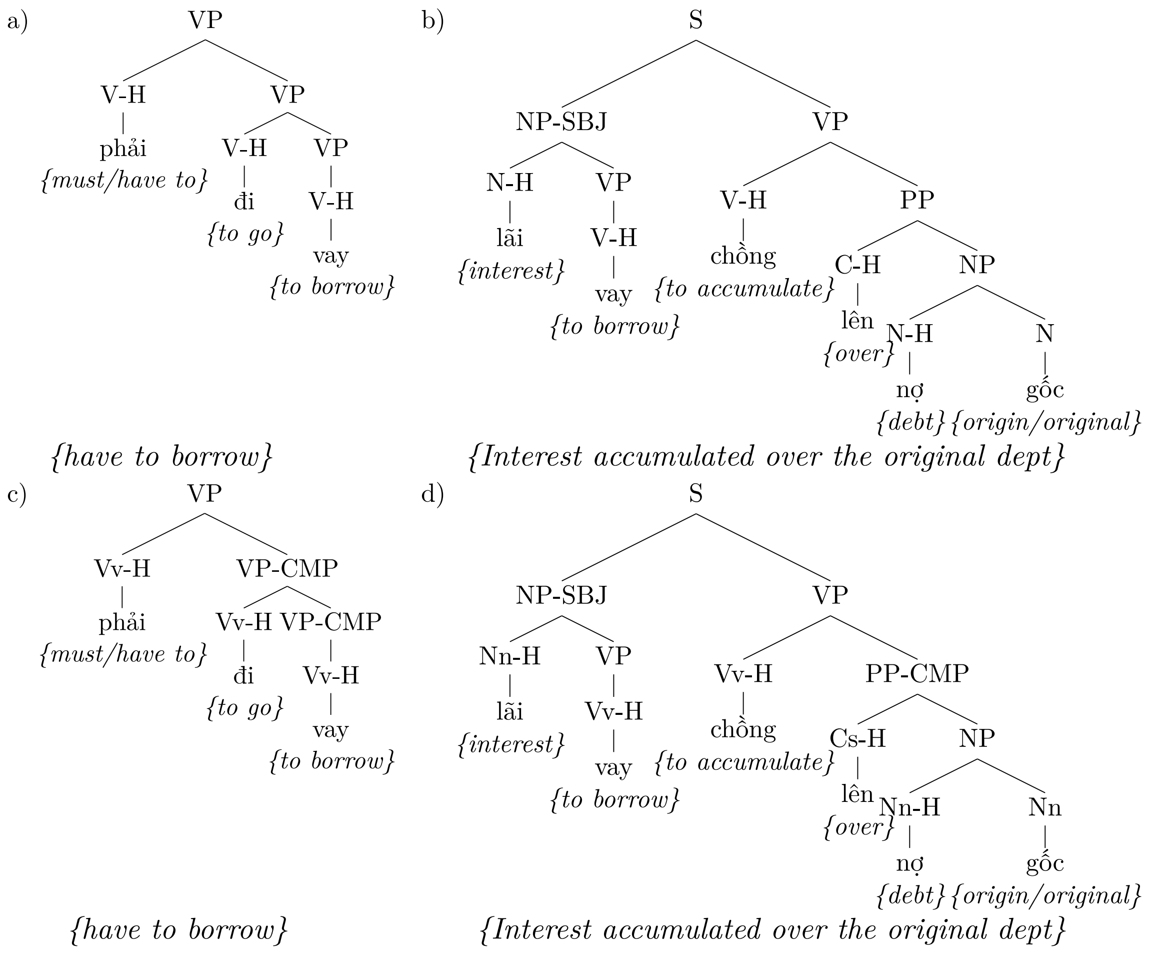


FIGURE 3.9: Examples show how complements are labeled in the VLSP treebank (Figures a and b) and our treebank (Figures c and d).

and added new tags. For example, Nguyen et al. [3] used the function tag *EXT* to mark verbs' complements, which are noun phrases expressing frequency or range [126]. Based on this definition, the verb phrases modifying the verbs $phải_{must/haveto}$ and $đi_{to\ go}$ in Figure 3.9a and the prepositional phrase modifying the verb $chồng_{to\ accumulate}$ in Figure 3.9b should not be labeled with the functional tags. However, we found that these verb phrases and the prepositional phrase are required by the previous head verb. We cannot understand the sentence's meaning if the modifying verb phrases and the prepositional phrase are removed. Therefore, in our bracketing guidelines, we have re-defined the complement tag as follows: *"A complement (CMP) can be a phrase or a clause which is required by the head verb. We cannot understand the sentence if the complement is deleted. However, complement is different from DOB and IOB"*. According to this definition, we mark the verb phrases modifying the verbs $phải_{must/haveto}$ and $đi_{to\ go}$ and the prepositional phrase modifying the verb $chồng_{to\ accumulate}$ with the function tag *CMP*, i.e., VP-CMP and PP-CMP, as illustrated in Figures 3.9c and d.

Figure 3.10 presents another example where we bracketed a Vietnamese sentence with a

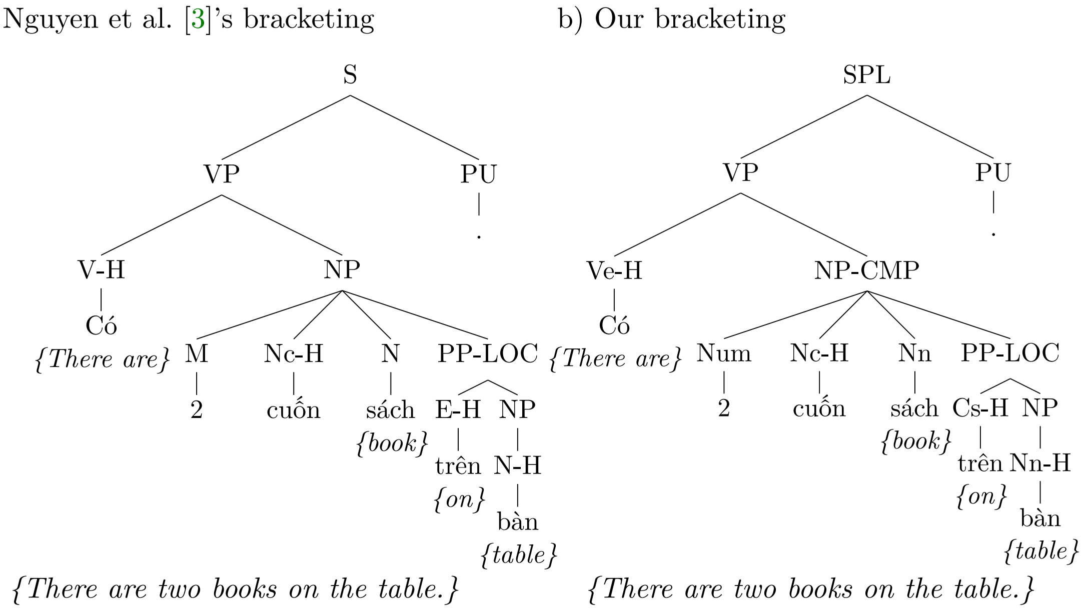


FIGURE 3.10: Example of bracketing a special sentence in our treebank (Figure b) in comparison with VLSP treebank (Figure a).

new added label—*SPL* (Figure 3.10b), which is different from Nguyen et al. [3] (Figure 3.10a). We can see from this figure that although Vietnamese is a SVO language, there are special sentences having only one main component. In our treebank, such special sentences are bracketed with the label *SPL* in order to distinguish them from normal sentences, which are composed by two main components (the subject and the predicate) and bracketed with the tag *S*. Meanwhile, Nguyen et al. [3] used only one tag *S* to annotate both of the two types of sentences.

Investigating the VLSP treebank, we have found many expressions that are bracketed with different structures from ours. For instance, in Nguyen et al. [3]'s treebank, a reduced relative clause modifying the head noun of the subject was annotated as a predicate (the bold expression in Figure 3.11a). We, in contrast, bracket reduced relative clauses as post-modifiers of the modified head words (Figure 3.11b).

3.4 Quality control

To ensure the annotation consistency and accuracy, we consider four issues: building good annotation guidelines, having an appropriate annotation process, training the annotators, and using tools to support the annotation. We have presented the annotation guideline development in Section 3.3 and our annotation process in Section 3.2. In this section, first, we will present our process to train the annotators and measure the inter-annotator agreement and accuracy. Then, we describe software tools are used to improve the annotation speed as well as to control the annotation quality.

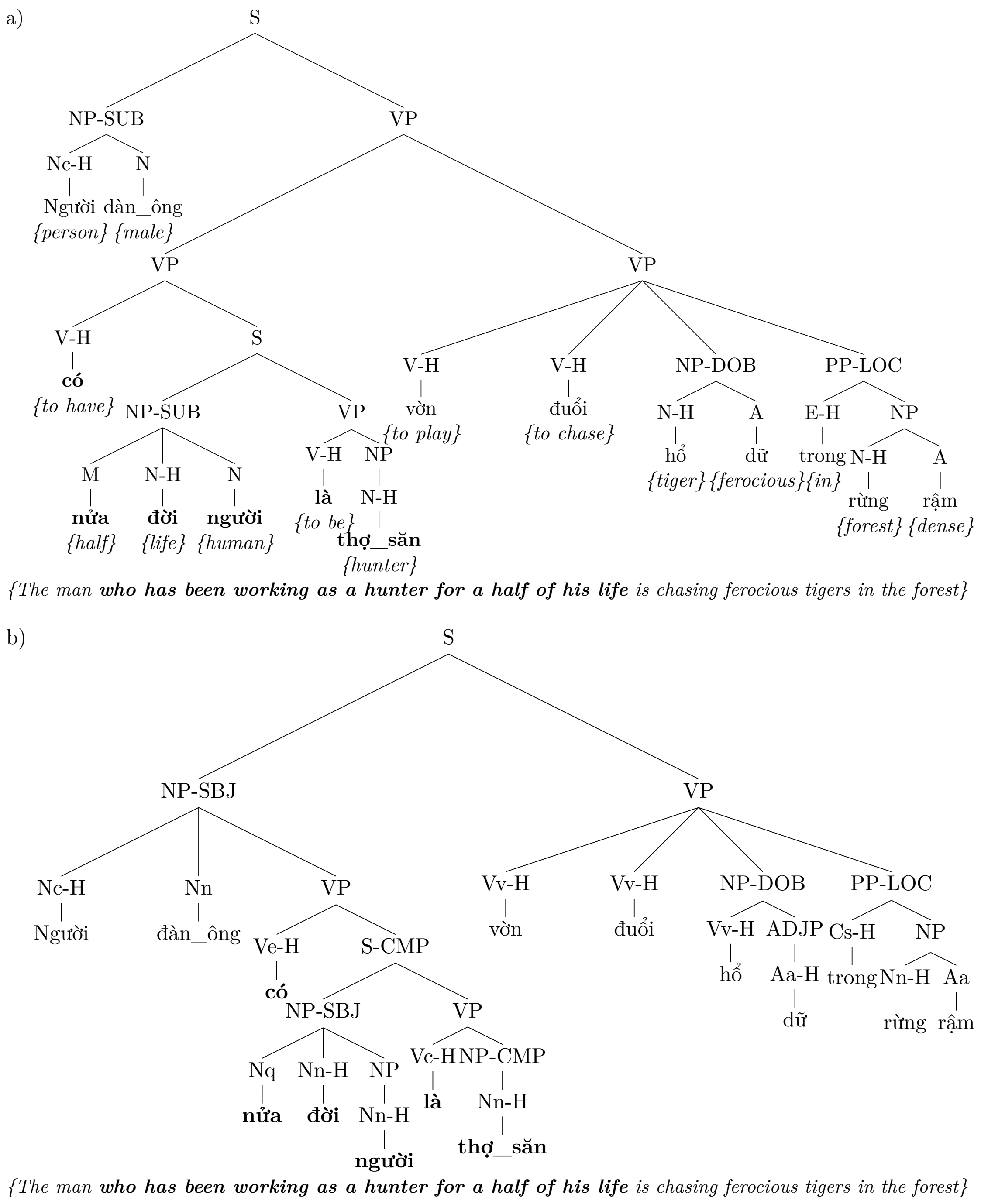


FIGURE 3.11: Examples to illustrate how the reduced relative clauses are bracketed in Nguyen et al. [3]'s treebank (Figure a) and our treebank (Figure b).

3.4.1 Training annotators, revising the guidelines and evaluating our treebank

Although we tried to construct the guidelines as completely as possible before the annotation process began, revising the guidelines during the annotation process is unavoidable since real text is far more complicated than examples mentioned in the literature. In this section, we will discuss our method to improve the quality of annotation guidelines and to ensure correct and consistent annotation. We also present the performance of word segmenting, POS tagging, and parsing tools trained on our treebank.

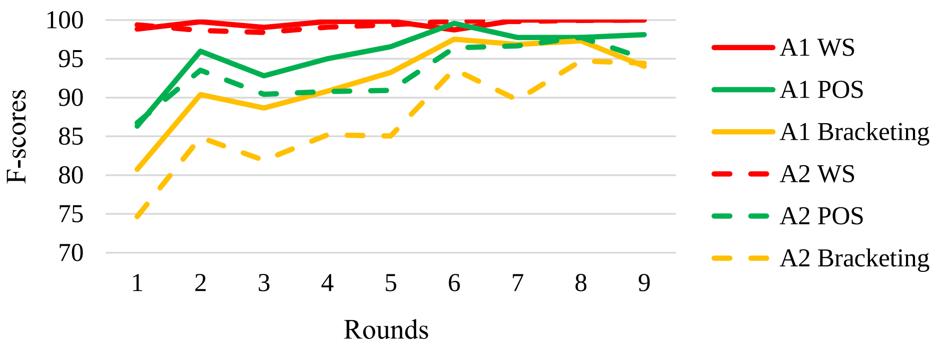
After finishing a draft of the annotation guidelines, we trained two annotators and asked them to annotate 600 texts (about 8,000 sentences) as preliminary annotation. In this preliminary stage, we received feedbacks from the annotators about constructions that they found difficult to annotate because of ambiguities or other reasons. Based on these feedbacks, we revised instructions that cannot be applied to new data and those that are not covered by the guidelines. After revising the guidelines, we retrained annotators with the second version. Then, we carried out nine measurement rounds to calculate accuracy and inter-annotator agreement. Each round includes the following steps:

- We randomly select three texts (about 40 sentences) from the results of the preliminary annotation;
- Each annotator re-annotates the texts independently;
- Accuracy is computed by comparing the annotations of each annotator to a benchmark data, while inter-annotator agreement is computed by comparing the annotations by one annotator to those by the other;
- We discuss with the annotators about errors and inconsistencies, and revise the annotation guidelines (if necessary).

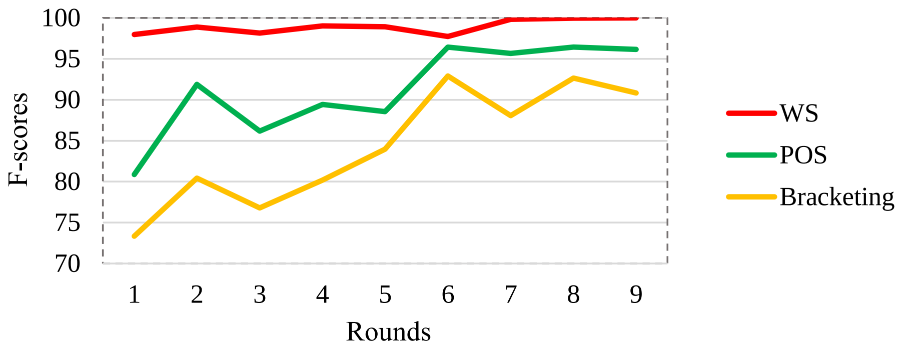
The benchmark data used to estimate the accuracy includes all texts selected in the above-mentioned nine rounds, i.e., about 360 sentences. The benchmark data were double-blind annotated by the two annotators. The resulting annotations were checked and revised (if necessary) during group meetings.

Figure 3.12 presents the accuracy and the inter-annotator agreement after the nine measurement rounds for three layers of annotation, including word segmenting, POS tagging and bracketing. Figure 3.12a shows the accuracy of each annotator (denoted by A1 and A2) compared to the benchmark data. Figure 3.12b shows the agreement between two annotators. We use F-scores for evaluating WS and bracketing, and use the precision scores for evaluating POS tagging. The figure shows that from the sixth round, the accuracy and the inter-annotator agreement were higher than 90%, which is satisfactory.

Moreover, we have also evaluated the quality of our annotators through new texts as follows. After the ninth round of measuring the accuracy and inter-annotator agreement



a) Accuracy



b) Inter-annotator agreement

FIGURE 3.12: Accuracy and inter-annotator agreement of the nine-round training process.

TABLE 3.24: Inter-annotator agreement measured on new texts.

Data	**WS**			**POS tagging**	**Parsing**		
	R	P	F	P	R	P	F
Test 1	99.72	99.44	99.58	97.18	90.54	90.70	90.62
Test 2	99.66	99.54	99.60	98.05	91.98	91.32	91.65

mentioned above, we randomly selected two tests from the preprocessed corpus[15], each test includes three texts (about 40 sentences) focusing on three different topics. Each test was then individually annotated by our annotators (A1 and A2). The inter-annotator agreement was presented in Table 3.24. We can see from this table that the agreement between our two annotators for three layers of annotation were higher than 90%. This again verifies that our treebank is reliable.

Our observations on the inconsistent annotations and errors have revealed that most

[15]Preprocessing steps includes cleaning data, topic classification, sentence segmentation, and annotating by using automatic tools.

Table 3.25: Performance of word segmenter, POS tagger and parser trained on the VLSP treebank and a subset of 10,000 sentences of our treebank.

Data	**WS**			**POS tagging**	**Parsing**		
	R	P	F	P	R	P	F
VLSP treebank	96.3	95.0	95.6	93.13	70.40	73.07	71.71
Our treebank	96.00	96.34	96.17	94.83	74.73	77.21	75.95

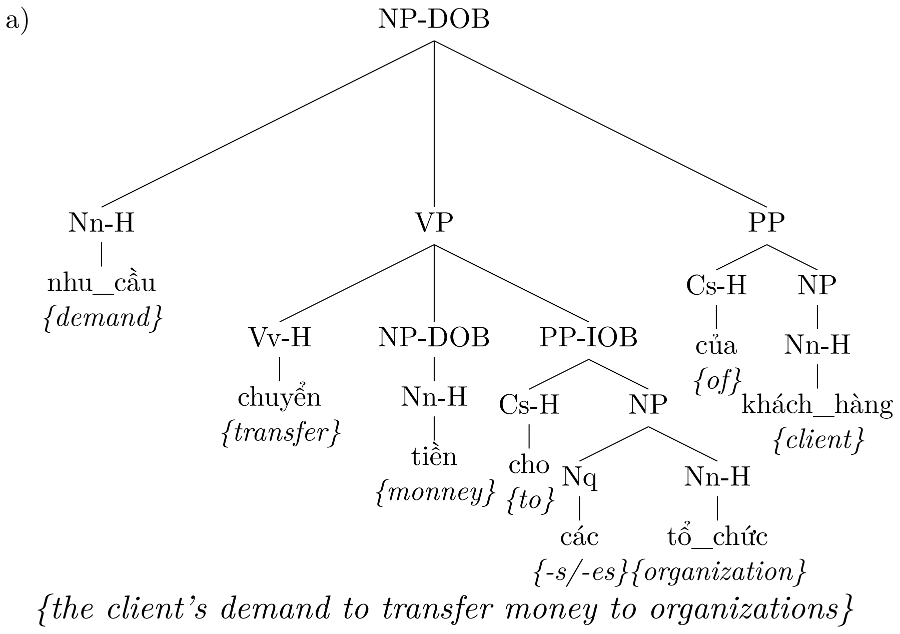
inconsistencies were caused by ambiguous expressions. Figure 3.13 shows an inconsistent example of the two annotators, in which the annotator A1 bracketed the prepositional phrase *của*$_{of}$ *khách_hàng*$_{client}$ *{of client}* as a post-modifier of the head noun *nhu_cầu*$_{demand}$ (Figure 3.13a). In contrast, the annotator A2 treated *của khách_hàng* as a post-modifier of the previous noun *tổ_chức*$_{demand}$ (Figure 3.13b). In fact, both annotators are correct since the whole expression can be interpreted in two different ways as illustrated in Figure 3.13. Although our guidelines contain many examples of ambiguous expressions as well as their correct annotations, they cannot cover all ambiguous cases. In real texts, ambiguous expressions appear in various forms and structures. As a result, it is very difficult to detect all of them and put them in the guidelines. Another cause of the inconsistencies is probably the carelessness of the annotators. Therefore, in order to achieve a high agreement ratio, we can (1) train the annotators carefully and let them practice more on the basis of real texts so that they become familiar with annotating and analyzing the texts, and (2) update the guidelines for new constructions through the annotation process.

In addition to measuring the accuracy and inter-annotator agreement, we have trained tools of word segmenting, POS tagging, and parsing on a subset of our treebank including 10,000 sentences. We randomly selected 1,000 sentences for testing. The rest, including 9,000 sentences, was used for training. For the word segmenter, as there is no Vietnamese word segmenting tool allowing us to re-train on a new corpus, we implemented our own method by using YamCha[16]. For the POS tagger, we used a method proposed by Nghiem et al. [33]. For the parser, we trained the Berkeley parser [13] with our treebank. The evaluation results presented in Table 3.25 show that tools trained on our treebank obtained better performance than those trained on the VLSP treebank [20, 34, 134].

3.4.2 Tools

This section first describes our annotation tools and then discusses several tools used to pre-process texts so that our annotation becomes a semi-automatic process. We also

[16]http://chasen.org/ taku/software/yamcha/



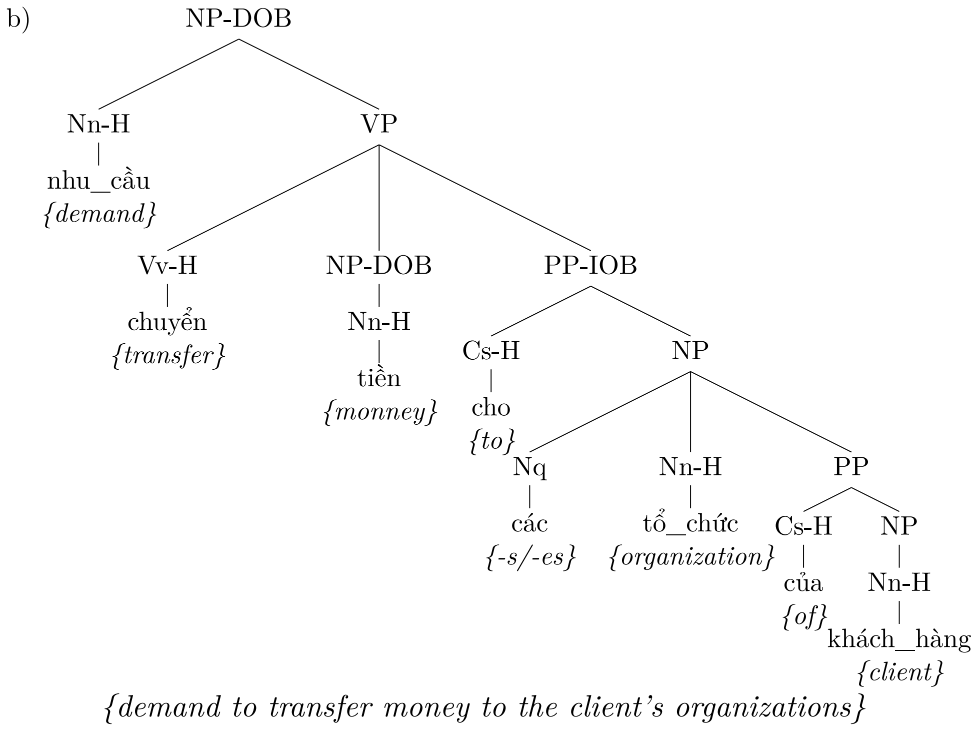
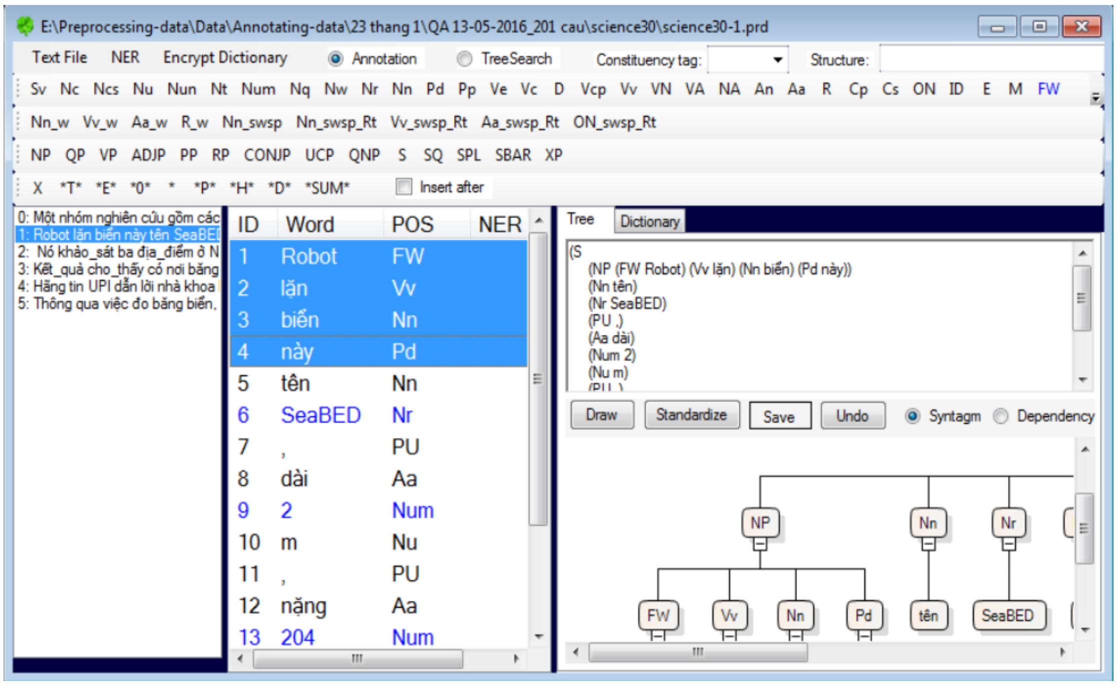


FIGURE 3.13: An inconsistent annotation between the two annotators in a polysemous expression. The left tree is annotated by A1, the right one is by A2.

show the improvement of the annotation speed when texts were pre-processed before the manual editing stage. Finally, we describe tools used to clean up our treebank.


E:\Preprocessing-data\Data\Annotating-data\23 thang 1\QA 13-05-2016_201 cau\science30\science30-1.prd
Text File NER Encrypt Dictionary Annotation TreeSearch Constituency tag: Structure:
Sv Nc Ncs Nu Nun Nt Num Nq Nw Nr Nn Pd Pp Ve Vc D Vcp Vv VN VA NA An Aa R Cp Cs ON ID E M FW
Nn_w Vv_w Aa_w R_w Nn_swsp Nn_swsp_Rt Vv_swsp_Rt Aa_swsp_Rt ON_swsp_Rt
NP QP VP ADJP PP RP CONJP UCP QNP S SQ SPL SBAR XP
X *T* *E* *0* * *P* *H* *D* *SUM* Insert after
Tree Dictionary
Draw Standardize Save Undo Syntagm Dependency
NP Nn Nr
FW Vv Nn Pd tên SeaBED

0: Một nhóm nghiên cứu gồm các
1: Robot lặn biển này tên SeaBEI
2: Nó khảo_sát ba địa_điểm ở N
3: Kết_quả cho_thấy có nơi băng
4: Hãng tin UPI dẫn lời nhà khoa
5: Thông qua việc đo băng biển,

ID	Word	POS	NER
1	Robot	FW	
2	lặn	Vv	
3	biển	Nn	
4	này	Pd	
5	tên	Nn	
6	SeaBED	Nr	
7	,	PU	
8	dài	Aa	
9	2	Num	
10	m	Nu	
11	,	PU	
12	nặng	Aa	
13	204	Num	

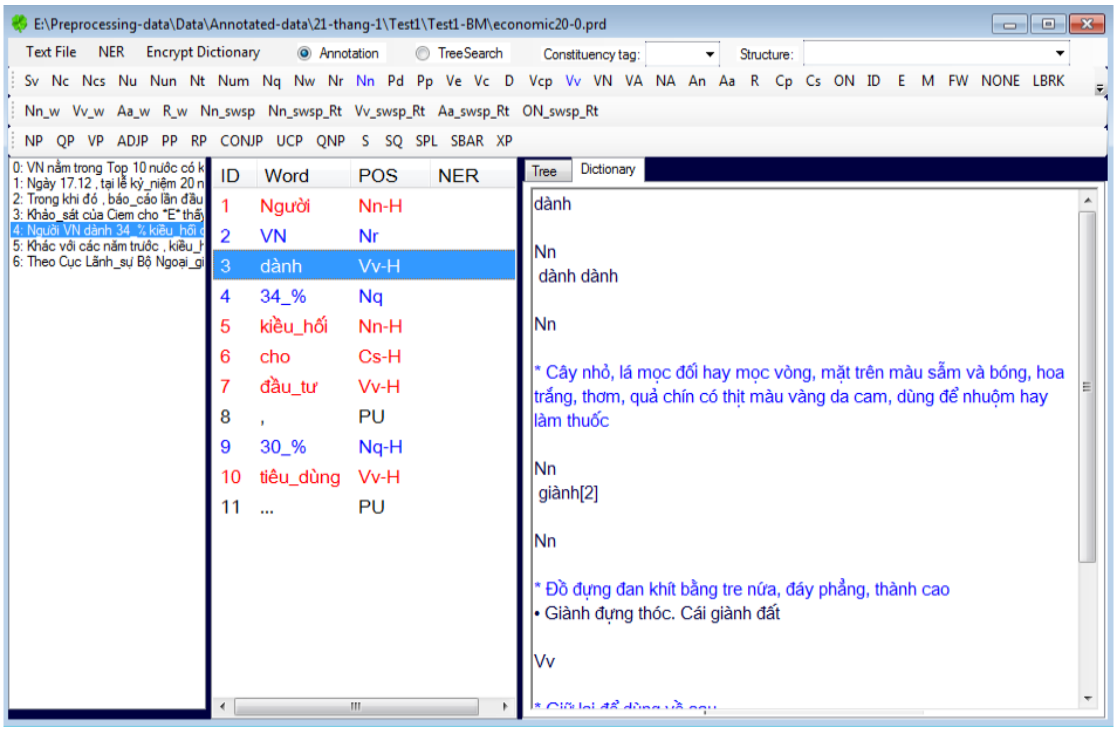
```
(S
    (NP (FW Robot) (Vv lặn) (Nn biển) (Pd này))
    (Nn tên)
    (Nr SeaBED)
    (PU ,)
    (Aa dài)
    (Num 2)
    (Nu m)
```

FIGURE 3.14: A screen shot of our annotation tool which shows how our annotation tool worked when the annotator bracketed a noun phrase.

3.4.2.1 Annotation tool

Nguyen et al. [127] developed a tool to support the annotation of Vietnamese Treebank. This tool allows annotators to visualize all annotation layers (word segmentation, part-of-speech tagging, and bracketing) in a textbox. Annotators have to type all tags and brackets. However, typing the tags not only wastes the time, but also causes annotation errors because of the carelessness of the annotators [19, 20].

To speed up the annotation as well as avoid typing errors, we have modified the annotation tool. In our annotation tool, each tag is presented by a button. The annotation and modification of all layers are implemented by using mouse. For example, to bracket a noun phrase, we select an expression that we want to bracket; then, we click the *NP* button. The *NP* tag and brackets are automatically inserted to the sentence as illustrated in Figure 3.14. In addition, we have also built a Vietnamese dictionary based on Hoang [122] and our annotation guidelines. This dictionary is applied to recommend word segmentation and POS tags. For instance, when we select the word W to annotate, all buttons that are potential POS tags of the word W will be highlighted. If W is not a word in the dictionary, all buttons are disabled. An annotator can also see all meanings of the word W and its corresponding POS tags in the editor. Figure 3.15 shows how our annotation tool worked when the annotator tagged POS for a word.


Text File
NER
Encrypt Dictionary
Annotation
TreeSearch
Constituency tag:
Structure:
Sv Nc Ncs Nu Nun Nt Num Nq Nw Nr Nn Pd Pp Ve Vc D Vcp Vv VN VA NA An Aa R Cp Cs ON ID E M FW NONE LBRK
Nn_w Vv_w Aa_w R_w Nn_swsp Nn_swsp_Rt Vv_swsp_Rt Aa_swsp_Rt ON_swsp_Rt
NP QP VP ADJP PP RP CONJP UCP QNP S SQ SPL SBAR XP
Tree
Dictionary

E:\Preprocessing-data\Data\Annotated-data\21-thang-1\Test1\Test1-BM\economic20-0.prd

0: VN nằm trong Top 10 nước có k
1: Ngày 17.12 , tại lễ kỷ_niệm 20 n
2: Trong khi đó , báo_cáo lần đầu
3: Khảo_sát của Ciem cho "E" thấy
4: Người VN dành 34 _% kiều_hối c
5: Khác với các năm trước , kiều_h
6: Theo Cục Lãnh_sự Bộ Ngoại_gi

ID	Word	POS	NER
1	Người	Nn-H	
2	VN	Nr	
3	dành	Vv-H	
4	34_%	Nq	
5	kiều_hối	Nn-H	
6	cho	Cs-H	
7	đầu_tư	Vv-H	
8	,	PU	
9	30_%	Nq-H	
10	tiêu_dùng	Vv-H	
11	...	PU	

dành

Nn
dành dành

Nn

* Cây nhỏ, lá mọc đối hay mọc vòng, mặt trên màu sẫm và bóng, hoa trắng, thơm, quả chín có thịt màu vàng da cam, dùng để nhuộm hay làm thuốc

Nn
giành[2]

Nn

* Đồ đựng đan khít bằng tre nứa, đáy phẳng, thành cao
• Giành đựng thóc. Cái giành đất

Vv

FIGURE 3.15: A screen shot of our annotation tool which shows how our annotation tool worked when the annotator tagged POS for a word.

TABLE 3.26: Annotation speeds of our annotators for three layers of annotation.

Annotator	**Word segmentation**	**POS tagging**	**Bracketing**
A1	19/6	14/11	53/74
A2	20/6	18/15	55/70

3.4.2.2 Speed up annotation with automatic tools

Additionally, we use several tools to automatically annotate the corpus before manually editing. We use a tool built by Dinh and Vu [135] for word segmentation. A POS tagging tool implemented by Nghiem et al. [33] is used to annotate POS. For the bracketing layer, as there is no available tool, we constructed a supporting tool by training a Berkeley parser [13] on the VLSP Treebank. Because the tag sets of the VLSP Treebank and ours are different, we automatically mapped the VLSP Treebank's tags to ours. Table 3.26 shows the annotation speed of each annotator (denoted by A1 and A2) for three layers of annotation (WS, POS tagging and bracketing), which were measured after we finished nine rounds of measurement of the accuracy and the inter-annotator agreement. In the table, each pair of numbers (x1/x2) is the speed (in minutes) to annotate 347 words (11 sentences) manually without using automatic tools and the speed to manually edit 347 words (11 sentences) after the automatic tools were applied to the raw texts respectively.

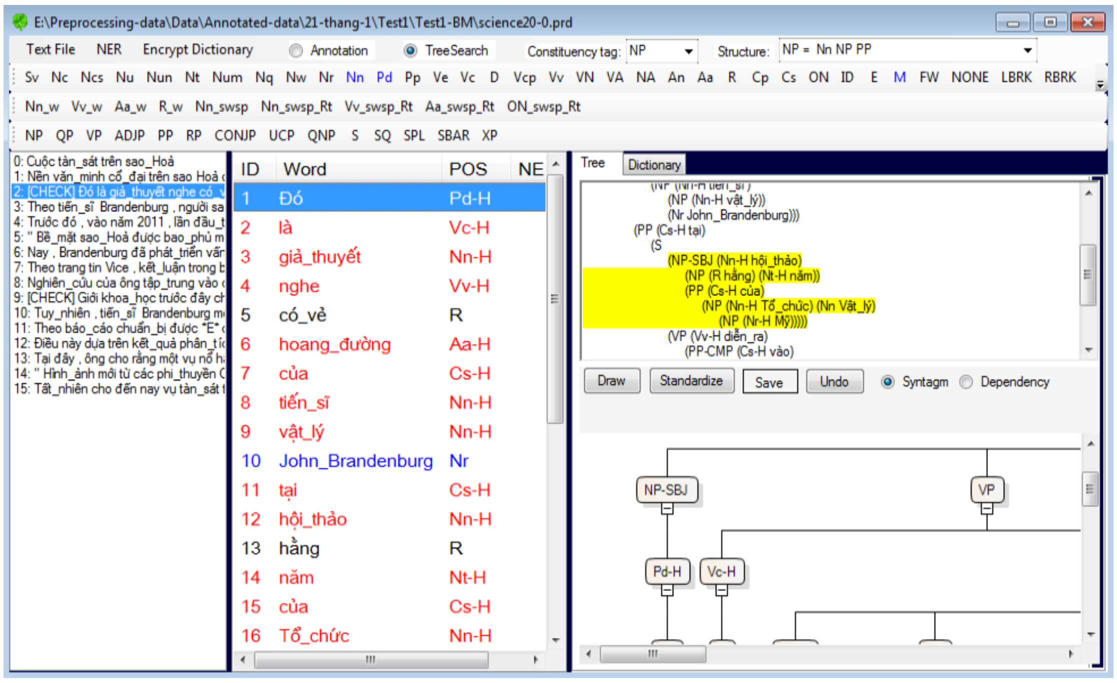
These results indicate that using tools to annotate texts before manually editing increases the annotation speed of WS and POS tagging significantly. However, this pre-process decreases the speed of bracketing. One of the reasons might be the low performance of the parser (71.71% of F-score). Since the parsing results contain many spans and crossing bracket errors, it takes time to delete spans and crossing bracket errors before re-bracketing.

3.4.2.3 Tools to clean up the treebank

As the final step of quality control, we run tools to detect annotation errors, and ask the annotators to manually correct them. Specifically, we use three different tools for three annotation layers of Vietnamese Treebank. To detect WS errors, we use a tool developed by us [19]. This tool detects expressions that have the same surface form or the same syntactic structure, but should be annotated in different ways. Such expressions are ambiguity and can potentially cause inconsistency between the two annotators. In order to identify true inconsistent annotations, we use heuristics based on n-gram sequences and phrase structures. Our tool was used to detect inconsistent annotations in the VLSP Treebank as well. The results showed that 99.2% of detected expressions are inconsistent annotations.

We also implemented a tool to detect POS inconsistent annotations [20]. We classified POS inconsistencies into two types. Nc-inconsistency is a sequence of a classifier noun and its modifier, in which the classifier noun has more than one way of POS annotations in the corpus. The second type of inconsistency is multi-POS inconsistency. We found that at each part (pre-modifier, head, or post-modifier) in each phrase category, each word that is not a classifier noun has only one POS tag. For example, the word $\textit{của}_{property}$, which is the head word of a noun phrase, has to be annotated as *Nn*. If $\textit{của}_{of}$ is the head word of a prepositional phrase, it has to be labelled as *Cs*. If a word is not classifier noun and has more than one POS tag at each part in each phrase category, it is considered as a multi-POS inconsistency. To detect inconsistent annotations for Vietnamese POS tagging, we used the position of words in phrases, phrase categories, and 2-gram sequences. The inconsistent POS annotations of the VLSP treebank produced by our tool is 97.5%.

After correcting annotation errors in WS and POS tagging, we run our TreeSearch tool to revise the bracketing layer. A screen shot of our tool is shown in Figure 3.16. The input of this tool is the treebank. All constituency tags in the corpus are automatically listed in the combo box *Constituency tag*. When users want to check and edit a structure, they first select a constituency tag, such as *NP* (noun phrase). All structures of noun


Text File
NER
Encrypt Dictionary
Annotation
TreeSearch
Sv Nc Ncs Nu Nun Nt Num Nq Nw Nr Nn Pd Pp Ve Vc D Vcp Vv VN VA NA An Aa R Cp Cs ON ID E M FW NONE LBRK RBRK
Nn_w Vv_w Aa_w R_w Nn_swsp Nn_swsp_Rt Vv_swsp_Rt Aa_swsp_Rt ON_swsp_Rt
NP QP VP ADJP PP RP CONJP UCP QNP S SQ SPL SBAR XP
Tree
Dictionary
Draw
Standardize
Save
Undo
Syntagm
Dependency
NP-SBJ
VP
Pd-H
Vc-H

E:\Preprocessing-data\Data\Annotated-data\21-thang-1\Test1\Test1-BM\science20-0.prd

(●) TreeSearch Constituency tag: NP Structure: NP = Nn NP PP

0: Cuộc tàn_sát trên sao_Hoả
1: Nền văn_minh cổ_đại trên sao Hoả
2: [CHECK] Đó là giả_thuyết nghe có_v
3: Theo tiến_sĩ Brandenburg , người sa
4: Trước đó , vào năm 2011 , lần đầu_t
5: " Bề_mặt sao_Hoả được bao_phủ m
6: Nay , Brandenburg đã phát_triển vấn
7: Theo trang tin Vice , kết_luận trong b
8: Nghiên_cứu của ông tập_trung vào
9: [CHECK] Giới khoa_học trước đây ch
10: Tuy_nhiên , tiến_sĩ Brandenburg m
11: Theo báo_cáo chuẩn_bị được "E"
12: Điều này dựa trên kết_quả phân_tíc
13: Tại đây , ông cho rằng một vụ nổ h
14: " Hình_ảnh mới từ các phi_thuyền
15: Tất_nhiên cho đến nay vụ tàn_sát t

ID	Word	POS	NE
1	Đó	Pd-H	
2	là	Vc-H	
3	giả_thuyết	Nn-H	
4	nghe	Vv-H	
5	có_vẻ	R	
6	hoang_đường	Aa-H	
7	của	Cs-H	
8	tiến_sĩ	Nn-H	
9	vật_lý	Nn-H	
10	John_Brandenburg	Nr	
11	tại	Cs-H	
12	hội_thảo	Nn-H	
13	hằng	R	
14	năm	Nt-H	
15	của	Cs-H	
16	Tổ_chức	Nn-H	

```
      (NP (Nn-H tiến_sĩ)
         (NP (Nn-H vật_lý))
         (Nr John_Brandenburg)))
   (PP (Cs-H tại)
      (S
         (NP-SBJ (Nn-H hội_thảo)
            (NP (R hằng) (Nt-H năm))
            (PP (Cs-H của)
               (NP (Nn-H Tổ_chức) (Nn Vật_lý)
                  (NP (Nr-H Mỹ)))))
         (VP (Vv-H diễn_ra)
            (PP-CMP (Cs-H vào)
```

FIGURE 3.16: A screen shot of our tool for revising the Vietnamese Treebank.

phrases in the corpus will be found and listed automatically in the combo box *Structure*. When they select a structure that they want to check and edit, e.g., *NP = Nn NP PP*, our tool will mark sentences containing a phrase structure of *NP = Nn NP PP* with the label *[CHECK]*. When users select a sentence marked with the label *[CHECK]*, all phrases having the structure *NP = Nn NP PP* will be coloured yellow. Users can edit the phrases easily by using the mouse. For example, if users want to change the tag *PP* in the yellow expression in Figure 3.16 to *VP*, they first click the mouse immediately before the tag PP and then select the button *VP* on the tool bar. By using our tool, searched structures are automatically extracted from the corpus. Such a feature not only saves time for creating queries but also ensures that no structure in the corpus is missed. Moreover, the search results are automatically highlighted in the editor, which is convenient for checking and editing.

3.5 Conclusion

This chapter described our efforts to build a high-quality Vietnamese Treebank while ensuring a reasonable annotation speed. We presented issues of the Vietnamese language and our methods to address them to develop the clear, consistent and complete annotation guidelines, including WS, POS tagging, and bracketing guidelines. These guidelines, which is used to train the annotators, are valuable sources that serve the use of the treebank. In addition to developing the annotation guidelines, this chapter also described other issues of ensuring the annotation quality, including an appropriate annotation process, an appropriate process of training annotators and software tools to support the annotation as well as to control the quality. Inter-annotator agreement ratios and accuracy are higher than 90%, which shows that the annotated treebank is reliable and satisfactory.

For the goal of a treebank with about 40,000 syntactic trees, we have completed 15,535 trees. The rest of the Vietnamese Treebank, which includes 24,465 trees, is now being annotated. We will also update the annotation guidelines for new constructions (if any). As mentioned in Section 3.4.2.2, we still have not had a good enough parser to pre-process texts in order to improve the annotation speed of the bracketing layer. Hence, we are planning to use our annotated syntactic trees to train a parser for Vietnamese. We hope that the results of our research are useful for not only Vietnamese but also other languages that have similar issues.

Chapter 4

An Empirical Investigation of Error Types in Vietnamese Parsing

We evaluated representative parsing models on the Vietnamese Treebank to find the most suitable parsing method for Vietnamese in this chapter. We then combined the advantages of automatic and manual analysis to investigate errors produced by the parsers and find the reasons for them. Our analysis focused on four possible sources of parsing errors, namely limited training data, word segmentation errors, part-of-speech tagging errors, and ambiguous constructions. This chapter is constructed as follows. Firstly, we introduce our research in Section 4.1. Section 4.2 presents the parsing evaluation and how word segmentation errors affect the performance of parsers. Our investigation on the behaviour of the parsers is presented in Section 4.3. Section 4.4 discusses the impact of the size of training data on parsing performance. Contributions of ambiguous POS tags and confusing constructions to parsing errors are described in Sections 4.5 and 4.6 respectively. Finally, we conclude our work in Section 4.7.

4.1 Introduction

Syntactic parsing plays a crucial role in improving the quality of natural language processing (NLP) applications and speech processing. Parsing has been broadly studied for languages such as English and Chinese hence the development of many parsing methods. For example, Klein and Manning [75], Petrov and Klein [70], Huang and Harper [101], Collins [102], etc. used a generative probabilistic model called probabilistic context-free grammar (PCFG) to develop the constituent parsers for English, Chinese, etc. However, the original PCFG uses coarse context-free grammar that cannot attain the high-quality parsing result due to the fact that there is not enough contextual information to distinguish the phrases having the similar behavior. Various strategies have been used to enrich contextual information for such generative parsers. For examples, Matsuzaki et al. [114] and Petrov et al. [13] employed automatic state-splitting; Collins [102] associated each category with a lexical item; Klein and Manning [75] applied structural annotation.

Other trend of parsing is to use discriminative probabilistic models in which syntactic information can be enriched with features. Recent research has shown that discriminative parsing has given superior performance to generative parsing. Two typical discriminative algorithms used in parsing are conditional random fields (CRF) and neural networks. Finkel et al. [103] and Hall et al. [72] reported that the feature-based CRF parsing attained an F-score of over 90% on English. The English parsers used neural network could also produce performances higher than 90% in F-score [73, 74].

The parsing problem has not yet been studied thoroughly for Vietnamese. Since the first Vietnamese treebank was built [127], a few researchers have adapted available constituent parsers to Vietnamese, such as AC. Le et al. [22] modified the Bikel's parser [128] and Le-Hong et al. [129] modified the LTAG parser developed by LORIA laboratory[1]. However, the parsing accuracies were far lower than the performances reported for English, Chinese, and French.

It is hard to parse Vietnamese due to its linguistic characteristics. Vietnamese does not have word delimiters and inflectional morphemes in comparison with English. While similar problems also occur in Chinese [36], parsing Vietnamese may be more difficult because the modern Vietnamese writing system is based on Latin characters, which represent the pronunciation but not the meanings of words. As a result, there are many polysemous expressions in Vietnamese, i.e., expressions having the same surface form but different interpretations. Difficulties in Vietnamese parsing are also caused by word orders. Although Vietnamese is a subject-verb-object (SVO) language like English and Chinese, its word orders are different from these languages. For example, the word order in noun phrases (NPs) in Vietnamese, is exactly the same as those in simple sentences, which leads to ambiguities in labelling these two types of expressions. In addition, other problems, such as word omission, cause many complications in parsing Vietnamese texts.

[1]http://www.loria.fr

A worthwhile effort would be to analyze parsing errors to improve the quality of parsers. We consider how to modify the parsers to capture the necessary information to model the syntax based on this analysis. Several studies [136, 137] have manually investigated the parsing errors. While manual investigations can find specific reasons causing parsing errors, they are limited to only a few syntactic phenomena. Moreover, they are not applicable to a large amount of parsing outputs. Kummerfeld et al. [110] developed a tool to automatically classify English parsing errors within a set of predefined error types such as NP attachment and VP attachment. However, based on these analyzing results, we cannot specify any reasons to explain why the error types occurred, which is important clues to improve the parser.

We firstly evaluate representative parsing models on the Vietnamese treebank in this research. We then investigate four possible sources of errors produced by parsers. The first source is the small size of the training data. In this phase, we evaluate the parsers on different sizes of the training data to check whether the current size of Vietnamese treebank is large enough for building a good parser. The second source may come from word segmentation errors. We evaluate the parsers on two different versions of the Vietnamese treebank, using gold word segmentation and word boundary predicted by an automatic word segmentation tool. By doing so, we can estimate how word segmentation errors affect the parsing performance. The third source is the confusing POS tags. In this investigation, by experimenting the input data with and without gold POS tags, we can isolate and quantify error types that can be solved by improving Vietnamese POS tagging. The final possible source is the ambiguous constructions. We used Kummerfeld et al. [110]'s tool in the automatic phase to quantify how often each type of errors occurred in parsers. By comparing the results obtained from this analysis, we could identify which types of errors could be addressed by different parsers, as well as which errors were difficult for parsers to tackle. We manually investigate parsing errors in the second phase based on the results obtained from analysis of the first phase to find the reasons for each error type.

4.2 Parsing evaluation

We evaluated the representative parsers in the literature for English and Chinese on the Vietnamese Treebank. For each parser, we used parameter settings that had the best accuracy in English.

Stanford-Unlex [75] is an unlexicalized probabilistic context free grammar (PCFG) parser where the coarse categories of PCFG, such as *NP* for noun phrases or *VP* for verb phrases, are manually enriched by several simple structural annotations that are state splitting (parent annotation and head lexicalization) and tag splitting.

Stanford-RNN [15] is a combination of a PCFG with a recursive neural network. Phrase representations in this model are learned via the recursive neural network that is conditioned on the coarse PCFG.

Berkeley parser [13] is an unlexicalized PCFG parser where the PCFG is automatically enriched and generalized by using informative latent annotations.

Epic-CRF [72] is a CRF parser in which the anchored rules are scored by using the sparse features of the surface spans.

Epic-NeuralCRF [73] is a CRF parser in which the anchored rules are scored using dense features that are computed via a feedforward neural network.

RNNGs [74] is a top-down transition-based neural model in which a recurrent neural network conditions on full input sentences to parameterize decisions. The RNNGs parser supports two training models, i.e., discriminative (RNNGs-D) and generative models (RNNGs-G).

We used the Vietnamese Treebank developed by Nguyen et al. [138] which includes 10,377 sentences. We randomly selected 1,000 sentences for the development (dev) set, 1,000 sentences for the test set, and the rest, including 8,377 sentences, was used for training. We evaluated parsers in two different versions of the input data: *(i) Raw:* applying automatic analysis for word segmentation[2] and POS tagging[3], *(ii) PredictedPOS:* using only gold word segmentation.

We use PARSEVAL for evaluating parsing performances on the *PredictedPOS* data set and use TEDEVAL [115] for evaluating the parsing results in both *Raw* and *PredictedPOS* scenarios. Although the PARSEVAL evaluation does not accept the different numbers of terminals between parse and gold trees, it was used in most parsing research. All parsers used in this research were also evaluated by the PARSEVAL metric. By using the same evaluation method with previous research, we can measure performances of the Vietnamese parsing in comparison with other languages such as English. Compared to the PARSEVAL evaluation, the TEDEVAL is not a popular evaluation method. However, it allows different terminals between parse and gold trees. In this

[2]We have trained a Vietnamese word segmentation tool using SVM yamcha (http://chasen.org/~taku/software/yamcha/) and left right maximum matching. The training data is the same as the one used for training parsers in this research. Our WS tool archives an F-score of 96.31% on the test set.

[3]The POS tags were predicted by the parsers. However, the RNNGs parser does not have that function. We have trained a Vietnamese POS tagger by using the method proposed by Nghiem et al. [33]. The training data is the same as the one used for training parsers in this research. Our POS tagger archives an accuracy of 94.65% on the test set.

research, the TEDEVAL evaluation not only provides another view of the performance of the Vietnamese parsing but also reveal the impact of word segmentation errors on the parsing performance.

4.2.1 PARSEVAL evaluation

Table 4.1 shows parsing results produced by Evalb[4] on the test set for the parsers. These results indicate that:

- Stanf-RNN archived a higher performance in comparison with Stanford-Unlex. The reason is that although Stanf-RNN was developed on the basis of the Stanford-Unlex, it used a recursive neural network to study phrase representations that can capture the lexical and semantic information. However, both Stanford-Unlex and Stanf-RNN have achieved low performances on the Vietnamese Treebank. That is because they used a manual method to enrich the contextual information for subcategories designed only in the Penn Treebank. It is difficult to apply these parsers to other treebanks such as the Vietnamese Treebank.
- Berkeley parser, similar to the Stanford-Unlex, is also an unlexicalized PCFG parser in which the grammar is enriched by the grammar splitting. However, the Berkeley parser produced much higher performance in comparison with the Stanford-Unlex as well as the Stanf-RNN. This is because grammars are splitted more heavily where needed in the Berkeley parser. This provides more effective subcategories for the parser. Moreover, the grammars are splitted automatically in the Berkeley parser, which makes it easy to apply to other treebanks than the Penn Treebank.
- The Epic-CRF and Epic-NeuralCRF that are CRF parsers with rich input features automatically extracted from the surface spans also work well on the Vietnamese treebank. However, the parsing results show that the dense features computed via a feedforward neural network [73] are not effective on the Vietnamese Treebank as the sparse features do [72].
- Rows RNNGs-D and RNNGs-G show that a transition-based neural model, in which top-down syntactic information was used, is also a suitable method for Vietnamese parsing. In this parser, a recurrent neural network that conditions on the entire syntactic derivation history was used to parameterize decisions. This has greatly relaxed context-free independence assumptions. As a result, it has improved the performance of parsers for not only Vietnamese but also for English

[4]http://nlp.cs.nyu.edu/evalb/

TABLE 4.1: PARSEVAL evaluations on the test set of the seven parsers.

Parsers	**Tagging accuracy**	**R**	**P**	**F**
Stanf-Unlex	89.92	52.15	56.38	54.18
Stanf-RNN	91.83	63.66	66.15	64.88
Berkeley	93.94	70.38	73.48	71.90
Epic-CRF	93.15	72.20	72.96	72.58
Epic-NeuralCRF	93.85	71.68	72.42	72.05
RNNGs-D	94.65	71.94	72.45	72.19
RNNGs-G	**94.65**	**71.71**	**74.21**	**72.94**

and Chinese [74]. However, these parsing results also show that the generative model (RNNGs-G) obtains a higher performance than the discriminative model (RNNGs-D). This is because the discriminative model conditioned on all history, stack, and input buffer, while the generative model only accessed the history and stack. Using larger and unstructured conditioning contexts (input buffer that contains unprocessed terminal symbols) in the discriminative model is difficult and provides opportunities to overfit.

As shown in Table 4.1, RNNGs-G, a top-down transition-based neural model, has achieved the highest performance on the Vietnamese Treebank, while the differences among the RNNGs-G, Berkeley parser, Epic-CRF, and Epic-NeuralCRF are not very significant. However, in comparison with English, these parsers still have achieved much lower performances. One possible reason is that although each parsing method has it-s own advantages, they are not good enough to address all Vietnamese constructions. Specifically, we can see that although the RNNGs-G did not use features, it conditions on the full input sentence that can capture the impact of a wide range of the surrounding words. This is beneficial for not only Vietnamese parsing but also English parsing. However, why does the RNNGs-G achieve much lower performance in Vietnamese than in English? One of the reasons is possibly that grammatical categories in English can be recognized on the basis of inflectional morphemes or function words, while Vietnamese does not have such inflectional morphemes. An additional characteristic of Vietnamese is that Vietnamese uses Latin characters in the writing system which represents the pronunciation but not meaning of the words. As a result, there are many words having the same surface form but different interpretations in Vietnamese. Moreover, dropping words including function words frequently occurs in the Vietnamese text. These indicate that simply conditioning on the surface form of the words cannot address all Vietnamese constructions.

The results of the Berkeley parser, Epic-CRF, and Epic-NeuralCRF have indicated that enriching the contextual information is beneficial for the Vietnamese parsing. The contextual information can be fine-grained categorizations or features extracted from the

Table 4.2: TEDEVAL evaluations on the test set of the seven parsers.

Parsers	**Labeled accuracy**	
	Raw	**PredictedPOS**
Stanf-Unlex	70.65	73.63
Stanf-RNN	72.34	75.33
Berkeley	80.55	81.32
Epic-CRF	80.60	83.59
Epic-NeuralCRF	80.51	83.75
RNNGs-D	81.22	88.06
RNNGs-G	81.81	88.45

surface spans of the anchor rules including the first and last words of the spans, the span length, span shape descriptions, the start, stop, and split indexes where the rule is anchored, etc. However, it seems that the contextual information extracted on the basis of CFG rules did not sufficiently capture the Vietnamese constructions. While the RNNGs have modeled sequences based on the full input sentence that could capture more useful contextual information.

Two conclusions that can be made from this parsing evaluation is that: (1) contextual information is very necessary for the Vietnamese parsing to capture the meaning of words in different contexts. The contextual information can be words playing the roles as inflectional morphemes in English, function words, fine-grained categorizations, etc.; (2) the contextual information is determined not only on the basis of CFG rules but also from the surrounding words.

4.2.2 TEDEVAL evaluation

Table 4.2 presents the TED accuracy on the test set of the seven parsers. Similarly to the PARSEVAL evaluation, RNNGs-G has also achieved the highest scores in this evaluation method in both *Raw* and *PredictedPOS* scenarios. However, although Epic-CRF has achieved a higher F-score than the Epic-NeuralCRF in the PARSEVAL evaluation, it has obtained lower TED accuracy than Epic-NeuralCRF.

Comparing the TED accuracy in *Raw* and *PredictedPOS* scenarios shows that the parsers have achieved higher performances on the *PredictedPOS* data set. This indicates that the word segmentation errors also contribute to the parsing errors. Specially, word segmentation errors have a significant impact on the sequential models. The TED accuracy of RNNGs-D and RNNGs-G have dropped 6.84% and 6.64% respectively when we used the predicted word boundaries. This indicates that to improve the quality of Vietnamese parsing, we should also improve the quality of the Vietnamese word segmentation.

TABLE 4.3: Average number of bracket errors per sentence for the development set of Vietnamese treebank. For example, Stanf-U produces output that has 2.23 note errors per sentence that is caused by VP attachment. Values in the "Worst" row are presented by full black bars. Values in the "Best" row are presented by empty bars.

Parser	F-score	1-Word Phrase	Unary	VP Attach	NP Attach	PP Attach	Clause Attach	NP Int.	Mod Attach	Diff Label	Co-ord	XoverX Unary	Other
Stanf-Unlex	53.49	■	■	■	■	■	■	■	◧	■	■	■	■
Stanf-RNN	64.16	◧	◧	◧	◧	◧	◧	◧	■	◧	■	◧	◧
Berkeley	71.79	◧	◧	◧	□	◧	◧	◧	◧	◧	◧	◧	◧
Epic-CRF	72.03	◧	◧	◧	◧	□	◧	◧	◧	◧	□	□	◧
Epic-NeuralCRF	71.66	◧	◧	◧	◧	◧	◧	◧	◧	◧	◧	◧	◧
RNNGs-D	70.86	◧	◧	◧	◧	◧	◧	◧	◧	◧	◧	■	□
RNNGs-G	72.22	□	□	□	□	◧	□	□	□	□	◧	◧	□
Best		0.86	0.48	1.36	1.10	0.87	0.81	0.38	0.38	0.70	0.03	0.02	1.78
Worst		1.33	0.86	2.23	1.86	1.80	1.06	0.68	0.96	0.92	0.07	0.03	4.06

The evaluation of the Vietnamese Treebank on different parsing methods has given an overview of the parsing performances on the Vietnamese language. While RNNGs-G has achieved the highest performance, this result is still much lower than that reported in English and Chinese, despite of using the gold word segmentation. What are the reasons for the low performance of Vietnamese parsing? Which clues can we use for improving the quality of Vietnamese parsing in the future? We will investigate the parsing errors in the following sections. This investigation is a valuable step towards improving the quality of the Vietnamese parsing.

4.3 Investigating behaviour of the parsers

This investigation was aimed at clarifying two main issues: (1) what are the most frequent errors in Vietnamese parsing and (2) which errors can be addressed by different parsing methods? To answer these two questions, we used the tool developed by Kummerfeld et al. [110] to classify the parsing errors into error types such as VP attachments, NP attachments, and Coordination[5]. We then computed the average number of bracket errors per sentence for each error type. The results from analysis on the development set of the Vietnamese treebank are presented in Table 4.3[6].

There are 11 error types (columns from "1-Word Phrase" to "XoverX Unary") detected by the analysis tool. The "Other" class contains about 20% of the Vietnamese parsing errors that cannot be classified by the analysis tool[7]. Values in the "Worst" and "Best" rows present the average number of bracket errors per sentence. For each parser, full black bars have the value presented in the "Worst" row. Values of the empty bars are presented in the "Best" row. For example, the full black bar at the row "Stanford-Unlex" and column "VP attach" of Table 4.3 indicates that the Stanford-Unlex produces the

[5]It should be noted that each VP attachment error, NP attachment error, etc. can include one or more node errors (bracketing errors) [110].

[6]The types of errors reported in this paper are the same as those in Kummerfeld et al. [110].

[7]The bracketing errors classified into "Other" class will be manually investigated in Section 4.6.

parsing output that contains, on average, 2.23 bracketing errors per sentence that are caused by VP attachments. However, the empty bar in the row “RNNGs-G” and column “VP attach” shows that the RNNGs-G produces, on average, 1.36 bracketing errors per sentence due to VP attachments.

By comparing 11 types of errors produced by the parsers (columns from “1-Word Phrase” to “XoverX Unary” in Table 4.3), we can see that bracket errors caused by VP attachments are the most frequent in Vietnamese parsing. In addition, we can see from this table that the errors that are due to NP attachments, PP attachments, 1-word phrase, and clause attachments also frequently appear in Vietnamese parsing. Moreover, Table 4.3 also indicates how the errors can be addressed by different parsing methods. For example, the RNNGs-G has created more Co-ordination (Co-or) errors than the Epic-CRF. It also created more XoverX Unary errors than the Berkeley and Epic-CRF but the differences are very small. However, for other error types, it is more productive than other parsers.

Hence, this analysis has shown the challenging constructions in Vietnamese parsing, in which the top three challenges are VP, NP, and PP attachments. This analysis has also indicated that the RNNGs-G is the best method for addressing almost error types in Vietnamese parsing. In addition, the quantification of error types also provides a chance of investigating challenges of Vietnamese parsing in comparison with other languages. A detailed comparison between Vietnamese and English parsing errors [110] is as follows:

- Most error types that appeared in English parsing also appeared in Vietnamese parsing, excepts the NP internal structure[8]. However, the frequency with which each error type occurred in Vietnamese parsing was much higher than that in English, except for Co-ordination.
- Although the most problematic constructions in English parsing, i.e., PP attachments, were not the most difficult constructions in Vietnamese parsing, PP attachment errors appeared more frequently in Vietnamese parsing than in English parsing.
- VP attachment errors were the most frequent errors in Vietnamese parsing. However, this error type did not appear in English parsing.

Kummerfeld et al. [121] also analyzed Chinese parsing errors. However, it was difficult for us to directly compare the results obtained from analysis for Vietnamese and Chinese parsing. This was because we used the method of error classification proposed

[8]The NP internal structure is not annotated in the Vietnamese Treebank.

by Kummerfeld et al. [110], which was substantially different from that of Kummerfeld et al. [121]. However, we drew three main conclusions from the observation of error analysis results: (i) the NP internal structure is the most problematic construction in Chinese parsing, while it is rare in Vietnamese parsing. (ii) Although VP, NP, and PP attachments are the top three difficult constructions in Vietnamese parsing, they are not frequent errors in Chinese parsing. No NP attachment errors have even appeared in Chinese parsing and (iii) one-word phrases are the most problematic constructions in both languages.

Hence, apart from containing the same complex constructions as English and Chinese such as PP attachments, Vietnamese parsing has its own constructions, such as VP attachments, that cannot be solved with the current parsing methods. A new parsing method of addressing specific issues in the Vietnamese language is required. In addition, by quantifying each error type produced by different parsing methods, our investigation provided an orientation for improving the quality of Vietnamese parsing. For example, solving the most frequent errors, such as VP attachment, NP attachment, and PP attachment errors can significantly improve the quality of parsers. However, we could not find reasons of error types from this investigation, which are very important clues to solving errors. We will investigate three possible sources for the poor performance of Vietnamese parsing in the following sections, viz., the impact of training data size (Section 4.4), impact of POS tagging errors (Section 4.5), and impact of ambiguous constructions (Section 4.6).

4.4 Impact of training data size

Our training data included 8,337 sentences, which were much fewer than the data used for English and Chinese parsing. To identify whether the small size of training data was the reason for the poor accuracy of the parsers, we evaluated them on different amounts of training data including 1,337, 2,337, 4,337, 6,337, and 8,337 sentences. The parsing results (F_1) on the test set of the Vietnamese Treebank are presented in Figure 4.1. We can see from this figure that the quality of the parsers increased when we increased the training data from 1,337 to 2,337, 4,337, and 6,337. However, the F-score is almost saturated when we increased the training data from 6,337 to 8,337 sentences. Therefore, we could not expect a significant improvement in accuracy by simply increasing the amount of data. Hence, the amount of training data was not the main reason for the poor performance of the parsers.

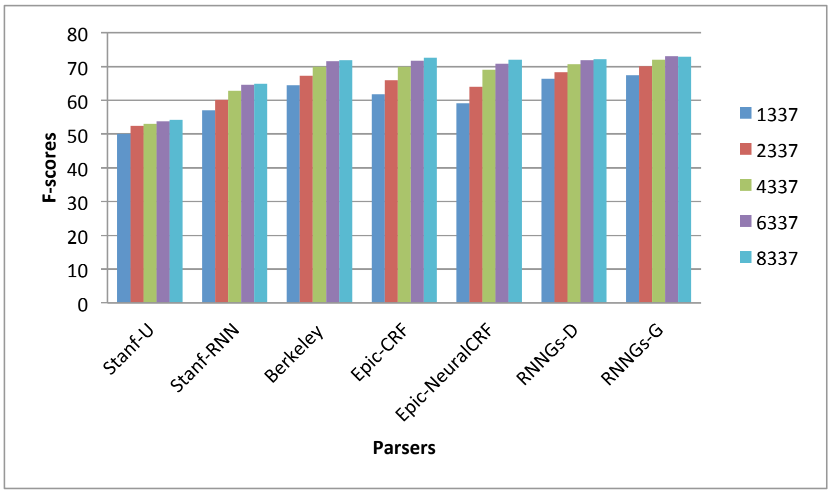


FIGURE 4.1: Parsing results from the parsers with different amounts of training data.

4.5 Impact of tagging errors

Figure 4.2 presents the PARSEVAL evaluations (F-scores) of the parsers on two different versions of the test set, *PredictedPOS* and *GoldPOS*, which used gold word segmentation and POS tags. These results indicate that using gold POS tags could improve the accuracies of the parsers. The F-score of RNNGs-G especially achieved 78.2%, which increased by 5.26% in comparison with the use of automatically tagged POSs (PredictedPOS). However, this figure does not indicate the error types that could be solved by improving POS tagging. We will investigate the contributions of tagging improvement to each error type in Section 4.5.1. Then, we explore how each ambiguous POS pair impacts parsing errors in Section 4.5.2.

4.5.1 Contributions of tagging improvement to error types

To find how the gold POS tags affected parsing errors, we compared the error types produced by the best parser, RNNGs-G, on two different versions of the dev set: (1) *Gold*: by using gold word segmentation and POS tags and (2) *Pred.*: by only using gold word segmentation. Table 4.4 presents a breakdown of these errors. In this table, the "Occurences" columns indicate the number of times that each error type occurs. The "Node errors" columns present the number of node errors that are caused by each error type. For example, VP attachment errors occur 392 times in the parsing output with automatically predicted POS tags. These 392 VP attachment errors have caused 1,362 bracketing errors. The "Node errors per sentence" columns indicate the average number of bracketing errors per sentence. "Gain" presents the gain (positive number) and loss (negative number) of node errors per sentence when the automatically predicted POS

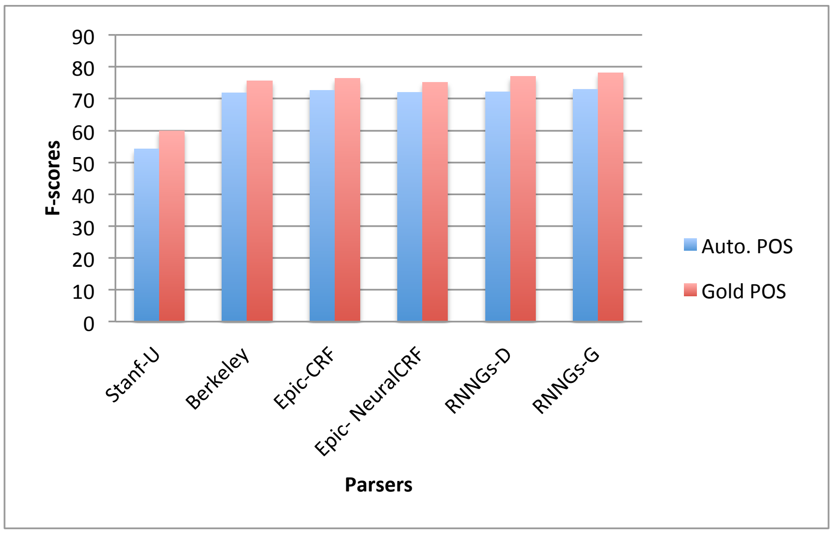


FIGURE 4.2: Results from the parsers on two different versions of test set, Predicted-POS and GoldPOS.

TABLE 4.4: Statistics on errors produced by RNNGs-G on two different versions of dev set, *Pred.* and *Gold*. "Gain" represents gain (positive number) and loss (negative number) of node errors per sentence when replacing automatically predicted POS tags with gold POS tags (i.e., Errors per sentence (Pred.) - Errors per sentence (Gold)).

Error type	**Occurrences**		**Node errors**		**Node errors per sentence**		**Gain**
	Pred.	**Gold**	**Pred.**	**Gold**	**Pred.**	**Gold**	
Single Word Phrase	765	517	857	544	0.86	0.54	0.32
Unary	475	386	475	386	0.48	0.39	0.09
VP Attachment	392	373	1362	1368	1.36	1.37	-0.01
PP Attachment	415	345	936	837	0.94	0.84	0.10
Different label	350	157	700	314	0.70	0.31	0.39
NP Attachment	333	339	1104	1101	1.10	1.10	0.00
Clause Attachment	327	277	814	696	0.81	0.70	0.11
Modifier Attachment	216	182	381	362	0.38	0.36	0.02
NP Internal Structure	245	211	379	324	0.38	0.32	0.06
Co-ordination	34	44	34	44	0.03	0.04	-0.01
XoverX Unary	24	25	24	25	0.02	0.03	0.01
Other	969	729	1779	1435	1.78	1.44	0.34
Sum	4545	3585	8845	7436			

tags were replaced by gold POS tags (i.e. Errors per sentence (Auto.) - Errors per sentence (Gold)).

The results in Table 4.4 indicate that improved tagging reduced the occurrence of some error types. However, these reductions were insignificant. Improved tagging did not especially help two of the most frequent error types in Vietnamese parsing, i.e., VP and NP attachments. We can see from Table 4.4 that occurrences of VP attachment errors reduce when we replace the automatically predicted POS tags by the gold POS tags. However, the gain does not occur for node errors caused by VP attachment errors.

TABLE 4.5: Top ten confusing POS pairs in Vietnamese.

No.	**Confused tags**	**Occurences**
1	Nn/Vv	106
2	Vv/Nn	87
3	Aa/Vv	77
4	Vv/Aa	69
5	Vv/Cs	62
6	Aa/Nn	51
7	Cs/Vv	40
8	Nc/Nn	38
9	Nn/Aa	37
10	R/Vv	32

The possible reason is that tagging improvement can help for several VP attachments. However, other VP attachment errors have occurred because of ambiguous constructions that caused more node errors.

4.5.2 Impact of ambiguous POSs on parsing errors

In this subsection, we investigate what are challenges of Vietnamese POS tagging, and how these challenges contribute to the parsing errors.

For the first issue, comparing the dev set with automatically predicted POS tags against the dev set with POS tags shows that there are 167 ambiguous POS pairs which occur 1,287 times in the dev set. The top ten confusing POS pairs and their frequency are presented in Table 4.5. Each POS tag pair x/y in this table indicates that the gold POS tag x is mispredicted as the POS tag y by the POS tagger. We can see from this table that distinguishing between the noun (Nn) and the verb (Vv) is the most ambiguous in Vietnamese POS tagging. Specifically, the confusion *Nn/Vv* occurs most often that is 106 times in the dev set. These challenges come from the characteristics of Vietnamese languages such as the lack of inflectional morphemes and using Latin characters in the writing system which represents the pronunciation but not the meaning of the word. These create a lot of polysemous words in Vietnamese text. In addition, other contextual information that can help for recognizing POSs such as adjuncts indicating tenses standing before verbs, special classifier nouns standing before verbs, or classifier nouns standing before nouns, are also frequently omitted in the real text.

For the second issue, to see how each ambiguous POS pair impacts the parsing errors, we used the same method as Kummerfeld et al. [121] to find how an ambiguous POS pair impacted parsing errors. We began from *Gold*, and replaced the gold tags with tags that were predicted by the automatic tagger (semi-gold input). We then parsed the semi-gold input by using RNNGs-G and obtained the results from the PARSEVAL evaluation and automatic analysis. The impacts of the top four confusing POS pairs

TABLE 4.6: Most frequently confusing POS tag pairs in Vietnamese POS tagging. Δ F1 indicates decreases in F-scores when gold POS tags were replaced with predicted POS tags. Meanings of POS tags are: *Nn*: common nouns, *Vv*: common verbs, and *Aa*: adjectives.

Confusing tags	**Occurrences**	**Bracketing Errors**	**F1**	**Δ F1**
Nn/Vv	106	7694	76.26	-1.94
Vv/Nn	87	7764	76.15	-2.05
Aa/Vv	77	7897	75.62	-2.58
Vv/Aa	69	7749	76.07	-2.13
Gold POSs		7436	78.20	

on overall bracketing errors and F-scores are presented in Table 4.6. We can see that all of these four confusing POS pairs were potential contributors to bracketing errors that caused significant decreases in F-scores. However, this table also indicates that the frequency of confusing POS pairs was not directly proportional to the contributions they made to parsing errors. For example, although *Aa/Vv* was the third most confusing POS pair, it contributed most to parsing errors and caused the highest decrease in F-scores (2.58). While occurrences of *Nn/Vv* were highest (106 times), they caused the lowest decrease in F-scores (1.94). The reason originates from different complexities of the phrases. The complexities of *NP* and *VP* are quite similar, for example both *Nn* and *Vv* can be modified by the *SBAR* or *PP*. Therefore, when a *Nn* was mispredicted as the *Vv*, the noun phrase where the *Nn* is the head word could be bracketed as the *VP*. Frequencies of parsing errors for these two constructions are not very different. While, the construction of *ADJP* is more simple than *VP*, for example, the *Aa* is not modified by a *PP*, except the *PP* with the head word $như_{as}$ which expresses the equal comparison, or the *Aa* cannot be modified by a *SBAR*. When a parsing error occurred in the *ADJP*, it caused few bracketing errors. However, when an *Aa* was mispredicted as a *Vv*, the *ADJP* with the head word *Aa* was bracketed as the *VP*. And the phrases such as *PP* and *SBAR*, which should be bracketed separately from the previous *Aa*, were bracketed as the modifier of the *Vv*. This has caused the significant difference of bracketing errors when the *Aa* was mispredicted as the *Vv* or reverse.

Table 4.7 summarizes the impact the top four confusing POS pairs had on each error type. The positive and negative numerals correspond to the number of bracketing errors that were created or reduced in parsing output when we replaced gold POS tags with predicted POS tags. Analysis based on five of the most frequent error types of VP attachment, NP attachment, PP attachment, clause attachment, and single word phrase revealed the following.

- **VP and NP attachments:** As was previously explained, improved POS tagging did not assist in preventing VP and NP attachment errors. We can see from

TABLE 4.7: Gains and Losses of bracket errors when the gold POS tags are replaced by the POS tags predicted by the automatic tagger. For example, the number 80 in row "Single Word Phrase" indicates that replacing the gold POS tag Nn (common noun) by the predicted POS tag Vv (common verb), number of Single Word Phrase error increases 80 errors.

Error type	**Nn/Vv**	**Vv/Nn**	**Aa/Vv**	**Vv/Aa**
Single Word Phrase	80	70	66	68
Unary	16	22	52	38
VP Attachment	-26	44	-45	17
NP Attachment	-65	-45	37	-43
PP Attachment	27	51	83	48
Clause Attachment	75	26	103	70
NP Internal Structure	69	21	50	30
Modifier Attachment	11	25	-11	-14
Different label	42	44	32	22
Co-ordination	-21	-14	-14	-21
XoverX Unary	-1	-1	-2	-1
Other	51	85	110	99

Table 4.7 that bracketing errors caused by VP attachment errors increased when we replaced gold POS tags *Vv* with predicted POS tags *Nn* or *Aa*. Bracketing errors caused by NP attachment errors also increased when we replaced gold POS tags *Aa* with predicted POS tags *Vv*. This indicated that gold POS tags were of help in these cases. However, using predicted POS tags was better for some constructions, e.g., using predicted POS tags *Vv* instead of gold POS tags *Nn* reduced 65 bracketing errors that were caused by NP attachment errors. This reveals that low performance of Vietnamese parsing is caused by other reasons.

- **PP attachments:** All four confusing POS pairs caused these types of parsing errors. However, *Aa/Vv* was the major contributor to parsing errors.
- **Clause attachments:** *Aa/Vv* was the major contributor to these types of parsing errors. In addition, ambiguous POS pairs, such as *Nn/Vv* and *Vv/Aa*, also created significant numbers of bracketing errors.
- **Single word phrase:** All four confusing POS pairs caused this type of parsing errors.

Hence, the quality of Vietnamese parsing could be improved by 5% through improving the Vietnamese POS tagging. We also found that all frequent ambiguous POS pairs contributed to parsing errors in which *Aa/Vv* was the major contributor. In addition, our investigations also found that improved POS tagging was beneficial for some constructions, such as PP attachment, clause attachment, and single word phrase. We should develop further processes for VP and NP attachments, which were two of the most confusing constructions, apart from improving POS tagging.

4.6 Ambiguous constructions in Vietnamese

An ambiguous construction is a tag sequence (including POS tags and phrase tags) that can be bracketed in different ways. For example, the sequence of POS tags *Nn Vv* can be bracketed as a noun phrase (NP) in which *Vv* is a modifier of the head noun *Nn*. This tag sequence can be also bracketed as a simple sentence where *Nn* is the subject and *Vv* is the predicate.

This section aims to investigate parsing errors that were caused by ambiguous constructions in Vietnamese. To isolate such type of errors, we eliminated impacts of small size of training data, word segmentation errors, and confusing POS tags from the parsing output. The above-mentioned evaluations showed that the parsing result by RNNGs-G did not change significantly when we increased the training data from 6,337 to 8,337 sentences. Therefore, by using the RNNGs-G parsing model trained on 8,337 sentences, we can eliminate the impact of small size of the training data. In addition, we also used the gold word segmentation and POS tags when we parsed the development set in the following experiment. As a result, parsing errors produced by the RNNGs-G in the development set are purely caused by ambiguous constructions in Vietnamese.

In this investigation, we manually analyze the error types produced by the analysis tool developed by Kummerfeld et al. [110]. This tool was designed to classify the English parsing errors. By directly applying it to the Vietnamese parsing output, we found about 20.1% of the errors placed in the "Other" type (for English, this ratio is about 11.3%). Although we do not modify the tool so that it can classify specific errors of Vietnamese parsing, we will manually analyze these errors to capture the reasons that caused them.

4.6.1 Classified constructions

The results obtained from a manual investigation based on 100 sentences randomly selected from the parsing output of the dev set indicated that Vietnamese included many ambiguous constructions. Table 4.8[9] presents several frequent ambiguous constructions in Vietnamese parsing that were found in this manual investigation. The column "Ambiguous construction" in this table presents frequent ambiguous tag sequences in Vietnamese. The error type, which is caused by ambiguous tag sequences, is indicated in the column "Error type". For example, the tag sequences *Nn Vv VP*, *Nn Aa VP*, *Nn Nn VP*, *Vv Vv VP*, and so on are candidates for VP attachment errors. For each

[9] *Nn* in Table 4.8 represents a noun that can be *Nn* (a common noun), *Nc* (a classifier noun), or *Ncs* (a special classifier noun). Each component of the tag sequence can be generalized as a phrase, e.g., instead of *Nn*, a noun phrase with head word *Nn* can be placed in the position of *Nn*

TABLE 4.8: Several frequent ambiguous constructions in Vietnamese parsing.

Ambiguous construction	**Error type**	**Frequency**
Nn|Vv|Aa|Cs Vv|Aa|Nn VP	VP attachment	19/34
Vv|Nn Vv|Aa|Nn NP	NP attachment	25/39
Nn|Vv Nn|Vv PP	PP attachment	17/29
NP VP NP VP	Clause attachment	10/26
NP|VP NP|VP VP	Clause attachment	6/26
Cs NP VP	Clause attachment	4/26

pair of *x/y* in the column “Frequency", the *x* represents the number of times that each ambiguous tag sequence caused errors in 100 investigated sentences. The *y* indicates the frequency of each error type in 100 investigated sentences. For example, we found 26 clause attachment errors in 100 investigated sentences, in which 10 clause attachment errors were caused by tag sequence *NP VP NP VP*. We will next present several examples of frequent ambiguous constructions in Vietnamese parsing.

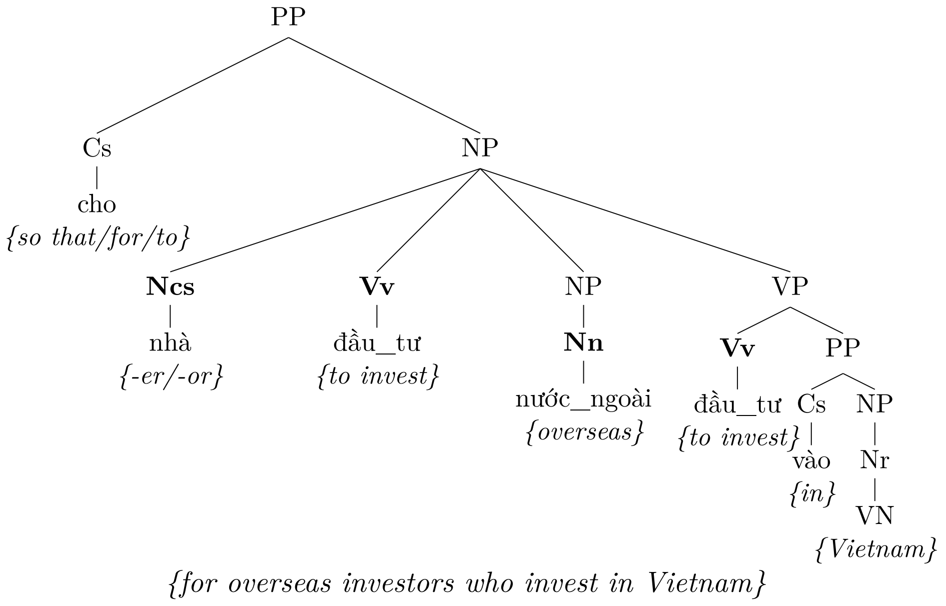
Figure 4.3[10] shows a VP attachment error caused by the ambiguous construction *Ncs Vv Nn VP*[11]. In this figure, RNNGs-G annotated the verb phrase *VP* as a modifier of the noun *nhà*$_{-er/-or}$ while this verb phrase was treated as a predicate of the simple sentence in the gold tree. This ambiguity occurred because Vietnamese does not have inflectional morphemes as well as supporting function words that are important clues to distinguish between the modifying verb phrase and the predicative verb phrase. This ambiguity is also caused by the problem of the word order. Unlike English and Chinese, where modifying lexical words stand in front of the head noun, in Vietnamese noun phrases, modifying lexical words follow the head word. It is difficult to distinguish between noun phrases and simple sentences because they have the same structure. Another possible reason for this ambiguity is that polysemous words occur frequently in Vietnamese. We can see in Figure 4.3 that the word *cho* can be understood as *so that*, which can be used to introduce a clause, or can be understood as *for* or *to* followed by a phrase.

Figure 4.4 presents another VP attachment error caused by the ambiguous tag sequence *Vv Nn Vv*. Similarly to this figure, the second *Vv* follows a *Nn* in Figure 4.3. However, in this case, the second *Vv* should be annotated as a modifier of the previous noun (gold tree) rather than the head of the phrase as in Figure 4.3a. In practice, similar to the example in Figure 4.3, RNNGs-G also considered the second *Vv* and the *Nn* in this example as two separate phrases (the parsing output). This confusion is also caused by

[10] Underscore “_” is used to link syllables of Vietnamese multi-syllable words. English translations of Vietnamese words are given as subscripts. If a Vietnamese word does not have a translatable meaning, the subscript is blank. The translation for the Vietnamese sentence is given in curly brackets below the original text.

[11] The sequence *Ncs Vv* is a combination of the special classifier noun *nhà*$_{-er/-or}$ and the verb *đầu_tư*$_{to\ invest}$ that means as the noun *investor*. Therefore, we can consider the construction *Ncs Vv Nn VP* as a similar form of the ambiguous construction *Nn Nn VP* mentioned in Table 4.8.

a) Parser output



b) Gold tree

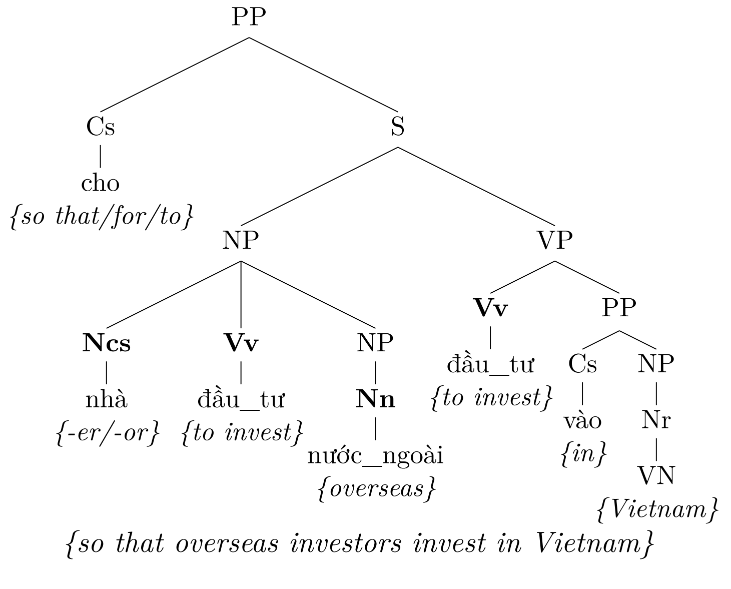


FIGURE 4.3: Examples illustrating the ambiguity between noun phrases and simple sentences in Vietnamese.

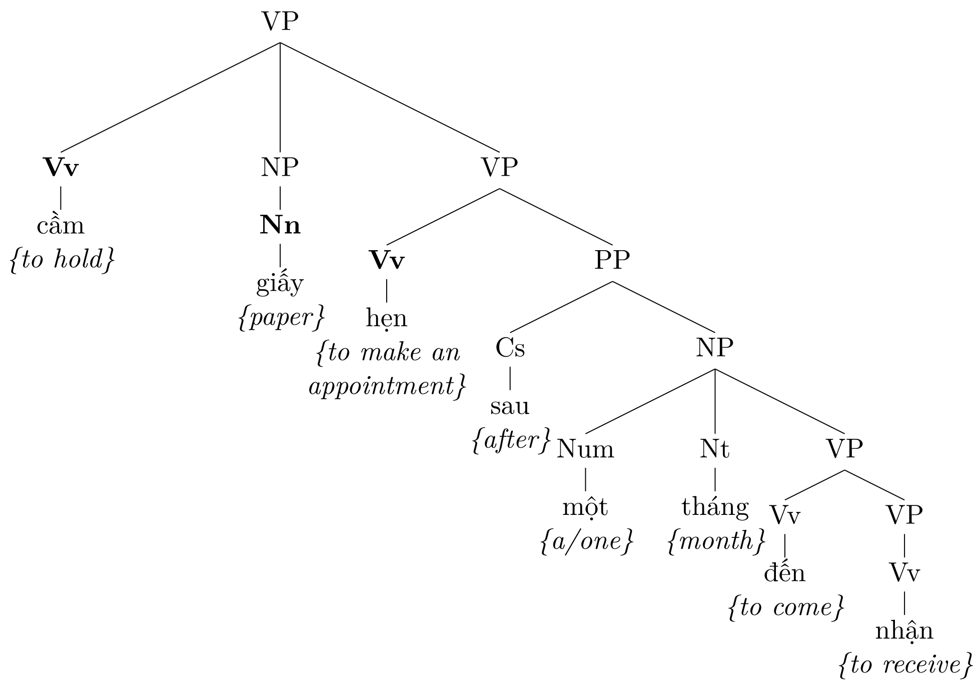
the phenomenon that the modifying verb ($hẹn_{to\ make\ an\ appoinment}$) follows the head noun ($giấy_{paper}$) in the noun phrase. In addition, there is no clue such as function words or inflection to distinguish the modifying verb from the head verb in a verb phrase. It is difficult to differentiate a noun phrase having a modifying verb (or verb phrase) from a separate structure including a noun phrase and a verb phrase because they have the same structure.

There is also ambiguity between coordination and subordinate constructions in Vietnamese. Figure 4.5 shows another VP attachment error caused by the structure of two adjacent verbs. This expression is bracketed as a subordinate construction by RNNGs-G, but it is a coordination in the gold tree. In this figure, we also cannot find any clue to determine whether this expression is a coordinate construction or a subordinate one. In English subordinate constructions, the second verb can be preceded by a *to* or added an inflectional morpheme *-ing*. In Vietnamese, we do not have such clues. In practice, there is a conjunction such as $và_{and}$ or $hoặc_{or}$ or a comma between two words in Vietnamese coordinate constructions. However, it is frequently dropped in real text. These characteristics lead to the situation that coordinate and subordinate constructions have the same structures.

Dropping words is also one of the reasons for the clause attachment error shown in Figure 4.6. We can see that the conjunction introducing the SBAR (gold tree) is dropped. This creates a POS sequence including two adjacent nouns $thành\_quả_{productt}$ $nhà\_nước_{government}$. It is ambiguous to determine the role of the noun $nhà\_nước_{government}$ because it can be a modifier of the previous noun or combines with the later words. We can also see from this example that the role of the coordinate conjunction $và_{government}$ is predicted incorrectly. It is because this conjunction can be used to link two clauses of a compound sentence (as the parsing output) or it can belong to a coordinating phrase (as the gold tree) in Vietnamese.

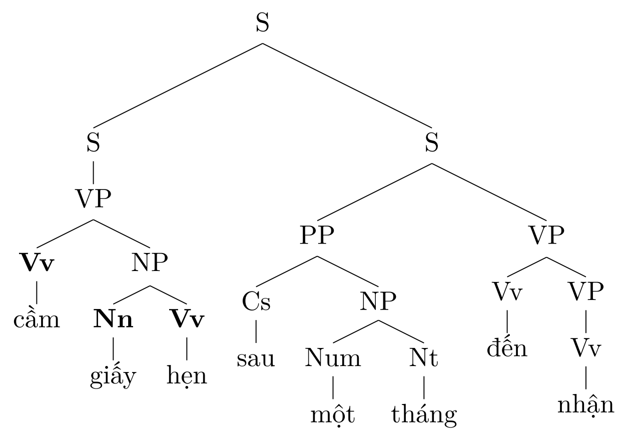
Figure 4.7 presents the construction *Vv Vv NP* that causes NP attachment error in Vietnamese parsing. When a noun phrase follows two verbs, it is difficult to determine that this noun phrase is the modifier of the first verb (as in the gold tree) or the second verb (as the parsing output). This ambiguity is caused by the common word order in Vietnamese, in which modifying lexical words follow the head word. In addition, it also results from the lack of inflectional morphemes. We can see in the figure that this is not an ambiguous construction in English parsing (as well as in Chinese) because cases like $mở\_rộng_{to\ extend}$ are expressed by adverbs standing before the head verb and cannot be combined with a noun. However, Vietnamese does not have adverbs; verbs and adjectives are used in such contexts. It is confusing to bracket these constructions because the noun phrase can modify both.

a) Parser output



{to hold the note of appointment, to come here to receive after a month}

b) Gold tree



{to hold the note of appointment, to come here to receive after a month}

FIGURE 4.4: Examples illustrating an ambiguity between subordinate and separate constructions in Vietnamese.

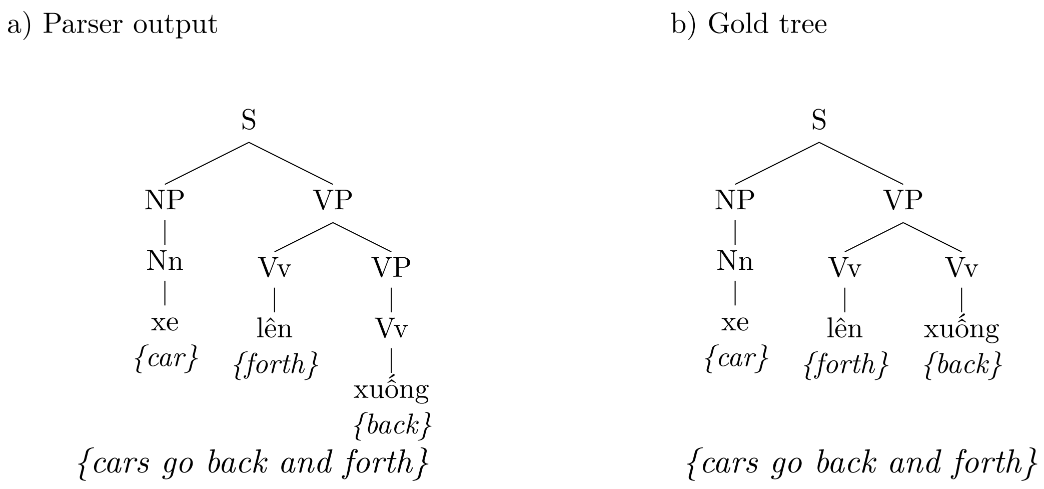


FIGURE 4.5: Examples illustrating ambiguity between coordination and subordinate construction in Vietnamese.

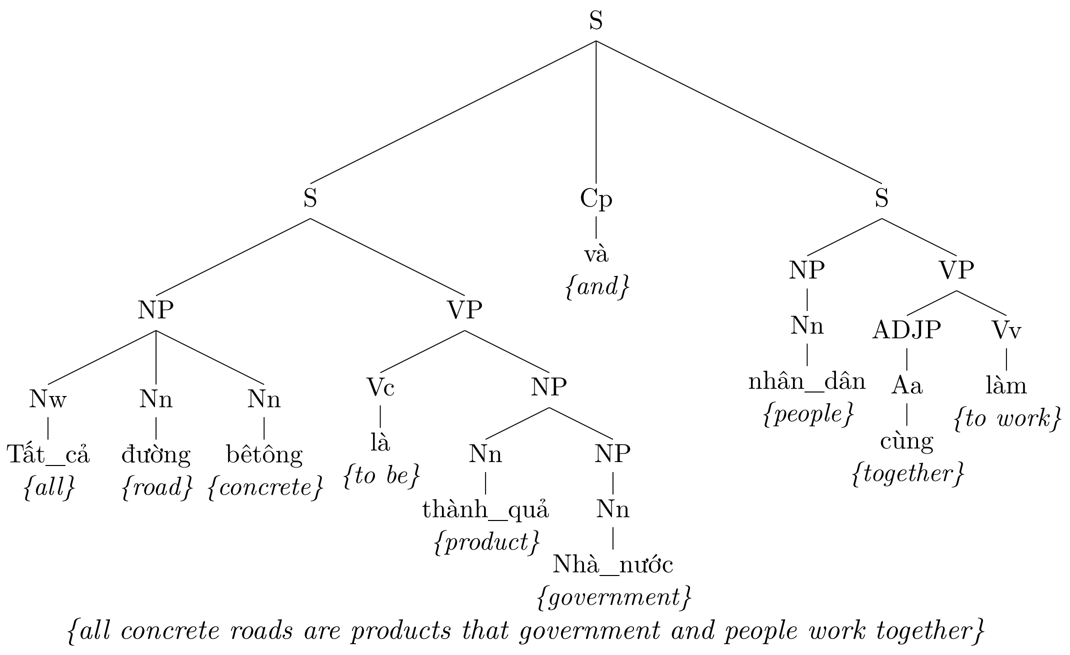
Figure 4.8 presents an example to illustrate how an adjective in Vietnamese plays the same role as an adverb in English. We can see that the adjective modifying the head verb is placed after the head verb. This causes the confusion in deciding whether a noun phrase should modify the previous adjective or the head verb. RNNGs-G considered this noun phrase as a modifier of the previous adjective (as shown in the parsing output). Meanwhile, it should annotate the adjective and the noun phrase as separate constructions (as in the gold tree).

Figure 4.9 presents a PP attachment error caused by the ambiguous construction *Vv Nn PP*. It is confusing because PP can modify the head verb or attached to the previous noun. This error results from the polysemous preposition *để*, which can be understood as *to* or *for*. While prepositional phrases having the head word *để*$_{to}$ are frequently attached to the head verbs, the other ones with the head word *để*$_{for}$ commonly modify the previous nouns.

4.6.2 "Other" class

The automatic analysis tool classified 20.1% bracketing errors into the "Other" class. Manually investigating these errors based on 100 output sentences shows that the major sources for them are similar to those of the above-mentioned error types. Several errors can be classified into the existing types such as clause attachment and VP attachment. For example, the error in Figure 4.10 can be considered as a clause attachment error, in which the clause S_2 should be attached to the root S (as in the gold tree) rather than the verb *về*$_{to\ come\ back}$ (as in the parsing output). In Vietnamese, the conjunction pair *nếu*$_{if}$ ... *thì*$_{then}$ is used to link two clauses of a compound sentence. Sometimes, *thì* can

a) Parser output



b) Gold tree

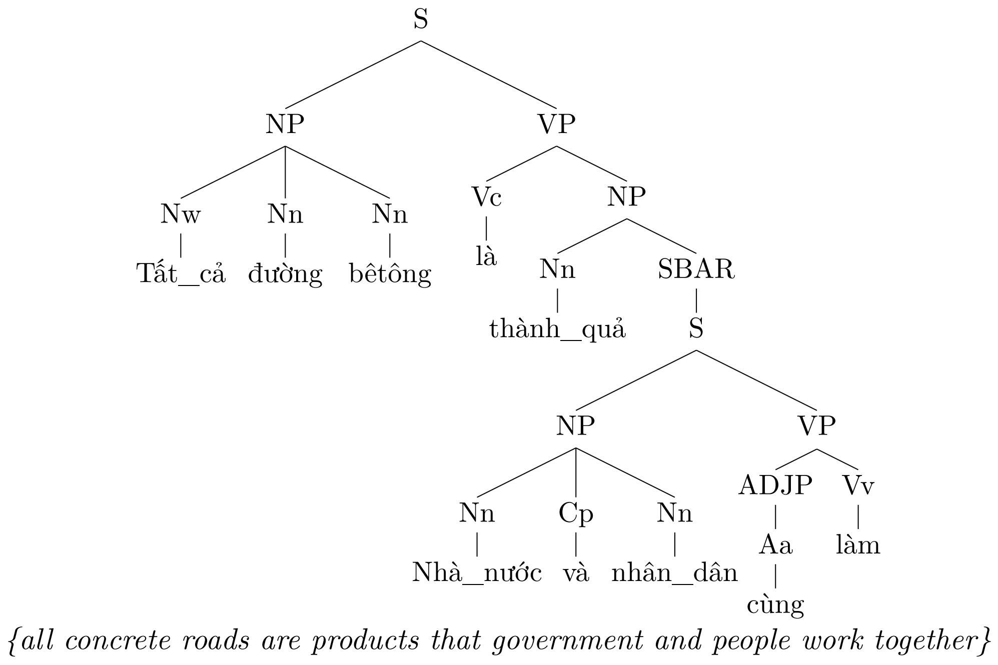


FIGURE 4.6: Examples illustrating a clause attachment error in Vietnamese.

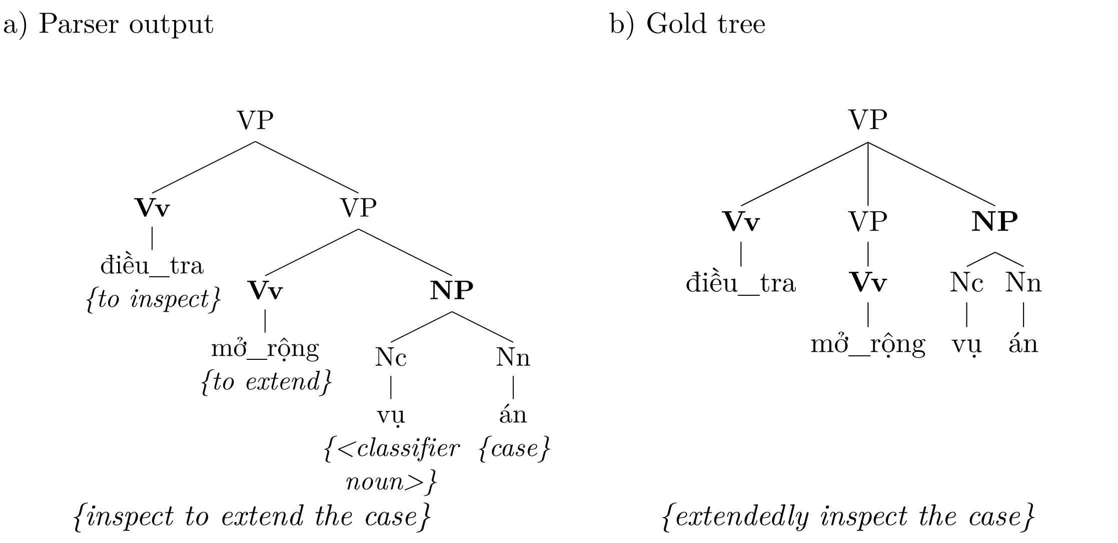


FIGURE 4.7: Examples illustrating the ambiguity between subordinate and separate constructions in Vietnamese.

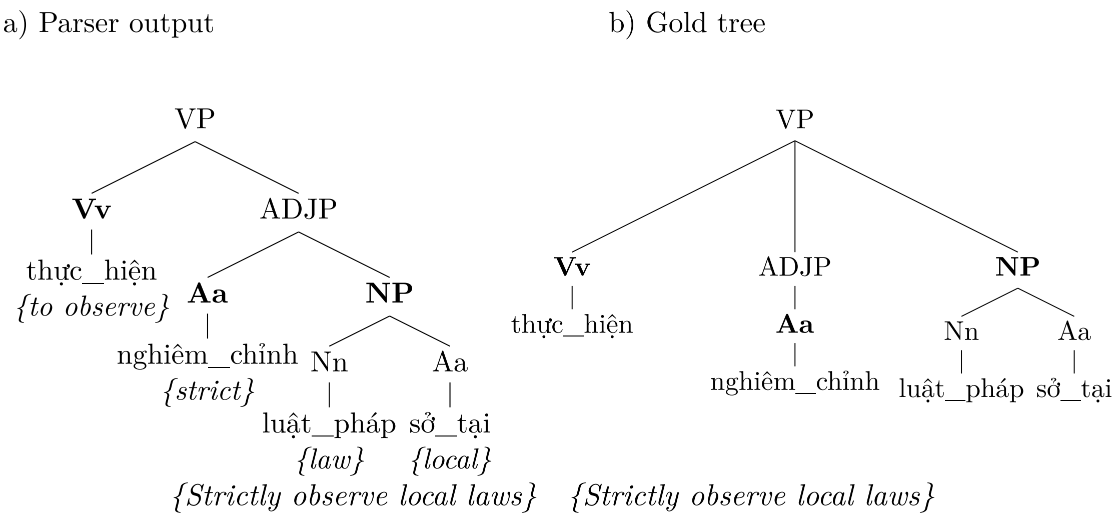
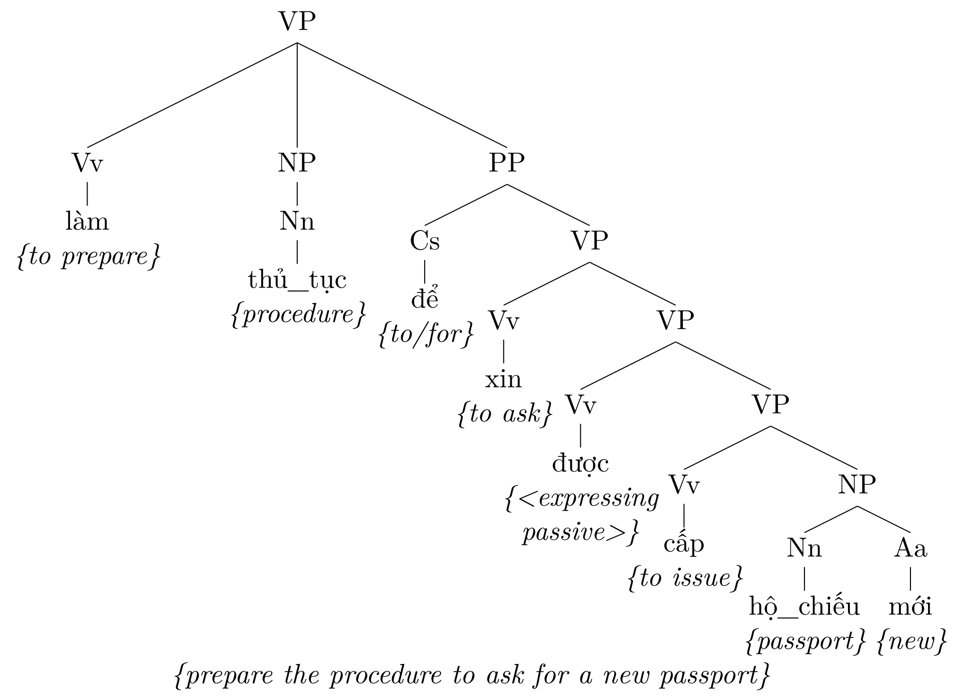


FIGURE 4.8: Examples illustrating the ambiguity between separate constructions and subordinate constructions that caused NP attachment errors in Vietnamese parsing.

be replaced by a comma. However, we can see from Figure 4.10 that the second linking word (*thì*$_{then}$ or comma) and the subject of the second clause were omitted. This created a construction similar to the verb phrase that includes two or more adjacent verbs, as mentioned above. It is ambiguous to determine whether the second verb modifies the previous verb or they should be annotated separately. One possible reason for such incorrect classification is that this construction does not appear in English.

In addition, we also found several errors caused by incorrect attachment of adjectives. Figure 4.11 presents an illustrating example for this. Similarly to the cases mentioned above, this error is also caused by the word order in which modifying lexical words (namely adjective phrases) follow the head noun in a noun phrase. Another reason is

a) Parser output



b) Gold tree

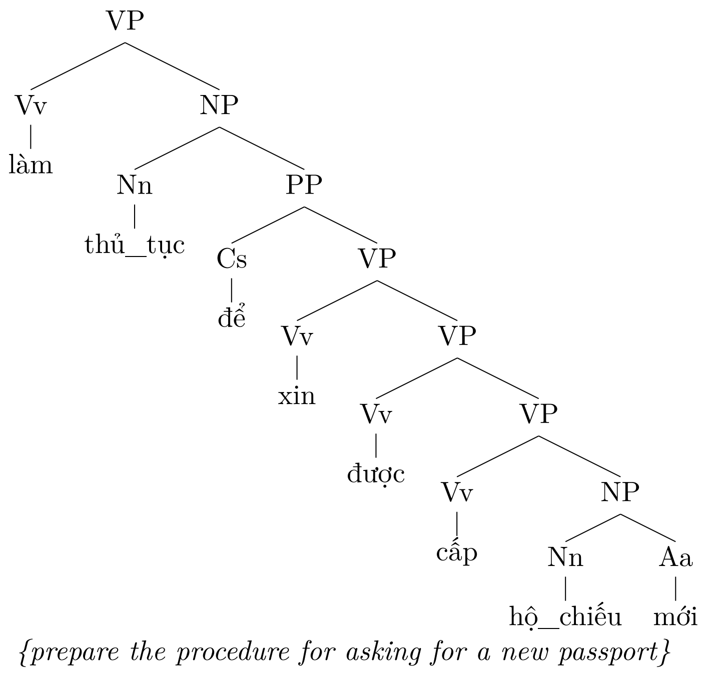
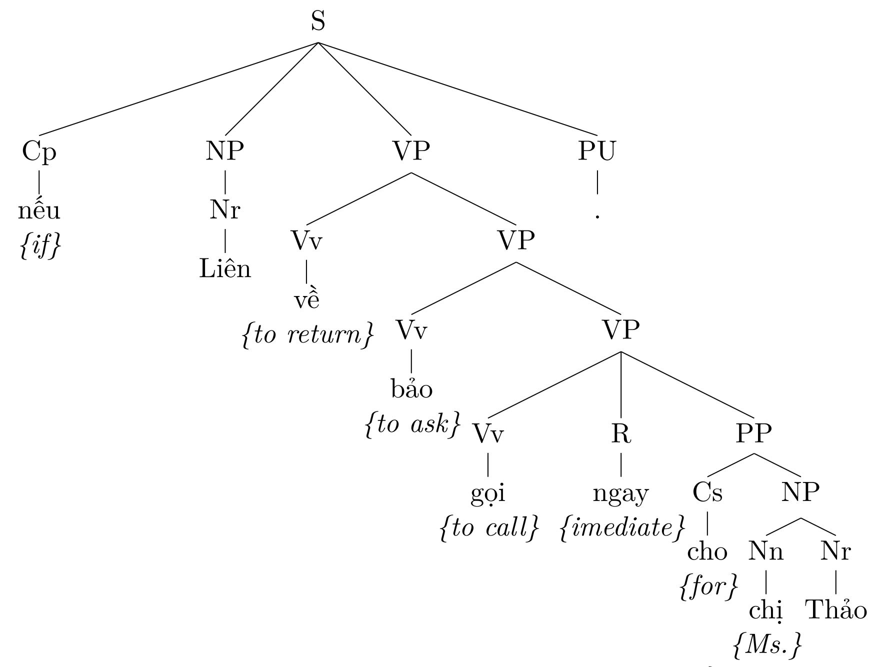


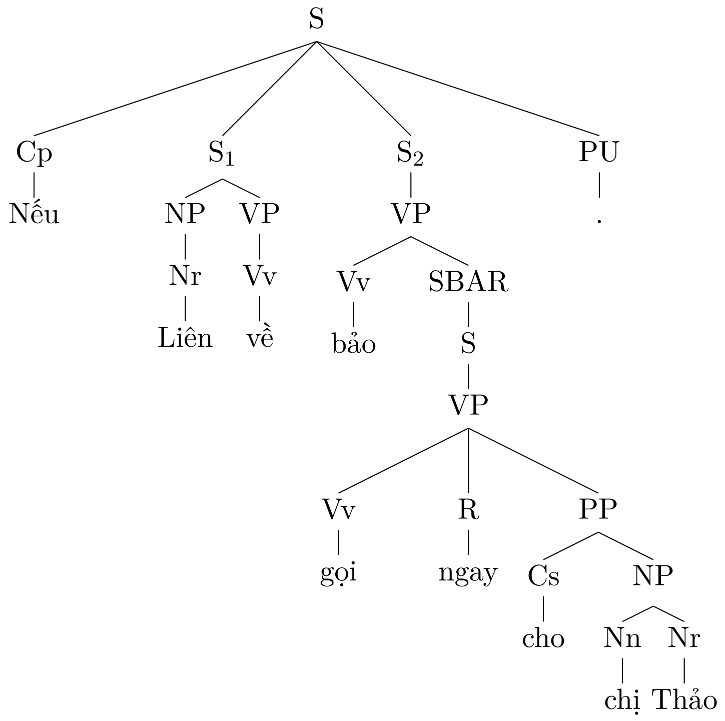
Figure 4.9: Examples illustrating the ambiguity in determining the head of a prepositional phrase.

a) Parser output



{If Lien returns, you ask her to call Ms. Thao immediately.}

b) Gold tree



{If Lien returns, you ask her to call Ms. Thao immediately.}

FIGURE 4.10: Examples illustrating a clause attachment error that cannot be recognized by the analysis tool.

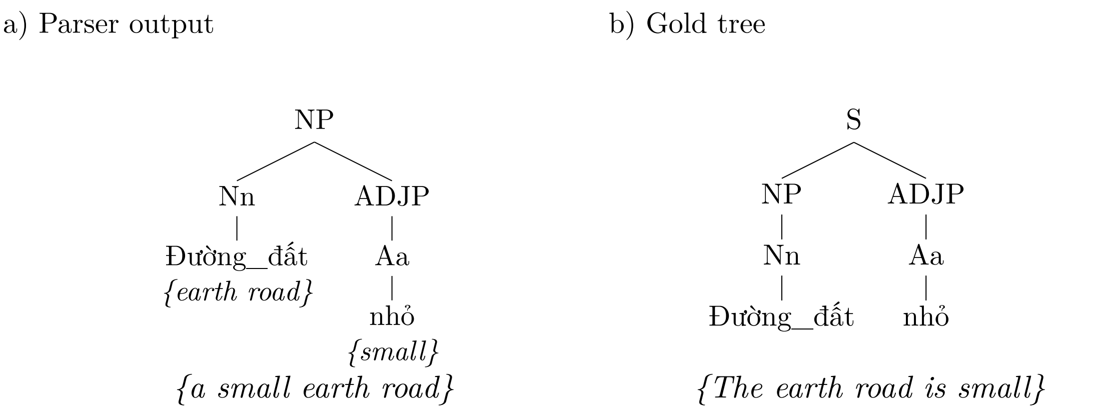
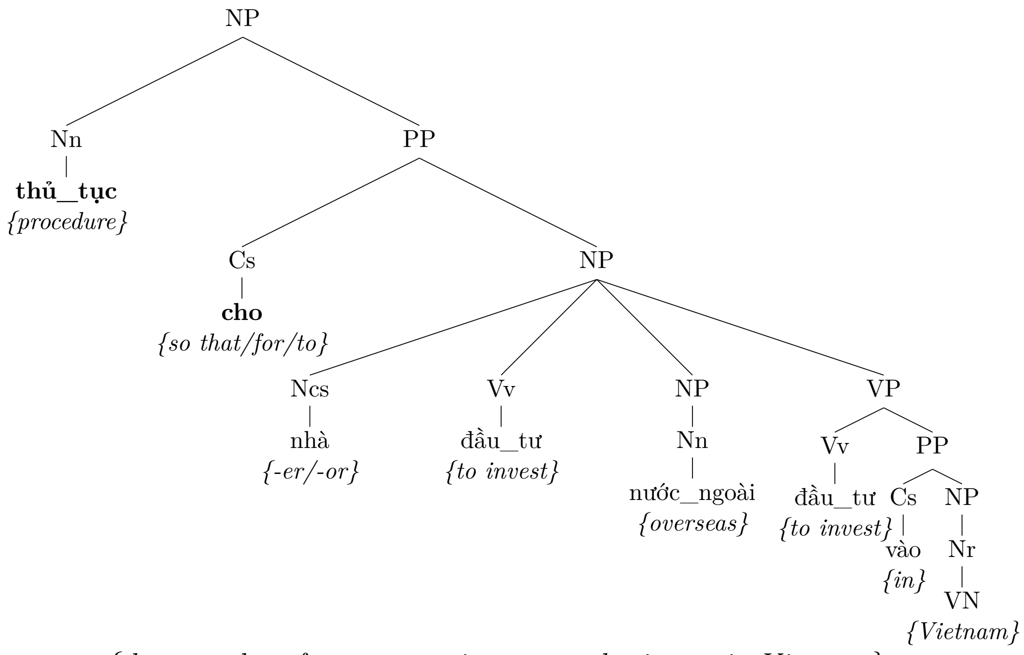


Figure 4.11: Examples illustrating an adjective attachment error that cannot be recognized by the annalysis tool.

that the predicate of a simple sentence is not necessarily a verb phrase; it can be an adjective phrase, a prepositional phrase, a noun phrase, or a clause. This causes the ambiguity between a noun phrase and a simple sentence because they have the same tag sequence.

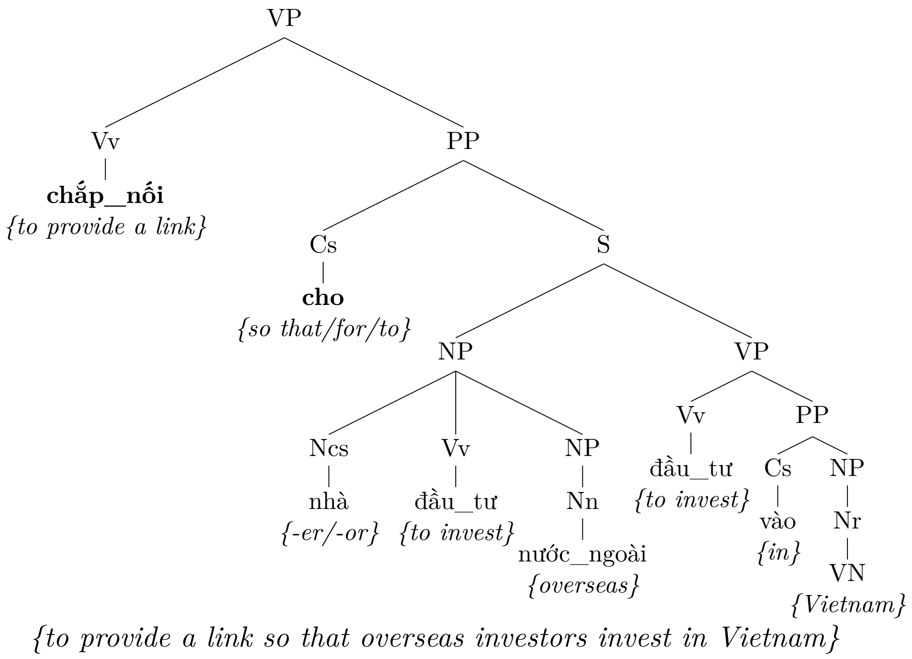
In summary, ambiguous constructions are the major contributor to errors in Vietnamese parsing. Statistics on top four frequent error types, VP, NP, PP, and clause attachments, indicated that there is about 63% of attachment errors caused by the frequent ambiguous constructions in Vietnamese. These ambiguities are resulted from the characteristics of Vietnamese language such as the lack of inflectional morphemes, polysemous words, dropping words, and word order in which modifying lexical words follow the head word. However, we find that these ambiguities can be addressed based on the context. For example, as we mentioned above, the polysemous word *cho* is one of the reasons that caused the ambiguity in Figure 4.3. This ambiguity will be solved if the meaning of the word *cho* is determined. If *cho* means *for* or *to*, the expression of *Ncs Vv Nn VP* is a noun phrase in which VP is a post-modifier. In the case that *cho* is interpreted as *so that*, this expression is a simple sentence where VP is the predicate. Determining the meaning of the word *cho* can be relied on the word that it modifies. For example, the word *cho* is interpreted as *for* when it modifies the word $\textit{thủ\_tục}_{\textit{procedure}}$ (Figure 4.12a). However, *cho* means *so that* when it modifies the word $\textit{chắp\_nối}_{\textit{to make a link}}$ (Figure 4.12b).

While solving the ambiguity in Figure 4.3 needs to consider the meaning of the surrounding words, the contextual information used to disambiguate the expression *Vv Nn Vv* in Figure 4.4 can be found within the expression itself. We can see that because Vietnamese does not have inflectional morphemes, the verb $\textit{hẹn}_{\textit{to make an appointment}}$ plays the same role as a specific noun that modifies the categorization noun $\textit{giấy}_{\textit{paper}}$ (as in the gold tree). Therefore, we will bracket the expression *Nn Vv* in Figure 4.4 as a subordinate a)



{the procedure for overseas investors who invest in Vietnam}

b)



{to provide a link so that overseas investors invest in Vietnam}

FIGURE 4.12: Examples illustrating how to determine the meaning of the preposition *cho*.

construction modifying the first verb if information to recognize the categorization noun *giấy*$_{paper}$ is provided. We found that categorization nouns can be distinguished from the other nouns by tag splitting. In the Vietnamese Treebank, although categorization nouns were annotated by the same POS tag as other common nouns (Nn), expressions of the categorization noun and specific noun were bracketed with the internal structure tag *Nn_swsp*. We can use this annotation for splitting POS tags.

Although Vietnamese has many confusing constructions, these ambiguities can be tackled on the basis of the contextual information such as words playing roles as prefixed or suffixes, function words, fine-grained categorizations, the head words of phrases, the main verbs of clauses, etc. Therefore, to improve the quality of Vietnamese parsing, we need to study new methods that can capture these contextual information.

4.7 Conclusion

To date, challenges of Vietnamese parsing have not been studied thoroughly. We tried to alleviate such situation by evaluating representative parsing models on the Vietnamese Treebank. Then by quantifying the error types produced by the parsers, we could capture the behaviour of parsers, viz parsing models based on CRF and neural network are good for Vietnamese; contextual information is very beneficial for improving the parsing performance. This investigation has revealed that there are four frequent errors in Vietnamese parsing: VP attachment, NP attachment, PP attachment, and clause attachment.

Our investigation has also shown that we could not expect a significant improvement of Vietnamese parsing by enlarging the training data. Although improvement of word segmentation is beneficial for parsing, the gains are insignificant. The POS tagging improvement can enhance the performance of the parser about 5% in F-score. However, it could not address two of the most frequent error types in Vietnamese parsing, VP attachment and NP attachment.

In addition, our investigation on parsing errors has revealed that ambiguous constructions are the major contributor to the parsing errors in Vietnamese. Statistics on the top four frequent error types, VP, NP, PP, and clause attachments, has indicated that there is about 63% of attachment errors caused by ambiguous constructions. These ambiguities resulted from the characteristics of the Vietnamese language such as the lack of inflectional morphemes, polysemous words, dropping words, and word order in which modifying lexical words follows the head word. However, these errors can be addressed by using contextual information such as fine-grained categorizations and objects of verbs. To improve the quality of Vietnamese parsing, we need to study new parsing methods that suit the characteristics of the Vietnamese language, so that parsers can capture the necessary contextual information for parsing Vietnamese constructions.

Chapter 5

Conclusion and future work

5.1 Conclusion

Treebanks and parsers play crucial roles in the growth of natural language processing (NLP) and linguistic research. To impulse the development of NLP on Vietnamese as well as other resource-inadequate languages, we built a high-quality Vietnamese treebank while ensuring a reasonable annotation speed. In addition, we evaluated our treebank on different parsing methods and investigated the sources of errors produced by the parsers.

Our efforts to build a high-quality Vietnamese treebank were presented in Chapter 3. In that chapter, we described challenges of Vietnamese language processing and our methods to address them. We developed three clear, consistent, and complete annotation guidelines, including WS guidelines (44 pages), POS tagging guidelines (73 pages), and bracketing guidelines (182 pages). These guidelines are valuable resources that serve the training of annotators and the use of the treebank. The quantity of rules in the guidelines is as follows.

- 9 rules for segmenting ambiguous expressions
- 34 rules for tagging ambiguous words
- 39 rules for bracketing ambiguous expressions

In addition to the annotation guidelines, Chapter 3 also described issues of ensuring the quality of the treebank, including appropriate processes of annotator training and annotation as well as software tools to support the annotation and control the quality.

Our treebank is reliable and satisfied. Inter-annotator agreement, intra-annotator agreement, and accuracy are higher than 90%. Our treebank is also more consistent and accurate than VLSP, the only existing Vietnamese treebank, although our annotation scheme is more complicated. Typically, we defined twice more POS tags than those of VLSP, but still, our treebank helped achieve better accuracy on important NLP tasks: automatic word segmentation, POS tagging, and parsing.

Evaluating representative parsing models on our treebank was described in Chapter 4. Among the tested parsers, RNNGs-G [74], a probabilistic model of phrase-structure trees based on recurrent neural network, achieves the highest accuracy. Other context-aware methods such as hierarchical state-splitting for unlexicalized parsing [13] and exploiting the rich input features of the surface spans for CRF parsing [72], also obtained competitive results. However, their F1 score (0.72) is lower than that of testing in English (0.90) and Chinese (0.86). The main reason is that those parsers cannot capture all contextual information in Vietnamese sentences.

To find the detailed reasons for the low performance of the Vietnamese parsing, we investigated the possible issues including a small size of treebank, word segmentation errors, confusing POS tags, and confusing constructions. The results showed that we could not expect a significant improvement by enlarging the training data. Although word segmentation errors and tagging confusions affect the parsing results, confusing constructions are the major cause of the low performance. Confusing constructions in Vietnamese appear in many forms, such as ambiguous POS sequences or ambiguous symbol sequences. They are caused by the characteristics of Vietnamese such as lack of inflectional morphemes, post-head modifying lexical words, and dropping words. Although Vietnamese has many confusing constructions, these ambiguities can be solved based on the contextual information such as the words playing the roles as prefixed and suffixes, function words, fine-grained categorizations, headwords of the phrases, main verbs of the clauses, etc.

Results of this thesis are worth contributions for the development of the Vietnamese language processing. Our treebank provides resources so that different algorithms can be tested and improved. Our parsing error analyses provide clues to modify parsers to capture the necessary information for syntax modeling. We envision that our research is not only beneficial for Vietnamese language processing but also for similar languages such as Thai, Laos, and so on.

5.2 Future work

As we mentioned in Chapter 2, a treebank can include three annotations: constituent annotation, functional annotation, and semantic annotation. Our treebank has been annotated with the constituent structure and grammar functions. In the future, we will extend our treebank with the semantic information so that it can serve a broader range of NLP problems. We will also convert our treebank into other structures such as dependency structure, HPSG [139], and CCG [54, 140].

For the parsing problem, we will study new parsing techniques as well as methods for enriching the contextual information to address the confusing constructions in Vietnamese. In addition, the quality of Vietnamese word segmentation and POS tagging needs to be improved. Given high-quality tools for word segmentation, POS tagging, and parsing, we can build advanced NLP systems for Vietnamese, such as machine translation and question answering.

Bibliography

[1] Quang-Ban Diep. *Vietnamese grammar.* Vietnam Education Publisher, 2005.

[2] SCSSV. *Vietnamese grammar.* Social Sciences Publishers, 1983.

[3] Phuong-Thai Nguyen, Anh-Cuong Le, Tu-Bao Ho, and Van-Hiep Nguyen. Vietnamese treebank construction and entropy-based error detection. *Language Resources and Evaluation*, pages 1–33, 2015.

[4] Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. Training a parser for machine translation reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 183–192. Association for Computational Linguistics, 2011.

[5] Jingsheng Cai, Masao Utiyama, Eiichiro Sumita, and Yujie Zhang. Dependency-based pre-ordering for chinese-english machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 155–160. Association for Computational Linguistics, 2014.

[6] Sho Hoshino, Yusuke Miyao, Katsuhito Sudoh, Katsuhiko Hayashi, and Masaaki Nagata. Discriminative preordering meets kendall's tau maximization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 139–144. Association for Computational Linguistics, 2015.

[7] Valentin Jijkoun, Maarten De Rijke, and Jori Mur. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1284. Association for Computational Linguistics, 2004.

[8] Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. Using syntactic information for improving why-question answering. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 953–960. Association for Computational Linguistics, 2008.

[9] Boris Galitsky, Dmitry I Ilvovsky, Sergei O Kuznetsov, and Fedor Strok. Matching sets of parse trees for answering multi-sentence questions. In *Proceedings of RANLP*, pages 285–293, 2013.

[10] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[11] Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Kazama. Learning with lookahead: can history-based models rival globally optimized models? In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 238–246. Association for Computational Linguistics, 2011.

[12] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[13] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics, 2006.

[14] Yusuke Miyao and Jun'ichi Tsujii. Feature forest models for probabilistic hpsg parsing. *Computational Linguistics*, 34(1):35–80, 2008.

[15] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *Proceedings of the ACL conference*. Citeseer, 2013.

[16] Chung-hye Han, Na-Rare Han, Eon-Suk Ko, and Martha Palmer. Development and evaluation of a korean treebank and its application to nlp. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 1635–1642, 2002.

[17] Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238, 2005.

[18] Anne Abeillé, Lionel Clément, and François Toussenel. Building a treebank for french. *Treebanks*, pages 165–187, 2003.

[19] Quy T. Nguyen, Ngan L.T. Nguyen, and Yusuke Miyao. Comparing different criteria for vietnamese word segmentation. In *Proceedings of 3rd Workshop on South and Southeast Asian Natural Language Processing (SANLP)*, pages 53–68. Citeseer, 2012.

[20] Quy T. Nguyen, Ngan L.T. Nguyen, and Yusuke Miyao. Utilizing state-of-the-art parsers to diagnose problems in treebank annotation for a less resourced language. In *Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*, pages 19–27. Association for Computational Linguistics, 2013.

[21] Hong-Phuong HP. Le, Thi-Minh-Huyen Nguyen, and Azim Roussanaly. Vietnamese parsing with an automatically extracted tree-adjoining grammar. In *Proceedings of Research, Innovation and Vision for the Future in Computing and Communication Technologies (RIVF)*, pages 1–6. IEEE, 2012.

[22] Anh-Cuong AC. Le, Phuong-Thai Nguyen, Hoai-Thu Vuong, Minh-Thu Pham, and Tu-Bao Ho. An experimental study on lexicalized statistical parsing for vietnamese. In *Proceedings of Knowledge and Systems Engineering*, pages 162–167. IEEE, 2009.

[23] Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, Phuong-Thai Nguyen, and Minh Le Nguyen. From treebank conversion to automatic dependency parsing for vietnamese. In *International Conference on Applications of Natural Language to Data Bases/Information Systems*, pages 196–207. Springer, 2014.

[24] Noam Chomsky. *Aspects of the Theory of Syntax*, volume 11. MIT press, 1965.

[25] Roger Garside, Geoffrey Leech, and Tamás Váradi. The lancaster parsed corpus. a machine-readable syntactically analyzed corpus of 144,000 words. available for distribution through icame. Technical report, Technical Report. Bergen: The Norwegian Computing Centre for the Humanities, 1992.

[26] Igor Aleksandrovič Melčuk. *Dependency syntax: theory and practice*. SUNY press, 1988.

[27] Sadao Kurohashi and Makoto Nagao. Building a japanese parsed corpus while improving the parsing system. In *Proceedings of The 1st International Conference on Language Resources & Evaluation*, pages 719–724. Citeseer, 1998.

[28] Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. Building a turkish treebank. *Treebanks*, pages 261–277, 2003.

[29] Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. The prague dependency treebank. In *Treebanks*, pages 103–127. Springer, 2003.

[30] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics, 1994.

[31] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467, 2004.

[32] A Moreno and S López. Developing a spanish treebank, 2003.

[33] Minh Nghiem, Dien Dinh, and Mai Nguyen. Improving vietnamese pos tagging by integrating a rich feature set and support vector machines. In *Proceedings of Research, Innovation and Vision for the Future in Computing and Communication Technologies (RIVF)*, pages 128–133. IEEE, 2008.

[34] Phuong Le-Hong, Azim Roussanaly, Thi Minh Huyen Nguyen, and Mathias Rossignol. An empirical study of maximum entropy approach for part-of-speech tagging of vietnamese texts. In *Traitement Automatique des Langues Naturelles-TALN 2010*, page 12, 2010.

[35] Hieu Le Trung, Vu Le Anh, and Kien Le Trung. An unsupervised learning and statistical approach for vietnamese word recognition and segmentation. In *Asian Conference on Intelligent Information and Database Systems*, pages 195–204. Springer, 2010.

[36] Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Fu-Dong Chiou, Shizhe Huang, Tony Kroch, and Mitchell P Marcus. Developing guidelines and ensuring consistency for chinese text annotation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, 2000.

[37] Juri Apresjan, Igor Boguslavsky, Boris Iomdin, Leonid Iomdin, Andrei Sannikov, and Victor Sizov. A syntactically and semantically tagged corpus of russian: State of the art and prospects. In *Proceedings of LREC*, pages 1378–1381, 2006.

[38] Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Genoa, 2006.

[39] Lingpeng Kong, Alexander M Rush, and Noah A Smith. Transforming dependencies into phrase structures. In *Human Language Technologies: The 2015 Annual*

Conference of the North American Chapter of the ACL, page 788–798. Association for Computational Linguistics, 2015.

[40] Sebastian Schuster and Christopher D Manning. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016.

[41] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, 2016.

[42] Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–92, 2014.

[43] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*, 2011.

[44] Daniel Zeman. Reusable tagset conversion using tagset drivers. In *LREC*, 2008.

[45] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168, 2002.

[46] Simonetta Montemagni, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, et al. Building the italian syntactic-semantic treebank. *Treebanks*, pages 189–210, 2003.

[47] Keh-Jiann Chen, Chi-Ching Luo, Ming-Chung Chang, Feng-Yi Chen, Chao-Jan Chen, Chu-Ren Huang, and Zhao-Ming Gao. Sinica treebank. *Treebanks*, pages 231–248, 2003.

[48] Cristina Bosco and Vincenzo Lombardo. Dependency and relational structure in treebank annotation. In *Proceedings of Workshop on Recent Advances in Dependency Grammar at COLING'04*, 2004.

[49] Anke Lüdeling. *Corpus linguistics*, volume 1. Walter de Gruyter, 2008.

[50] Carl Pollard and Ivan A Sag. *Head-driven phrase structure grammar*. University of Chicago Press, 1994.

[51] Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D Manning. Lingo redwoods. *Research on Language and Computation*, 2(4):575–596, 2004.

[52] Kiril Simov, Gergana Popova, and Petya Osenova. Hpsg-based syntactic treebank of bulgarian (bultreebank). *A rainbow of corpora: Corpus linguistics and the languages of the world*, pages 135–142, 2002.

[53] Mark Steedman and Jason Baldridge. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*, 2011.

[54] Julia Hockenmaier and Mark Steedman. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396, 2007.

[55] Julia Hockenmaier. Creating a ccgbank and a wide-coverage ccg lexicon for german. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 505–512. Association for Computational Linguistics, 2006.

[56] Weiwei Sun and Jia Xu. Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979. Association for Computational Linguistics, 2011.

[57] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. Gated recursive neural network for chinese word segmentation. In *ACL (1)*, pages 1744–1753, 2015.

[58] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*, pages 1197–1206, 2015.

[59] Nobuhiro Kaji and Masaru Kitsuregawa. Accurate word segmentation and pos tagging for japanese microblogs: Corpus annotation and joint modeling with lexical normalization. In *EMNLP*, pages 99–109, 2014.

[60] Quang Thang Dinh, Hong Phuong Le, Thi Minh Huyen Nguyen, Cam Tu Nguyen, Mathias Rossignol, and Xuan Luong Vu. Word segmentation of vietnamese texts: a comparison of approaches. In *Proceedings of 6th international conference on Language Resources and Evaluation*, 2008.

[61] Nguyen Thi Minh Huyen, Azim Roussanaly, Hô Tuong Vinh, et al. A hybrid approach to word segmentation of vietnamese texts. In *International Conference*

on Language and Automata Theory and Applications, pages 240–249. Springer, 2008.

[62] Chaluemwut Noyunsan, Choochart Haruechaiyasak, Seksan Poltree, and Kanda Runapongsa Saikaew. A multi-aspect comparison and evaluation on thai word segmentation programs. In *JIST (Workshops & Posters)*, pages 132–135, 2014.

[63] Chenchen Ding, Ye Kyaw Thu, Masao Utiyama, and Eiichiro Sumita. Word segmentation for burmese (myanmar). *ACM Transactions on Asian and Low-Resource Language Information Processing*, 15(4):22, 2016.

[64] Jinho D Choi. Dynamic feature induction: The last gist to the state-of-the-art. In *Proceedings of NAACL-HLT*, pages 271–281, 2016.

[65] Pascal Denis, Benoît Sagot, et al. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art pos tagging with less human effort. In *PACLIC*, pages 110–119, 2009.

[66] Weiwei Sun and Xiaojun Wan. Towards accurate and efficient chinese part-of-speech tagging. *Computational Linguistics*, 2016.

[67] Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 213–216. Association for Computational Linguistics, 2009.

[68] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 529–533. Association for Computational Linguistics, 2011.

[69] Nobuhiro Kaji and Masaru Kitsuregawa. Efficient word lattice generation for joint word segmentation and pos tagging in japanese. In *IJCNLP*, pages 153–161, 2013.

[70] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *HLT-NAACL*, volume 7, pages 404–411. Association for Computational Linguistics, 2007.

[71] Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. An annotation scheme for free word order languages. In *Proceedings of the fifth conference on*

Applied natural language processing, pages 88–95. Association for Computational Linguistics, 1997.

[72] David Hall, Greg Durrett, and Dan Klein. Less grammar, more features. In *Proceedings of the 52rd Annual Meeting of the Association for Computational Linguistics*, pages 228–237, 2014.

[73] Greg Durrett and Dan Klein. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, page 302–312. Association for Computational Linguistics, 2015.

[74] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. In *Proceedings of NAACL-HLT 2016*, page 199–209. Association for Computational Linguistics, 2016.

[75] Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[76] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015.

[77] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, 2014.

[78] Zhiguo Wang, Haitao Mi, and Nianwen Xue. Feature optimization for constituent parsing via neural networks. In *Proceedings of ACL*, pages 1138–1147, 2015.

[79] Pi-Chuan Chang, Michel Galley, and Christopher D Manning. Optimizing chinese word segmentation for machine translation performance. In *Proceedings of the third workshop on statistical machine translation*, pages 224–232. Association for Computational Linguistics, 2008.

[80] Alexandre Allauzen, Lauriane Aufrant, Franck Burlot, Elena Knyazeva, Thomas Lavergne, and François Yvon. Limsi@ wmt'16: Machine translation of news. In *Proceedings of the First Conference on Machine Translation*, pages 239–245. Association for Computational Linguistics, 2016.

[81] Fuchun Peng and Xiangji Huang. Machine learning for asian language text classification. *Journal of Documentation*, 63(3):378–397, 2007.

[82] Alex Chengyu Fang and Jing Cao. Enhanced genre classification through linguistically fine-grained pos tags. In *Proceedings of PACLIC*, pages 85–94, 2010.

[83] Maria Chinkina, Madeeswaran Kannan, and Detmar Meurers. Online information retrieval for language learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics—System Demonstrations*, pages 7–12, 2016.

[84] Cory Barr, Rosie Jones, and Moira Regelson. The linguistic structure of english web-search queries. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1021–1030. Association for Computational Linguistics, 2008.

[85] Beatrice Santorini. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *University of Pennsylvania*, 1990.

[86] Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. Bracketing guidelines for treebank ii style penn treebank project. *University of Pennsylvania*, 1995.

[87] Fei Xia. The segmentation guidelines for the penn chinese treebank (3.0). technical report ircs 00-06. *University of Pennsylvania*, 2000.

[88] Fei Xia. The part-of-speech tagging guidelines for the penn chinese treebank (3.0). technical report ircs 00-07. *University of Pennsylvania*, 2000.

[89] Nianwen Xue, Fei Xia, Shizhe Huang, and Anthony Kroch. The bracketing guidelines for the penn chinese treebank (3.0). technical report ircs 00-08. *University of Pennsylvania*, 2000.

[90] Jarborg Jerker. Manual for syntaggning. *Technical report, Gothenburg University, Department of Swedish.*, 1986.

[91] Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 system demonstrations*, pages 169–174. Association for Computational Linguistics, 2012.

[92] Gertjan Van Noord, Gosse Bouma, Frank Van Eynde, Daniel De Kok, Jelmer Van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste. Large scale syntactic annotation of written dutch: Lassy. In *Essential Speech and Language Technology for Dutch*, pages 147–164. Springer, 2013.

[93] Gülşen Eryiğit. Itu treebank annotation tool. In *Proceedings of the Linguistic Annotation Workshop*, pages 117–120. Association for Computational Linguistics, 2007.

[94] Markus Dickinson and W Detmar Meurers. Detecting inconsistencies in treebanks. In *Proceedings of TLT*, volume 3, pages 45–56, 2003.

[95] Alexander Volokh and Günter Neumann. Automatic detection and correction of errors in dependency tree-banks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 346–350. Association for Computational Linguistics, 2011.

[96] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: an overview. In *Treebanks*, pages 5–22. Springer, 2003.

[97] Tugba Pamay, Umut Sulubacak, Dilara Torunoglu-Selamet, and Gülsen Eryigit. The annotation process of the itu web treebank. In *The 9th Linguistic Annotation Workshop held in conjuncion with NAACL 2015*, page 95, 2015.

[98] Daniel Gildea. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202, 2001.

[99] Chenhui Chu, Toshiaki Nakazawa, Daisuke Kawahara, and Sadao Kurohashi. Sctb: A chinese treebank in scientific domain. In *The 12th Workshop on Asian Language Resources*, page 59, 2016.

[100] Geoffrey Sampson. English for the computer: The susanne corpus and analytic scheme, 1995.

[101] Zhongqiang Huang and Mary Harper. Self-training pcfg grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 832–841. Association for Computational Linguistics, 2009.

[102] Michael Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003.

[103] Jenny Rose Finkel, Alex Kleeman, and Christopher D Manning. Efficient, feature-based, conditional random field parsing. In *ACL*, volume 46, pages 959–967, 2008.

[104] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[105] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[106] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[107] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781, 2015.

[108] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[109] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

[110] Jonathan K Kummerfeld, David Hall, James R Curran, and Dan Klein. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059. Association for Computational Linguistics, 2012.

[111] E Black, S Abney, D Flickinger, C Gnadiec, R Grishman, P Harrison, D Hindle, R Ingria, F Jelinek, J Klavans, et al. A procedure for quantitatively comparing the syntactic coverage of english. In *Proceedings DARPA Speech and Natural Language Workshop, Pacific Grove, Morgan Kaufmann*, 1991.

[112] Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics, 1997.

[113] Slav Petrov and Dan Klein. Discriminative log-linear grammars with latent variables. In *NIPS*, pages 1153–1160, 2007.

[114] Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic cfg with latent annotations. In *Proceedings of the 43rd annual meeting on Association for Computational Linguistics*, pages 75–82. Association for Computational Linguistics, 2005.

[115] Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. Joint evaluation of morphological segmentation and syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 6–10. Association for Computational Linguistics, 2012.

[116] Geoffrey Sampson and Anna Babarczy. A test of the leaf-ancestor metric for parse accuracy. *Natural Language Engineering*, 9(04):365–380, 2003.

[117] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[118] Dekang Lin. Dependency-based evaluation of minipar. In *Workshop on the Evaluation of Parsing Systems, Granada, Spain*, 1995.

[119] Sandra Kübler and Heike Telljohann. Towards a dependency-oriented evaluation for partial parsing. In *Proceedings of Beyond PARSEVAL–Towards Improved E-valuation Measures for Parsing Systems (LREC 2002 Workshop)*. Citeseer, 2002.

[120] Ines Rehbein and Josef Van Genabith. Evaluating evaluation measures. 2007.

[121] Jonathan K Kummerfeld, Daniel Tse, James R Curran, and Dan Klein. An empirical examination of challenges in chinese parsing. In *ACL (2)*, pages 98–103, 2013.

[122] Phe Hoang. *Vietnamese Dictionary.* Scientific & Technical Publishing, 1998.

[123] DCS LacViet Corp. *Vietnamese Dictionary.* LacViet Corp., 2011.

[124] Thi-Minh-Huyen Nguyen, Thi-Tuyen-Linh Hoang, and Xuan-Luong Vu. Vietnamese word segmentation guidelines. technical report sp8.2. *Ministry of Education and Training (Vietnam)*, 2010.

[125] Phuong-Thai PT. Nguyen, Xuan-Luong Vu, and Thi-Minh-Huyen Nguyen. Vietnamese part-of-speech tagging guidelines. technical report sp 7.3. *Ministry of Education and Training (Vietnam)*, 2010.

[126] Phuong-Thai PT. Nguyen, Xuan-Luong Vu, Thi-Minh-Huyen Nguyen, Minh-Thu Dao, Thi-Minh-Ngoc Dao, and Kim-Ngan Le. Vietnamese bracketing guidelines. technical report sp7.3. *Ministry of Education and Training (Vietnam)*, 2010.

[127] Phuong-Thai Nguyen, Xuan-Luong Vu, Thi-Minh-Huyen Nguyen, Van-Hiep N-guyen, and Hong-Phuong Le. Building a large syntactically-annotated corpus of vietnamese. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 182–185. Association for Computational Linguistics, 2009.

[128] Daniel M Bikel. *On the parameter space of generative lexicalized statistical parsing models*. PhD thesis, Citeseer, 2004.

[129] Phuong Le-Hong, Thi-Minh-Huyen Nguyen, and Azim Roussanaly. Vietnamese parsing with an automatically extracted tree-adjoining grammar. In *Computing*

and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on, pages 1–6. IEEE, 2012.

[130] Zhiguo Wang and Nianwen Xue. Joint pos tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of ACL*, pages 733–742, 2014.

[131] Ryan McDonald, Kevin Lerman, and Fernando Pereira. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220. Association for Computational Linguistics, 2006.

[132] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135, 2007.

[133] Anne-Marie Di Sciullo and Edwin Williams. *On the definition of word*, volume 14. Springer, 1987.

[134] Phuong Le-Hong, Thi Minh Huyen Nguyen, Azim Roussanaly, and Tuong Vinh Ho. A hybrid approach to word segmentation of vietnamese texts. In *Proceedings of the 2nd International Conference on Language and Automata Theory and Applications*, 2008.

[135] Dien Dinh and Thuy Vu. A maximum entropy approach for vietnamese word segmentation. In *Proceedings of Research, Innovation and Vision for the Future in Computing and Communication Technologies*, pages 248–253. IEEE, 2006.

[136] Roger Levy and Christopher Manning. Is it harder to parse chinese, or the chinese treebank? In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 439–446. Association for Computational Linguistics, 2003.

[137] Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. Descriptive and empirical approaches to capturing underlying dependencies among parsing errors. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1162–1171. Association for Computational Linguistics, 2009.

[138] Quy T. Nguyen, Yusuke Miyao, Ha T.T. Le, and Ngan L.T. Nguyen. Challenges and solutions for consistent annotation of vietnamese treebank. In *Proceedings of the Language Resources and Evaluation Conference*, 2016.

[139] Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *International Conference on Natural Language Processing*, pages 684–693. Springer, 2004.

[140] Daniel Tse and James R Curran. Chinese ccgbank: extracting ccg derivations from the penn chinese treebank. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1083–1091. Association for Computational Linguistics, 2010.