SN Computer Science (2021) 2:323
https://doi.org/10.1007/s42979-021-00715-4

ORIGINAL RESEARCH

Sentential Semantic Dependency Parsing for Vietnamese

Tuyen Thi-Thanh Do[1] · **Dang Tuan Nguyen**[2]

Received: 16 March 2021 / Accepted: 19 May 2021 / Published online: 5 June 2021

Abstract
Semantic dependency parse is the dependency graph of a sentence. This graph shows the grammatical dependencies between words in a sentence clearer than the dependency parse because it can present dependencies in which one word is possibly the dependant in two or more dependencies in a sentence. Therefore, it has been used to represent the meaning of a sentence. To represent the meaning of a sentence in Vietnamese, a method of parsing the sentence into semantic dependencies is proposed in this paper. This rule-based method transforms the result of Vietnamese dependency parser using semantic constraints defined in a lexicon ontology called Vietnamese lexicon ontology (VLO). The test result shows that the proposed method can capture more dependencies than the state-of-the-art Vietnamese dependency parser with the precision of 0.5328 and of 0.3113, respectively, and it can capture the indirect object dependency type which Vietnamese dependency parser cannot capture in our test.

Keywords Semantic dependency parsing · Rule-based dependency parsing · Semantic representation · Lexicon ontology

Introduction

Semantic dependency parse is similar to dependency parse [1] except that it is a graph instead of a tree. It has been used to represent the semantic of the sentence [2–4] because it is able to represent more dependencies of a sentence than the dependency parse. Therefore, the semantic of a sentence can be identified by parsing the sentence into semantic dependency parse. At this time, the state-of-the-art Vietnamese dependency parser [5] can parse a Vietnamese sentence into dependency parse. Thus, a research question is how to transform a dependency parse into a semantic dependency parse to identify the semantic of a Vietnamese sentence. For conveniences, we will use SDP, DP, D-parser and SD-parser for semantic dependency parse, dependency parse, dependency parser and semantic dependency parser, respectively.

Figure 1 shows the differences between the DP and the SDP of the same Vietnamese sentence "*Hayes và Lighthill đã đề_xuất một mô_hình vật_lý*" ("*Hayes and Lighthill proposed a physic model*"). In Fig. 1, (a) and (b) are the DP and SDP of the sentence, respectively. The DP has a dependency "*conj*(*và-2, Lighthill-3*)" which means "*Lighthill*" is the dependant of "*và*" (*and*) in "*conj*" dependency. This dependency did not present the true role of "*Lighthill*" which is a subject of verb "*đề_xuất*" (*proposed*) in the sentence. Therefore, the SDP of the sentence should be identified to present the true dependencies of the sentence.

The SDP of a sentence is possibly generated using an enhanced dependency parser [6, 7] or by transforming the DP of the sentence. The SDP of a Vietnamese sentence has to present all semantic relations, therefore, if a SDP is generated from a DP, many semantic dependencies will be generated from DP. However, the invalid dependencies are possibly generated if we do not have any semantic constraints between two words. This is the open problem addressed by Schuster [4] when converting dependency treebanks

This paper is a revised and expanded version of our paper entitled "Sentential Semantic Dependency Parsing for Vietnamese" presented at 7th International Conference on Future Data and Security Engineering, Quy Nhon, Binh Dinh, Vietnam, November 25–27, 2020.

This article is part of the topical collection "Future Data and Security Engineering 2020" guest edited by Tran Khanh Dang.

✉ Dang Tuan Nguyen
dangnt@sgu.edu.vn

Tuyen Thi-Thanh Do
tuyendtt@uit.edu.vn

[1] University of Information Technology, VNU-HCM, Ho Chi Minh City, Vietnam

[2] Saigon University, Ho Chi Minh City, Vietnam

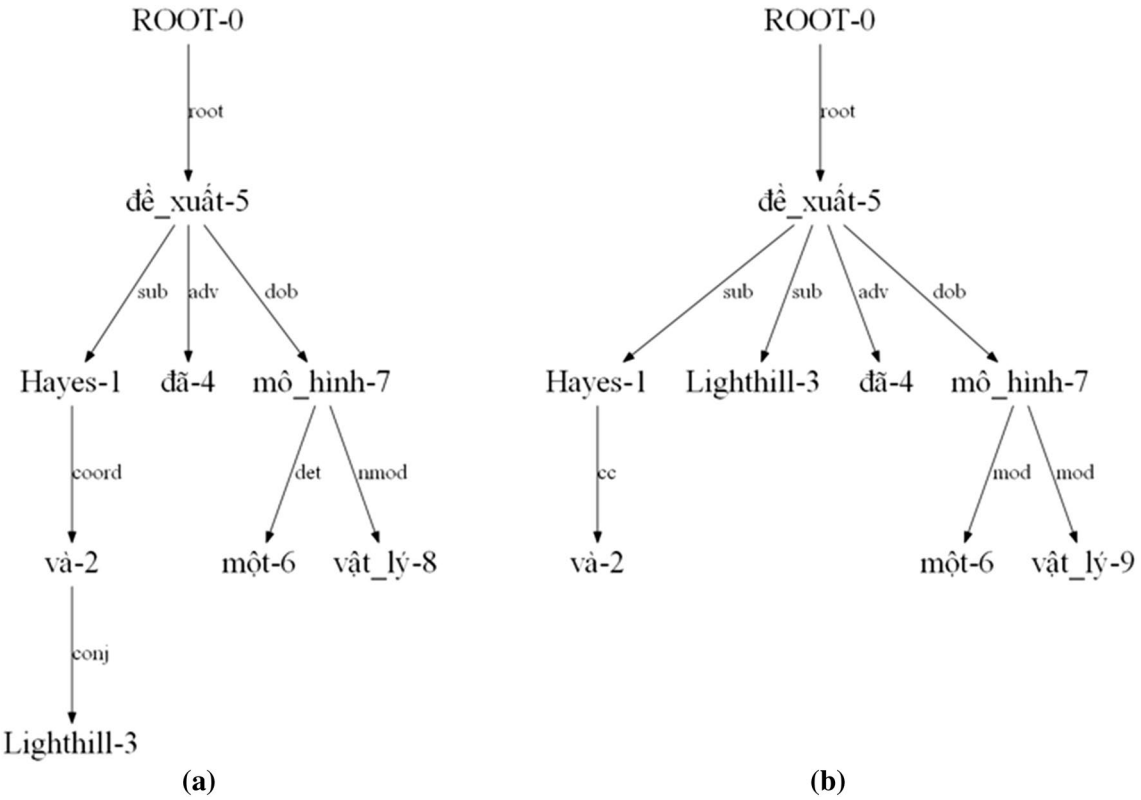

Fig. 1 **a** The DP with [5] and **b** the SDP of the same Vietnamese sentence

into dependency graphbanks. In this paper, which is the extended version of our paper [8], we would like to present our transformation-based approach in generating the SDP of a Vietnamese sentence with semantic constraints to solve the problem of generating invalid dependencies and the problem of correcting the wrong dependencies. In our approach, we use semantic constraints defined in a lexicon ontology for dependency validation and use transformation rules for generating semantic dependencies and correcting wrong dependencies from DP. The sentential semantic parsing for Vietnamese method proposed in this paper can identify implicit predicate argument dependencies, especially direct object and indirect object dependencies, while the Vietnamese dependency parser [5] rarely captures the indirect object dependencies and cannot present all semantic dependencies as shown in this paper. This is the main contribution of this paper.

This paper presents the sentential semantic parsing for Vietnamese method in five following sections. The first section introduces the problem to be solved in this paper. The second section presents backgrounds of semantic dependency parsing. Th third section proposes the sentential semantic dependency parsing method for Vietnamese. The fourth section presents the evaluation of the proposed method in comparison with the Vietnamese dependency parsing. Finally, some conclusions and future works are presented in the last section.

Backgrounds

Semantic Dependency Parsing

Semantic dependency parsing [2, 3] is the task of generating the semantic dependencies which are the relation of a word pair in a sentence. The types of semantic dependency are defined differently from each representation scheme such as Abstract Meaning Representation [9], Bilexical Semantic Dependency [10] and Universal Dependency [11, 12]. However, there are three types of dependency in common.

The first type of dependency is the predicate argument dependency. Predicate argument dependency type presents the argument structure of a verb in a sentence. There are three sub-types which are the subject dependency, direct object dependency and indirect object dependency indicating the relation between a verb and its subject, the relation

between a verb and its object and the relation between verb and its indirect object, respectively.

The second type of dependency is the modifier dependency. The modifier dependency type presents the relation between a word and its modifiers in a sentence. There are many sub-types of the modifier dependency type such as adjective modifier and adverb modifier [11, 12].

The third type of dependency is the complement dependency. The complement dependency type presents the relation between a word and its complement in a sentence. There are also many sub-types of the complement dependency type such as object of preposition and adjective complement [11, 12].

The types of semantic dependency are very similar to types of dependency as in Universal Dependency (UD) [11, 12], therefore, semantic dependency parsing can be seen as the extension of dependency parsing which identifies all dependencies in a sentence [13] according to the meaning of that sentence. For example, the DP (a) in Fig. 1 contains only one subject type dependency *sub*(*đề_xuất-5, Hayes-1*) while the SDP (b) in Fig. 1 contains two subject type dependencies *sub*(*đề_xuất-5, Hayes-1*) and *sub*(*đề_xuất-5, Lighthill-3*) because both Hayes and Lighthill are the author of the proposal according to the meaning of the sentence.

Although the semantic representation schemes are different, the semantic dependency parsers (SD-parser) are very similar in building process. Herschcovich et al. [13, 14] proposed parsers for Universal Conceptual Cognitive Annotation (UCCA) [15] scheme, Dozat [16] and Qi [7] proposed parsers for Universal Dependency scheme. These parsers are transition-based parsers trained on appropriate semantic dependency graphbanks. These graphbanks can be built on scratch or converted from dependency treebanks [4]. When converting dependency treebanks into dependency graphbanks, it is very important to ensure that the converted dependencies satisfy the semantic constraints between their governor and dependant in the way of Head-driven Phrase Structure Grammar [17, 18]. This is the problem that Schuster [4] has left to future works and we would like to solve this problem in this paper.

Semantic Constraints

Words of the same syntactic category may have different combinations in a phrase or in a sentence because of semantic constraints. In the example of Schuster [4], the sentence "*the store buys and sells cameras*" implies that "*cameras*" is the object of "*buys*" and "*sells*" while the sentence "*she was reading or watching a movie*" shows that "*a movie*" is the object of "*watching*" only. These two example sentences show that two sentences having identical syntactic structures may have different semantic dependency structures because the different words have different semantic constraints. In this case, "*buys cameras*", "*sells cameras*" and "*watching a movies*" satisfy the semantic constraints while "*reading a movie*" does not.

Head-driven Phrase Structure Grammar (HPSG) [17, 18] is a framework to ensure the semantic constraints in the syntactic parse tree of a sentence. In HPSG, every lexicon is represented in a syntactic–semantic combination structure, called typed feature structure (TFS). In TFS, syntactic and semantic constraints are explicitly presented. A HPSG parser uses respective TFSs to decide which words are possible to make syntactic relations and which word is the head (governor) in every syntactic relation. To define the TFS, an ontological category [18] containing semantic labels is needed to specify the sense of every word. These senses are the core of semantic constraints.

Sentential Semantic Dependency Parser for Vietnamese

The approach of this paper is to define rules to transform the DPs of Vietnamese D-Parser [5] into SDPs because there are not any large Vietnamese graphbanks to train a SD-parser at this time. Besides, the SDP can be generated from DP [4] if there are semantic constraints to confirm the validity of the generated semantic dependencies. Therefore, we would like to utilize the Vietnamese dependency parser [5] for reducing the cost of building a SD-parser from scratch. Another reason of studying transformation rules for converting DPs into SDP is that these transformation rules are possibly applied to build large graphbanks from Vietnamese dependency treebanks [19] to train SD-parser with deep-learning models.

In our approach, the SDP of Vietnamese sentence may contain up to five types of dependency which are *hasActor*, *hasDObj*, *hasIDObj*, *hasPComp* and *hasMod*. These dependency types have corresponding relations described in Vietnamese Lexicon Ontology (VLO), which will be described in "Vietnamese lexicon ontology". All types of dependency in Universal Dependency [11, 12] except conjunct dependency are possibly converted into these five types with a mapping table. Similarly, all types of dependency used in Vietnamese D-parser [5] are possibly converted into five types of dependency using Table 1.

We use only the five types of dependency in SDP for two reasons. First, they can present all important relation between word senses in a sentence without losing the semantic of the dependencies. Second, a small number of

Table 1 Mapping from types of dependency used in [5] to types of semantic dependency

Ord	Types of dependency used in [5]	Types of semantic dependency
1	All types of dependency except the below types of dependency	*hasMod*
2	pob	*hasPComp*
3	sub	*hasActor*
4	dob	*hasDObj*
5	idob	*hasIDObj*

dependency label is better for annotating graphbanks and parsing.

To parse Vietnamese sentence to SDP, we need an ontological category to present the Vietnamese word senses and the semantic constraints to ensure the validity of every semantic dependency.

Vietnamese Lexicon Ontology

Vietnamese Lexicon Ontology (VLO) [20, 21], demonstrated in Fig. 2, is an ontology of word senses. In VLO, individuals are word senses and similar senses are grouped into classes, called semantic classes. VLO is used as ontological category in HPSG [18] as following:

- Semantic classes are used for annotating the meaning of words instead of word senses because the number of word senses is noticeably larger than the number of semantic classes and the semantic class are appropriate to annotate the meaning of synonyms. In Fig. 2, the two words "*rượt*" (*chase*, a dialect of South Vietnam) and "*đuổi*" (*chase*, a dialect of North Vietnam) are annotated by semantic class "*Đuổi*" although their senses, respectively, are "*w_Rượt*" and "*w_Đuổi*".
- The relations between semantic classes are semantic constraints. The semantic constraints are defined at building VLO stage and possibly inferred from the relations

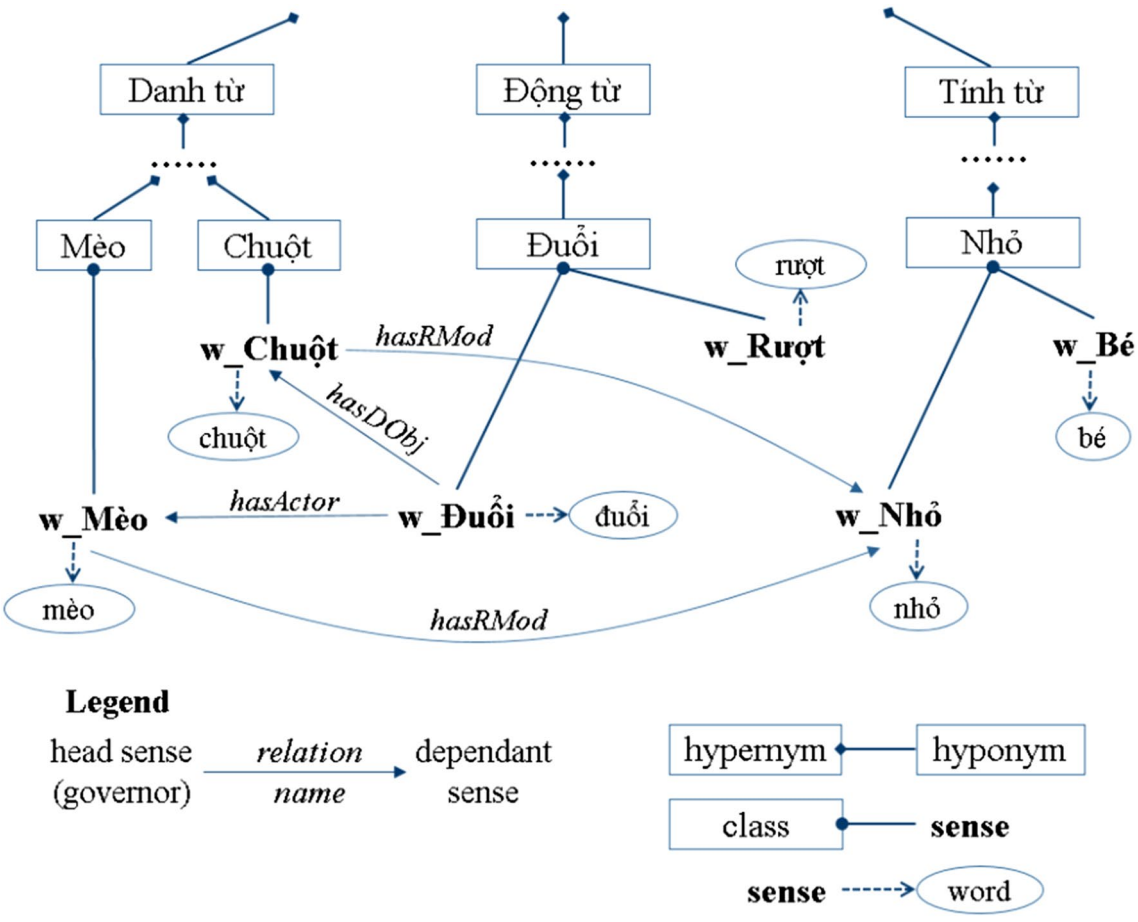

Fig. 2 The demonstration of VLO including words, word senses, semantic classes and relations (semantic constraints) of the sentence "*mèo nhỏ đuổi chuột nhỏ*" ("*a small cat chases a small mouse*")

between word senses. In the latest version V1.1 of VLO [21], there are eight important types of relation which cover the verb frame dependencies, conjunction dependency, modifier dependencies and complement dependencies:

- *hasActor*($sense_a$, $sense_b$) is the relation between the sense of a verb $sense_a$ and the sense of its subject $sense_b$. In Fig. 2, there is a relation *hasActor*(*w_Đuổi*, *w_Mèo*) indicating that the word "*mèo*" (*cat*) is possibly a subject of the verb "*đuổi*" (*chase*).
- *hasDObj*($sense_a$, $sense_b$) is the relation between the sense of a verb $sense_a$ and the sense of its direct object $sense_b$. In Fig. 2, there is a relation *hasDObj*(*w_Đuổi*, *w_Chuột*) indicating that the word "*chuột*" (*mouse*) is possibly a direct object of the verb "*đuổi*" (*chase*).
- *hasIDObj*($sense_a$, $sense_b$) is the relation between the sense of a verb $sense_a$ and the sense of its indirect object $sense_b$.
- *hasConj*($sense_a$, $sense_b$) is the relation indicate that $sense_a$ and $sense_b$ have the same category. There are six categories are used for confirming *hasConj* relation: nominal (including noun, proper noun, pronoun), verb, adjective, adverb, preposition and number.
- *hasPComp*($sense_a$, $sense_b$) is the relation between the sense of a word $sense_a$ and the sense of its complement $sense_b$ where the position of the word with $sense_b$ is on the left side of the position of the word with $sense_a$ in a phrase. This type of relation can be seen as the generalization of all types of complement dependency where the dependant word is on the left side of head (governor) word. This generalization will not lose the semantic of the dependency because the types of complement dependency are featured by the senses of words in the dependencies [11, 12] and *hasPComp* is used for annotating the dependency of two senses. This relation type is used for verifying *hasPComp* dependencies
- *hasRPComp*($sense_a$, $sense_b$) is the relation between the sense of a word $sense_a$ and the sense of its complement $sense_b$ where the position of the word with $sense_b$ is on the right side of the position of the word with $sense_a$ in a phrase. This relation type is used for verifying *hasPComp* dependencies.

Table 2 Collapsing rules for Vietnamese DP

Ord	Dependencies to be converted	Converted dependencies	Examples
Collapsing the object of preposition dependencies			
1	vmod(X, Y), pob(Y, Z)	vmod(X, Z), case(Z, Y)	vmod(*nghiên_cứu-1*, *về-2*), pob(*về-2*, *đặc_tính-4*) *collapsed to:* vmod(*nghiên_cứu-1*, *đặc_tính-4*), case(*đặc_tính-4*, *về-2*)
2	nmod(X, Y), pob(Y, Z)	nmod(X, Z), case(Z, Y)	nmod(*lý_thuyết-5*, *của-7*), pob(*của-7*,*newton-8*) *collapsed to:* nmod(*lý_thuyết-5*, *newton-8*) case(*newton-8*, *của-7*)
3	loc(X, Y), pob(Y, Z)	loc(X, Z), case(Z, Y)	loc(*góc-11*, *ở-12*), pob(*ở-12*,*đỉnh-13*) *collapsed to:* loc(*góc-11*, *đỉnh-13*) case(*đỉnh-13*, *ở-12*)
Collapsing the coord dependencies			
4	coord(X, Y), conj(Y, Z)	conj(X, Z), cc(Z, Y)	coord(*lực-16*,*và-19*), conj(*và-19*,*mômen-21*) *collapsed to:* conj(*lực-16*,*mômen-21*) cc(*mômen-21*, *và-19*)
Collapsing the dependencies of passive voice verb The word "*được*" and "*bị*" ("*be*") are denoted by *wpass*			
5	$\boldsymbol{R}$(U, wpass), $\boldsymbol{R}_i$(wpass,X), vmod(wpass, Y)	$\boldsymbol{R}$(U, Y), $\boldsymbol{R}_i$(Y, X), vmod(Y, wpass)	root(*ROOT*, *được-3*) sub(*được-3*, *năng_lượng-1*), adv(*được-3*, *có_thể-2*), vmod(*được-3*,*giải_phóng-4*) *collapsed to:* root(*ROOT*,*giải_phóng-4*) sub(*giải_phóng-4*, *năng_lượng-1*), adv(*giải_phóng-4*, *có_thể-2*), vmod(*giải_phóng-4*, *được-3*)

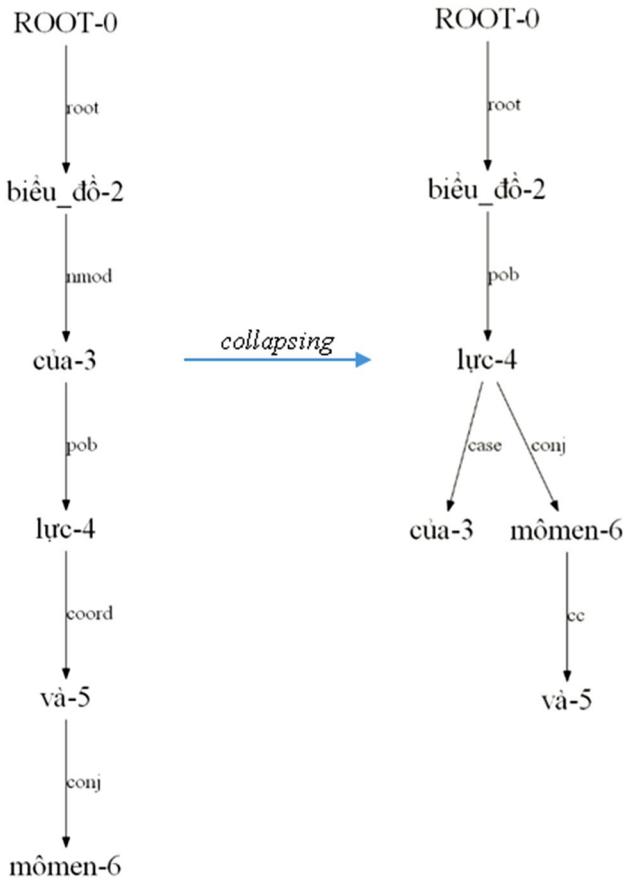

Fig. 3 Collapsing the object of preposition dependency and the coord dependency in DP of the phrase "*biểu đồ của lực và mômen*" ("*the chart of force and moment*")

- *hasMod*($sense_a$, $sense_b$) is the relation between the sense of a word $sense_a$ and the sense of its modifier $sense_b$ where the position of the word with $sense_b$ is on the left side of the position of the word with $sense_a$ in a phrase. This relation type is used for verifying *hasMod* dependencies.
- *hasRMod*($sense_a$, $sense_b$) is the relation between the sense of a word $sense_a$ and the sense of its modifier $sense_b$ where the position of the word with $sense_b$ is on the left side of the position of the word with $sense_a$ in a phrase. This relation type is used for verifying *hasMod* dependencies.

In our approach, we use VLO for verifying if two words $word_a$ and $word_b$ are possibly combined in a dependency $Rel(word_a, word_b)$. If $word_a$ and $word_b$ have their senses $sense_a$ and $sense_b$, respectively, and there is a relation $Rel(sense_a, sense_b)$ in VLO then dependency Rel(worda, wordb) is a semantic relation and is possibly generated when transforming a DP into SDP. VLO return the result of dependency verification in three values. The value YES means the dependency is valid, the value NO means the dependency is invalid and the value UNKNOWN means there are not enough information about the dependency.

An important problem when using VLO is the word sense identification method. When building VLO, we have also annotated the sense for each word in the sentences that we used when building VLO. The result of this annotation is a sense tagged dataset for Vietnamese and we have used this dataset to train a sense tagger for sense prediction. The sense tagger is a transformation-based learning model because the number of sense tags is big while the number of annotated sentences is small so that we cannot apply a complex sequential labelling method to build a sense tagger.

Transformation Rules

The transformation rules are defined follow collapsing rules [22] and enhanced universal dependencies in English [4]. There are four types of transformation rules: the collapsing rule, the adjusting dependency label and head rule, the adjusting non-projectivity graph rule and the expanding dependencies rule.

The first transformation type includes collapsing rules. These rules are defined after the collapsing rules in English and Germany [22]. In DP, A constituent composed of a constituent C_1 with head word c_1, a preposition or a conjunction k and a constituent C_2 with head word c_2 will be presented by two dependencies $r_1(c_1, k)$ and $r_2(k, c_2)$. These dependencies do not explicitly show the real dependency between c_1 and c_2. Therefore, the collapsing rules will be applied to DPs for making the dependencies containing prepositions and conjunctions clearer in SDPs. These rules are shown in Table 2 and the results of applying collapsing rules in dependency parses are illustrated in Figs. 3 and 4.

The second transformation type includes rules of adjusting dependency label and head. There are dependencies with wrong dependency label or with wrong head word in DP because there is no the semantic constraints or the semantic restrictions when parsing a sentence to DP. Rules of this type will correct these errors and they are divided into three sub-types: the adjusting modifier verb rule, the expanding dependencies rule and the adjusting wrong label and head word rule.

The first sub-type of adjusting dependency label and head transformation includes adjusting modifier verb rules. There are dependencies showing that verb is a modifier of a noun but the noun is actually the subject of the verb. In this case, these dependencies are adjusted with the rules in following

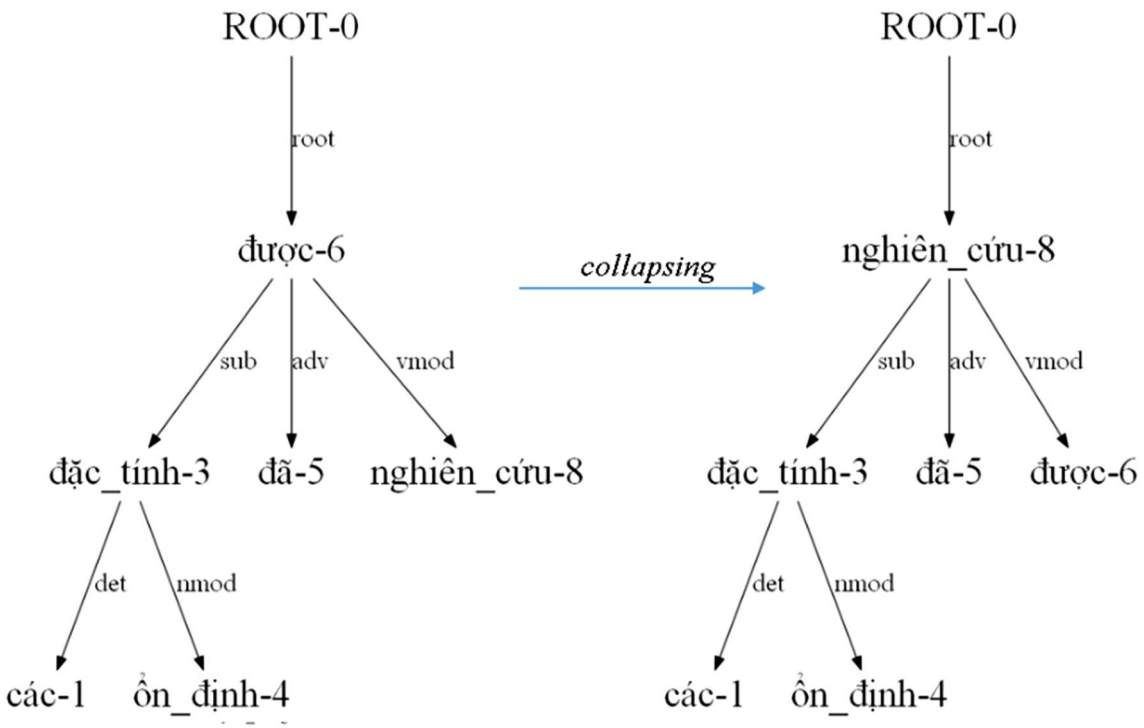

Fig. 4 Collapsing the dependency of the verb *nghiên_cứu-8* in passive voice in the DP of the sentence "*các đặc tính ổn định đã được nghiên cứu*" ("*the stable characteristics were studied*")

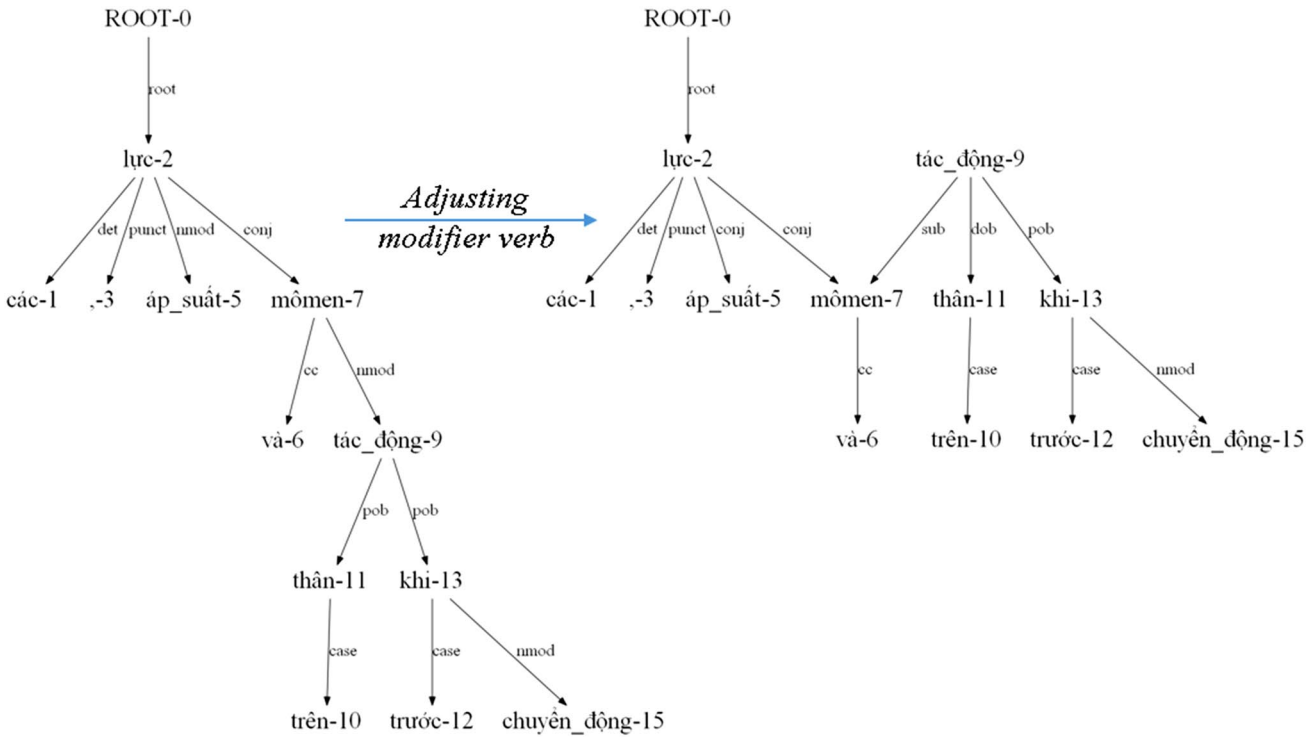

Fig. 5 Adjusting the dependency of modifier verb (*tác_động-9*) in the DP of the sentence "*các lực, áp suất và mômen tác động trên thân trước khi chuyển động*" ("*the force, pressure and moment impact the body before the motion*")

Algorithm 1. The result of applying adjusting modifier verb rules to a DP is shown in Fig. 5.

Algorithm 1. Adjusting modifier verb

Input:

- dependency $R(w_i, w_j)$
- noun w_i, verb w_j,
- sen_i and sen_j are respectively word sense of w_i and w_j,
- i and j are word indices in the phrase or sentence.
- *VLO* containing semantic constraints

Output: dependency $Rs(A, B)$ is adjusted from $R(w_i,w_j)$

```
1   Rs ← R, A ← wi, B ← wj
2   if (i < j)
3     if (wj is passive)
4       if (hasIDObj(senj, seni)∈VLO)
5         Rs ← iob, A ← wj, B ← wi
6       else
7         if (hasDObj(senj, seni)∈VLO)
8           Rs ← dob, A ← wj, B ← wi
9         else
10          if (hasDObj(senj, seni)∈VLO)
11            Rs ← dob, A ← wj, B ← wi
12    else
13      if (hasIDObj(senj, seni)∈VLO)
14        Rs ← iob, A ← wj, B ← wi
15  else
16    if (wj is passive)
17      if (hasDObj(senj, seni)∈VLO)
18        Rs ← dob, A ← wj, B ← wi
19      else
20        if (hasIDObj(senj, seni)∈VLO)
21          Rs ← iob, A ← wj, B ← wi
22    else
23      if (hasActor(senj, seni)∈VLO)
24        Rs ← sub, A ← wj, B ← wi
25  return Rs(A, B)
```

The second sub-type of adjusting dependency label and head transformation includes adjusting the predicate argument rules. Subject dependencies, direct object dependencies and indirect object dependencies in passive voice may be wrong identified because the D-parser does not have the semantic constraints. Therefore, the rules of this sub-type included in the following Algorithm 2 will correct the label of these dependencies. The result of applying rules of adjusting the predicate argument to a DP is shown in Fig. 6.

Algorithm 2. Adjusting the predicate argument

Input:

- dependency $R(w_i, w_j)$,
- verb w_i is passive, nominal w_j,
- sen_i and sen_j are respectively word sense of w_i and w_j,
- i and j are word indices in the phrase or sentence,
- *VLO* containing semantic constraints.

Output: dependency $Rs(w_i, w_j)$ is adjusted from $R(w_i, w_j)$.

```
1   Rs ← R
2   if (i < j and R is dob)
3     if (hasIDObj(seni, senj)∈VLO)
4       Rs ← iob
5   else
6     if (hasActor(seni, senj)∈VLO)
7       Rs ← iob
8     else
9       if (i > j and R is sub)
10        if (hasIDObj(seni, senj)∈VLO)
11          Rs ← iob
12      else
13        if (hasDObj(seni, senj)∈VLO)
14          Rs ← dob
15  return Rs(wi, wj)
```

The third sub-type of adjusting dependency label and head transformation includes adjusting wrong label and head word rules. The D-parser does not have semantic constraints, therefore, the modifier, complement and conjunction dependencies generated may have wrong head or wrong label. The rules of this sub-type will correct these errors with the following Algorithm 3. Given every dependency $R(w_i, w_j)$, if it does not satisfy the semantic constraints, the Algorithm 3 will find an appropriate word w_t in range $[min(i, j), max(i,j)]$ which is the descendant of word w_i in DP and have semantic constraint $Rs(w_t, w_j)$. *Rs* is a dependency label chosen from a ordered list {*hasActor*, *hasDObj*, *hasIDObj*, *hasMod*, *hasRMod*, *hasPComp*, *hasRPComp*, *hasConj*}in which *hasActor* has the top most priority. If the appropriate word w_t is not found, find it in the ancestors of word w_i in the same manner. The results of applying rules of adjusting wrong label and head word to a DP are illustrated in Figs. 5 and 7.

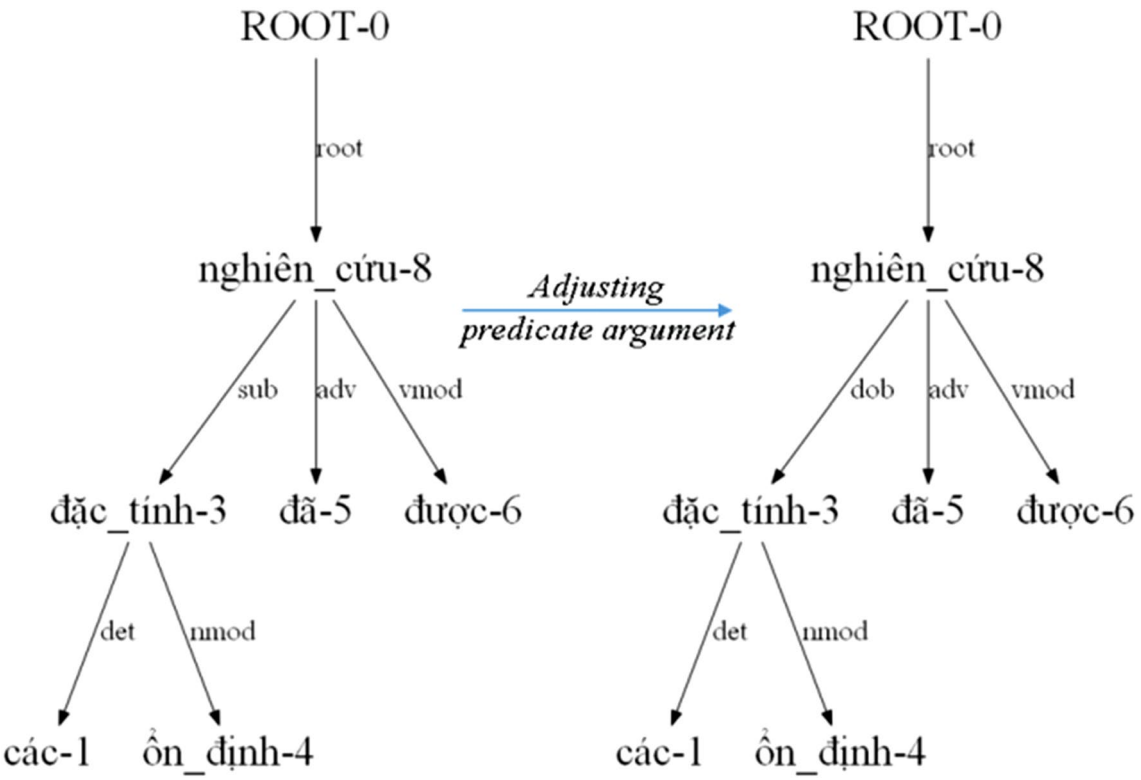

Fig. 6 Adjusting the predicate argument of passive verb (*nghiên_cứu-8*) in the DP of the sentence "*các đặc tính ổn định đã được nghiên cứu*" ("*the stable characteristics were studied*")

Algorithm 3. Adjusting wrong label and head word

Input:

- dependency parse *T*,
- dependency $R(w_i, w_j) \notin$ VLO,
- w_i and w_j are arbitrary words
- sen_i and sen_j are respectively word sense of w_i and w_j,
- *i* and *j* are word indices in the phrase or sentence,
- *VLO* containing semantic constraints.

Output: dependency $Rs(A, B)$ is adjusted from $R(w_i, w_j)$.

```
//DPLabel is a function converting semantic dependency
//        label to dependency label
//getSense is a function returning the sense of a word
  Rs ← R, A ← wi, B ← wj
  RSET ← {hasActor, hasDObj, hasIDObj, hasMod,
          hasRMod, hasPComp, hasRPComp, hasConj}
  s ← min(i, j), e ← max(i,j)
  CLIST ← {wk|wk is descendant of wi in T,k∈(s,e)}
  for t = s to e
    for Rt in RSET
      if(wt∈CLIST and Rt(getSense(wt),senj)∈VLO)
        Rs ← DPLabel(Rt), A=wt, B=wj
        return Rs(A, B)
  PLIST ← {wi}
  while (PLIST[length(PLIST)] has parent wk in T)
    PLIST ← append(PLIST, {wk})
  for t = 1 to length(PLIST)
    for Rt in RSET
      if (wt∈CLIST and Rt(getSense(wt),senj)∈VLO)
        Rs ← DPLabel(Rt), A=wt, B=wj
        return Rs(A, B)
```

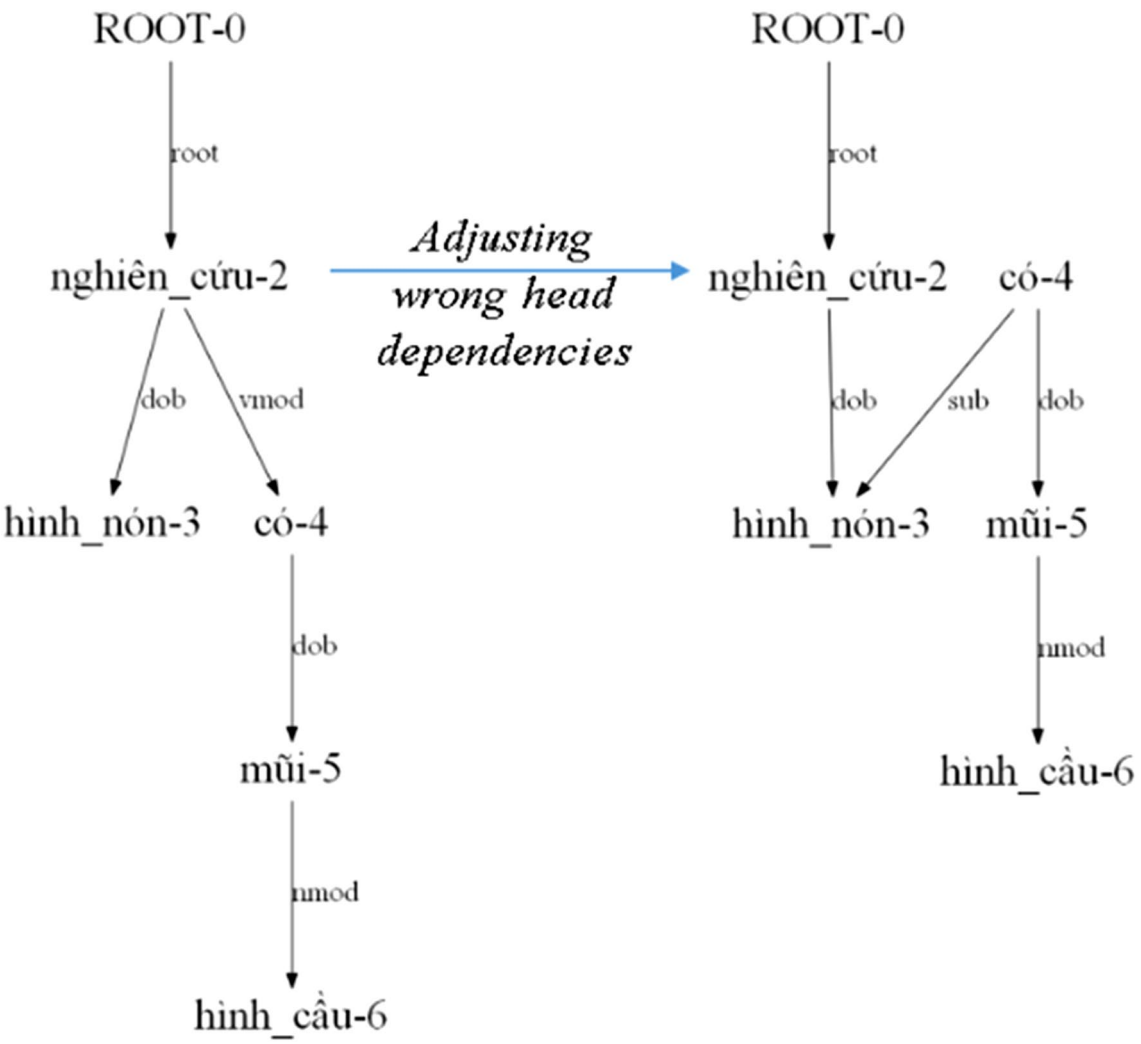

Fig. 7 Adjusting the wrong head word of dependency *vmod*(*nghiên_cứu-2*, *có-4*) in the DP of the sentence "*nghiên cứu hình nón có mũi hình cầu*" ("*study the cone having a modified spherical nose*")

In Fig. 5, the dependency *nmod*(*lực-2*, *áp_suất-5*) was adjusted to *conj*(*lực-2*, *áp_suất-5*) in which the dependency label "*nmod*" was replaced by dependency lable "*conj*". In this case, the algorithm 3 has adjusted the wrong label case. In Fig. 7, the dependency *vmod*(*nghiên_cứu-2*, *có-4*) was adjusted to *subj*(*có-4*, *hình_nón-3*) in which the word "*có-4*" was adjusted from the dependant of the word "*nghiên_cứu-2*" to the governor of the word "*hình_nón-3*". In this case, the algorithm 3 has adjusted the wrong head case.

The third transformation type includes adjusting non-projectivity [23] graph rules. After adjusting dependencies with the above transformation rules, the dependency graph may be non-projectivity. The rules of this type will correct the dependencies which break the projectivity constraint with the following Algorithm 4. The result of applying rules of adjusting non-projectivity graph to the dependency graph in Fig. 8 is shown in Fig. 9.

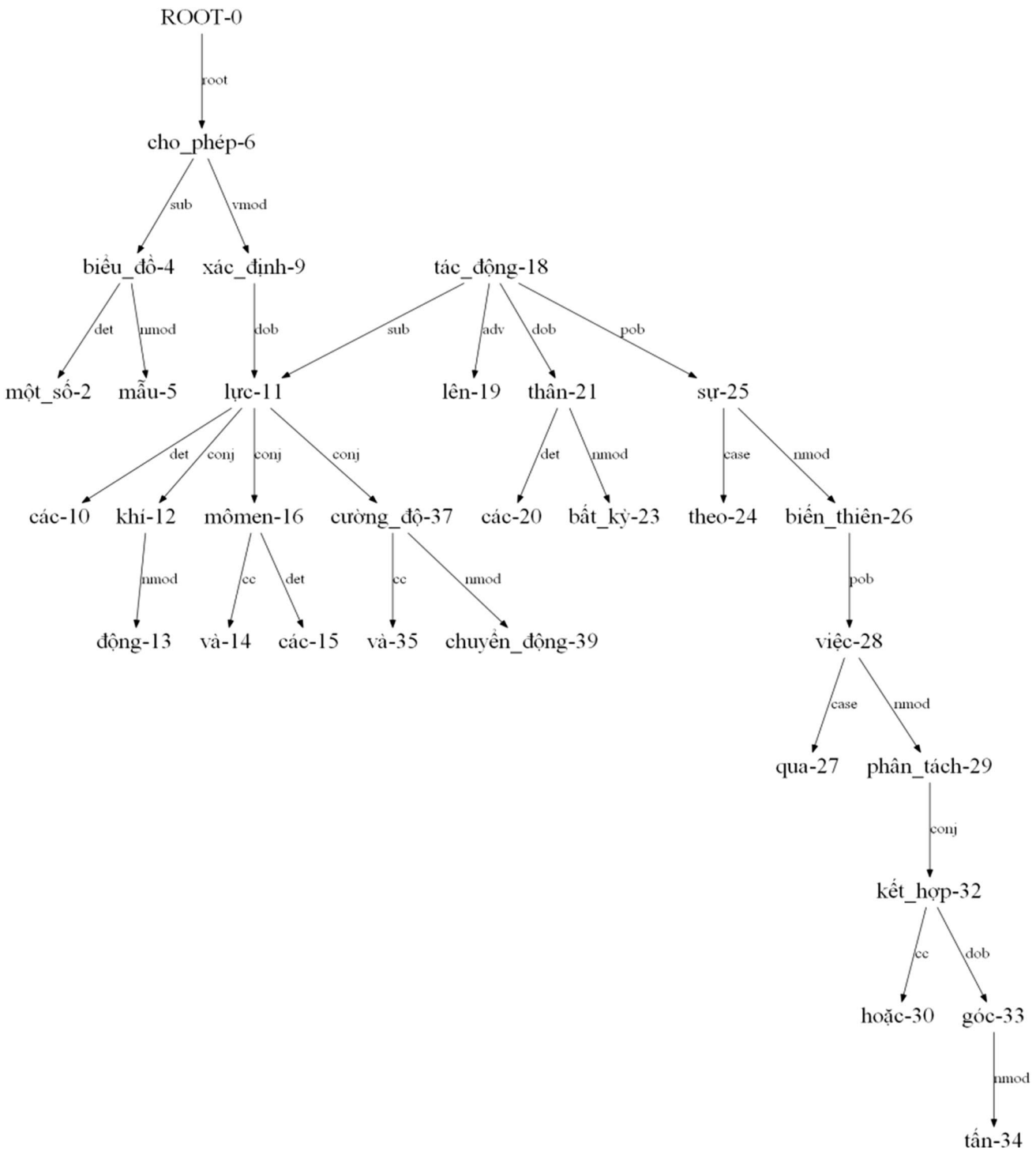

Fig. 8 A non-projectivity graph with dependency *conj*(*lực-11*, *cường_độ-37*)

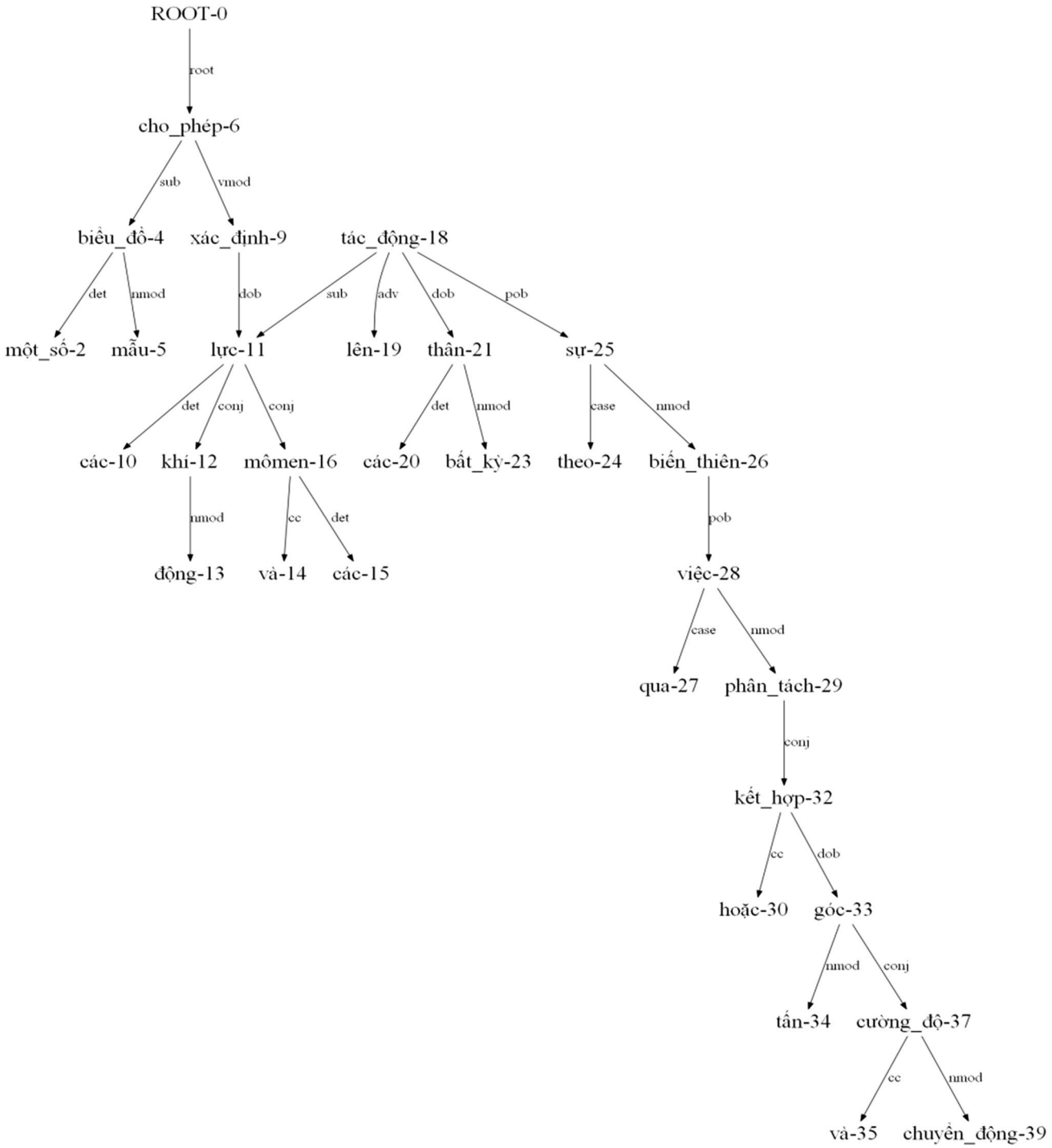

Fig. 9 A projective graph adjusted from the non-projectivity graph in Fig. 8

Algorithm 4. Adjusting non-projectivity graph
Input:

- non-projectivity dependency graph *G,*
- dependency $R(w_i, w_j)$ breaks the projectivity constraint,
- w_i and w_j are arbitrary words
- sen_i and sen_j are respectively word sense of w_i and w_j,
- *i* and *j* are word indices in the phrase or sentence,
- *VLO* containing semantic constraints.

Output: projectivity dependency graph G.

```
1   wp ← an ancestor of wi that i<p<j or i>p>j
2   RSET ← {hasActor, hasDObj, hasIDObj, hasMod,
            hasRMod, hasPComp, hasRPComp, hasConj}
3   if (i < p and p < j)
4     CLIST←{wk|wk is descendant of wi in T,k∈(p,j)}
5     for t = j to p
6       for Rt in RSET\{hasMod, hasPComp}
7         if(wt∈CLIST and Rt(getSense(wt),senj)∈VLO)
8           delete R(wi, wj) from G
9           Rs = DPLabel(Rt)
10          add Rs(wt, wj) to G
11    return G
12  if (j < p and p < i)
13    CLIST←{wk|wk is descendant of wi in T,k∈(j,p)}
14    for t = j to p
15      for Rt in RSET\{hasRMod, hasRPComp}
16        if(wt∈CLIST and Rt(getSense(wt),senj)∈VLO)
17          delete R(wi, wj) from G
18          Rs = DPLabel(Rt)
19          add Rs(wt, wj) to G
20    return G
```

The fourth transformation type includes expanding dependencies rules. The conjunction dependencies do not exist in SDP because they are syntactic dependencies to group words having similar semantic function. The rules of this type will replace the conjunction dependencies with appropriate dependencies. These rules are defined in the following Algorithm 5. For illustration, the results of applying expanding dependencies rules are shown in Figs. 10 and 11.

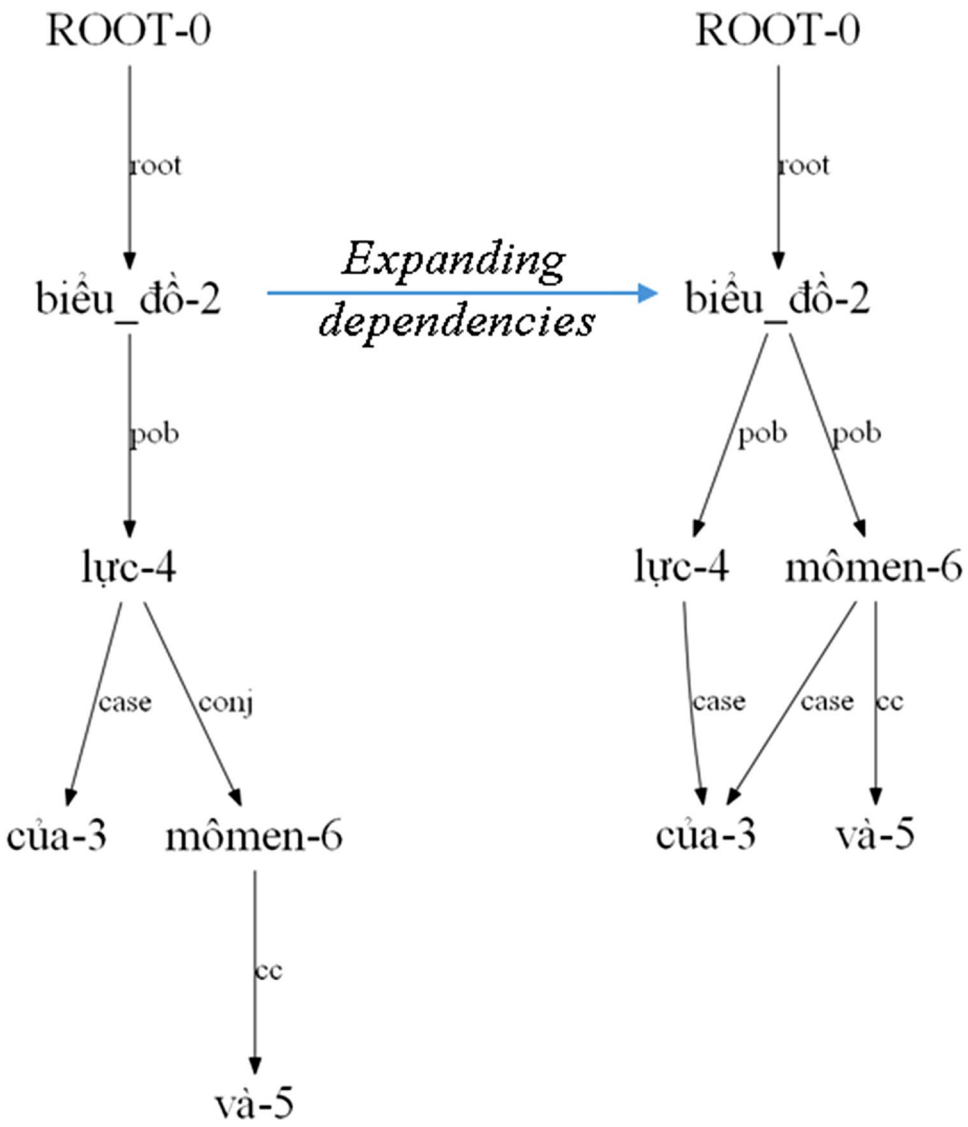

Fig. 10 The result of expanding dependency *conj*(*lực-4*, *mômen-6*) in dependency graph of the phrase "*biểu đồ của lực và mômen*" ("*the chart of force and moment*")

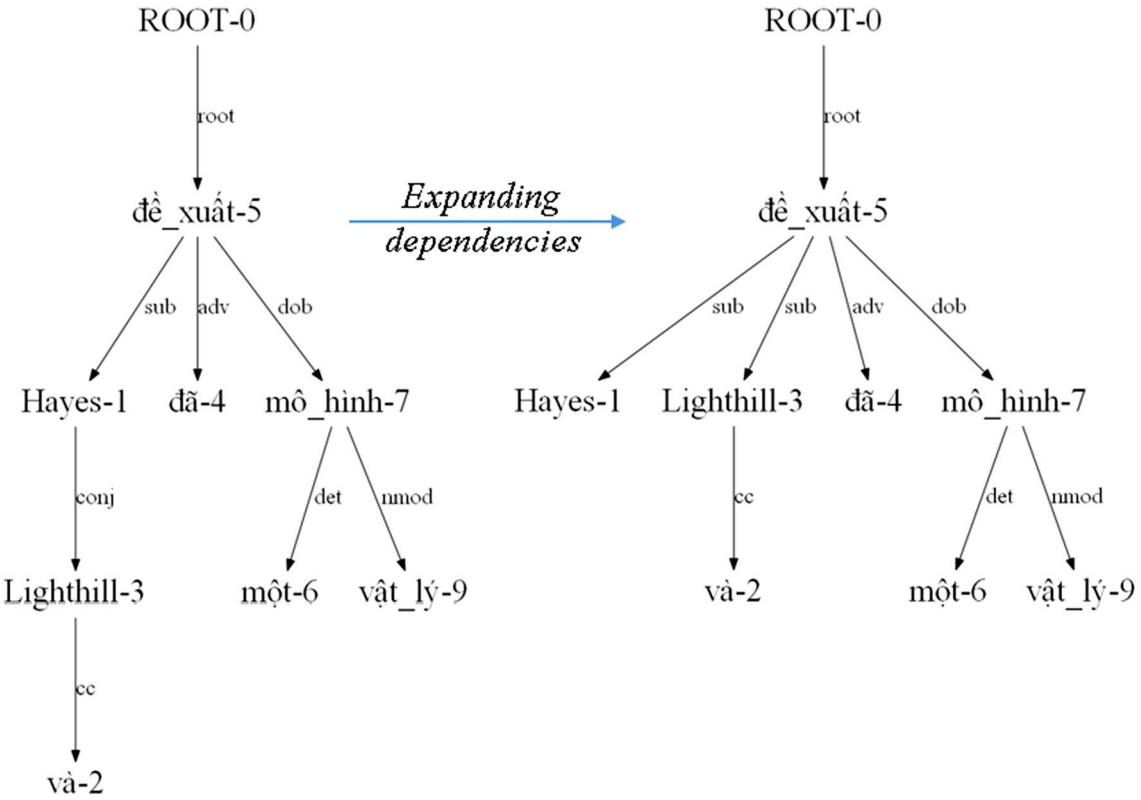

Fig. 11 The result of expanding dependency *conj*(*Hayes-1*, *Lighthill-3*) in dependency graph of the sentence "*Hayes và Lighthill đã đề xuất một mô hình vật lý*" ("*Hayes and Lighthill proposed a physic model*")

Algorithm 5. Expanding dependencies

Input:

- dependency graph *G,*
- *VLO* containing semantic constraints.

Output: expanded dependency graph *G*.

```
//SDPLabel is a function converting dependency label to
//          semantic dependency label
//getSense is a function returning the sense of a word
   1   for R(wp,wi) in G
   2     PLIST ← {wk|k=p or conj(wp,wk)∈G}
   3     CLIST ← {wk|k=i or conj(wi,wk)∈G}
   4     Rs ← SDPLabel(R)
   5     for wk in PLIST
   6       for wl in CLIST
   7         if (k≠p and l≠i and (i-p)*(i-k)>0 and
                 Rs(wk,wl)∈VLO)
   8           add R(wk,wl) to G
   9   return G
```

Converting Dependency Label

After applying all transformation rules, the DP is transformed into semantic dependency graph but many dependency labels of the semantic dependency graph are still the same as in the DP therefore these dependency labels have to be replaced with the appropriate semantic dependency labels. The dependency label replacement follows Algorithm 6.

Algorithm 6. Replacing dependency labels
Input:

- dependency graph ***G***,
- hashtable ***Lbl***(key: dependency label, value: semantic dependency label) extracted from **Table 1**.

Output: expanded dependency graph ***G***.

```
1   for R(wp,wi) in G
2     SR ← Lbl.get(R)
3     G ← G \ {R(wp,wi)}
4     G ← G ∪ {SR(wp,wi)}
5   return G
```

Evaluation

For evaluation, we need the VLO for semantic constraint validation and a sense tagger for word sense labelling. At this time, the VLO is not very large and the sense tagged dataset is small therefore we can experiment our method on a small dataset only. The result of our method will be compared to the result of Vietnamese dependency parser [5] to show the efficiency of our method in capturing semantic dependencies because we do not have any Vietnamese semantic graphbank for training a semantic dependency parser or any Vietnamese semantic dependency parser for directly comparing to our parser at this time.

Test Dataset

The test dataset includes 343 sentences manually annotated in SDP. These sentences are extracted from the online news website VnExpress[1] then they were annotated in a 4-step process as following:

- First step, we used the Vietnamese word tokenizer from VnCoreNLP [24] for Vietnamese word segmentation then we manually checked the results.
- Second step, we annotated label *hasMod* for modifier dependencies in noun phrases, verb phrases and adjective phrases.
- Third step, we annotated label *hasPComp* for complement dependencies in noun phrases and verb phrases.
- Final step, we annotate labels *hasActor*, *hasDObj* and *hasIDObj* for appropriate predicate argument dependencies.

The annotated result of each sentence is a dependency graph as shown in Figs. 10 and 11. In this process, the annotator has read carefully to clearly understand each sentence and set down all semantic dependencies of the sentence. The statistic of dependencies in the test dataset are shown in Table 3

Sense Tagging

Our approach is using VLO to verify semantic dependencies before adding them into SDP when transforming DP. Therefore, sense tagging is an obvious step before transforming DP into SDP. In this experiment, we use Transformation-based Learning method to train a sense tagger because our sense tagged training dataset contains 835 sense tagged sentences which is not large enough to apply state-of-the-art POS tagging model. Our sense tagging model has the accuracy of 0.7949 on our test data.

[1] http://vnexpress.net

Table 3 The statistic of dependencies in test dataset

Type of dependency	Number	Ratio (%)
hasActor	407	5.7
hasDObj	841	11.7
hasIDObj	64	0.9
hasMod	4509	62.7
hasPComp	1363	19.0
Sum	7184	100

Table 4 The results of our SD-parser and Vietnamese D-parser

Type of dependency	Our SD-parser		Vietnamese D-parser	
	P	R	P	R
hasActor	0.3852	0.4619	0.2492	0.2015
hasDObj	0.6334	0.4602	0.5351	0.2176
hasIDObj	0.6400	0.2500	0	0
hasMod	0.4884	0.5962	0.3736	0.5006
hasPComp	0.4588	0.3925	0	0
Average	0.5328	0.5681	0.3113	0.3512

Evaluation Results

We have implemented our SD-parser follow "Sentential semantic dependency parser for Vietnamese" and used it to generate semantic dependency graphs from 343 sentences of the test dataset. Then, we used Vietnamese D-parser [5] to generate DPs from 343 sentences of test data and converted the dependency labels of the result DPs using Table 1. The evaluation results shown in Table 4 indicate that the transformation rules with semantic constraints of VLO are really effective for converting DPs to SDPs with the precision and recall of generating predicate argument dependencies increased 10–24% when compare with the result of the Vietnamese D-parser.

The results in Table 4 show that Vietnamese D-parser cannot capture the indirect object dependency type while our SD-parser capture this dependency type with precision of 0.64 and recall of 0.25. This dependency type is important for presenting the semantic of a sentence. Therefore, the transformation-based approach using semantic constraints is reasonable to build Vietnamese SD-parser.

The overall precision and recall of the SD-parser are only 0.5328 and 0.5681, respectively. There are two reasons for these results. The first reason is that the F_1 score of the underlying D-parser of our SD-parser is 0.7353 which should be improved because the DP generated from a D-parser strongly affects to the result of generating SDP. The second reason is that the accuracy of our sense tagger is acceptable with 0.7949. It should be improved significantly to show the effective of the transformation with semantic constraints.

Conclusions and Future Works

In this paper, we would like to present our work about sentential SD-parser for Vietnamese. The SD-parser uses transformation rules with semantic constraints to convert a DP to SDP. This is an reasonable approach to generating SDPs from Vietnamese sentences. In the experiment, our SD-parser can capture more semantic dependencies than Vietnamese dependency parser. It can capture the indirect object dependency type which is not captured using Vietnamese dependency in our test data. Since our SD-parser uses a lexicon ontology, the VLO, to verify the validity of dependencies, a resulted SDP will present the semantic of a sentence better than a DP. In our approach, there are two problems which affect the transformation results. The first problem is the effective of Vietnamese dependency parser. Although a DP generated from a dependency parser cannot present all semantic dependency of a sentence, it is the base for generating the SDP. The second problem is the accuracy of sense tagging model, which is 0.7949 in our experiment, because SD-parser needs word senses to verify the validity of dependencies. If word senses are wrongly identified, the results of verifying dependencies are wrong and the generated SDPs are not good.

The results of SD-parser are better than Vietnamese dependency parser in presenting semantic of sentence according to our experiment, however, SD-parser's precision of 0.5328 and recall of 0.5681 are low and need improvements. The first improvement is to enrich the VLO and to build a large Vietnamese sense tagged dataset. The second improvement is to apply deep neural network in sense tagging task for state-of-the-art results.

Although the Vietnamese dependency parser should be improved also, we have focussed in building VLO and Vietnamese sense tagged dataset for semantic related tasks first. In addition, when we have large VLO and sense tagged dataset, we can also develop a semantic dependency parser for Vietnamese.

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Nivre J. Dependency grammar and dependency parsing. Technical Report MSI report 05133; 2005.
2. Oepen S, Kuhlmann M, Miyao Y, Zeman D, Cinkova S, Flickinger D, Hajic J, Uresova Z. SemEval 2015 Task 18: broad-coverage

semantic dependency parsing. International workshop on semantic evaluation. ACL, Denver; 2015. p. 915–926.
3. Oepen S, Kuhlmann M, Miyao Y, Zeman D, Flickinger D, Hajic J, Ivanova A, Zhang Y. Semeval 2014 task 8: broad-coverage semantic dependency parsing. International workshop on semantic evaluation. ACL, Dublin; 2014. p. 63–72.
4. Schuster S, Manning CD. Enhanced english universal dependencies: an improved representation for natural language understanding tasks. International conference on language resources and evaluation; 2016. p. 2371–2378.
5. Nguyen DQ, Dras M, Johnson M. An empirical study for Vietnamese dependency parsing. Australasian language technology association workshop, Melbourne, Australia; 2016. p. 143–149.
6. Grünewald S, Friedrich A. RobertNLP at the IWPT 2020 shared task: surprisingly simple enhanced UD parsing for english. International conference on parsing technologies and the IWPT 2020 shared task on parsing into enhanced universal dependencies; 2020. p. 245–252.
7. Qi P, Dozat T, Zhang Y, Manning CD. Universal dependency parsing from scratch. The CoNLL 2018 shared task: multilingual parsing from raw text to universal dependencies. ACL, Brussels, Belgium; 2018. p. 160–170.
8. Do TT-T, Nguyen DT. Sentential semantic dependency parsing for Vietnamese. Future data and security engineering. Binh Dinh: Springer; 2020. p. 429–447.
9. Banarescu L, Bonial C, Cai S, Georgescu M, Griffit K, Hermjakob U, Knight K, Koehn P, Palmer M, Schneider N. Abstract meaning representation for sembanking. 7th linguistic annotation workshop and interoperability with discourse; 2013. p. 178–186.
10. Oepen S, Kuhlmann M, Miyao Y, Zeman D, Cinková S, Flickinger D, Hajic J, Ivanova A, Uresova Z. Towards comparability of linguistic graph banks for semantic parsing. Tenth international conference on language resources and evaluation; 2016. p. 3991–3995.
11. De Marneffe M-C, Manning CD. Stanford typed dependencies manual. Technical report, Stanford University; 2016.
12. Nivre J, de Marneffe M-C, Ginter F, Goldberg Y, Hajic J, Manning CD, McDonald R, Petrov S, Pyysalo S, Silveira N, Tsarfaty R, Zeman D. Universal dependencies v1: a multilingual treebank collection. Tenth international conference on language resources and evaluation. European Language Resources Association (ELRA); 2016. p. 1659–1666.
13. Hershcovich D, Abend O, Rappoport A. Multitask parsing across semantic representations. 56th annual meeting of the association for computational linguistics, vol. 1. Melbourne: ACL; 2018. p. 373–385.
14. Hershcovich D, Abend O, Rappoport A. A Transition-based directed acyclic graph parser for UCCA. 55th annual meeting of the association for computational linguistics, vol. 1. Vancouver: ACL; 2017. p. 1127–1138.
15. Abend O, Rappoport A. Universal conceptual cognitive annotation (UCCA). Annual meeting of the association for computational linguistics, vol. 1. Sofia: Association for Computational Linguistics; 2013. p. 228–238.
16. Dozat T, Manning CD. Deep biaffine attention for neural dependency parsing. International conference on learning representations; 2017.
17. Levine RD, Meurers WD. Head-driven phrase structure grammar: linguistic approach, formal foundations, and computational realization. Encyclopedia of language and linguistics, 2nd ed; 2006. p. 237–252.
18. Pollard C, Sag IA. Head-driven phrase structure grammar. Chicago: University of Chicago Press; 1994.
19. Nguyen DQ, Pham SB, Nguyen P-T, Le Nguyen M. From treebank conversion to automatic dependency parsing for Vietnamese. International conference on applications of natural language to data bases—information systems. Springer; 2014. p. 196–207.
20. Do TT-T. Building a Vietnamese lexicon ontology for syntactic parsing and document annotation. iiWAS. Vienna: ACM; 2013. p. 619–623.
21. Do TT-T, Nguyen DT. VLO V1. 1-A Vietnamese lexicon ontology for universal dependency parsing. International conference on advanced computing and applications. IEEE; 2020. p. 94–100.
22. Ruppert E, Jonas K, Martin R, Chris B. Rule-based dependency parse collapsing and propagation for German and English. German society for computational linguistics and language technology. University of Duisburg-Essen; 2015. p. 58–66.
23. Covington MA. A fundamental algorithm for dependency parsing. Proceedings of the 39th annual ACM southeast conference; 2001. p. 95–102.
24. Vu T, Nguyen DQ, Dras M, Johnson M. VnCoreNLP: a Vietnamese natural language processing toolkit. Conference of the North American chapter of the association for computational linguistics: demonstrations. New Orleans: Association for Computational Linguistics; 2018. p. 56–60.