ViPa Project - Vietnamese Parsing

Tuan Do - Alex Luu

I. Vietnamese language overview

According to Vajda (http://pandora.cii.wwu.edu/vajda/ling201/test1materials/typology.htm), three basic structural techniques for showing syntactic relations:

- word order
- function words
- inflections

Besides, it is observed that there is potential counter-correlations between word order and inflections from syntactic perspective.

Consequently, we can classify Vietnamese language, like any other natural languages in virtue of Morphological and word order typologies as below.

1. Morphological typology

a. How much syntactic information is contained in the average word?

Vietnamese is an analytic language which conveys syntactic information only by word order and function words.

b. How do morphemes usually build words?

Vietnamese is an isolating language which has no derivational affixes

(It is noticed that all isolating languages are analytic.)

The implication here is that the phenomenon of syntactic feature agreement is not obvious in Vietnamese language.

2. Word Order typology

Word Order in Vietnamese language is SVO, the most common order among the world's languages.

The implication here is that Vietnamese does share the common types of structural ambiguity (such as attachment and coordination ambiguity) with other SVO languages such as English.

II. Input Data

Our input data is a shortened version of Vietnamese Cinderella fairy tale named "Tấm (broken rice) Cám (rice bran)" shown as below.

Table 1. Input Data

Vietnamese (original)	**English** (word-by-word oriented translation)
Tấm Cám là hai chị em cùng-cha-khác-mẹ	Tam Cam are two same-father-different-mothers sisters
Bố Tấm đã mất	Tam's father died
Tấm ở với mẹ con Cám	Tam lives with Cam and her mother
Họ suốt-ngày hành-hạ Tấm	They night-and-day torture Tam
Tấm rất cô-đơn	Tam is very desolated
Tấm chỉ có cá bống làm bạn	Tam only has a goby to be friends with
Cám phát-hiện ra Bống	Cam discovers the goby
Cám làm-thịt Bống	Cam cooks the goby
Tấm không tìm thấy Bống	Tam doesn't see the goby
Tấm khóc	Tam sobs
Bụt hiện lên	Buddha appears
Bụt giúp Tấm tìm xương Bống.	Buddha helps Tam to find the goby's bones
Tấm chôn xương Bống dưới chân giường	Tam buries the goby's bone under the bed legs
Vua mở hội	The King holds a festival
Mẹ con Cám đi xem	Cam and her mother go to attend
Họ không muốn Tấm đi	They don't want Tam to go
Mẹ-ghẻ trộn thóc với gạo	The stepmother mixes fine rice grains with coarse ones
Mẹ sai Tấm tách-riêng chúng ra	The mother forces Tam to separate them
Tấm không-thể đi dự hội	Tam cannot go to attend the festival
Tấm khóc	Tam sobs
Bụt lại hiện lên	Buddha again appears
Bụt nhờ chim giúp Tấm	Buddha asks the birds to help Tam
Tấm không có quần áo đẹp đi dự hội	Tam doesn't have beautiful clothes to go to attend the festival
Bụt bảo Tấm đào xương bống lên	Buddha tells Tam to dig the goby's bones out
Xương bống hóa-thành quần áo đẹp	The goby's bones become the beautiful clothes
Tấm mặc quần áo đi dự hội	Tam gets dressed to go to attend the festival

Table 1. Input Data (cont.)

Vietnamese (original)	**English** (word-by-word oriented translation)
Tấm làm-rơi giày trong lễ hội	Tam drops her slippers at the festival
Vua nhặt được đôi giày	The King picks up the slippers
Tấm ướm vừa giày	Tam fits her feet perfectly into the slippers
Tấm trở thành hoàng-hậu làm mẹ con Cám tức-giận	Tam becomes the Queen, which makes Cam and her mother angry
Tấm về giỗ cha	Tam visits home for her father’s death anniversary
Tấm nghe lời mẹ-ghẻ trèo lên hái cau	Tam obeys the stepmother’s words to climb up to gather areca nuts
Mẹ-ghẻ chặt cây cau	The stepmother chops the areca tree
Tấm rơi xuống ao chết-đuối	Tam falls into the pond, drowning
Cám thế Tấm vào cung làm hoàng-hậu	Cam replace Tam to go to the palace to become the Queen
Tấm hóa-thành chim bay vào cung	Tam becomes a bird to fly to the palace
Vua linh-cảm chim là Tấm	The King has the sense that the bird is Tam
Vua quấn-quýt bên chim	The King becomes close friends with the bird
Cám ghen-tức	Cam envies
Cám giết chim	Cam kills the bird
Chim chết hóa-thành cây thị	The bird dies, becoming a diospyros decandra tree
Vua hái quả thị đem vào phòng	The King gathers the diospyros decandra fruit to bring to his room
Cám về nhà thăm mẹ	Cam visits home to see her mother
Tấm chui ra từ quả thị gặp vua	Tam emerges from the diospyros decandra fruit to meet the King
Vua lại phong Tấm làm hoàng-hậu	The King again proclaims Tam to become the Queen
Cám trở-về cung	Cam return to the palace
Tấm sai người dội nước-sôi giết Cám	Tam designates someone to pour boiling water down onto Cam
Tấm đem xác Cám làm nước mắm	Tam uses Cam's body to make fish sauce
Tấm tặng mắm cho mẹ-ghẻ	Tam presents the fish sause to the stepmother
Mẹ-ghẻ ăn khen ngon	The stepmother eat, praising its taste

III. Grammar

We choose the descriptive approach to develop a context-free grammar for our Vietnamese syntactic parser. In other words, the phrase-structure rules of our grammar optimally capture all the empirical traits of the above-mentioned text (see Table 1), and therefore embodies a extremely simplified version of the complete grammar for Vietnamese language (see Table 2). Particularly, this grammar only takes into consideration the simple sentences which “consists of two immediate constituents - the subject announcing a topic and the predicate which provides a comment on that topic” (Nguyen, Đ. (1997). London Oriental and African Language Library: Vietnamese. Amsterdam, NLD: John Benjamins Publishing Company. Retrieved from http://www.ebrary.com). To facilitate the computational process we model this structure of simple Vietnamese sentences in a more flexible formulation which allows the two mentioned constituent to break down into the ordered list of one obligatory noun phrase, one optional auxiliary, and one or more verb phrases.

In addition, as Vietnamese share the similar SVO word order with English, the lexical entries for verb phrases in our lexicon will vividly reflect the structures of English verb phrases which are explored in our Problem Set #4 except for the augmentative verbs, tagged as AUG, appearing after the head verbs in verb phrases to denote the completeness, the direction or the orientation of the actions denoted by the head verb. It is noticed that these augmentative verbs shares the similar characteristics with Chinese direction and result verbal complements present at http://resources.allsetlearning.com/chinese/grammar/Direction_complement and http://resources.allsetlearning.com/chinese/grammar/Result_complement.

Nevertheless, depart from English, Vietnamese has a different pattern of noun phrases which is demonstrated, for example, by the occurrence of complement adjective phrases in the posterior position of the head nouns.

Table 2. Grammar

Phrase-Structure Rules of our Context-free Grammar (excluding those rules whose right-hand sides are terminal node)					
S	→	NP	(AUX)	VP	(VP)*
NP	→	(CD)	(MW)	CN+	(AP)
VP	→	VP			
VP	→	VP	NP		
VP	→	VP	AP		
VP	→	VP	PP		
VP	→	VP	AUG		
VP	→	VP	NP	VP	
VP	→	VP	NP	PP	
VP	→	VP	NP	AUG	
VP	→	VP	AUG	NP	
VP	→	VP	AUG	PP	
VP	→	VP	NP	AUG	NP
AP	→	ADJ			
PP	→	PREP		NP	

Notes:

S – Start symbol	ADJ – adjective
NP – noun phrase	AUG – co-verb
VP – verb phrase	AUX – auxiliary
AP – adjective phrase	PREP – preposition
PP – prepositional phrase	CD – cardinal number
CN – common noun	MW – measure word

(CD and MW are referred from http://www.cis.upenn.edu/~chinese/posguide.3rd.ch.pdf)

Finally, as mentioned above, Vietnamese is an extremely isolating language; therefore, the concept of syntactic feature agreement deems irrelevant to it. As a result, we only keep in our feature list mainly the features of tense (and aspect), modality and negation (see Table 3).

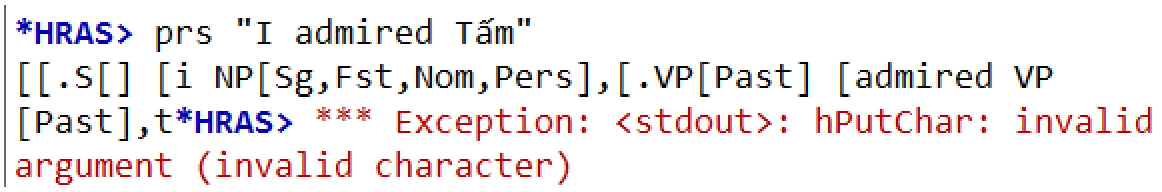
Table 3. Features of Auxiliaries

Auxiliary Feature		**Denotation**	**Example**
Tense and aspect	past	*Past*	đã
	progressive	*Prog*	đang
	future	*Futu*	sẽ
	present	*Pres*	(other cases)
Modality		*Moda*	không-thể (impossible)
Negation		*Nega*	không

IV. Challenges

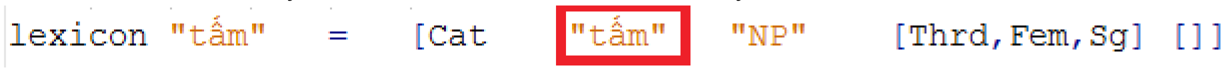
1. Output Data

As noted in http://book.realworldhaskell.org/read/characters-strings-and-escaping-rules.html that the interactive **GHCI** interpreter may not be able to deal with international characters, we encounter the message error when trying to get the parsing result.



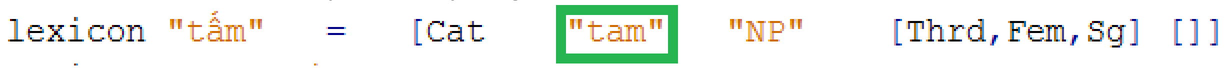
```
*HRAS> prs "I admired Tấm"
[[.S[] [i NP[Sg,Fst,Nom,Pers],[.VP[Past] [admired VP
[Past],t*HRAS> *** Exception: <stdout>: hPutChar: invalid
argument (invalid character)
```

This error is caused by the *Phon* value of the lexical entry such as "tấm" in this case.



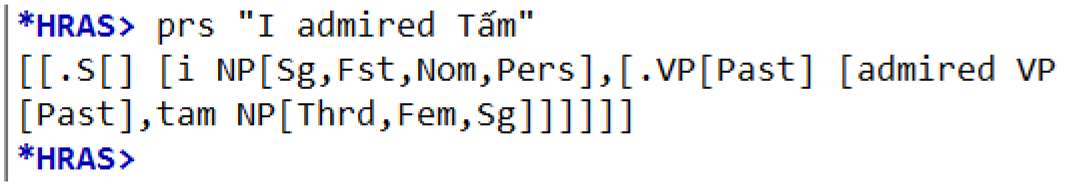
```
lexicon "tấm"    =    [Cat    "tấm"    "NP"    [Thrd,Fem,Sg]  []]
```

To overcome this shortcoming of the interactive **GHCI** interpreter, we decide to encode the Phon value of each lexical entry with only English letters as show below.



```
lexicon "tấm"    =    [Cat    "tam"    "NP"    [Thrd,Fem,Sg]  []]
```

As a result, the interactive **ghci** well returns the parsing result.

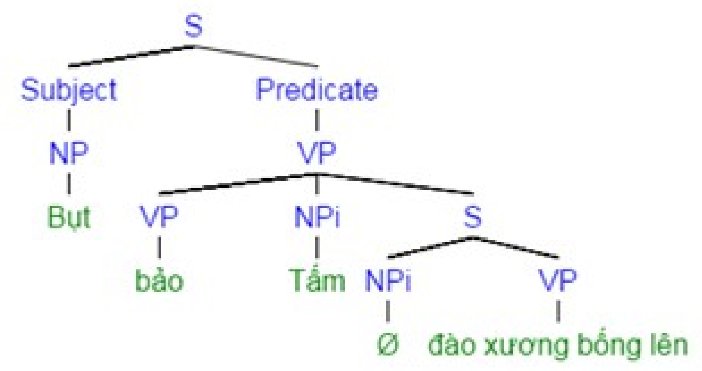


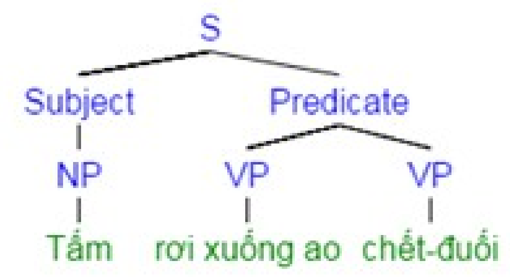
```
*HRAS> prs "I admired Tấm"
[[.S[] [i NP[Sg,Fst,Nom,Pers],[.VP[Past] [admired VP
[Past],tam NP[Thrd,Fem,Sg]]]]]]
*HRAS>
```

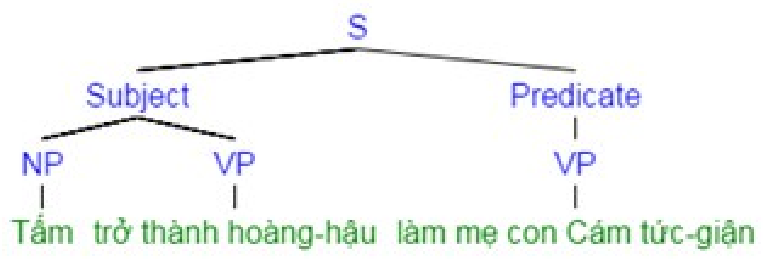
2. Structural Ambiguity

Here we only mention the most obvious structural ambiguity characterizing our examined Vietnamese simple sentences through the following examples which have the same surface structures:

Sentence 1	Bụt bảo Tấm đào xương bống lên

Surface structure	**NP**	**VP**	**VP**
	Bụt *Buddha*	bảo Tấm *tells Tam*	đào xương bống lên *to dig the goby's bones out*
Deep structure	 S Subject Predicate NP VP Bụt VP NPi S bảo Tấm NPi VP Ø đào xương bống lên		

Sentence 2	Tấm rơi xuống ao chết-đuối		
Surface structure	**NP**	**VP**	**VP**
	Tấm *Tam*	rơi xuống ao falls into the pond	chết-đuối drowning
Deep structure	 S Subject Predicate NP VP VP Tấm rơi xuống ao chết-đuối		

Sentence 3	Tấm trở thành hoàng-hậu làm mẹ con Cám tức-giận		
Surface structure	**NP**	**VP**	**VP**
	Tấm *Tam*	trở thành hoàng-hậu *becomes the Queen*	làm mẹ con Cám tức-giận (which) makes Cam and her mother angry
Deep structure	 S Subject Predicate NP VP VP Tấm trở thành hoàng-hậu làm mẹ con Cám tức-giận		

To disambiguate sentence 1 and sentence 2, we create a phrase-structure rule for sentence 1 as in Table 2:

VP → VP NP VP

However, our current grammar cannot disambiguate sentence 2 and sentence 3. In other words, it cannot define whether the first VP in these sentences belongs to the subject or the predicate. This is the reason why we decide to parse our sentences with a more flexible formulation which does not make any conclusion about the subject and the predicate of the examined sentences. Instead, our parser returns the sentence-initial NP and one or more following VPs.

V. Parser

The following changes have been made to the Parser (P.hs) in order to accommodate our Vietnamese grammar:

- Addition of wild-card type Parser function, in addition to *many*:
    - zeroOrOne :: Parser a b -> Parser a [b] : Parser to parse Phrase?
    - oneOrMore :: Parser a b -> Parser a [b]: Parser to parse Phrase+
- parseCD, parseMW: correspondingly to parse CD and MW lexicon.
- Modification of parseNP:
    - npRule = \ xs ->

```
        [ (Branch (Cat "_" "NP" [] []) (cd++mw++cns++ap),us) |
        (cd,ys) <- (zeroOrOne parseCD) xs,
        (mw,zs) <- (zeroOrOne parseMW) ys,
        (cns,ts)  <- (oneOrMore parseCN) zs,
        (ap, us) <- (many parseAP) ts ]
```

- npRule is modified to take into account for the cases appeared in the data. Noun phrase in Vietnamese is actually far more complex. Measure words could be multiples; before CD there could be also multiple types of modifiers. Determiner particles are generally put at the end, but they are not frequently used in writing, so we decided to leave it out of the data. The occurrences of multiple nouns at the core of NP could have different semantics. The typical case could be multiple noun-modifier to the right of the head. Another case could be a pair of possessor-thing possessed. The parser however cannot distinguish the case when the first noun is topic of the whole sentence, and the predicate is actually a whole S. For example:

a. Cái nồi cán đã hỏng rồi

Cái	nồi	cán	đã	hỏng	rồi
Mw	pot	handler	PAST-part	V-broken	PERF-complement
[NP		][VP			]

b. Cán cái nồi đã hỏng rồi

Cán	cái	nồi	đã	hỏng	rồi
handler	Mw	pot	PAST-part	V-broken	PERF-complement
[NP			][VP		]

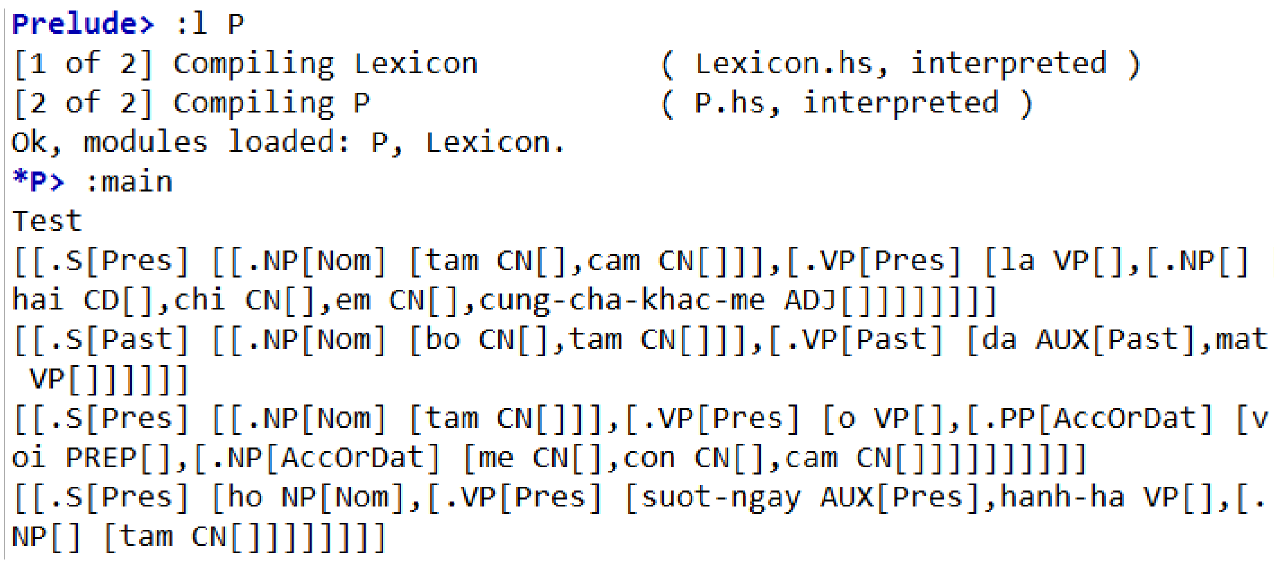
- Modification of parseVP:
    - vpRule = \xs ->

```
[ (Branch (Cat "_" "VP" fs []) (aux++[vp]++xps),ts) |
(aux, ys)   <- (many parseAux) xs,
(vp,zs)     <- leafP "VP" ys,
subcatlist  <- [subcatList (t2c vp)],
(xps,ts)    <- parseNPorPPorAPorAugorVPMany zs,
fs       <- getFeatureFromAux aux, match subcatlist (map t2c xps) ]
```

  - parseVP firstly parses the auxiliary at the beginning of VP. The auxiliary then is used to extract the tense/modality/negation feature, and these features are then lifted to VP.
  - The subcategorization list of verbs consists of NP, PP, AP, Aug or VP. This is actually a simplification/flattening of VP structure.
- Modification of parseSentence:
  - Sentence structure is modified to accommodate multiple Verb phrases.

VI. Project Demo

The project's code includes two haskell files named *P.hs* and *Lexicon.hs*. We integrate the test code in *main* function of module *P* which shows the parsing results of all 50 sentences in our text. Here is the screenshot of the corresponding testing process:



```
Prelude> :l P
[1 of 2] Compiling Lexicon          ( Lexicon.hs, interpreted )
[2 of 2] Compiling P                ( P.hs, interpreted )
Ok, modules loaded: P, Lexicon.
*P> :main
Test
[[.S[Pres] [[.NP[Nom] [tam CN[],cam CN[]]],[.VP[Pres] [la VP[],[.NP[]
hai CD[],chi CN[],em CN[],cung-cha-khac-me ADJ[]]]]]]]]
[[.S[Past] [[.NP[Nom] [bo CN[],tam CN[]]],[.VP[Past] [da AUX[Past],mat
 VP[]]]]]]
[[.S[Pres] [[.NP[Nom] [tam CN[]]],[.VP[Pres] [o VP[],[.PP[AccOrDat] [v
oi PREP[],[.NP[AccOrDat] [me CN[],con CN[],cam CN[]]]]]]]]]]
[[.S[Pres] [ho NP[Nom],[.VP[Pres] [suot-ngay AUX[Pres],hanh-ha VP[],[.
NP[] [tam CN[]]]]]]]]
```