Constituency Parsing of Complex Noun Sequences in Hindi

Arpita Batra[†], Soma Paul[†], and Amba Kulkarni[‡]

[†] Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India
arpita.batra@students.iiit.ac.in,soma@iiit.ac.in
[‡] Department of Sanskrit Studies, University of Hyderabad, India
apksh@uohyd.ernet.in

Abstract. A complex noun sequence is one in which a head noun is recursively modified by one or more bare nouns and/or genitives[1] Constituency analysis of complex noun sequence is a prerequisite for finding dependency relation (semantic relation) between components of the sequence. Identification of dependency relation is useful for various applications such as question answering, information extraction, textual entailment, paraphrasing.

In Hindi, syntactic agreement rules can handle to a large extent the parsing of recursive genitives (Sharma, 2012)[12].This paper implements frequency based corpus driven approaches for parsing recursive genitive structures that syntactic rules cannot handle as well as recursive compound nouns and combination of gentive and compound noun sequences. Using syntactic rules and dependency global algorithm, an accuracy of 92.85% is obtained.

Keywords: constituency parsing, bracketing, complex noun sequence, compound noun, genitives

1 Introduction

A noun can have various pre-modifiers such as adjective, adjectival phrase, bare noun (henceforth, compound noun), genitive noun. The case becomes complex when a head noun[2] is modified recursively as in

1. (*ladake* *kA* (*mittI* *kA* *ghar*))
   "boy" genitive-marker "mud" genitive-marker "house"
2. (*jilA* (*nirvAchan* *adhikArI*))
   "district" "election" "officer"

Or a head noun is modified by a complex modifier. Example:

[1] Genitive markers in Hindi are kA, and its allomorphic variations ke and kI.
[2] Hindi is a head final language

3. ((*AdamI* *ke* *bete*) *kA* *ghar*)
   "man" genitive-marker "son" genitive-marker "house"
4. ((*krishi* *prasanskaraNa*) udyog)
   "agriculture" "processing" "industry"

Complex noun sequence is a sequence having multiple nouns. Nouns may or may not be separated by genitive markers. When no genitive marker is present in between the nouns, then such sequence is known as compound noun. A noun sequence can be represented as the following regular expression[1]:

(*n*oun+ [3] *g*enitive-marker)* [4] *n*oun+

Binary constituency parsing of noun with complex modifier is an important requirement for determining the semantic relation between noun and its modifier. (Sharma, 2012)[12] uses agreement rules for parsing nouns having recursive genitive modifiers and reports an accuracy of 80%. Syntactic agreement rules alone fail to determine the constituents when the allomorphic forms of genitive are same as in:

5. (*ladake* *kA* (*patthar* *kA* *ghar*))
   "boy" genitive-marker "stone" genitive-marker "house"
6. ((*vimAnoM* *kI* *kharId*) *kI* *yojanA*)
   "aircraft" genitive-marker "purchases" genitive-marker "plan"

Syntactic rules also fail for bare noun sequences. For handling cases where rules cannot be applied, we use the frequency based method. Detail approach and results are discussed in Section 3 and 4 respectively.

2 Related Work

Corpus driven approaches are prevalent for handling constituent analysis of compound nouns. (Pustejovsky et. al., 1993)[11] has used bigram frequency for bracketing compound nouns having three noun constituents. For compound noun a-b-c, where "a", "b" and "c" are noun constituents, if frequency of a-b is greater than frequency of b-c, then the result is ((a-b)-c), else the result is (a-(b-c)). (Lauer, 1995)[9] has used conceptual association instead of lexical association for finding the better pair. This helps in handling sparsity. Conceptual association is found in between noun categories, obtained from Roget's thesaurus. In this paper, the concept of adjacency and dependency pairs was used for the first time. In adjacency model, semantically better adjacent pairs, a-b or b-c is found. If a-b is semantically better than b-c, then the output is ((a-b)-c), otherwise the output is (a-(b-c)). For dependency model, we find whether a modifies b or a modifies c. If a-b is semantically better than a-c, then the result is left bracketed

[3] x+ matches any sequence that contains atleast one x
[4] x* matches any sequence that contains zero or more occurrences of x

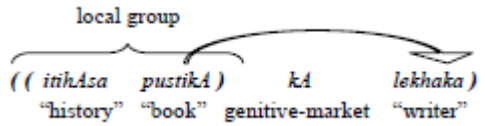
((a-b)-c), otherwise the result is right bracketed (a-(b-c)). (Keller and Lapata, 2005)[8] has also used both the adjacency and dependency based models. But association is compared using frequencies of the adjacency and dependency pairs. These frequecies are obtained from interpolation of frequency obtained from web and corpus. (Girju et. al., 2005)[3] has used the conceptual association to find the parse. Noun classes from WordNet are used for determining the output.

(Kulkarni and Kumar, 2011)[6] has used conditional probability to decide the compatibility between the two words. For noun compound a-b-c, it compares the probability of occurrence of a-b and b-c. If occurrence of a-b is more probable, then the ouput is ((a-b)-c), else the ouptut is (a-(b-c)). This approach is also used in (Kulkarni et. al., 2012)[7] for bracketing compounds of Indian languages like Sanskrit, Hindi, Marathi. (Kavuluru and Harris, 2012)[5] has used both non-greedy and greedy based approach for bracketing compound nouns having four constituents. These compound nouns are from biomedical domain. Greedy approach is based on comparing frequencies of the possible components which can be joined. Initially, a and b, b and c, c and d are the possible components which can be joined. After this combination, second best combination is found. For greedy approach, we have used lexical association[10], jaccard index[4] to find the best pair. This helps in normalizing the frequency. In non-greedy approach, cohesion measure is calculated. Cohesion values are found for all possible trees. This is obtained by summing the relatedness of each node and relatedness is found by using jaccard index. Then, we choose the output with the highest cohesion value.

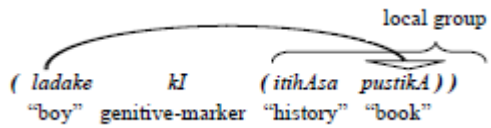
3 Approaches

In this paper, we handle complex noun constructions where the modifier can be a series of bare nouns or a combination of genitive and bare nouns. As a first step, we locally group bare sequences of noun within a complex sequence. Such group is called here local group. The head of the local group can either modify a noun outside the group (as in 7) or it can itself be modified by a noun outside the group (as in 8):

7.



8.



After the formation of local group (if there is any), the syntactic rules (Sharma, 2012)[12] are applied for genitives. Finally, frequency based approach is applied for the cases where syntactic rules fail. Same approach is also followed for further constituent analysis of local groups. The detail of this step is discussed below.

Frequencies for both adjacency and dependency pairs are used to parse the noun sequences. Adjacency pairs refer to the two sequences of noun adjacent to each other in a complex sequence. Dependency pairs are the two sequences of noun which are not necessary to be adjacent to each other but their relative positions are maintained. Let a-b-c-d be the sequence with a, b, c, d as the constituent nouns or sequence of nouns. Adjacency pairs in this compound are: a-b, b-c and c-d. Dependency pairs in this compound are: a-b, a-c, a-d, b-c, b-d and c-d.

We have used both greedy and global approaches for finding the parsed output. Greedy approach refers to the concept in which we find the best pair from all possible pairs and then we find the second best pair. If a wrong pair is chosen, then the error percolates up in the whole parse. Consider an example with a-b-c-d as the sequence. Let the expected output be (a((bc)d)). If a-b is found to be the best pair initially, then possible generated output are ((ab)(cd)) and (((ab)c)d). Due to wrong selection of (ab), output (a((bc)d)) cannot be obtained. To avoid this, global approach has been used. Global approach refers to the concept in which we consider all possible bracketing of a compound. Sequence a-b-c-d have five possible bracketings which are (((ab)c)d), ((ab)(cd)), ((a(bc))d), (a((bc)d)), (a(b(cd))[5]. For all five cases, cohesion value is calculated. We choose the output which results in maximum cohesion value [5].

In total, four approaches have been used: adjacency global, adjacency greedy, dependency global and dependency greedy for parsing sequences where rules do not apply. For all approaches, if jaccard index[4] measured for the pair is found to be zero, then we take left bracketed result as default because it has been observed (see Table 1) that left bracketing is more frequent than right bracketing in Hindi. Bare nouns in a corpus are found to be very less. Therefore, to get better frequencies of the sequence, we have considered both bare as well as genitive nouns because of their same nature. Ex: "*kavi sammelan*" is equivalent to"*kavi kA sammelan*" (meeting of poets). This helps in dealing with sparsity to an extent.

[5] For sequence having 'n+1' number of noun constituents, number of possible bracketing is equal to catalan number, C_n[6]

Table 1. Distribution of left bracketed and right bracketed sequences having three noun constituents. (*a, b, c in below table denotes noun and gen denotes genitive marker.*)

Type of Complex Noun Sequence	Left Bracketed	Right Bracketed
Compound Noun (a b c)	159 ((a b) c)	78 (a (b c))
Nouns separated by one genitive-marker (a gen b c) or (a b gen c)	683 ((a b) gen c)	306 (a gen (b c))
Nouns separated by two genitive-markers (a gen b gen c)	502 ((a gen b) gen c)	42 (a gen (b gen c))

3.1 Adjacency Greedy Approach

It is the approach in which we have considered bracketing adjacent pairs greedily. Lexical association[10] for adjacent pairs is found using jaccard index[4]. In this approach lexical association of all pairs are compared and the best pair is chosen. After this step, another best pair is chosen till the whole parse tree is formed. Algorithm in detail is described below:

Algorithm:

(a) Let there be 'n' noun components: W_i, i ranging from 1 to n.
(b) Compute the lexical association for each adjacent pair of constituents.
(c) Select the pair with the highest association. Let the pair be W_{k-1} and W_k.
(d) Join W_{k-1} and W_k nodes.
(e) Remove W_{k-1} and W_k from the constituent list.
(f) Insert W_{k-1}-W_k as the single constituent in the list.
(g) Repeat steps "a" to "f" till all the nodes are joined.

$$Association(p,q) = \frac{freq(pq)}{freq(p) + freq(q) - freq(pq)} \tag{1}$$

where, "p" and "q" are the elements of constituent list.

3.2 Adjacency Global Approach

It is the approach in which cohesion value[5] for all possible bracketing for adjacent pair is considered and one with the highest cohesion value is chosen. Cohesion for a tree is measured by summing the association[10] corresponding each node of a tree. Association for a node in a tree is obtained by using jaccard index[4]:

$$Association(i,j) = \frac{freq(ij)}{freq(i) + freq(j) - freq(ij)} \tag{2}$$

where, “i” and “j” are the nodes which are combined to form a single node.

Consider (a((bc)d)) as an example. To find the cohesion of this tree, lexical association for b-c, bc-d and a-bcd are added. For more clarification, association for each node corresponding (a((bc))d)) tree is shown below. Tree representation for (a((bc)d)) is shown in Fig. 1.

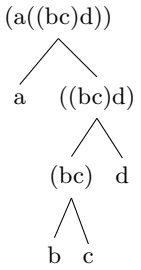


Fig. 1. Tree Representation for bracketed (a((bc)d) sequence

$$Association(b,c) = \frac{freq(bc)}{freq(b) + freq(c) - freq(bc)} \tag{3}$$

$$Association(bc,d) = \frac{freq(bcd)}{freq(bc) + freq(d) - freq(bcd)} \tag{4}$$

$$Association(a,bcd) = \frac{freq(abcd)}{freq(a) + freq(bcd) - freq(abcd)} \tag{5}$$

$$Cohesion([a[[bc]d]]) = Association(b,c) + Association(bc,d) + Association(a,bcd) \tag{6}$$

3.3 Dependency Greedy Approach

It is the approach in which we have considered bracketing dependency pairs greedily. Lexical association[10] for dependency pairs is found using jaccard index (see Formula 1)[4]. Lexical association of all pairs are compared and the best pair is chosen. After this step, another best pair is chosen till the whole parse tree is formed. For approaches which use dependency pairs, the concept of head percolating up has been used. Since, Hindi is a head final language, therefore the last noun is percolated up. This approach is better than adjacency greedy because it is difficult to find the sequence having more than two nouns in a corpus. However, it is relatively easier to find the frequency of the sequence having two nouns from the corpus.

Algorithm:

(a) Let there be 'n' noun components: W_i, i ranging from 1 to n.
(b) Compute the lexical association for each dependency pair of constituents.
(c) Select the pair with the highest association. Let the pair be W_{k-1} and W_k.
(d) Join W_{k-1} and W_k nodes.
(e) Remove W_{k-1} from the constituent list. (head percolation)
(f) Repeat steps "a" to "e" till all the nodes are joined.

3.4 Dependency Global Approach

It is the approach in which cohesion value[5] for all possible bracketing is measured and one with the highest is chosen. In this approach, lexical association for a node depends on the head. It does not depend on all the constituents. Formula for calculating association value for each node is shown below:

$$Association(i,j) = \frac{freq(head(i)head(j))}{freq(head(i)) + freq(head(j)) - freq(head(i)head(j))} \tag{7}$$

Where, "i" and "j" are the nodes being joined. head(i) is the function which returns the head of a node. Since, Hindi is a head final language, therefore the last noun is returned.

Consider (a((bc)d)), cohesion is measured by adding jaccard index value for b-c, c-d and a-d. In adjacency model, jaccard index for b-c, bc-d and a-bcd were measured. Here, instead of bc-d, we take c-d, since c is the head of bc which is already been bracketed. Association for each node of (a((bc)d)) is shown below. For more clarification, tree representation is already shown in Fig. 1.

$$Association(b,c) = \frac{freq(bc)}{freq(b) + freq(c) - freq(bc)} \tag{8}$$

$$Association(bc,d) = \frac{freq(cd)}{freq(c) + freq(d) - freq(cd)} \tag{9}$$

$$Association(a,bcd) = \frac{freq(ad)}{freq(a) + freq(d) - freq(ad)} \tag{10}$$

$$Cohesion([a[[bc]d]]) = Association(b,c) + Association(bc,d) + Association(a,bcd) \tag{11}$$

4 Experiments and Results

For experiment, we have extracted complex noun sequences from Hindi Dependency TreeBank that has been released in (Coling, 2012)[2] for the shared task on machine translation and parsing. We have obtained total 2322 noun sequences. Out of them, the number of noun compounds is 258. There are 603 sequences of genitives of the structure [noun1 kA noun2 (kA noun3) +]). Rest of the data

contains both genitive as well compound sequences. Table 2 shows the distribution of complex noun sequences in the data used for conducting the experiments. The third column in the table shows the number of cases that cannot be handled by syntactic agreement rules of genitives and therefore required to be parsed by our proposed approach.

Table 2. Data used for the experiments and distribution of various types of complex nouns

Type of Complex NounSequence	**Total number of Sequences**	**Total number of sequences where agreement rules not help**
Compound nouns [a-b-c+]	258	258
No compound noun in genitives [a gen b (gen c)+]	603	228
Noun sequence [(a+ gen)* b]	2322	654

The following table presents the result of applying the syntactic agreement rules and the four frequency based experiments conducted in the present work on all kinds of complex nouns.

Table 3. Accuracy for (a) compound noun, (b) no compound noun in genitives and (c) all complex noun sequences

Method	**Accuracy for compound nouns [a b (c)+]**	**Accuracy for the case where no compound noun is in genitives [a gen b (gen c)+]**	**Accuracy for complex noun sequences [(a+ gen)* b+]**
Adjacency greedy	65.89	88.05	89.62
Adjacency global	61.62	89.05	89.92
Dependency greedy	67.82	88.22	89.70
Dependency global	**68.60**	**93.03**	**92.85**

We find from the above table that dependency global approach outperforms the other three approaches for all kinds of complex noun sequences. However the result of the analysis of only compound nouns is the poorest as is evident from the above table.

In Hindi, noun sequence with three noun constituents occur more often than others. Therefore, detail results for sequences containing three noun constituents is shown in Table 4. Accuracy obtained for compound nouns using dependency global approach is 71.30% while for genitives, it is 93.93%. Here also, we can see that dependency global is performing better than adjacency global and adjacency greedy. However, in case of genitives, much difference is not seen in adjacency greedy method and adjacency global method. Their accuracies are 89.15% and 89.52% respectively.

Table 4. Accuracy for sequence [a b c] and [a gen b gen c] containing three noun constituents

Method	**Accuracy for compound nouns [a b c]**	**Accuracy for the case where no compound noun is in genitives [a gen b gen c]**
Adjacency greedy	67.08	89.15
Adjacency global	63.29	89.52
Dependency greedy	69.19	89.15
Dependency global	**71.30**	**93.93**

Table 5 shows f-score for left bracketed and right bracketed bare nouns. Table 6 shows f-score for left bracketed and right bracketed sequence containing three noun constituents and two genitive markers. Left bracketed sequence is having better f-score than right bracketed sequence in all the four methods. F-score of 81.42 and 52.5 is obtained for left bracketed and right bracketed nominal compounds respectively. For the case of genitives, f-score is better than that of compound nouns. F-score measured for left bracketed and right bracketed genitives is 96.70 and 62.06 respectively. Accuracy for nominal compounds and genitives with four noun constituents is shown in Table 7.

Table 5. Precision, recall and f-score for three constituent compound nouns [a b c]

Method	((a b) c)			(a (b c))		
	Precision	**Recall**	**f-score**	**Precision**	**Recall**	**f-score**
Adjacency greedy	76.12	74.21	75.15	50	52.56	51.25
Adjacency global	72.78	72.32	72.55	44.30	44.87	44.58
Dependency greedy	80.71	71.06	75.58	52.57	64.38	52.28
Dependency global	71.98	93.71	**81.42**	51.21	53.84	**52.5**

Table 6. Precision, recall and f-score for complex sequence containing three nouns separated by two genitive-markers[a gen b gen c]

Method	((a gen b) gen c)			(a gen (b gen c))		
	Precision	**Recall**	**f-score**	**Precision**	**Recall**	**f-score**
Adjacency greedy	96.63	91.43	93.96	37.68	61.90	46.84
Adjacency global	96.64	91.83	94.17	38.80	61.90	47.70
Dependency greedy	96.63	91.43	93.96	37.68	61.90	46.84
Dependency global	96.99	96.41	**96.70**	60	64.28	**62.06**

Table 7. Accuracy for compound nouns and genitives containing four noun constituents

Method	Accuracy for compound nouns	Accuracy for genitives
Adjacency greedy	57.89	76.78
Adjacency global	47.36	83.92
Dependency greedy	57.89	78.57
Dependency global	42.10	83.92

5 Analysis of Results

We see in Table 3 that accuracy for constituency parsing of recursive compound nouns is less than that of the genitives. This is because syntactic agreement rules resolves the parse for genitives for 71.83% cases. We also observe that approaches

based on dependency pairs perform better than approaches based on adjacency pairs. Adjacency pairs even include sequences with more than two nouns. Such sequences are difficult to be found in corpus[6]. For the same reason, not much difference is seen in adjacency greedy and adjacency global methods. But for the models based on dependency pairs, frequency for two noun constituents is needed, which is found relatively easier in the corpus. This helps in improving accuracy to an extent.

Further, we see that global based approaches are better than greedy based approaches. This is because in global based approaches, whole context is considered in deciding the output. While in greedy based approaches, only nearby sequence is considered. Whole context is not seen in this case. Therefore, during computation, some information gets missed.

We see that right bracketed sequence has better precision than left bracketed sequence. This means that there are less number of cases where ((ab)c) is the expected output but the result obtained is (a(bc)). Reason for this is lexical association for b-c is found to be better than association of a-b in adjacency based approach. And for dependency based approach, the association of a-c or b-c is better than association of a-b. This type of error was found in the sequence,
rakshA *shodha* *sevA*
"defense" "research" "service"
For the above sequence, expected output is ((rakshA shodha) sevA), but the output obtained is (rakshA (shodha sevA)). For this sequence, lexical association of "shodha sevA" and "rakshA sevA" is found to be better than the association value of "rakshA shodha". Therefore, in both adjacency and dependency based approach result is (rakshA (shodha sevA)).

Less precision is observed for the right bracketed sequence because there are large number of cases where (a(bc)) is the expected output and the result obtained is ((ab)c). One of the reasons for this is non-occurrence of a-b, a-c and b-c in the corpus from which frequency is obtained and therefore, left bracket is taken as the default output. Another reason is lexical association for a-b is found to be better than the association of b-c in adjacency based approach. And for dependency based approach, the association of a-b is better than a-c and b-c. This type of error was observed in the sequence,
rAjya *pashupAlana* *vibhaAga*
"state" "herding" "department"
For the above sequence, expected output is (rAjya (pashupAlana vibhAga)). But the result obtained is ((rAjya pashupAlana) vibhAga). Here, association value of "rAjya pashupAlana" is better than association value of "pashupAlana vibhAga" and "rAjya vibhAga". Therefore, left bracket is obtained as the result. There are more such similar cases and hence resulting in less precision of right

[6] corpus was collected by crawling web pages

bracketed sequence.

6 Conclusion

This paper presents four approaches for constituency parsing of complex nouns in Hindi. Dependency models are found more efficient than adjacency models in handling sparsity of noun sequences and therefore performs better parsing. As part of future work, we attempt to perform the global dependency experiment on a bigger corpus. We also aim to use similarity measures to handle data sparsity issues.

References

1. Chapter 9: Regular expressions. (2004), the Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition, The Open Group.
2. Hindi dependency treebank (2012), workshop on MT and Parsing in Indian Languages. 24th International Conference on Computational Linguistics.
3. Girju, R., Maldovan, D., Tatu, M., Antohe, D.: On the semantics of noun compounds. Computer Speech and Language (2005)
4. Jaccard, P.: The distribution of the flora in the alpine zone. The New Phytologist (1912)
5. Kavuluru, R., Harris, D.: A knowledge-based approach to syntactic disambiguation of biomedical noun compounds. In: Proceedings of COLING (2012)
6. Kulkarni, A., Kumar, A.: Statistical constituency parser for sanskrit compounds. In: Proceedings of ICON (2011)
7. Kulkarni, A., Paul, S., Kulkarni, M., Kumar, A., Surtani, N.: Semantic processing of compounds in indian languages. In: Proceedings of COLING (2012)
8. Lapata, M., Keller, F.: Web-based models for natural language processing. ACM (2005)
9. Lauer, M.: Corpus statistics meet the noun compound: Some empirical results (1995)
10. Pecina, P.: Lexical association measures. Institute of Formal and Applied Linguistics (2009)
11. Pustejovsky, J., Bergler, S., Anick, P.: Lexical semantic techniques for corpus analysis. Association for Computational Linguistics (1993)
12. Sharma, S.: Disambiguating the parsing of hindi recursive genitive constructions. IIIT Hyderabad, India (2012)