TYPE-IDENTIFIER FOR NAVYA-NYĀYA EXPRESSIONS

Amba Kulkarni* and Arjuna S.R.**

Navya-Nyāya (NN) school of philosophy, an offshoot of Nyāya school of philosophy, apart from philosophical contributions, has contributed a scientific methodology and terminology to express the philosophical thoughts and debates in an unambiguous way. This scientific terminology and methodology leads to linguistic expressions which are typically hard to comprehend due to their linear structure, long compounds and extensive usage of derivational suffixes (Taddhitas). In order to make the comprehension easy, we have built a semi-automatic computational tool. This tool has four components viz. a segmenter, compound parser, type identifier and a graph renderer. In this paper we describe the compound type identifier.

1 Introduction

At the outset, let me[1] take this opportunity to congratulate Prof. Pahi for receiving much deserving lifetime achievement award. I also take this opportunity to thank him for fruitful discussions I had with him at several occasions at Hyderabad and Jaipur on mutually interested topics ranging from linguistics, vaiśeṣika ontology to Navya Nyāya and logic, and I wish in future also I'll have more such occasions of academically engaging discussions with him.

Navya-Nyāya (Neo-Logic), an offshoot of Nyāyaśāstra (Indian Logic), is one of the fundamental schools of Indian philosophy. The seeds of *Navya-Nyāya* (NN) School of Philosophy are found in the Udayanāchārya's work[5]. Gangeśa (century), the founder of NN in his

Tattvacintāmaṇi introduced the technical language of NN. This language deals with the verbal cognition, logic and epistemology. It introduces a few conceptual terms and provides a mechanism to express the underlying cognitive structure corresponding to a linguistic expression in an unambiguous way.

An NN expression (NNE) is a compound, written as a single word without any break or hyphen in between the components. Components when joined to form a compound undergo some changes. For example, the case information of the components joined is lost. The components also loose their accents, and there is mandatory euphonic change (sandhi operation) involved during compound formation. This also makes a compound ambiguous. Anil Kumar et al.[17] describe the steps involved in processing Sanskrit compounds and also discuss the associated computational complexity. The steps involved in the analysis of compounds are as follows.

1. Splitting a compound into components.

   This involves undoing euphonic transformations.

2. Analysing its constituent structure.

   At this stage a compound is analysed showing how the components are grouped together.

3. Identifying relations between the components.

   Now the relation between the components thus grouped is made explicit.

4. Providing a paraphrase of the compound.

   Finally a paraphrase of the compound is generated.

We illustrate these steps with two examples: an English one followed by a NN expression [2]

Example 1: Consider an English compound 'lake water pollution reduction log'. We skip the first step, since the components here are already split.

1. Constituency analysis for this compound is

   ((((lake-water)-pollution)-reduction)-log)

2. Type of a compound is now marked which specifies the relations between the components.

   ((((lake-water)**T7**-pollution)**T6**-reduction)**T7**-log)**T6**

   Here **T** stands for *Tatpuruṣa* (an endo-centric) compound and the numbers 6 and 7 indicate the genitive and the locative case markers.

3. The paraphrase of a compound is obtained by supplying the missing prepositions corresponding to the compound type. Thus the paraphrase of the above compound is

   Log of the reduction in pollution of water in lake.

Example 2: Consider now an NN expression which defines earth as a substance which has smell as its characteristic property.

gandhatvāvacchinnagandhaniṣṭhādheyatānirūpitādhikaraṇatāvatī. (1)

1. After splitting the compound into its components, we get

   gandhatva-avacchinna-gandha-niṣṭha-ādheyatā-nirūpita-adhikaraṇatā-vatī.

   Here the components are separated by hyphen and the derivational suffix '-vatī' is separated by a caret(^) sign.

2. The constituency parse of this compound is

   ((((gandhatva-avacchinna)-((gandha-niṣṭha)-ādheyatā))-nirūpita)-adhikaraṇatā)^vatī

3. After identifying the relations between the components, we get

   ((((gandhatva-avacchinna)**T3**-((gandha-niṣṭha) **T7**-ādheyatā)**K**) **K**-nirūpita) **T3**-adhikaraṇatā)^vatī

   where *K*, *T*3, and *T*7 stand for *Karmadhāraya*, *Tatpuruṣa* with instrumental case and *Tatpuruṣa* with locative case suffix respectively. These are all endo-centric compounds, with a requirement of nominative, instrumental and locative case suffixes during paraphrasing.

4. Finally the paraphrase of this compound is

Sanskrit: *gandhatvena avacchinnā, gandhe niṣṭhā yā ādheyatā, tannirūpitā adhikaraṇatāvatī*

Gloss: by smell-ness delimited in smell residing which substratum-ness determined by that superstratum-ness possessing

English: An object which has substratum-ness which is determined by the superstratum-ness that is residing in the smell and is delimited by the smell-ness.

We notice mainly two problems in understanding this technical language. The first one is understanding the semantics of the basic technical vocabulary and the second one is understanding the underlying syntactic structure of the linguistic expression. While there have been several works in understanding the technical vocabulary by eminent scholars such as Ingalls[9], Matilal[20], Shaw[23], Bhattacharyya[4], and others, there have been very few attempts towards understanding the syntactic structure of NN expressions. A few notable efforts in this direction are by Bhattacharyya[4] and Ganeri[6]. Further the graphical representation of these expressions by Jha[10], Kitagawa[25] and Kulkarni[11] have resulted in better understanding of them. With the advent of technology, and with a better understanding of the syntax of NN expressions, now it is possible to build computational tools which can render the NN expressions graphically. *Nyāyacitradīpikā* is a semi-automatic computational tool for rendering the expressions graphically built by the authors.

Nyāyacitradīpikā, consists of four components - NN Segmenter, NN Parser, Graph generator and NN type-identifier. The NN Segmenter splits a compound into its components. The NN Parser, a semi-automatic tool, parses the segmented NNEs. The Graph generator generates two types of graphs - a Conceptual Graph(CG) and a Compressed CG from the parsed NNE. Detailed descriptions of these tools are reported in Arjuna and Kulkarni[3].

In this article, we introduce the NN type-identifier, a component that identifies the type of a compound in the parsed NN expression. In the next section, we give a brief summary of earlier efforts in building a compound type-identifier. In the third section, we define a context-free

grammar for type identifier and describe important cues for deciding the type of a compound. In the last section, we conclude with our observations on the utility of this endeavour.

2. Earlier efforts

Semantically Pāṇini classifies the Sanskrit compounds into four major types: a) Avyayībhāva, b) Tatpuruṣa, c) Bahuvrīhi, d) Dvandva. This classification is not sufficient for complete understanding of a compound. For example, the paraphrase of a compound *gandhatvāvacchinna* is *gandhatvena avacchinna*, while the paraphrase of *ghaṭābhāvaḥ* is *ghaṭasya abhāvaḥ*, and both of them belong to the same class of *Tatpuruṣa*. In the given instances, the paraphrases are different due to the semantic differences and it happens in all the types of compound. Based on their semantic differences, these compounds are further sub-classified into 59 sub-types. The types and sub-types of compound are based on standards evolved by the project entitled *"Development of Sanskrit Computational Tools and Sanskrit-Hindi Machine Translation System"* sponsored by Ministry of Information Technology, Government of India, New-Delhi.

There are very few efforts in the automatic detection of a compound type and sub-type. Anil et al.[17] discuss the steps involved in the analysis of Sanskrit compounds and show how the corpus statistics helps in building a type identifier. Later in 2013, Kulkarni and Anilkumar[16] used clues from Pāṇini's Aṣṭhādhyāyī in identifying the types of certain compounds. They have critically gone through the Pāṇinian sūtras related to compound formation providing semantic clues. They used a proper combination of both statistical and rule-based methods to decide the compound type semi-automatically. Kulkarni et al.[13] extend this work further for modern Indian languages. They discuss how semantic classification of Pāṇini can be applied to modern Indian languages, including English. According to them, access to the semantic content of the components and also wider context is needed to decide the type of a compound. This work was carried out with focus on Hindi and Marathi.

The NNEs have specific vocabulary and the compound type is predictable in such cases. We decided to take advantage of this fact and build a domain-specific type-identifier for NNEs.

3 Analysis of NNE compounds

Compounding of words is always between two components at a time, except in *Dvandva*, *Bahupada-Bahuvrīhi* and *Bahupada-Tatpuruṣha* compounds. Regarding the components of an NN compound, Matilal[18] observes: 'The NN expression involves a small number of technical terms together with a non-logical vocabulary'. Thus, the components in a NN compound are of two types: 1) a technical term, and 2) a non-technical term. Examples of technical terms are *avacchinna*, *nirūpita*, *niṣtha*. Technical and non-technical terms both can appear either as a first (*pūrvapada*) component or as a second (*uttarapada*) component. For example, in a compound *ghaṭatva-avacchinna*, the term *avacchinna* is the second term. Now consider a compound with three components *ghaṭatva-avacchinna-ādheyatā*, which is parsed as *((ghaṭatva-avacchinna)-ādheyatā)*. Here, *avacchinna* is the second component of a compound *ghaṭatva-avacchinna*, and is the semantic head of the compound. This compound is further combined with *ādheyatā* to form a complex compound[3]. Since *avacchinna* is the semantic head of the first compound-component, instead of calling *ghaṭatva-avacchinna* the first component, we may briefly refer *avacchinna* as the first-component. Similarly, we can find examples where the words *niṣtha*, *nirūpita*, etc. also can appear either as the second term or as the first term of a component.

3.1 Semantics of compounds with *avacchinna* as one component

The term *avacchinna*, as Ganeri[6] describes it, is a sentence forming operator. It joins two terms say 'X' and 'Y' into a sentence 'X-avacchinna-Y'. Now 'X-avacchinna-Y' is always to be parsed as *((X-avacchinna)-Y)*. And this will be then paraphrased as 'Y which is delimited by X'. For example, the three component compound *((ghaṭatva-avacchinna)-ādheyatā)* will be paraphrased as *ghaṭatvena-avacchinnā yā sā ādheyatā* 'The super-stratum-ness, which is delimited by pot-ness'. Generalising this, the paraphrase of *((X-avacchinna)-Y)*

will be 'X{3} avacchinna yat tat Y', where {3} indicates the instrumental case suffix. Now, following the annotation scheme developed by the Sanskrit consortium, we represent this as '((X-avacchinna)T3-Y)K1', where 'T3' stands for an endo-centric compound with a relation expressed by the instrumental case suffix (*Ṭṭīyā-Tatpuruṣa*), and 'K1' stand for a copulative compound with adjective as the first component (*Viśeṣaṇa-pūrvapada-Karmadhāraya*).

3.2 Semantics of compounds with *niṣṭha* as one component

The term *niṣṭha* is another sentence forming operator as described by Ganeri. Similar to *avacchinna*, 'X-niṣṭha-Y' is always to be parsed as *((X-niṣṭha)-Y)*. And this will be then paraphrased as 'Y (which is) residing in X'. For example, the three component compound *((ghaṭa-niṣṭha)-ghaṭatvam)* will be paraphrased as *ghaṭe-niṣṭham yat tat ghaṭatvam*. If we generalise, the paraphrase of *((X-niṣṭha)-Y)* will be 'X{7} niṣṭha yat tat Y', where {7} indicates the locative case suffix. Now, following the same annotation scheme, we represent this as '((X-niṣṭha)T7-Y)K1', where 'T7' stands for an endo-centric compound with a relation expressed by the locative case suffix (*Saptamī Tatpuruṣa*).

3.3 Semantics of compounds with *nirūpita* as one component

The term *nirūpita* is termed as a conditioning operator by Ganeri. It also takes two arguments 'X' and 'Y' to generate a new term 'X-nirūpita-Y'. This expression is always parsed as *((X-nirūpita)-Y)*. It's paraphrase is 'Y (which is) described by X'. For example, the three component compound *((ādheyatā-nirūpita)-adhikaraṇatā)* will be paraphrased as *ādheyatayā-nirūpitā yā sā adhikaraṇatā*. If we generalise, the paraphrase of *((X-nirūpita)-Y)* will be 'X{3} nirūpitā yat tat Y', where {3} indicates the instrumental case suffix. Now, following the same annotation scheme, we represent this as '((X-niṣṭha)T3-Y)K1', where 'T3' stands for an endo-centric compound with a relation expressed by the instrumental case suffix (*Ṭṭīyā-Tatpuruṣa*).

Now we look at the context-free grammar of this tool.

4 Context-free grammar

If a compound has only two components, then the extraction of both the component is trivial. However, when a compound has more than two

components, we need to have a parse of the compound showing how the components are joined one at a time. With a compound with 'n' components, there will be 'n-1' compounds. Each compound will have two components, which may be compound themselves. Thus given such a parsed structure, expressed linearly, now we need to scan this parsed structure and identify the first and second components at each level. This problem is similar to evaluation of a mathematical expression involving nested parenthesized expressions. We give below the context-free grammar to analyse such an expression. For each nested compound, the head is synthesized from its components. The context free grammar with rules for synthesizing the attributes is given in Table-1.

NNE	:	compound
		↑.head = ↓.head
	;	
compound	:	Ppada '-' Upada
		↑.head = Upada.head
	;	
Ppada	:	'(' **pada**
		↑.head = pada.head
	;	
Upada	:	**pada** ')'
		↑.head = pada.head
	;	
pada	:	compound
		↑.head = ↓.head
	|	concept
		↑.head = ↓.head
	;	

Table 1: Production rules with attributes

In this grammar, 'Ppada' stands for a *pūrvapada* 'the first component' and 'Upada' stands for an *uttarapada* 'the second component'. The input for this grammar is a parsed compound. For instance, input will be *((gandha-niṣṭha)-ādheyatā)*. In this input,

'*(gandha*' is the *pūrvapada* and '*niṣṭha)*' is the *uttarapada* for the first level of compounding. Then at next level, '*((gandha-niṣṭha)*' is the *pūrvapada* and '*ādheyatā)*' is the *uttarapada*. The word or pada after the symbols (,) is the key to identify the types of the compound in any NNE. In the above instance, *gandha* and *ādheyatā* are non-technical terms and *niṣṭha* is a technical term. Hence, as discussed above, when *niṣṭha* is *uttarapada*, then type of that compound is inevitably T7 - *Saptamī-Tatpuruṣha*. If it is the head of the *pūrvapada*, then the type of compound will be undoubtedly K1-*Viśeṣaṇa-pūrvapada-Karmadhāraya*. Similarly when *avacchinna* is a *uttarapada*, compound type will be always T3 - *Tṛtīyā-Tatpuruṣha* and if it is head of *pūrvapada*, then it will be beyond any doubt K1 - *Viśeṣaṇa-pūrvapada-Karmadhāraya*.

The constituency parse of the expression *((gandha-niṣṭha)-ādheyatā)* following this grammar is shown in Figure 1.

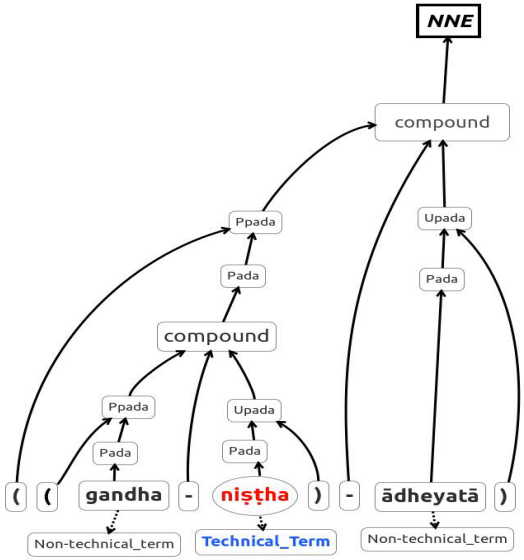

Figure 1: Constituency parse corresponding to the grammar

We will see how the head gets computed in Fig 2 and Fig 3 with the same example.

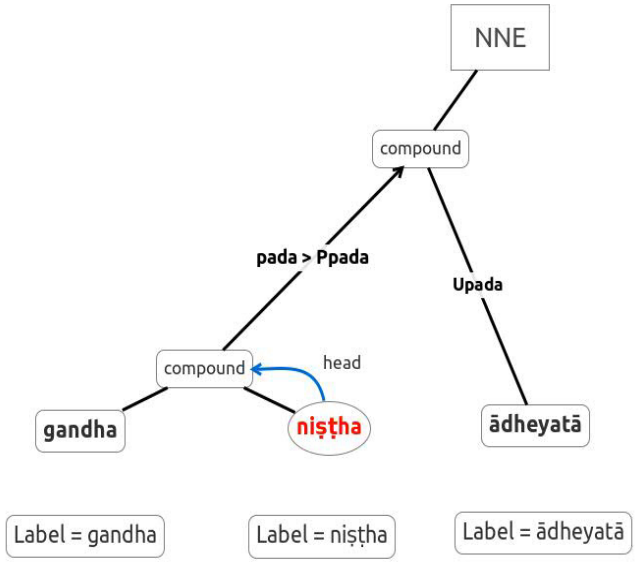

Figure 2: Head-info computed according to the grammar - step 1

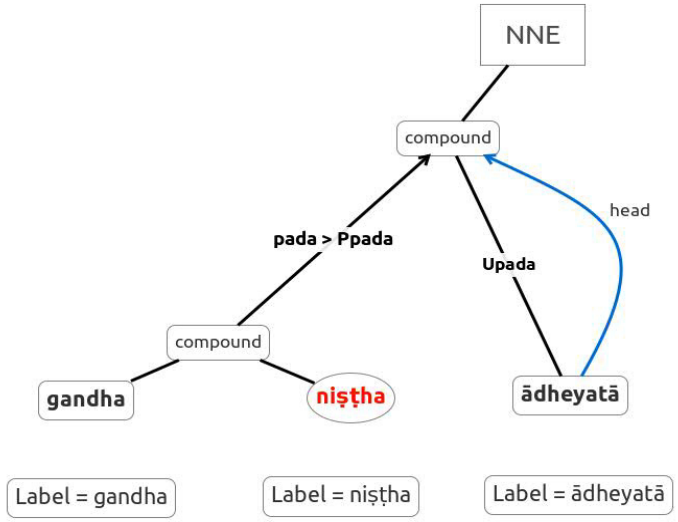

Figure 3: Head-info computed according to the grammar - step 2

5 Analysis of the result

We tested this type identifier on 352 NNEs collected from Māthurīpañcalakṣhaṇīsarvasvam. Each NNE has more than two components. Minimum number of components were 3 and the longest NNE has 40 components. Thus the minimum number of binary compounds in these NNEs were 2 and maximum number of binary compounds were 39. The 352 NNEs together had 2329 binary compounds. Of these 1340 compounds could not be identified, since none of the components of these compounds was from the NN technical vocabulary. Remaining 989 compounds are recognised correctly. Among the 59 fine-grain compound types, only 8 types of compound are used with NN technical vocabulary. These types are - K1 (*Viśeṣaṇa-pūrvapada-Karmadhāraya*), T3 (*Tṛtīyā-Tatpuruṣha*), T6 (*Śaṣṭhī-Tatpuruṣha*), Bs6 (*Śaṣṭhyartha-Bahuvrīhi*), T7 (*Saptamī-Tatpuruṣa*), K6 (*Sambhāvanā-pūrvapada-Karmadhāraya*), T5 (*Pañcamī-Tatpuruṣha*), Tds (*Samāhāra-Dvigu*).

Table 2 gives the classification of these compounds along with their frequency of occurrence. Our grammar could recognise all of them correctly.

K1	283
Bs6	203
T6	186
T3	178
T7	74
K6	45
T5	13
Tds	7
Total	989

Table 2: Frequency distribution of identified compound types

6 Conclusion

This is an effort to ease the process of understanding NNEs with the assistance of computational device. Once the compound types are identified, it is possible to generate the paraphrase also automatically. In order to identify the compound types of the compounds which do not involve NN technical terms, we need the lexical semantics. Anil Kumar[15] and Pavankumar[22] have listed various semantic features that are needed for the analysis as well as generation of the compounds. Further the advances in machine learning will also help in identifying suitable parameters for correct identification of the compound types. We are currently working on how to use the lexical semantics effectively for identifying the compound type automatically in case of compounds consisting of non-technical terms.

References

[1] S. R. Arjuna and Gérard Huet. Semi-automatic analysis of Navya-Nyāya compounds, SALA-30, Hyderabad, 2014.

[2] Arjuna.S.R and Amba Kulkarni. Segmentation of Navya-Nyāya Expressions. In *ICON-2014*, 2014.

[3] Arjuna.S.R and Amba Kulkarni. Analysis and Graphical Representation of Navya-Nyāya Expressions. In *Sanskrit and Computational Linguistics, 16th World Sanskrit Conference, Bangkok, Thailand*, 2016.

[4] Sibajiban Bhattacharya. Some features of the Technical Language of Navya-Nyaya. *Philosophy East and West*, 40(2):129–149, 1990.

[5] N. S. Dravid. *Nyāyakusumāñjaliḥ of Udayana*. ICPR, New Delhi, 1996.

[6] Jonardon Ganeri. Towards a formal regimentation of the Navya-Nyāya technical language-I. *Logic, Navya-Nyāya & Applications*, pages 109–124, 2008.

[7] Brendan S. Gillon. Tagging classical Sanskrit compounds. In Amba Kulkarni and Gérard Huet, editors, *Sanskrit Computational Linguistics 3*, pages 98–105. Springer-Verlag LNAI 5406, 2009.

[8] Gérard Huet. A Functional Toolkit for Morphological and Phonological Processing, Application to a Sanskrit Tagger. *J. Functional Programming*, 15,4:573–614, 2005.

[9] Daniel H H Ingalls. *Materials for the Study of Navya-Nyāya Logic*. Harvard University Press, 1951.

[10] V. N. Jha. *Viṣayatāvāda of Harirāma Tarkālaṅkāra*. University of Pune, Pune, 1987.

[11] Amba Kulkarni. Navya-Nyāya for Scientists and Technologists: A first step. Master's thesis, Indian Institute of Technology, Kanpur, 1994.

[12] Amba Kulkarni and Anil Kumar. Statistical constituency parser for Sanskrit compounds. In *Proceedings of ICON 2011*. Macmillan Advanced Research Series, Macmillan Publishers India Ltd., 2011.

[13] Amba Kulkarni, Soma Paul, Malhar Kulkarni, Anil Kumar, and Nitesh Surtani. Semantic Processing of Compounds in Indian Languages. In *COLING-2012 Proceedings*, pages 1489–1502, 2012.

[14] Tirumala Kulkarni and Jaideep Joshi. *The language of Logic - Navyanyāya Perspectives*. Manipal university Press, Manipal, India., 2013.

[15] Anil Kumar. *An Automatic Compound Processing*. PhD thesis, Department of Sanskrit Studies, University of Hyderabad, 2012.

[16] Anil Kumar and Amba Kulkarni. Clues from Aṣ. tādhyāyī for compound type. In Malhar Kulkarni and Chaitali Dangarikar, editors, *Recent Researches in Sanskrit Computational Linguistics Fifth International Symposium Proceedings*, pages 62–83. D. K Printworld (P) Ltd, New Delhi, 2013.

[17] Anil Kumar, Vipul Mittal, and Amba Kulkarni. Sanskrit compound processor. In *Proceedings of the 4th International Sanskrit Computational Linguistics Symposium*, 2010.

[18] Bimal Krishna Matilal. *The Navya-Nyaya Doctrine of Negation*. Harvard University Press, Cambridge, Massachusetts, 1968.

[19] Bimal Krishna Matilal. *Nyāyasiddhāntadīpa of Śaśadhara*. Lalbhai Dalpatbhai Institute of Indology, 1976.

[20] Bimal Krishna Matilal. *Nyāya-Vaiśeṣika*. Harrassowitz, 1977.

[21] Bimal Krishna Matilal. *Epistemology, Logic and Grammar in Indian Philosophical Analysis Ed. Jonardon Ganeri*. Oxford University Press, 2005.

[22] Pavankumar Satuluri. *Sanskrit Compound Generation: With Focus on the Order of Operations*. PhD thesis, Department of Sanskrit Studies, University of Hyderabad, 2016.

[23] J L Shaw. The Nyāya on cognition and negation. *Journal of Indian philosophy*, 8(4):279–302, 1980.

[24] Badarinatha Shukla. *Māthurī Pañcalakṣaṇī*. Rajasthan Hindi Granth Academy, Jaipur, 1984.

[25] Toshihiro Wada. *The Analytical Method of Navya-Nyāya*. Egbert Forsten, Groningen, 2007.

Endnotes

1. Amba Kulkarni
2. These examples are taken from Arjuna and Kulkarni[3].
3. samastapada-garbhita-samāsa

Professor
Department of Sanskrit Studies,
Hyderabad Central University
Hyderabad.
Email : ambapradeep@gmail.com
M. 8106783000

Research Fellow
Department of Sanskrit Studies,
Hyderabad Central University
Hyderabad
Email : arjunsr1987@gmail.com
Mob : 8106783000