Chapter 1

Overview

1.1 Introduction

Sanskrit has more than 2500 years old almost exhaustive grammar in the form of Pāṇini's Aṣtādhyāyī which has the features of computability. However, only recently Sanskrit Computional Linguistics[1] has gained a momentum.

The गणकाष्टाध्यायी[2] software based on Panini's अष्टाध्यायी, provides various search options to get the information with पद-पाठ, अनुवृत्ति, English traslation etc. of rules of अष्टाध्यायी. It also shows the process of generating various nominal declansions and verbal conjugations following Pāṇini's rules.

The French scholar Gérard Huet has done a significant work on the segmentation of Sanskrit texts. His work is purely ruled based. His website

[1] International Symposium on Sanskrit Computational Linguistics held in 2007, 2008, 2009 and 2010

[2] http ://www.taralabalu.org/panini/

"Sanskrit Heritage site"[3] provides an automatic Lemmatizer, Tagger, Sanskrit Reader and Sanskrit Parser.

The German scholar Oliver Hellwig has developed a website DCS, the Digital Corpus of Sanskrit[4]. It provides a searching facility for collection of lemmatized Sanskrit texts. It also provides an automatic segmentation and tagging of Sanskrit texts. He follows the statistical methods.

A Special Center for Sanskrit Studies (SCSS) at Jawaharlal Nehru University has developed various tools on Languages processing, Lexical resources, E-learning etc. for Sanskrit under the supervision of Dr. Girish Nath Jha. All the tools are available online at http://sanskrit.jnu.ac.in .

A consortium of Seven institutions[5] has been engaged in developing tools for analysis of Sanskrit. Under this project guidelines and standards for annotation at various levels such as sandhi, samāsa, kāraka, POS etc were developed and a substantial amount of manually tagged data following these standards is generated. The consortium has developed various tools such as sandhi, sandhi-splitter, morphological analyser and generator, POS tagger, sentential parser under this project.

All these tools handle only morphological analysis and segmentation to a large extent. Some of these tools for example Huet's and Hellwig's processors also do sentential parsing. However, there have been almost negligible effort in handling Sanskrit compounds automatically beyond

[3]http://sanskrit.inria.fr/

[4]http://kjc-fs-cluster.kjc.uni-heidelberg.de/dcs/index.php

[5]University of Hyderabad- Hyderabad , Jawaharlal Nehru University- New-Delhi , IIIT-Hyderabad, Sanskrit Academy-Hyderabad, Poornaprajna Vidyaapetha- Bangalore, Rashtriya Sanskrit Vidyapeetha-Tirupati, JRRSU-Jaipur.

segmentation.

A worth noting contribution in the field of Sanskrit compounds is by the Department of Indology of French Institute Pondichery. It has developed a CD version of पाणिनीयोदाहरणकोशः (Volum II - समासप्रकरणम्). It contains a searchable database of compound generation(रुपसिद्धि) of approximately 4,400 compounds from महाभाष्यम्, काशिकावृत्ति, भाषावृत्ति and सिद्धान्तकौमुदी.

On the theoretical side, there are again notable efforts by Gillon and K. V. Ramakrishnamacharyulu towards developing a tagging scheme for compounds. Gillon (Gillon, 2009) suggests tagging of compounds by enriching the context free rules. He does so by specifying the vibhakti, marking the head and also specifying the enriched category of the components. He also points out how certain components such as `na' provide a clue for deciding the type of a compound.

1.2 Goal of research

Sanskrit is very rich in compound formation unlike modern Indian Languages. The compound formation being productive it forms an open-set and as such it is also not possible to list all the compounds in a dictionary. The compound formation involves a mandatory sandhi[6]. But mere sandhi splitting does not help a reader in identifying the meaning of a compound, since typically a compound does not code the relation between its components explicitly. To understand the meaning of a compound, it

[6]Sandhi means euphony transformation of words when they are consecutively pronounced. Typically when a word w_1 is followed by a word w_2, some terminal segment of w_1 merges with some initial segment of w_2 to be replaced by a ``smoothed" phonetic interpolation, corresponding to minimising the energy necessary to reconfigure the vocal organs at the juncture between the words.(Huet, 2006)

is necessary to identify its components and discover the relation between them.

The goal of my research is to evolve a methodology for automatic analysis of Sanskrit compounds. The automatic analysis of a Sanskrit compound involves four major tasks viz. segmentation or component identification, deciding the grouping of components or constituency parsing, identifying the type of a compound and finally paraphrase generation. For building a compound processor, we explore the possibility of both the rule based approach as well as use of statistical methods wherever possible.

1.3 The organisation of thesis

Chapter one (the current chapter) gives a survey of the work in the field of computational linguistics followed by a survey of various works in the field of Sanskrit compounds.

Second chapter gives a brief summary of the features of compounds, the semantics involved, its classification both semantic as well as syntactic. This sets a ground for building the compound processor. We also describe the tagset developed by the Sanskrit Consortium[7] for manual tagging of Sanskrit compounds.

The third chapter describes an architecture of a compound processor. Processing a compound involves four major tasks viz. segmentation (समासपदच्छेदः), constituency parsing (सामर्थ्यनिर्धारणम्), type-identification (समा-

[7]Sanskrit Consortium is a consortium of 7 institutes funded by TDIL programme of DIT for the development of tools for analysis of Sanskrit text and Sanskrit-Hindi Machine Translation.

सभेदनिर्धारणम्) and paraphrase generation (विग्रहवाक्यनिष्पादनम्)(See figure 1.1). These four tasks form the natural modules of a compound processor. The output of one task serves as an input for the next task until the final paraphrase is generated.

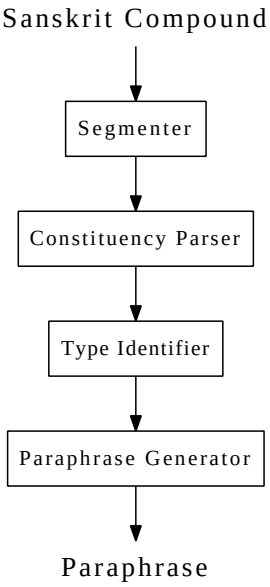

Figure 1.1 : Compound Analyser

The task of a segmenter is to split a compound into its constituents. For instance, the compound

sumitrānandavardhanaḥ

is segmented as

sumitrā-ānanda-vardhanaḥ

Each of the constituent component except the last one is typically a compounding form (a bound morpheme)[8].

[8]with an exception of components of an `aluk' compound.

The constituency parser parses the segmented compound syntactically by pairing up the constituents in a certain order two at a time. For instance,

sumitrā-ānanda-vardhanaḥ

is parsed as

<<sumitrā-ānanda>-vardhanaḥ>

The type-identifier determines the type on the basis of the components involved. For instance,

<<sumitrā-ānanda>-vardhanaḥ>

is tagged as

<<sumitrā-ānanda>T6-vardhanaḥ>T6 .

where T6 stands for compound of type ṣaṣṭī-tatpuruṣa. This module needs an access to the semantic content of its constituents, and possibly even to the wider context.

Finally after the tag has been assigned, the paraphrase generator generates a paraphrase for the compound. For the above example, the paraphrase is generated as :

sumitrāyāḥ ānandaḥ = sumitrānandaḥ,

sumitrānandasya vardhanaḥ = sumitrānandavardhanaḥ.

The fourth chapter describes the first task viz. segmentation. In this chapter, we discuss the computational complexity involved in splitting of a compound into components and then describe how the optimality theory, which is based on Generation-Constraint-Evaluation paradigm, may be used for splitting of compounds. We generate all possible splits following the sandhi rules from Aṣṭādhyāyī and constraint the splits using

morphological analyser. The output is evaluated using simple statistical methods. The algorithm with tested data is explained at the end.

The fifth chapter explains a constituency parser which takes an output of the segmenter and produces a binary tree showing the syntactic composition of a compound corresponding to each of the possible segmentation. In this chapter we describe an algorithm for parsing the components using manually tagged corpus of compounds. We describe how simple conditional probablity may be used for parsing the components. Results of five-fold-testing are given at the end of the chapter.

In the sixth chapter, we deal with an automatic type-identification of compounds. We describe how some of Pāṇini's compound related aphorisms may be used for type-identification. When the rules fail, we use probablity methods to identify the type of compound. We identify the semantic clues for identification of compound type following the aphorisms. We also point out the aphorisms where the clues are difficult to compute or use computationally in the present context.

The seventh chapter is on paraphrase generation, the final task of a compound processor. In this chapter, we give around 55 rules for automatic paraphrase generation corresponding to various sub-types and also describe the algorithm.

The eighth chapter contains the conclusion and provides future directions for improving the automatic sanskrit compound processor further. The use of this analysis for other Indian Languages is also discussed in brief.