Chapter-1

Computational Morphology and Sanskrit

1.0 Introduction

Morphology is a branch of linguistics that deals with word formation, analysis and generation. It is concerned with how words are formed or created in a language from smaller units in a systematic way. It has to find as well as to describe the mechanism behind this process. Each language has morphemes which are its smallest meaningful units. Realization of morphemes as part of a word is called morph. In formation of words from these smaller units, certain processes such as inflection, derivation and compounding etc. are involved. Also involved is morphotactics that determines how morphs should be put together to form words.

Computational Morphology (CM) is application of morphological rules in the field of computational linguistics. CL is an emerging area in Artificial Intelligence that is related to accomplish linguistic tasks through computational methods. CM deals with the processing of words in both their graphemic and phonemic forms. Its most basic task can be defined as taking a string of characters or phonemes as input and delivering an analysis as output. It has various practical applications from low-level applications to speech and language processing systems.[1]

Low-level applications

Low-level applications of CM are simpler tasks such as hyphenization, spelling correction, stemming etc.

[1] Harald Trost, Entry under "Morphology" in Oxford Handbook of Computational Linguistics

Hyphenation is almost exclusively done automatically. Segmenting words correctly into their morphs helps to solve the task. The major problem is spurious segmentations.

Spelling correction, in its simplest form, can be a mere comparison of the input against a list of word forms. But such a list will never contain all the words occurring in a text and enlarging the list will include more and more obscure words that will match with typos thus preventing their detection. Therefore, most systems use a root lexicon, plus a relatively small set of affixes and simple rules to cover morphotactics.

Stemmers are used in information retrieval to reduce as many related words and word forms as possible to a common canonical form which can then be used in the retrieval process. However, this canonical form is not necessarily the base form. The main requirement is – like in all the above tasks – robustness.

Another application is **text-segmentation** in Chinese, Japanese or Korean. In these languages words in a sentence are not separated by blanks or punctuation marks. Morphological analysis can be used to perform the task of word separation.

Natural language applications

Natural language processing (NLP) is a subfield of artificial intelligence and linguistics. It studies the problems of automated generation and understanding of natural human languages. Natural language generation systems convert information from computer databases into normal-sounding human language, and natural language understanding systems convert samples of human language into more formal representations that are easier for computer programs to manipulate.[2] CM has a wider scope of being applied in natural language processing systems involving parsing and/or generating natural language utterances in written or spoken form. Such applications can range from message

[2] *http://en.wikipedia.org/wiki/Natural_language_processing*

and information extraction to dialogue systems and machine translation. For many current applications, only inflectional morphology is considered.

In a parser, morphological analysis of words is an important prerequisite for syntactic analysis. Properties of a word the parser needs to know are its part-of-speech category and the morphosyntactic information encoded in the particular word form. Another important task is lemmatization, i.e., finding the corresponding dictionary form for a given input word, because for many applications a lemma lexicon is used to provide more detailed syntactic (e.g, valency) and semantic information for a deep analysis.

In generation, on the other hand, the task is to produce the correct word form from the base form plus the relevant set of morphosyntactic features.

Speech applications

A text-to-speech system takes (electronically stored) text as input and produces speech from it. Morphological analysis helps to solve two different tasks in such systems. One is to guide the grapheme-to-phoneme conversion. Characters are often ambiguous with respect to their translation into phonemes. Finding out the underlying morphological structure is necessary for solving the task correctly.

A less obvious application is the use of morphological analysis to help in determining the part-of-speech category of words. This is an important prerequisite of syntactic analysis which is the basis for coming up with a correct prosody.

Speech recognition is a field where morphological analysis can become ever more important. At the moment most available systems make use of full form lexicons and perform their analysis on a word basis. Increasing demands on the lexicon size on the one hand and the need to limit the necessary training time on the other hand will make morph-based recognition systems more attractive.

1.1 Morphological Analysis and Morphological Analyzers

Morphological analysis is the process in which a word is analyzed into its root word and associated morphemes. Morphological analyzers are programs used to provide morphological analysis of a language. They include recognition engine, lexicon/thesaurus, and algorithms to find out stem within an input word and identifying its suffixes.[3] They take as input a sentence, i.e. a sequence of words and retrieve related grammatical information such as lexical category, gender, number, person, tense etc. of every word. For example, for nouns they provide its gender, number and case information along with the root word. For verbs, information related to tense, aspect, mood etc. is retrieved. If a word has more than one meaning, the analyzer is supposed to return grammatical information for every meaning separately. [4]

It is the basic tool used in spell checker, grammar checker, parser and machine translation systems. To build a syntactic representation of the input sentence, a parser must map each word in the text to some canonical representation and recognize its morphological properties.

For the purpose of analysis of morphologically rich languages, the root and the morphemes of each word has to be identified. The combination of a surface form and its analysis as a canonical form and inflection is called a lemma. The main problems involved in the process are:

1. **Morphological alternations**: In a language the same morpheme may be realized in different ways depending on the context.

2. **Morphotactics:** Stems, affixes, and parts of compounds do not combine freely, a morphological analyzer needs to know what arrangements are valid.

---

[3] *http://www.acroterion.ca/Morphological_Analysis.html*

[4] Akshar Bharati, Vineet Chaitanya and Rajeev Sangal. "Natural Language Processing: A Paninian Perspective", Prentice-Hall of India, New Delhi, 1995.

There are many approaches for morphological analysis. A popular approach to morphological alternation is the cut-and-paste method in which the canonical form is derived by removing and adding letters to the end of a string. Another one is the finite state technology, which was introduced in early 1980s and is applied for automatic recognition and generation of word forms. It maintains that rules for morphological alternations can be implemented by finite-state transducers. It was also widely recognized that possible combinations of stems and affixes can be encoded as a finite-state network. An automaton containing inflected word forms can be upgraded to a morphological analyzer, for example, by adding a code to the end of the inflected form that triggers some predefined cut-and-paste operation to produce the lemma. Instead of cutting and pasting it at runtime, the entire lemma can be computed in advance and stored as a finite-state transducer whose arcs are labeled by a pair of forms. The number of nodes in this type of network is small, but the number of arc-label pairs is very large as there is one symbol for each morpheme-allomorph pair. A more optimal lexical transducer can be developed by constructing a finite-state network of lexical forms, augmented with inflectional tags, and composing it with a set of rule transducers. Lexical transducers can be constructed from descriptions containing any number of levels. This facilitates the description of phenomena that are difficult to describe within the constraints of the two-level model. Because lexical transducers are bidirectional, they are generally non-deterministic in both directions. If a system is only to be used for analysis, a simple finite-state network derived just for that purpose may be faster to operate. [5]

A morphological generator performs the reverse function of an analyzer. For every root and its features provided as input, it generates the word-forms in the given paradigms. Morphological analyzer and morphological generator are two essential and basic tools for building any language processing application for a natural language.

[5] *http://www.languageinindia.com/aug2006/parsingrajendran.pdf*

1.1.1 India's linguistic scenario

The Indian subcontinent is remarkably rich in languages. Linguistic diversity is a unique component of the cultural and political structure of this area. The 200 languages enumerated in the Census are a linguistic subtraction of over 1,600 mother tongues reported by the people indicating their perception of their linguistic identity and linguistic difference. [6] The linguistic diversity in India is marked by the fluidity of linguistic boundaries between dialect and language, between languages around state borders and between speech forms differentiated on cultural and political grounds. In spite of diversity, linguistic identity is thin because of the large size of population of the country. Some of the minor languages have more speakers than many European languages. The linguistic differences between the Indian languages, particularly at the grammatical and semantic levels, are less than expected, given their different historical origins. The mutual convergence of these languages due to their intensive and extensive contact have made India one linguistic area. Inter-translatability between those languages is therefore very high.

The languages of India can be historically categorized into four major language families:

Indo–European : This is the most prominent language family. Most of the major Indic languages leaving Dravidian languages belong to it. The Indo – European has the sub–families, Indo–Aryan and Dardic/Kashmiri. Major languages are Hindi, Sanskrit, Punjabi, Sindhi, Gujarati, Marathi, Bengali, and many other languages.

Dravidian: There are four prominent Dravidian languages – Telugu, Kannada, Tamil and Malayalam. All of them are spoken in the southern most part of the country.

Austro-Asiatic: It has Munda and Mon-Khmer/Khasi as sub-families.

Sino – Tibetan: Tibeto-Burman and Thai/Kempti are sub-families of Sino-Tibetan.

The Indo-European which is commonly called 'Indo-Aryan' has the largest number of speakers followed by Dravidian, Austro-Asiatic, also called 'Munda' and Sino-Tibetan which is commonly called the 'Tibeto-Burman'.

[6] *http://www.languageinindia.com/july2005/morphologynortheast1.html*

Andamanese: Andamanese is the fifth language family of India.[7] It comprises of the four language groups represented by the four speech communities of the Great Andamanese, the Jarawa, the Onge and the Sentinelese. Great Andamanese arguably is language family in itself.[8] All these languages are primitive languages and the speaking community of these languages are the primitive tribes who have been living in these islands for thousands of years, untouched from civilized society. They are people of the Negrito stock and it is not clearly known where they came here from.

1.1.2 Development of Language Technology in India

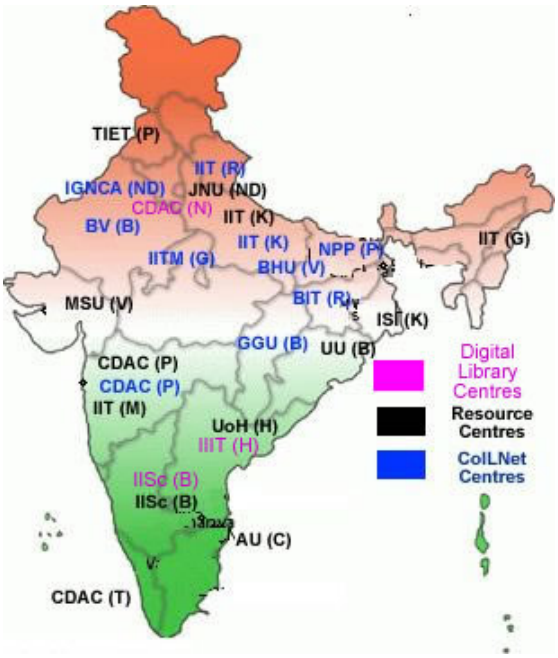
Technology Development in Indian Languages (TDIL)[9] was initiated by the Department of Information Technology, Ministry of ICT, Government of India with the objective of developing Information Processing Tools and Techniques to facilitate human-machine interaction without language barrier; creating and accessing multilingual knowledge resources; and integrating them to develop innovative user products and services. Since then, TDIL has been facilitating and supporting the development of language technology resources in all Indian languages, and promoting their dissemination. Towards this goal it has established 13 Resource Centers for Indian Language Technologies (RCILTS) in March 2000 at different Universities and Institutes.[10] These Resource centers are aimed at:

- Improving the quality of life of people of India by enabling to use Information Technology through Indian Languages..
- Developing new products and services for processing information in Indian Languages.
- Facilitating research in technology areas such as Machine Translation (MT), Optical Character Recognition (OCR), Text-to-Speech (TTS), and Speech Recognition in Indian Languages that leads to product development.

[7] Abbi, Anvita. 2006. Endangered Languages of Andaman Islands. Lincom Europa.
[8] Ibid.
[9] http://tdil.mit.gov.in/
[10] http://tdil.mit.gov.in/languagetechnologyresourcesapril03.pdf

List of Resource Centers

Resource Centre	Languages Associated With
Indian Institute of Technology, Kanpur	Hindi, Nepali
Indian Institute of Technology, Mumbai	Marathi, Konkani
Indian Institute of Technology, Guwahati	Assamese, Manipuri
Indian Institute of Science, Bangalore	Kannada, Sanskrit (Cognitive Models)
Indian Statistical Institute, Kolkata	Bengali
Jawaharlal Nehru University, New Delhi	Foreign Languages (Japanese, Chinese) & Sanskrit (Language Learning Systems)
University of Hyderabad, Hyderabad	Telugu
Anna University, Chennai	Tamil
MS University, Baroda	Gujarati
Utkal University,	Oriya
Department of Computer Science and Application	Punjabi
Thapar Institute of Engg. & Tech., Patiala	
ERDCI, Trivendrum	Malayalam
CDAC, Pune	Urdu, Sindhi, Kashmiri



A map showing main resource centers of TDIL across the country[11]

[11] Source: TDIL website, *http://tdil.mit.gov.in/*

A few other funding agencies are also supporting the language technology efforts. Several individuals in India and abroad and some industries have also been developing different language products/resources. An example of a morphological verb analyzer can be cited here that has been developed at JNU by a research scholar[12] for an extremely endangered language of Great Andamanese. As there are so many distributed efforts and with so many languages it is very difficult to keep track of the activities in this area. So, TDIL has tried to evolve an indexing system for the language technology resources, so that a suitable portal can be created to offer a variety of services to persons concerned with Indian language technologies. If all the resource centres follow this indexing and share their documents with TDIL, it would be easy to keep track of the progress and would be possible to encourage more people to participate in the development activities.

1.1.3 Survey of Morphological Tools for Indian languages

Morphological analyzer is an integral part of any Natural Language Processing system, especially in the context of Indian languages where morphology plays significant role due to high inflectional and derivational nature of these languages. For fixed word-order languages, the semantics of a word are primarily governed by its absolute and relative position within a sentence. But for free word-order languages, the position of words in the sentence cannot provide much clue about its semantics. As in the case of Indian languages, which are mostly free order, the semantics (part of speech and other subtleties) are heavily dependent on the surface realization of the word. Therefore, morphological analysis is inevitable to develop any NLP tool for Indian languages. Unlike English and various foreign languages, most of the Indian languages can be characterized by a rich system of inflections (*vibhakti*), derivation and compound formation. The numbers of word are derived from the root word by some specific orthographic rule in the Indian languages. A competent morphological analyzer is needed for a machine to deal with the lexicons of these languages.

[12] Choudhary, Narayan Kumar, 2006, "Developing a Computational Framework for the Verb Morphology of Great Andamanese", Unpublished dissertation at Centre for Linguistics, JNU.

The task of the morphological analyzer is to identify the structural components of a word, and glean information about it. While Indian languages in general are morphologically richer than languages like English, Dravidian languages are a lot more complex. Clearly, there is no way to list all forms of all words in a dictionary. Therefore, morphology is essential.

The development of Natural Language Processing in India has done significant progress since its beginning. A variety of tools have been developed for different purposes. These include machine translation systems, morphological analyzers and generators, POS taggers, spell checkers, text processors, word nets and many other tools. Given the morphological richness of these languages, numerous NLP tools related to morphology have been developed. The range covered is from major languages like Hindi, Telugu and Bengali to languages like Manipuri and Assamese. The survey of morphological analyzers, generators and other tools for morphological understanding of various Indian languages is given below. It mainly focuses on morphological analyzers and generators, taggers, spell-checkers and text-processors. However, efforts and tools of government institutions only have been covered leaving behind the initiatives of private sector which also is putting a lot of effort in this field.

1.1.3.1Morphological Analyzers for Indian languages:

Morphological Tagger for Hindi

Vasu Renganathan from University of Pennsylvania has developed a Hindi Morphological Tagger[13]. One has to enter a Hindi word or sentence and press 'Tag' to get tagged words as output. It is available in Unicode and ITRANS compatibility. The source code is written in PERL. The site also provides some of the data files

Morphological Analyzer for Telugu

[13] *http://ccat.sas.upenn.edu/plc/tamilweb/hindi.html*

Telugu is highly inflectional and agglutinative language. It exhibits features like vowel harmony, sandhi etc. Clitics, particles, vocatives are all part of the word. Auxiliary verbs are used in various combinations to indicate complex aspects.

RCILTS for Telugu, Univ. of Hyderabad claims to develop the only large scale morphological analyzer system built for Telugu.[14] It has been built, tested against corpora and refined over the past 10 years. The system uses a root word dictionary of 64,000 entries and a suffix list categorized into a number of paradigm classes. The basic methodology is to look for suffixes, remove them taking care of sandhi changes and then cross checking with the dictionary. Inflection, derivation, external sandhi are all handled. There is also a separate morphological generator that can put together the roots and affixes to construct complete word forms.

For POS Tagging to Telugu, a morphological analyzer developed by G. Uma Maheswara Rao from University of Hyderabad is being used. It gives the way to analyze each and every word in the Telugu corpus with the different possibilities.

Kannada Morphological Analyzer

RCILTS, Hyderabad has also developed a Kannada morphological analyzer and generator using their *Network and Process Model*.[15] A finite state network captures in a declarative and bidirectional fashion all the affixes, their ordering and the various permitted combinations. The process component takes care of sandhi changes when affixes are added or removed. This model makes it possible to develop a morphological analyzer, test it against a corpus and then get a generator of comparable performance with no extra effort since the same network is used both for analysis and generation. In this model, as it is claimed, a complete and detailed analysis is made at the level of each affix.

[14] TDIL magazine '*vishvabharata'* July 2003, accessible@ *tdil.mit.gov.in/Telugu-UOHJuly03.pdf*
[15] Ibid.

Morphological Analyzer and Synthesizer for Hindi [16]

Media Lab Asia Research Laboratory at IIT Kharagpur, [17] WB has developed a morphological analyzer for Hindi. It can identify the tense, aspect, modality, person, gender and number along with the root (*'dhaatu'*) of an inflected verb form. For this, the morphological analyzer uses a Directed Acyclic Morphological Structure. For nouns, the analyzer has to determine its *vibhakti* (inflection), suffixes and prefixes. It analyzes the lexical word groups corresponding to noun and determines the *karaka* (semantic role).

The group also aims to perform the decompositions for sandhi and samaasa (conjugating and compounding words) so as to have a powerful vocabulary for the system, and a generalized prefix and suffix handler. The DAM structure developed is based on the fact that possible inflections of a word can be identified by scanning the word backward from the end, one step at a time.

A morphological synthesizer has also been developed to generate the structural word given the root of the word, and the tense, aspect, mode, number, gender, and person. Hindi is a GNP language - i.e. the inflected forms of the verbs are mainly inflected using the gender, number and person information.

Morphological Analyzer and Synthesizer for Bangla [18]

Media Lab Asia Research Laboratory at IIT Kharagpur [19] has also designed a morphological analyzer and synthesizer for Bengali. The morphological analyzer can identify the tense, aspect, modality and person of an inflected verb form.

Bengali is a classifier language - where the inflection takes place mainly due to the tense, aspect and mode. For second and third persons, there is the concept of familiarity. The morphological analyzer has been modeled as finite state transducers which are given a

[16] *http://www.mla.iitkgp.ernet.in/technology.html*
[17] *http://www.mla.iitkgp.ernet.in/Resource/index.html*
[18] *http://www.mla.iitkgp.ernet.in/technology.html*
[19] *http://www.mla.iitkgp.ernet.in/Resource/index.html*

root and TAM and GNP information as input. The transducer's job is to generate the correct inflected form of the word by adding the appropriate suffix for Bengali.

Verbs are classified according to the vowels present in them. The modus operandi of the synthesizer is to determine the category of the root, given an input. It then accordingly synthesizes the ultimate form of the word depending on the input TAM GNP parameters. The number of exceptions is limited, and therefore easily traceable.

Oriya Morphological Analyzer (OMA)

PG Department of Computer Science and Application, Utkal University, Bhubaneswar has developed Oriya Morphological Analyzer (OMA), a computational model for the analysis and generation of Oriya language.[20] The major contents of OMA are:

i) Pronoun Morphology (PM),
ii) Inflectional Morphology (IM) and
iii) Derivational Morphology (DM).

The OMA has many constituent parts [21] such as OriNet Database (OD), which stores the Oriya lexicon (Only root words) whereas OMA Engine (OE) processes the system and Morphological Parsing (MP) parses the word according to orthographic rule. Decision Tree (DT) decides to classify the morphemes and different functional programmes by use of OMA.

The OMA system has been designed on the basis of Object-Oriented Approach (OOA) so that different functions can be easily added to or deleted from it. Pronoun Morphology and Inflection Morphology have been implemented in the OMA in such a manner that it successfully runs with the OriNet system, Oriya Spell Checker (OSC) and Oriya Grammar Checker (OGC). The OSC handles any type of word (derived, inflectional or root) using the OMA. It also provides sufficient interface for applications involved in Oriya Machine Translation (OMT), Word-Net for Oriya (OriNet), Oriya Spell Checker

[20] http://www.ilts-utkal.org/oriyamorphological.htm

(OSC) and Oriya Grammar Checker (OGC). All these developments have been worked out on the basis of the syntactic approach of Sanskrit language.

Developers have developed and implemented the Decision Tree (DT) and its respective algorithm of each type of morphology, through which OMA runs successfully. While performing morphological analysis, OMA not only deals with the study of words but also its morphemes. The OMA system is useful for various applications in developing NLP tools. For example, in OMT, there is need of root words, which is obtained through the OMA. Similarly, it has typical use in OriNet, the Word-Net for Oriya language, for searching any type of lexicon. Its Morphological Parser (MP) successfully handles ambiguous morphemes and provides different alternatives for them.

Morph Analyzer for Assamese and Manipuri[22]

RCILTS at Department of Computer Science & Engineering, Indian Institute of Technology Guwahati Assam[23] has developed morphological analyzers for Assamese and Manipuri**.** They are knowledge tools and work as an aid to MT system and Spell Checker.

Assamese Morphological Analyzer is based on the Stemming technique in which affixes are added/ deleted according to the linguistic rules. The derived words are verified with the existing Corpus/Dictionary to treat as valid words. This analyzer currently works with twenty linguistic rules, though more rules can be added without alteration in code. Modules are available in the form of API's for customization.

Manipuri Morphological Analyzer also uses stemming technique and does morphological processing with the help of the grammatical rules and the dictionaries of roots and affixes. A root dictionary containing 3000 root words and an affix dictionary containing 55 affixes are being used. Identification of linguistic rules for nouns and pronouns has also commenced. A model tagger is added to tag and analyze word. The

[21] http://www.ilts-utkal.org/publication/NLP/Abstract/abs-12.pdf
[22] *tdil.mit.gov.in/TDIL-OCT-2003/morph%20analyzer.pdf*

tagger tags the lexical category of the root and the grammatical category of the affixes. The code is written using Perl script and the user interface has been developed using Perl/Tk. The dictionaries are stored in MS-Access database.

Morphological Analysis in Tamil

Tamil is a verb final, relatively free-word order, morphologically rich and agglutinative language. Computationally, each root word in Tamil can take a few thousand inflected word-forms. Subject-verb argument is required for the grammaticality of a Tamil sentence. Tamil allows subject and object drop as well as verb less sentences. In addition, the subject of a sentence or a clause can be a possessive Noun Phrase (NP) or an NP in nominative or dative case. As Tamil is an agglutinative language, each root word can combine with multiple morphemes to generate word forms.

The following is the list of computational morphological analysis attempted and/or implemented for Tamil:[24]

1. Rajendran's Morphological Analyzer for Tamil: This was one of the very first efforts towards building a morphological analyzer for Tamil. It was initiated by *anusaraka* group of researchers under whose guidance Rajendran, Tamil University prepared this morphological analyzer for Tamil for Translating Tamil into Hindi at the word level.

2. AUKBC Morphological Parser for Tamil: [25] AUKBC NLP team under the supervision of Rajendran prepared a Morphological parser for Tamil. The API Processor of AUKBC makes use of the finite state machinery like PC Kimmo. It parses, but does notgenerate.

---

[23] *http://www.iitg.ernet.in/rcilts*
[24] "Parsing in Tamil: Present State of Art", S. Rajendran, Ph.D.
http://www.languageinindia.com/aug2006/parsingrajendran.pdf
[25] *http://www.au-kbc.org/frameresearch.html*

3. Vaishnavi's Morphological Generator and Analyzer for Tamil: Vaishanvi has built generators and analyzers for Tamil morphology. The generator implements the item and process model of linguistic description. It works by the synthesis method of PC Kimmo. The analyzer uses a hybrid model for Tamil. It is theoretically rooted in a blend of IA and IP models of morphology. It constitutes an in-built lexicon and involves a decomposition of words in terms of morphemes within the model to realize surface well-formed words-forms. Thus it tries to define a transformation depending on the morphemic nature of the word stem. The analysis involves a scanning of the string from the right to left periphery scanning each suffix at a time stripping it, and reconstructing the rest of the word with the aid of phonological and mopho-phonemic rules exemplified in each instance. This goes on till the string is exhausted. For the sake of comparison AMPLE and KIMMO models are implemented.

5. RCILTS-T's Morphological analyzer for Tamil:[26] Resource Centre for Indian Language Technological Solutions-Tamil, Anna University, Chennai has prepared a morphological analyzer for Tamil named as *Atcharam*. It takes a derived word as input and separates it into root word and associated morphemes. It uses a dictionary of 20,000 root words based on fifteen categories. It has two modules - noun and verb analyzer based on 125 rules. It uses heuristic rules to deal with ambiguities. It can handle verb and noun inflections. It is also used for extracting the root words from inflections in Tamil Search Engine and Online Tamil Dictionary. The system was developed using Java on Windows 2000 platform. It is expandable as more morphological rules can be added and Dictionary size can be increased. It has been tested with a file from CIIL Corpus producing approximately 75% result.

6. RCILTS-T's Morphological generator for Tamil:

Resource Centre for Indian Language Technological Solutions-Tamil has also prepared a morphological generator for Tamil. It is named as *Atchayam*. It generates words when Tamil morphs are given as input. It has two major modules – noun and verb generators. The noun section handles suffixes like plural markers, oblique form, case markers and

[26] *http://ns.annauniv.edu*

postpositions. The verb section takes tense and PNG makers, relative and verbal participle suffixes, and auxiliary verbs. It uses sandhi rules and125 morphological rules. It handles adjectives and adverbs. It has word and sentence generator interfaces.

Others

- **Ganesan's Morphological Analyzer for Tamil** to analyze CIIL corpus. It uses phonological and morphophonemic rules and takes into account morphotactic constraints of Tamil. An efficient morphological parser has also been built.
- **Kapilan's Morphological Analyzer** is especially built for analyzing Tamil Verbal Forms.
- **Deivasundaram's Morphological parser** built for a Tamil Word Processor. He too makes use of phonological and morphophonemic rules and morphotactics constraints of the language for developing his parser.
- **Ramasamy's Morphological Generator for Tamil:** Ramasamy has prepared a morphological generator for Tamil.
- **Winston Cruz's Parsing and Generation of Tamil Verbs** makes use of GS morph method for parsing Tamil verbs which does morphotactics by indexing. The algorithm simply looks up two files to see if the indices match or not. The processor generates as many forms as it parses and uses only two files.
- **Dhurai Pandi's Morphological Generator and Parsing Engine for Tamil Verb Forms:** It is a full-fledged morphological generator and a parsing engine on verb patterns in modern Tamil.

Punjabi Morphological Analyzer and Generator

The Advanced Centre for Technical Development of Punjabi Language, Literature and Culture, Punjabi University, Patiala has developed a Morphological Analyzer and Generator for Punjabi.[27] As generator, the software displays the list of all the possible word forms of any Punjabi root word, along with their respective grammatical information. As analyzer, it identifies the grammatical attributes of any Punjabi word and

[27] *http://www.advancedcentrepunjabi.org/punjabi_mor_ana.asp*

can also be used to search for any Punjabi word in it to know its root and other grammatical information.

The application areas of the software include, automatic spelling and grammar checking, natural language understanding, machine translation, speech recognition, speech synthesis etc part of speech tagging, parsing. The common man can also get in-depth information about the Punjabi words from the software. Knowing the grammatical information of a word helps in its proper and error free use in sentences. It can also help the beginners to learn new words and the specialists to create new terminology.

The database used in the software consists of more than 1.72 lakh Punjabi words, grouped into 22 word classes such as noun, personal pronoun, reflexive pronoun, verb, inflected and uninflected adverb, inflected and uninflected adjective, conjunction, interjection etc.

Morphological Analysers for multiple languages

Akshara Bharathi Group [28] at Indian Institute of Technology, Kanpur, India and University of Hyderabad, Hyderabad, India has developed morphological analyzers for different languages. These are available for online download for Hindi, Marathi, Telugu, Kannada and Punjabi. Downloaded system needs Linux Operating System, Perl, Perl enabled vim(only for Telugu), GDBM, Flex. The site claims that Telugu has 95% coverage (for arbitrary text in modern standard Telugu) and Hindi has 88% coverage.

IIT Bombay

RCILTS (CFILT - Marathi), IIT-Bombay, led by Dr. Pushpak Bhattacharya is a happening place for NLP in India. With an objective to enable Indian language information processing on internet, this project funded by MIT, Govt. of India has been working on creating Lexical Knowledge Bases like Wordnet , tools like MorphA and POS taggers for Hindi and Marathi and MTS with Universal Networking Language

[28] *http://ltrc.iiit.net/showfile.php?filename=onlineServices/morph/index.htm*

(UNL) as interlingua. Wordnet for Hindi and Marathi with lot of literature on it can be downloaded from the CFILT portal[29]. It also offers an overview of the UNL based Interlingua approach to MT with its online versions of English to Hindi MTS, Hindi to UNL conversion system, UNL to Hindi generation system. This centre is developing a POS tagger for Hindi using a data driven approach, making use of graphical algorithm Conditional Random Fields. This algorithm based tagger makes use of the spelling of the words, their lexical attributes and the suffixes to achieve an accuracy of about 90% on the BBC Hindi news corpora. It has a tagset of 30 tags with two pseudo tags S-START and S-END marking the sentence boundaries.[30]

1.1.3.2 Spell-Checkers for Indian languages

Tel-Spell: Spell Checker for Telugu

RCILTS-Telugu, Univ. of Hyderabad has developed a spell checker for Telugu named Tel-Spell.[31] The system has been tested on large scale corpora and enhancements and refinements have been going on for years. The new version is far simpler, more transparent, portable, well documented, and conforms to standards. Research work has also been taken up on developing stemming algorithms for Telugu. A pure corpus based statistical stemming algorithm has been developed. The performance of this stemmer for the spell checking application has been studied in various combinations with dictionary and morphology based approaches. A 10 Million word corpus developed has been used to build and test the performance. The performance of the system has been found to be satisfactory both in terms of detection and correction of spelling errors. The Telugu spell checker has also been integrated into AKSHARA advanced multilingual text processing system.

Bangla Spell-checker

[29] Resource Centre for Indian Language Technology Solutions (CFILT) - http://www.cfilt.iitb.ac.in/ (accessed: 14.10.2006)
[30] Srivastava, Manish et al., 2006, 'Conditional Random Field Based POS Tagger for Hindi', in the Proceedings of MSPIL, IIT-B, Mumbai
[31] *tdil.mit.gov.in/Telugu-UOHJuly03.pdf*

RCILTS-Bengali Indian Statistical Institute, Kolkata has developed a **Bangla Spell-Checker**.[32] In this spell-checker, only non-word errors are considered and not word errors like substitution, deletion, insertion, and transposition error. In Bangla, errors may occur due to phonetic similarity of characters or typographical mistakes. In this spell-checker, the main technique of error detection is based on matching the candidate string in the normal as well as in the *reversed dictionary.* Moreover, this approach is combined with a phonetic similarity key based approach where phonetically similar characters are mapped into a single symbol and a nearly-phonetic dictionary of words is formed. Using this dictionary, phonetic errors can be easily detected/ corrected. Here a candidate string first passes through the phonetic dictionary. If the word is not found in the dictionary and also failed to give suggestion then it tries to divide the word in root part and suffix part by separately verifying both. If an error is found, the spell-checker tries to provide suggestions. If it fails, it checks whether the string is a conjunct word generated by appending two noun words and suffix. Option for adding new words permanently or temporarily is provided in the spell checker.

The spell-checker uses several files containing root words and suffix words. The main dictionary contains about 60,000 root-words and 100,000 inflected words. Noun and verb suffix files are also used. The spell-checker works fast and almost correctly detects the non-word errors. However, it makes about 5% false alarm due to conjunct words formed by euphony and assimilation as well as proper nouns in the corpus.

Gujarati Spell Checker

RCILTS-Gujarati, Maharaja Sayajirao University, Baroda is developing a Gujarati Spell Checker.[33] It is based on a morphological analysis of Gujarati language. Morphological analyzer covers the analysis of Nouns and Verbs of the language and generates effective suggestions not greater then ten. All the non-Gujarati words along with different symbols are ignored and are not checked. The Gujarati Spell Checker is integrated along with Multi-Lingual Text Editor for the convenience to user. Currently the software is under

[32] *tdil.mit.gov.in/Bengali-ISIKolkataJuly03.pdf*
[33] *tdil.mit.gov.in/Gujarati-MSUniversityBarodaJuly03.pdf*

rigorous testing to find all the faults in the spell checker and morphological analyzer. Developers have plans to increase the size of the root dictionary covering the maximum possible words of the language and improving the algorithms for the correctness and efficiency of the spell checker. They also aim at building an independent spell checker which can be used on Unicode or ISCII compatible text.

NERPADAM - Malayalam Spell Checker

RCILTS-Malayalam, C-DAC, Thiruvananthapuram has designed a software subsystem named ***Nerpadam*** [34] that can be integrated with Microsoft word as a macro or the Malayalam editor style-pad developed by them, to check the spelling of words in a Malayalam text file. While running as a macro in word, it functions as an offline spell checker and can be use with a previously typed text file only. Both off line and online checking are possible when it is integrated with the text editor. It generates suggestions for wrongly spelt words. The system adapts a rule cum dictionary-based approach for spell checking.

It incorporates a fully developed morphological analyser for Malayalam. This module splits the input word into root word, suffixes, post positions etc. and checks the validity of each using the rule database. Finally it checks the dictionary to find the root word. If anything goes wrong in this checking it is detected as an error and the error word is reprocessed to get 3 to 4 valid words, which are displayed as suggestion. The user can add new words into a personalized data base file, which can be added to the dictionary.

Tamil Spell Checker

RCILTS for Tamil at Anna University, Chennai has developed a spell checker for Tamil that takes care of the rich morphological structure of Tamil. After tokenizing the document into a list of words, each word is passed to the morphological analyzer which analyzes only the correct words. The morphological analyzer tries to split the suffix. In

[34] *tdil.mit.gov.in/Malayalam-CDACThiruvananthapuramJuly03.pdf*

case of error, it passes the word to spelling verification and correction phase to correct the mistake. When the correction of errors is completed, root word and all components are sent to morphological generator (for word forming), which then generate the possible corrected words as suggestions.

The Spell checker for Tamil helps the user to identify most of errors, which may occur while typing. The tasks implemented in Tamil Spell checker are Case marker, postposition checking and adjective checking for nouns, PNG marker checking for verbs, Adverb checking, and adjacent key error checking.

Spell-checker for Punjabi

RCILTS-Punjabi at Thapar Institute of Engineering & Technology, Patiala [35] has developed a spell-checker for Punjabi. Developers have retained multiple spellings for the commonly used words and have used Harkirat Singh's *Shabad Jor Kosh* as the base. Additional words were taken from the dictionaries published by Punjabi University and the State Language Department and the corpus developed by CIIL, Mysore. For generating the suggestion list, a study was conducted to discover the most common errors made by Punjabi typists. A list of similar sounding words and consonants was also complied and a suggestion list using this knowledge was generated based on reverse edit distance. In most cases, the right suggestion is presented as a default suggestion and the user just needs to confirm the default suggestion and proceed with the next error. Otherwise, the user needs to scroll through a list of suggestions and pick one as the right one. The spell checker supports text typed in any of the popular Punjabi fonts as well as ISCII encoded files. It is nearly complete now.

1.1.3.3 Text-processors for Indian languages

AKSHARA: An Advanced Multi-Lingual Text Processor

AKSHARA[36] is an advanced multi-lingual text processing tool. Various dictionaries, morphological analyzers, spell checkers, OCR systems, TTS systems; text processing tools have been included in it. Several text processing tools, Telugu spell checker and Telugu TTS have also been integrated. AKSHARA encodes texts in a standard character encoding scheme such as ISCII or UNICODE. Mapping to fonts is done only for the purposes of display and printing - all other operations are performed on character encodings. Attributes are included in an open XML style markup language called Extensible Document Definition Language (XDL) developed by us. This makes it easy to convert to and from various other encoding schemes thereby ensuring highest levels of portability and platform independence.

A unique feature of AKSHARA is that it understands the script grammar and warns if one tries to build ungrammatical syllables. AKSHARA has been successfully used to clean up all the corpora at CIIL, Mysore.

AKSHARA is platform independent and also claimed to be also robust and reliable. AKSHARA has been successfully used to develop a 10 Million word corpus of Telugu. Full support for Regular Expressions and Finite State Machines is being integrated. AKSHARA is unique in providing multilingual email sending as well as receiving facilities. AKSHARA also enables one to develop interactive web pages in Indian languages and English. These web pages work across platforms and browsers. All this is made possible through WILIO technology. AKSHARA is freely available.

Bilingual Punjabi/English Word Processor:

A bilingual Punjabi/English word processor named **Likhari**[37] has been developed by RCILTS-Punjabi at Thapar Institute of Engineering & Technology, Patiala. **Likhari** supports word processing under the windows environment and allows typing and

[35] *http://tdil.mit.gov.in/Punjabi-TIETPatialaJuly03.pdf*
[36] *http://ildc.gov.in/telugu/htm/Akshara.htm*

processing in Punjabi Language through the common typewriter keyboard layout. It has MS-Word compatible features and commands. It provides a number of features that make the use of Punjabi Language on a computer easy and provides a number of tools to increase the efficiency of the user. These tools include Bilingual Spell Checker with suggestion list, on-screen keyboard layouts with composition reference for Punjabi language typing, bilingual search and replace, sorting as per the language, alphabetical order, technical glossaries and onscreen Bilingual dictionaries.

The main features of **Likhari** are:

- Very simple user interface
- Online active Keyboard for users who do not know how to type in Punjabi.
- Choice of Phonetic, Remington and Alphabetic Keyboard layouts with composition reference.
- Bilingual Spell Checker for Punjabi and English.
- Bilingual Search and Replace.
- Support for sorting the text in English or Punjabi as per alphabetical order.
- Extensive help at various levels to make it easy for the user to learn.

Nashir – Word processor for Urdu and other languages:

RCILTS-Urdu, Sindhi & Kashmiri, Centre for Development of Advanced Computing, Pune has developed a word-processor named ***Nashir***.[38] It is said to be capable of creating documents in Perso-Arabic languages, and at the same time to layout complete newspapers and magazines in Urdu, Sindhi, Kashmiri, Arabic and Persian. Each document of the Nashir consists of a number of pages on which text blocks, graphics, etc. can be placed. Nashir is also well suited for publishing segment. Spellchecker support which uses a base-dictionary is added in the full version.

Nashir supports Nastaliq True Type fonts (presently 2 fonts) and Naskh fonts (presently 12 fonts) along with fonts for Sindhi and Kashmiri. It supports C-DAC & Phonetic Keyboards as well as user-defined keyboards. It provides various drawing objects and

[37] *http://tdil.mit.gov.in/Punjabi-TIETPatialaJuly03.pdf*
[38] *tdil.mit.gov.in/UrduSindhiKashmiri-CDACPuneJuly03.pdf*

also supports the OLE Automation. *Nashir* supports Urdu, Sindhi & Kashmiri along with English. A transliteration engine (uTRANS) has also been implemented which converts an ".aci" (ISCII file) into transliterated version in Urdu script (Naskh/ Nastaliq). Rule based transliteration has been developed for Hindi & Punjabi. The user can save the document as HTML page, and thus Naskh as well Nastaliq scripts can be viewed on the Internet.

Phonetic keyboard is supported. Both Horizontal and Vertical kerning is provided to manually adjust a text. It has Horizontal and Vertical rulers in the GUI, dynamic font settings for the Urdu and English fonts.

1.2 Sanskrit Morphology

1. 2.0 Sanskrit morphology

Sanskrit has two-fold morphology- nominal and verbal. In Sanskrit, a syntactic unit is called *pada.* A Sanskrit sentence, according to **Cordona[39] (1988),** can be represented in following formula

$$(N\text{-}E^{n})p\ldots(V\text{-}E^{v})p.$$

Pada can be nominal (*subanta*) or verbal (*tiṅanta*). These forms are produced by inflecting the stems and hence they are part of Sanskrit inflectional morphology. The derivational morphology in Sanskrit includes formation of derived nominal and verbal stems. Both of them can be formed by either nominal stems or verb roots.

1. 2.1 Inflection

Sanskrit is very rich in inflections. The inflection morphology involves formation of two kinds of words or *padas*: *subanta padas* (nominal words) and *tiṅanta padas* (verb forms). The Pāṇinian analysis for Sanskrit has categorized each and every usable word under these two categories. A word cannot be used in the language unless it is one of them. Even the indeclinable words (*avyayas*) are first assigned to the first category, and later the characteristics of it are deleted. The two categories of *padas* are named so according

to the set of affixes which is affixed to them. These are called sup suffixes and tiṅ suffixes respectively, thus creating *sup+anta = subanta* and *tiṅ+anta = tiṅanta padas*.

Nominal inflection

Nominal inflection in Sanskrit includes all non-verb categories, i.e. *subanta-padas.* Sanskrit *subanta* forms can be potentially very complex. They can include primary (*kṛdanta*) and secondary (*taddhitānta*), feminine forms (*strīpratyayānta)* and compound nouns (*samāsa).* They can also include *upasargas* and *avyayas* etc. According to Pāṇini, there are 21 morphological suffixes called sup suffixes to be attached to the nominal bases (*prātipadika*).

Pāṇini has listed 21 *sup* suffixes:[40]

su, au, jas, am, auṭ, śas, ṭā, bhyām, bhis, ṅe, bhyām, bhyas, ṅasi, bhyām, bhyas, ṅas, os, ām, ṅi, os, sup.

These suffixes are grouped in seven sets of three affix each- (***su, au, jas***) (***am, auṭ, śas***) (***ṭā, bhyām, bhis***) (***ṅe, bhyām, bhyas***) (***ṅasi, bhyām, bhyas***) (***ṅas, os, ām***) (***ṅi, os, sup***).[41] The 7 sets basically indicate seven cases and the *vibhakti*-s used for their denotation. The three in each set are meant to indicate three numbers singular, dual and plural[42] respectively. These suffixes are added to the *prātipadikas*[43] (any meaningful form of a word, which is neither a root nor a suffix) to obtain inflected forms (*subanta padas*). The further modifications of the sup affixes appended to the *prātipadika* depends mainly on the gender and end character of the base.

prātipadikas are of two types: primitive and derived. The primitive bases are stored in g*aṇapāṭha* (collection of bases with similar forms) while the latter are formed by adding the derivational suffixes. They denote unity, duality and plurality respectively. Some words are only in the singular always, like *ekaḥ*(one), some are always dual like *dvi* (two), *akṣi* (eyes) etc. and some are always plural like *apaḥ* (water), *dārāḥ* (wife) etc.

---

[39] **George Cardona, 1988** Pāṇini, His Work and its Traditions, vol. i (Delhi: **MLBD**, **1988)**

[40] svaujasamauṭchasṭābhyāmbhisṅebhyāmbhyasṅasibhyāmbhyasṅasosāmṅyossup

[41] supaḥ

[42] dvyekayordvivacanaikavacane

[43] arthavadadhāturapratyayaḥ prātipadikam 1/2/45; kṛttaddhitasamāsāśca 1/2/46

Verbal inflection

The *tiṅ* terminations are 18 in number. They are added to the verb root which can be primitive or derived. The *tiṅ* terminations are divided in 2 equal sets in two *padas: parasmai* and *ātmane.* The terminations in each set are again viewed as 3x3, i.e combination of three persons (first, second and third) with three numbers.

1.2.2 Derivation

Sanskrit is rich in derivation. Both *subanta padas* and *tiṅanta padas*, before they are inflected, generally undergo certain derivational operations. Derivation includes suffixes which are added to both nominal stems and verbal roots to form new words, again nominal or verbal. In other words, newly derived nominal words or verb roots can be formed by nominal stems as well as from verbal stems. The suffixes are well categorized with their meanings and environments.

Nominal Derivatives:

Nominal derivatives are the words formed by adding derivational suffixes to nominal words. These new words can be nominal stems as well as verbal stems. The nominal stems which are derived from the nominal stems can be mainly categorized under taddhitāntas and strīpratyayāntas. The nouns derived from verb roots are called *kṛdantas*. These are derived by affixing kṛt suffixes to the verb roots.

The nominal forms formed from other nominal stems are called *taddhitāntas*- derived by adding taddhita suffixes to nominal stems. The secondary derivative affixes are called *taddhita,* which derive secondary nouns from *prātipadikas.* For example - *dāśarathī, gauṇa* etc. Pāṇini has listed many *taddhita* suffixes some of which are- *a, akañc, ac, añ, aṇ, at, iṣṭhan, īyasun, kan, ḍhak, ḍhañ, tamap, tarap, tayap, tal, tyap, tral, dvayasac, fak, matup, mātrac, yak, yat, yañ, ḍāc, kha, gha, cha, uraca, ṭhak, ṭhañ, ṭhan, na, ha, va, vatup* etc. For example, *dākṣī, kva, aśvakaḥ, viśvajanīnam, kṣatriyaḥ, mālīyaḥ, raivatikaḥ, dāṇḍikaḥ, laghutamaḥ, gurutaraḥ, gārgyāyaṇaḥ, iha, balavān* etc.

Sanskrit also has eight feminine suffixes *ṭāp, cāp ḍāp, ṅīṣ, ṅīn, ṅīp, uṅ* and *ti* etc. and the words ending in these suffixes are called *strīpratyayānta.* For example - *ajā, gaurī, mūṣikā, indrāṇī, gopī, aṣṭādhyāyī, kurucarī, yuvatī, karabhorū* etc.

The nominal morphology of Sanskrit distinguishes nouns and adjectives on the one hand and pronouns on the other. There is no absolute distinction between nouns and adjectives with respect to inflection, but modifiers take the number and gender of the terms they qualify.

Pronouns are personal (first and second person), demonstrative (deictic), interrogative and relative. Pronouns other than personal pronouns observe gender distinctions. There are also endings particular to pronouns.

Sanskrit also has indeclinable terms, including particles such as the connective *ca* 'and' (sentence and nominal connective) and the negative particle *na*, as well as terms such as *yad, yadi* 'if', *tad, tarhi* 'then', *karhi* 'when?', *tataḥ* thence, *yataḥ* whence, *kutaḥ* whence?, *yadā* when, *tadā* then, kadaa when?, yatra where, tatra there, kutra where?, derived from pronominals.

There are also preverbs, which regularly occur immediately preceding a verb or another preverb – although in vedic they can be separated from a verb or follow it- as well as pre- and post-positional terms like adhi, anu, which co-occur with particular case-forms.

Indeclinable NPs

avyaya subanta-padas remain unchanged under all morphological conditions [44] . According to Pāṇini [2.2.82][45], affixes *cāp, ṭāp, ḍāp,* (feminine suffixes) and *sup* are deleted by *luk* when they occur after an *avyaya*. Pāṇini gives definitions for identifying certain word forms as avyayas which also include compound forms.

[44] sadṛṁ triṣu liṅgeṣu sarvāsu ca vibhaktiṣu ।
vacaneṣu ca sarveṣu yanna vyeti tadavyayam ॥ - gopatha brāhmaṇa
[45] avyayādāpsupaḥ 2/4/82

Verbal Derivatives:

Verbal derivatives may be nominal stems or verbal stems derived from verb roots or stems. The nouns that are derived immediately from verbs comprehend a great variety of terms. There are two principal classes of their classification:

- Adjectives (Attributives)
- Names (Substantives)

These different nouns are formed by affixing certain terminations to the verb root, which is modified in a greater or lesser degree, and then forms the inflective base. These terminations are numerous. The greater number has a very limited application while others comprehend a large class of words. The primary derivatives are called *kṛdanta*. The primary affixes are to be added to verbs to derive substantives, adjectives or indeclinable *kṛt*. For example *paṭhitavyam, pātavya, paṭhanīya, pacelima, jeyam, deyam, karttā, kumbhakāraḥ, janamejayaḥ, pāṭhakaḥ, paṭhantī, gantum, khāditum, svapnam gatiḥ, gatvā,vihāya, ādāya* etc.

Verbal derivatives also include participles, infinitives, gerund etc.

1.2.3 Compounding

Simple words (*padas*), whether substantives, adjectives, verbs or indeclinables, when added with another *subanta pada* form *samāsa* (compound). Sanskrit *samāsas* are divided into four categories, some of which are divided into sub-categories. The four main categories of compounds are as follows:

(1) Adverbial or *avyayībhāva*,

(2) Determinative or *tatpuruṣa,*

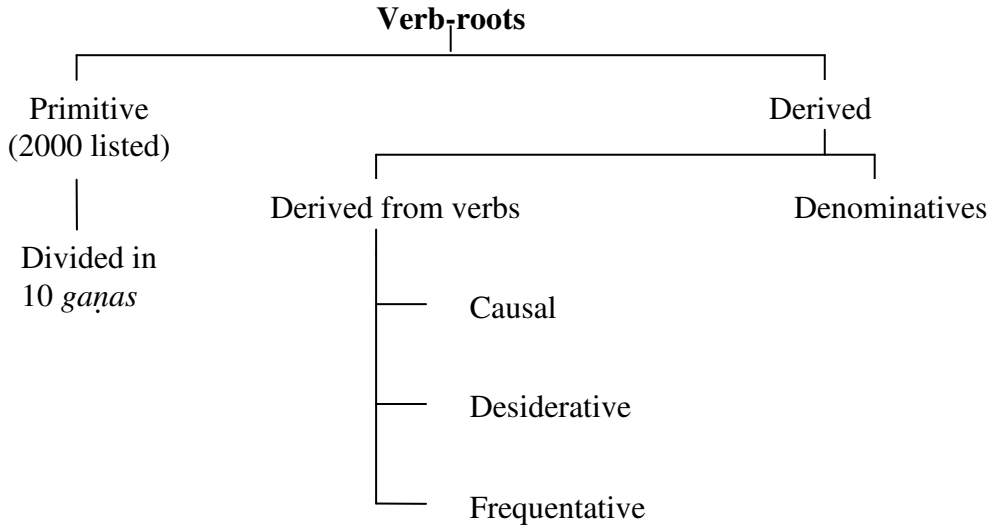
(3) Attributive or *bahuvrīhi* and

(4) Copulative or *dvandva*.

dvandva and *tatpuruṣa* compounds may be divided into sub-categories also. tatpuruṣa has dvigu and karmadhāraya as its sub-categories.

1.2.4 Sanskrit Verb-Morphology:

The verb forms are derived from verb-roots or *dhātus*. These *dhātus* are encoded with the core meaning of the verb. These can be primitive[46] or derived[47]. Primitive verb-roots, which are around 2000 in number, have been listed in a lexicon named *'dhātupāṭha'*. They are divided in 10 groups/classes called *gaṇas*. All the verb-roots of a group undergo somewhat similar inflectional process. Derived verb-roots may be derived from primitive verb-roots or from nominal forms. Prefixes also play an important role as they can change the meaning of a verb root. These roots then have to undergo various inflectional suffixes that represent different paradigms. In this process, the base or root also gets changed. The chart given on the next page gives an overview of Sanskrit verb roots.



Derived Verb-roots:

Derived from verb-roots:

1. **Causals *(nijanta)*-** Causals are formed by adding *'ṇic'* suffix to a primitive verb root. They convey the sense that a person or thing causes another person or thing to perform the action or to undergo the state denoted by root.

[46] *bhūvādayo dhātavaḥ* (*Pāṇini* 1/3/1)

2. **Desideratives *(sannanta)-*** Desiderative of a primitive verb root is formed by adding *'san'* affix to it. It conveys the sense that a person or thing wishes to perform the action or is about to undergo the state indicated by the desiderative form. Any basic verb-root or its causal base may have a desiderative form.

3. **Frequentatives *(yaṅanta)-*** Frequentatives verbs import repetition or intensity of the action or state expressed by the root from which it is derived. Can be of two types:

    a. *Ātmanepada* Frequentative *(yaṅanta)* – *'yaṅ'* affix added
    b. *Parasmaipada* Frequentative (*yaṅluganta*) – '*yaṅ*' is added but deleted

An illustration is given below of formation of derived verb-roots from a primitive verb root *bhū*.

bhū (to be)----------
- --- (+*ṇic*) ---- *bhāvay* (to cause someone or something to be)
- --- (+ *san*) ---- *bubhūṣ* (to desire to be)
- --- (+ yaṅ) ---- *bobhūy* (to be repeatedly)
  - (yaṅ deleted) ---- *bobho/bobhav*

These derived verb-roots, however, undergo similar operations, with some specifications, to form verb forms.

Roots derived from nominal words:

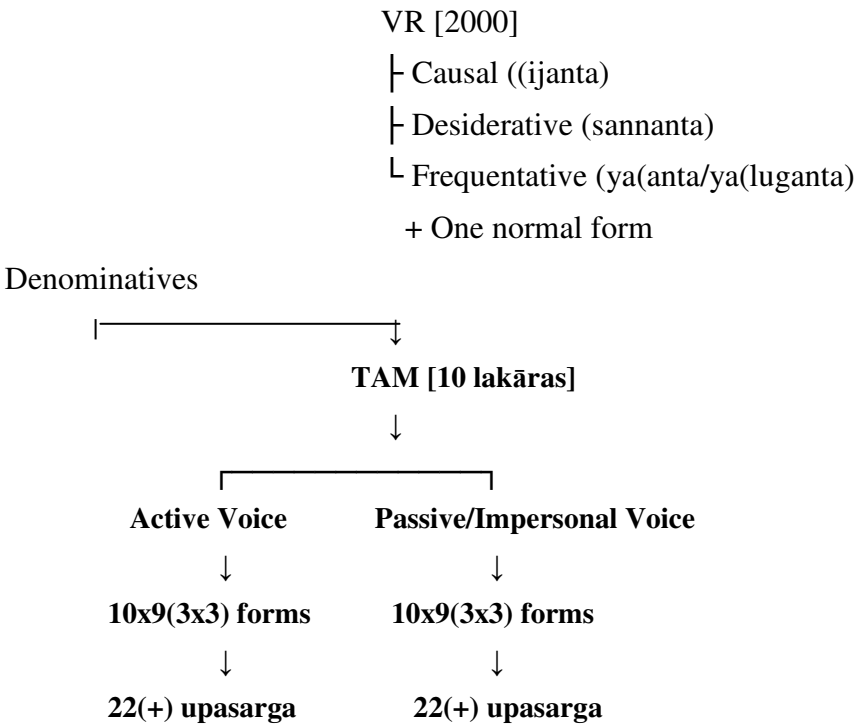
A large number of Sanskrit verb-forms can be derived from nominal words. These are known as 'nāmadhātus' (denominatives). Taking a nominal word as head, various derivational suffixes are added to these to form nominal verb-roots. The sense conveyed by the nominal verb root depends upon the suffix added to it. Yet, denominatives commonly import that a person or thing behaves or looks upon or wishes for or resembles a person or thing denoted by the noun. These denominatives, however, can be innumerable as there is no end to nominal words in Sanskrit.

---

[47] *sanādyantā dhātavaḥ* (*Pāṇini* 3/1/32)

Process of formation of Sanskrit verb forms:

A Sanskrit verb root may take various forms. There are ten lakāras that represent Tense, Aspect and Mood. Inflectional terminations are 18 in number. These are divided in two groups – Parasmaipada and Ātmanepada, each having 9 affixes which is a combination of 3 persons x 3 numbers. A verb is conjugated in either pada, though some of the roots are conjugated in both. For each different lakāra, a root is affixed with these 9 terminations. Again, there are three voices- Active, Passive and Impersonal. Transitive verbs are used in the Active and Passive voices while intransitive verbs are conjugated in the Active and Impersonal voices. Addition of one or more of 22 prefixes (upasargas) to verb roots can result in more variety of forms. Derivative verb roots, both derived from verb roots as well as nominal words, also follow same process to form verb forms. There can be some specific rules and exceptions in some cases. A chart given here gives a rough estimate of all the possible verb-forms of Sanskrit. ┐



[48] **Mishra Sudhir K., Jha, Girish N.,** 2004**,** ***Identifying Verb Inflections in Sanskrit morphology*****,** In proc. of SIMPLE 04, IIT Kharagpur, pp. 79-81.

The verb roots of *gaṇa* 1, 4, 6 and 10 adopt certain terminations when *tiṅ* affixes are added to them. Consequently, the verb roots of these class forms base ending in *'a'*. The *tiṅ* affixation also influences the verb root and it undergoes several morpho-phonemic changes such as its end vowel is gunated. The verb root can adopt certain more operations resulting in the final verb-form.

1.2.5 Survey of Morphological tools for Sanskrit

C-DAC

Deśika[49] (Natural Understanding System), a Software Package, developed by Indian Heritage Group, C-DAC, Bangalore led by P. Ramanujan, claims to generate and analyze plain and accented written Sanskrit texts using grammar rules of Pāṇini's Aṣṭādhyāyī, with an exhaustive database of Amarakośa and heuristics of semantics and contextual processing from Nyāya and Mīṃāmsā Śāstra-s. It also claims to analyze Vedic texts. The analysis module of the Software is called as a general purpose Sanskrit parser which claims to dissolve and identify the compound and combined word forms, though the present version of the Software downloadable from the TDIL (Technology Development for Indian Languages) website has Subanta Generation Module only. This module has two modes: **choose mode** declines for the *prātipadika*-s already present in the list and **edit mode** declines for the entries which are not present in the list for which suitable gender and paradigm should be selected. Deśika is also supposed to include Vedic processing and śābdabodha[50].

Sanskrit Authoring System (VYASA)[51] claims to be a robust multilingual document editor with transliteration facility among Indian Languages and Roman, sorting and

[49] Deśika (Natural Language Understanding System), http://tdil.mit.gov.in/download/Desika.htm
[50] C-DAC R&D Activies: Development of Desika A Natural language Understanding (NLU) system for Sanskrit. [1990 – 1994], http://www.cdac.in/html/ihg/activity.asp, accessed on July 16, 2006
[51] C-DAC, R&D Activities: Developing Sanskrit Authoring System (VYASA), http://www.cdac.in/html/ihg/activity.asp and The House Magazine of C-DAC, Pen to Paper Developing Sanskrit Authoring System – VYASA, http://www.cdac.in/html/connect/3q2000/art10a.htm

searching facilities, indexing, and concordance. It also claims to provide tools for analyses at morphological, syntactic and semantic levels. Tools for searching/indexing/sorting, lexical updation, lexical tagging, extraction/indexing of quotations in commentaries/explanations, transliteration facility, word split programs for *sandhi* and *samāsa*, poetry analysis (textual/metric/statistical), statistical tools like concordance, thesauri, and electronic dictionaries are also included. However, the system is not available anywhere to evaluate or check.

Academy of Sanskrit Research, Melkote

Academy of Sanskrit Research (ASR), Melkote under the directorship of Prof. M. A. Lakshmitatachar has been working on NLP in Sanskrit and other Indian languages for more than 10 years. The software 'śābdabodha' developed at ASR, Melkote is available on TDIL website. It is claimed to be an interactive application built to analyze the semantic and syntactic structure of Sanskrit sentences. The software works on DOS 6.0 or higher with GIST (Graphic based Intelligence Script Technology) shell on Windows 95 platform. The software is said to include a user interface. For an input Sanskrit sentence or a file the program will check for the syntactic compatibility and otherwise gives all the morphological details with POS information having nouns, pronouns, adjectives, participles, indeclinables, indeclinable participle and verbs tags of the components of the sentence. It claims to process all types of sentences of Sanskrit, and can handle generation and analysis of *subanta* of more than 26,000 stems, *tiṅanta* conjugations of roots, in two voices, ten *lakāras* and three modes viz. *kevala tiṅanta, ṇijanta* and *sananta*. It also handles the generation and analysis and identification of case inflected forms of 11 types of *kṛdanta* of 150 roots. Apart from this it is said to have a database of 690 avyayas, 26,000 nominal stems, 600 verbal roots, *kṛdanata* forms of 600 verbal roots, 5 *taddhita* suffixes. For handling the semantic analysis, a matrix of 52 sets of nouns with their synonyms amounting to 300 nouns, 27 actions denoted by nearly 200 verbs are said to have been prepared.[52] This institution is also working with 20 odd software tools like **Bodha** (Sentence disambiguation system according to *śābdabodha* of

[52] Language Processing Tools: TDIL website, http://tdil.mit.gov.in/nlptools/ach-nlptools.htm

navīna nyāya system), **Śemuṣī** (*subanta* generator/analyser), **Prajñā** (*tiṅanta* generator/analyser), **Cetanā** (*kṛdanta* generator/analyser), **Pāṇini** (*sandhi* joiner according to Pāṇinian rules) etc. which are to be released.[53]

Rashtriya Sanskrit Vidyapeetha (RSV), Tirupati

RSV Tirupati has put much effort in developing linguistic resources for NLP in Sanskrit. Prof. K.V. Ramakrishnamacharyulu, (presently V.C. of Rashtriya Sanskrit Sansthan, Jaipur) and Dr. Srinivasa Varkhedi along with Prof. Vineet Chaitanya and Amba P. Kulkarni have initiated many projects and have developed many tools like *pada-ccheda*, which isolates Sanskrit compound words into its components, which works on Sanskrit ISCII text in Linux environment. It is also working on developing Sanskrit morphological analyzer. An initial analyzer developed in collaboration with IIIT-H is already online. Apart from this it is also concentrating on *kṛdanta* and *tiṅanta* analyzers and also generators for *subanta, tiṅanta* and *samāsa*. RSV Tirupati along with C-DAC Bangalore, Ahobila Mutt Sanskrit College Madhurantakam Tamil Nadu, Poorna Prajna Samshodhana Mandiram Bangalore, Chinmaya International Foundation Veliyanad Kerala, ASR Melkote Karnataka, IIIT-H, Dept. of Sanskrit H.S.Gour University Sagar Madhya Pradesh have combined initiative to develop a large Sanskrit Corpus (presently this is not online).[54]

RCILTS – Utkal University

Resource Centre for Indian Language Technology Solutions (RCILTS) – Oriya Centre at the Department of Computer Science and Application, Utkal University, led by Prof. Sangamitra Mohanty and funded by Ministry of Information Technology (MIT), has been working on Oriya-English-Hindi Trilingual Dictionary, Oriya and Sanskrit WordNet, Trilingual Word Processor, Grammar and Spell checkers and MorphAs for Oriya and

[53] Academy of Sanskrit Research, Melkote, http://www.sanskritacademy.org/About.htm
[54] RSV Tirupati, http://rsvidyapeetha.ac.in and http://www.sansknet.org

also English to Oriya Machine Translation System. It has received IPR for four of their products from Ministry of Human Resource and Development (MHRD), Govt. of India.

San-Net (Sanskrit WordNet)

RCILTS-Utkal[55] claims to have developed a proto-type of Sanskrit WordNet using Navya-Nyāya and Pāṇinian Grammar. It has 300 Sanskrit words (250 Nominal words and 50 Verbal words) having synonymy, antonym, hyponymy, hypernymy, holonymy and meronymy relations with their analogy, etymology, and definitions. It also claims that a standard Knowledge Base (KB) that has been developed for analyzing syntactic, semantic and pragmatic aspects of any lexicon.

The Sanskrit Heritage Site

Dr. Gerard Huet, Director, INRIA has developed various computational tools for Sanskrit which are available online[56]. The **Declension Engine** declines all the nominal inflectional forms with the 'compound base' for a query word given with its gender information. It also gives the 'compound base' of the query word. The **Conjugation Engine** gives all the possible forms of the verb root in its *ātmane* and/or *parasmai* terminations, in *kartari* and *karmaṇi/bhāve* voices with its Desiderative forms in eight *lakāra*-s. It also gives few participle and indeclinable *kṛt* forms[57]. He claims that from 535 roots, his engine generates 14554 noun forms, 113935 root verbal forms, 203281 root participial forms, 14311 *iic* and periphrastic forms, and 737 indeclinable forms, totaling roughly half a million forms[58]. **Lemmatizer** and **Sanskrit Readers** are the analyzers. While the lemmatizer tries to tag a given simple inflected noun or a verb (without *upasarga*-s), the Sanskrit Reader Companion analyses a given phrase or a simple sentence, segments it into individual words, tags each word and parses the input. Modular transducers are applied to constraint the lexical analyzer to recognize the stream of forms as a regular expression over 14 phases, specifying Sanskrit's morphology geometry to get

[55] RC-ILTS – Computer Science & Application, Utkal University, http://www.ilts-utkal.org/sanskrit.htm
[56] The Sanskrit Heritage Site, Huet, Gerard. http://sanskrit.inria.fr/
[57] Ibid.

right composition of compound chunks. Over generation of segmentation and ambiguity of tags are checked by semantic role analysis similar to Pāṇini's kāraka theory and also by governance patterns of verbs.[59]

The Sanskrit Library

The Sanskrit Library Project, directed by Dr. Peter M. Scharf, Classics Dept., Brown University, has developed a web-based reading room holding Kramapāṭha, a Sanskrit independent-study reader on texts like Pañcatantra, Rāmopākhyāna and Pāṇini's Aṣṭādhyāyi. The reader gives the text in Devanagari with Roman transliteration, *sandhi* analysis, a detailed grammatical analysis and lexical analysis of the text, with notes and translation. As part of independent-study texts, a detailed classification of grammatical categories (tag set) is made.

A nominal inflection generator tool is also available which generates nominal inflections for a given word with gender and class information.[60]

IIIT Hyderabad

Language Technology and Research Centre (LTRC), IIIT-H, is one of the leading NLP centres in India. IIT Kanpur and later LTRC, IIIT-H, have been the pioneers in the field of language technology and to initiate Pāṇinian approach to NLP in India. LTRC has several ongoing activities with Govt. of India, Carnegie Mellon University's Language Technology Institute, UPENN, HP Labs, Google, Nokia and TCS besides several academic institutions in India. Prof. Rajeev Sangal and Prof. Vineet Chaitanya with the Akshar Bharati[61] group have developed many NLP tools for Indian languages like MorphA-s (morphological analyzers), Anusaraka-s[62] (language access tool), Shakti

---

[58] Huet, Gerard, 2006, Parsing Sanskrit by Computer, (abstract) in the Proceeding of the 13th World Sanskrit Conference, Edinburgh, UK

[59] Ibid.

[60] The Sanskrit Library Reading Room - http://cgi-user.brown.edu/cgi-user/sanskrit/login

[61] Akshar Bharati, is a personification of groups (@ IITK, IIIT-H, University of Hyderabad, etc.) working on NLP with special emphasis to Indian Languages giving due attention to Indian theories of grammar and language.

[62] Anusaaraka or a Language Accessor is a computer software which renders text from one Indian language into another (Kannada-Hindi, Marathi-Hindi, Punjabi-Hindi and Telugu-Hindi). It produces output not in

Machine translation System. MorphA-s for Hindi, Telugu, Marathi, Kannada and Punjabi were developed as part of Anusaraka-s and now presently they are made available online and also can be downloaded from their website. Telugu MorphA is said to have coverage of 95% for any arbitrary modern standard Telugu text and Hindi MorphA has 88% coverage.[63] MorphA-s for these languages were developed based upon the paradigm model.[64] For a given word the MorphA gives the root word with its feature information like gender, number, person, tense etc.[65]

Sanskrit MorphA was developed by Vinish Jain, an M.Tech student of IIIT-H under the guidance of Amba P. Kularni, which was a further development of an earlier working Sanskrit MorphA developed at ASR Melkote. This MorphA was developed using paradigm approach in the model of other MorphA-s at IIIT-H which used a lexicon derived from Monier William's dictionary. This could only handle *subanta*-s. Later with the collaboration of RSVP Tirupati, the present Sanskrit MorphA[66] was developed by Amba P. Kulkarni and V.Sheeba. This MorphA has separate modules to handle *subanta*, *tiṅanta, kṛdanta* and *samāsa*. Each word is filtered through all the four modules and all possible answers are produced. *subanta* module contains 222 paradigms for nouns and pronouns with a root dictionary of around 1.5 M words extracted from Monier Williams dictionary. *tiṅanta* module has a database of ~10 M verb forms (no prefixed verbs). *kṛdanta* module can handle ~42 K of 20 types of *kṛdanta*. This approach has the advantage of developing further without disturbing the main program, as the program is independent of data and also more modules can be added.[67] This MorphA can handle only *sandhi*-free text and cannot filter the multiple tags and disambiguate it for a single word in a given context.

---

the target language but close to it, which is in comprehensible to the reader who is trained to read the output.

[63] Morphological Analysers – IIIT Hderabad – http://www.iiit.net/ltrc/morph/morph_analyser.html

[64] Bharati, Akshar et al, 1999, Natural Language Processing: A Paninian Perspective, Prentice Hall Pvt. Ltd. New Delhi, pp.39-43.

[65] Ibid..

[66] RSVP, http://rsvidyapeetha.ac.in/~anusaraka (accessed: 13.10.2006)

Jawaharlal Nehru University (JNU)

The RCILTS – Sanskrit, Japanese, Chinese unit of JNU, under the leadership of Prof. G.V.Singh claims to have developed web based Sanskrit Language Learning System for the use of scholars for designing Knowledge based systems based on the Indian traditions. The unit has developed a computational module of Aṣṭādhyāyī of Pāṇini, Sanskrit-English lexicon, English-Sanskrit lexicon and a lexicon of Nyāya terms. It also says that it has made some efforts on the *sandhi* analysis system.[68]

Girish Nath Jha[69] developed a Nominal Inflection Generator for Sanskrit for his M. Phil. dissertation. The program, written in prolog, generates all the inflections of *subanta* given a Sanskrit word with gender and ending letter information.

Special Centre for Sanskrit Studies, JNU:

Special Centre for Sanskrit Studies, JNU, established in 2002, has undertaken the task of developing NLP tools for Sanskrit. Under the guidance of Dr. Girish Nath Jha, the students have developed certain useful programs for computational processing of Sanskrit texts. A list of works is given below:

The NLP team at the centre has developed an Online Multilingual Amarakosha (OMA) under a project funded by UGC under UPOE program. It is Unicode based software which supports seven languages- Sanskrit, Hindi, Kannada, Punjabi, Bangla, Oriya and English and allows the user to search the synonym from one language to another. The output displays the grammatical and semantic category of the word, its base word, reference and ontological information. The software also provides the facility to enter and

---

[67] Bharati Akshar, Amba Kulkarni, V Sheeba, "Building a Wide Coverage Sanskrit Morphological Analyzer: A Practical Approach", in the Proceedings of First National Symposium on Modelling and Shallow Parsing of Indian Languages, 2-4 April 2006, IIT Bombay, Mumbai

[68] RCILTS, JNU – Achievements: http://tdil.mit.gov.in/SanskritJapaneseChinese-JNUJuly03.pdf (accessed 15.10.2006)

[69] Jha, Girish Nath, 1993, 'Morphology of Sanskrit Case Affixes: A Computational Analysis', M.Phil., submitted to JNU, New Delhi.

edit the data by language experts. The software is expected to be extended as a multilingual interface, search engine and text processing tool in near future.[70]

R. Chandrashekhara[71], developed tag-sets for POS tagging of Sanskrit text as part of his Ph.D. thesis. This tagger (POST) which is an online system run on Apache Tomcat platform using Java Servlet takes *sandhi*-free classical Sanskrit prose text as input and provides the tagged text as output. The system is very important for the further R&D on the Sanskrit-Indian Languages Machine Translation Systems (MTS).

Sudhir Kumar Mishra[72], has recently completed his Ph.D. research on Kāraka Analyzer for Laukika Sanskrit prose text based on Pāṇini's Kāraka formulations. Kāraka rules are of central importance in Sanskrit syntactic structure. As part of this research work he also worked on identification of verb inflections in Sanskrit morphology[73].

Subash Chandra[74] developed a Sanskrit *Subanta* Recognizer and Analyser System (SRAS) for his M.Phil. dissertation. SRAS is also an online system run on Apache Tomcat platform using Java Servlet and MSSQL server 2005 as back end. This system has been developed according to Pāṇinian formulation which accepts only non-joint (*sandhi-rahita*) Sanskrit text in Devanāgarī script and fully depends on both the rule base, example base and a database of other linguistic resources. The system has been tested with some selected corpora and claims to give an average accuracy of 91.65%.

Research work is also being done on learning Sanskrit language using e-learning approach[75], and *sandhi* analyzer applying Pāṇinian and some heuristic rules[76], online

[70] System is accessible @ http://sanskrit.jnu.ac.in
[71] Chandrashekhar, R, 2006, 'POS Tagging for Sanskrit', submitted for Ph.D degree at SCSS, JNU.
[72] Mishra, Sudhir Kumar, 2007, 'Sanskrit Karaka Analyzer for Machine Translation', submitted for Ph.D. degree at SCSS, JNU
[73] Mishra, Sudhir Kumar & Girish Nath Jha, 2005, 'Identifying Verb Inflections in Sanskrit Morphology', in the proceedings of SIMPLE-05, IIT-Kharagpur, pp.79-81
[74]Chandra, Subash, 2006, 'Machine Recognition and Morphological Analysis of Subanta-padas', submitted for M.Phil degree at SCSS, JNU
[75] Bhowmik, Preeti & Jha, Girish Nath, 2006, 'Sanskrit Language Pedagogy: an e-learning approach', In the Souvenir Abstracts of 28th AICL, BHU, Varanasi, p.150.
[76] Kumar, Sachin & Jha, Girish Nath, 2006, 'Issues in sandhi processing of Sanskrit', In the Souvenir Abstracts of 28th AICL, BHU, Varanasi, p.129.

indexing of Ādiparva of Mahābhārata[77], computational analysis of Sanskrit gender and analysis of derived nouns in Sanskrit.[78]

Sanskrit to English Translator:

Aparna Subramanian of School of Computer Science, Devi Ahilya Vishwavidyalaya, Indore worked on a project titled "Sanskrit to English Translator"[79] for her M.Sc. degree. It performs sandhi viccheda of Sanskrit words and then translates them to English. It has two components: morphological parser and translation generator. The morphological parser module consists of a set of transducers that transform the provided input text to a set of acceptable output. It also gives the parse of these words. The second part of the parser is a Viccheda module, which applies reverse sandhi rules to split the combined Sanskrit words into separate basic words. The translator is also a two part module, which first structures the English sentence according to the grammar using the parse information. The second generates equivalent English words according to the morphological details. The two combined together give the needed translated sentence.

[77] Mani, Diwakar, & Jha, Girish Nath, 2006, 'Online indexing of Ādiparva of Mahābhārata', In the Souvenir Abstracts of 28th AICL, BHU, Varanasi, p. 125.
[78] Singh, Surjit Kumar & Jha, Girish Nath, 2006, 'Strategies for Identifying and Processing Derived Nouns in Sanskrit', In the Souvenir Abstracts of 28th AICL, BHU, Varanasi, p. 131.
[79] Subramanian, Aparna, Jan 2005, "Sanskrit to English Translator", Language in India, Vol. 5:1.