Lexicon-directed segmentation and tagging in Sanskrit

GÉRARD HUET

Abstract

We propose a methodology for Sanskrit processing by computer. The first layer of this software, which analyses the linear structure of a Sanskrit sentence as a set of possible interpretations under sandhi analysis, is operational. Each interpretation proposes a segmentation of the sentence as a list of tagged segments. The method, which is lexicon directed, is complete if the given (stem forms) lexicon is complete for the target corpus. It uses an original design for a finite-state transducers toolkit, based on functional programming principles. Further layers of this computational linguistics architecture are discussed.

1. COMPUTATIONAL LINGUISTICS AND SANSKRIT

Descriptive linguistics is the study of natural language phenomena. Theoretical linguistics strives to provide formal models of linguistic activity. Noam Chomsky initiated modern theoretical linguistics with successive theories (context-free and transformational grammars, government and binding, minimalism). These theories usually provide a *generative grammar* paradigm, by which means any valid sentence in the language may be generated. With the advent of computers a still more formal approach is attempted, where executable programs process a digital representation of natural language into information structures, which provide some degree of understanding

of the given text or discourse. In the pioneer days of the fifties, automated translation systems were thus attempted, although difficulties were largely underestimated, leading to doubts about the feasibility of natural language understanding by computers. 50 years later, however, the situation is more promising, due to thorough investigations of linguistic models, the availability of fast processors and large memories, the development of large-scale finite-state methods and the availability of large corpuses where statistical methods could train generic methods by tuning their parameters on real data.

Linguistic studies were part of Indian culture from the early ages. Grammar (*vyākaraṇa*), phonetics (*śikṣā*), prosody (*chandas*), hermeneutics (*nirukta*), were recognised as fields of knowledge (*vedāṅga*). A stream of grammarians set to describe in exact terms the structure of Sanskrit utterances. Thus the grammarian Pāṇini, in the 4th century B.C., wrote a treatise in eight parts (*Aṣṭādhyāyī*) which may be considered a full-fledged generative grammar for classical Sanskrit. Further grammarians of this tradition (Kātyāyana, Patañjali, Bhartṛhari) are basically commentators on Pāṇini. They do not question his basic model, but rather explain it, refine it and complete it. Actually, the existence of this reference generative grammar had a strong influence on the historical development of the language: whereas the common language (*prākṛta*) kept evolving into modern (North Indian) languages, Sanskrit was frozen in a pristine state of preservation, with very regular phonetic, morphological, and syntactic paradigms. On the other hand, the availability of recursive rules in Pāṇini's model allowed poets to develop artificially iterative uses of certain constructs, such as compound formation of unbounded depth, making Sanskrit a quite unique 'natural' language.

However, the availability of Pāṇini's grammar, with its associated lexicons (*dhātupāṭha, gaṇapāṭha*), is not sufficient for the direct implementation of a Sanskrit mechanical analyser. Let us now explain this point.

It is recognized that linguistic modelling proceeds by successive layers, from an external surface form (speech or written text) to an internal informational net of semantic concepts. One traditionally distinguishes these layers as the successive submodels of linguis-

tic activity, from surface speech or text to conceptual structures and plans:

- Phonetics (speech recognition)
- Phonemics (discretization of utterances as streams of phonemes)
- Morphology (structure of words)
- Syntax (structure of sentences)
- Rhetorics (structure of discourse)
- Semantics (structure of concepts and information updating)
- Pragmatics (structure of causality)

A generative grammar such as Pāṇini's gives an account of the generation of correct speech from an intended meaning, going in reverse in the above list, in a deterministic synthesis direction. The process corresponds roughly to the SENS-TEXTE theory of Igor Mel'cuk. On the other hand, a computer program analysing a surface phonetic stream will have to traverse the layers in the opposite direction, leading to an intrinsically non-deterministic analysis. Since the crucial information about semantics and pragmatics is not available in lower layers, ambiguities arise at the various stages, leading to a combinatorial explosion of possible interpretations. The choice between these interpretations depends on knowledge about the subject matter, tradition, and common sense, all processes which are hard to correctly model by computation. Actually, poets and the composers of esoteric treatises took advantage of the inherent ambiguity of the language to compose Sanskrit texts with multiple meanings, which may be deciphered only by separate commentaries. The Indian tradition of frequent metaphorical descriptions adds a supplementary degree of difficulty to an already formidable task. Thus, on one hand Pāṇini's grammar can only be used as a gold standard for any Sanskrit computational processor, and on the other hand we cannot hope, in the current state of knowledge engineering, to provide more than a computer-assisted Sanskrit analyser, usable as a tool for text mark-up assistance by knowledgeable Sanskrit scholars (*paṇḍita*). Still, such a tool would be extremely useful, for facilitating the organisation of critical editions, creating concordance indexes, comparing historical

and regional variants of the literary works, organising layers of commentaries, and otherwise computing statistical information which is, in the present state of the art, impossible to achieve rigorously.

2. OVERALL DESIGN OF A LINGUISTICS PLATFORM

The above layer description of linguistics activity determines the overall architectural design of a computational linguistics platform. Of course, the layers are not independent: for instance, agreement and valence constraints from syntax must be consistent with morphological features (gender, number, person); also euphony causes interference between phonemics and morphology. And notions of topic and focus from communicative semantics relate to prosody, which is actually part of phonetics. Nonetheless, this layered architecture gives a first approximation to the structure of informatics processing modules.

In parallel to this sliced cake model of a linguistics flow processor, we find the pervasive *lexicon*, which holds all the word-dependent parameters of the various levels. The lexicon (and its derivatives, such as the inflected forms lexicon, containing conjugated forms of verbs and declined forms of substantives and adjectives) is the central repository of knowledge about words, their phonemic representation, their etymology and morphology analysis, their grammatical role, their semantics, etc.

This general model is fairly generic over language families, but the importance of the various layers, and their mutual interference, vary from language to language. For instance, Sanskrit has a specially complex morphology, compensated by a relatively undemanding syntax. But, first of all, euphony is systematic, with a complex set of morphemic rules, both at the level of morphology derivation of inflected words (internal sandhi) and at the level of shallow syntax for the formation of compounds, as well as for the glueing of words into sentences (external sandhi), leading conversely to a complex segmentation problem. This exposes the first serious difficulty for the mechanical processing of Sanskrit: the problem is to guess the sequence of words which by external sandhi rewriting would yield the

given stream of phonemes. This problem occurs either when processing a phoneme stream issued from a speech recognition module or when processing a written sentence. This sandhi decomposition is by no means unique, and actually only semantic considerations will ultimately curb the overgeneration of this non-deterministic analysis.

3. LEXICON

We started by constructing a Sanskrit lexical data base from a Sanskrit to French dictionary which we developed over the years as a glossary of Indian culture. The structure of the lexical data base is explained in (Huet 2000; Huet 2001).

The database comprises 12,500 entries at the time of writing. These include: 520 roots, 7,500 word lemmas, 2,800 compounds, and 1,300 idiomatic expressions. Printable versions of the lexicon are generated in two formats: book PDF format, using a *devanāgarī* font with ligature computation by Velthuis' DEVNAG preprocessor; and HTML hypertext format, as a Web site with search engines.

The lexicon is structured as an inductive datatype rather than as a relational database format. The structure makes explicit the various parameters such as gender, number, etc. in order to facilitate morphology computations. Specific etymological links express the derivation of a word, ultimately to its roots. Various invariants are checked when the lexicon is processed, namely when it is compiled into internal representations. This compilation process is fast, and thus the source form of the database is the reference document which gets corrected and updated as a text file by the lexicographer.

4. SANDHI

External sandhi means euphony transformation of words when they are consecutively uttered. 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, which corresponds to minimizing the energy necessary to reconfigure the vocal organs at the juncture between the words. Sandhi af-

fects also the morphological derivation of words and their inflected forms, using a more complex *internal sandhi* process - long term usage propagated the smoothing inside the words, for instance by transforming the nasal *n* into the retroflex *ṇ* and similarly the sibilant *s* into the retroflex *ṣ*.

We model sandhi computation by a two-tape transducer. The transducer has two input tapes, one for storing the stream of phonemes issued from w_1 read from right to left, the other for storing the stream of phonemes issued from w_2 read from left to right. According to the interaction between the two boundary phonemes, proper output is generated on the output tape. This process is basically deterministic, although some choices and variants exist. We have two versions of the sandhi automaton, respectively for internal and external sandhi.

The internal sandhi processor is used systematically by the morphological engine described in the next section. External sandhi is iterated over all possible combinations of interactions by phoneme strings of length one or two - no deeper transformations occur in the case of external sandhi. This iteration computes a compiled form of a set of sandhi rules of the form $u|v \rightarrow w$, where u, v and w are words (standing for strings of phonemes). Such a rule represents the regular relation which holds between all pairs of strings $\lambda u|v\rho$ and $\lambda w\rho$, for any strings λ and ρ, where $|$ marks word boundaries. Such rewrite rules are used for constructing the segmentation automaton described below in Section 6. We remark that the set of such rules, owing to their atomic character, is surprisingly large (we generate 2,790 such rules).

5. MORPHOLOGY

First of all, let us emphasize that we do not deal here with derivational morphology, neither primary (*kṛt*) nor secondary (*taddhita*) - we assume that corresponding entries are explicitly listed in the lexicon. We consider only inflexion (*vyaya*), which derives declensions of substantives and conjugation of finite verbal forms. The rationale is that derivational morphology is of interest mostly to semantic layers, that it is far from being as regular as inflexion, and that over-

generation would be hard to control. The price that we have to pay is that the lexicon is cluttered with quality substantives in *-tva* and *-tā*, agents in *-tṛ*, processes in *-na* and so on. On the verbal side, this obliges us to list derived conjugations (causative, desiderative, intensive) and verbal adjectives, nouns and adverbs (participles, infinitives, absolutives). On the other hand, we shall obtain compounds of any depth without the need to list them in the lexicon, since we consider compound analysis as a special case of segmentation.

The second remark is that we *generate* such inflected forms, using internal sandhi generation, as opposed to attempting to analyse them at the morpheme level. We thus avoid stemming, a complex process since it would involve internal sandhi analysis.

We entered the morphological laws for inflexion as paradigm tables. For instance, for nouns, adjectives and pronouns, each table maps a pair (number, case) into a set of possible suffixes. A dispatch switch takes as parameters a word stem and a gender, and determines the corresponding paradigm table. This information may now be used in several ways. First of all, when navigating in the hypertext form of the lexicon, one may click on gender annotations. Each such gender annotation is in the scope of exactly one stem form of the surrounding entry, giving the pair (stem, gender) of parameters of the dispatch switch, yielding a paradigm table. The paradigm table may be filled with the specific declensions, by computing the internal sandhi of the stem and the suffix. The corresponding table of inflected forms may then be displayed in the browser window, as an interactive grammatical aid. We may iterate this computation over all pairs (stem, gender) recorded in the lexicon, and this constructs a static list of all possible inflected forms. We record this list, as a mapping from such forms to the list of all morphological tags (stem, gender, number, case) which yield the given inflected form. From the 12,500 entries of our dictionary (out of which 8,000 are non-compound entries) we produced a total list of 150,000 inflected forms[1], using 88 paradigm tables.

The morphology of verbs is similar, but still more complex, be-

[1]This data is freely available for downloading as an XML resource (given with its validating DTD) at our site `http://sanskrit.inria.fr/`

cause of the many tenses, aspects, voices, and moods. The present system (present, imperfect, optative and imperative, at active, passive and middle voices) has been implemented for the 10 present classes, as well as the future (periphrastic and autonomous), perfect (reduplicating and periphastic), and aorist systems (with the 7 formation schemes of the latter). The derived conjugations (causative, intensive, and desiderative) are not constructed by generative paradigms, but entered in the lexicon on a by need basis. Each root is recorded with the sets of preverb sequences which it allows. These preverb annotations anchor hypertext pointers to the corresponding verb entries.

6. SEGMENTATION OF NOUN PHRASES

The first level of analysis of a Sanskrit written text is segmentation. Words are not separated by blanks and punctuation symbols, but are merged together with external sandhi, in a continuous phonemic stream which faithfully reflects the speech flow. Thus, recognition of written text is actually similar to speech processing, and segmentation must be solved in order to reveal the first level of the linear structure, that is the representation of a sentence as a list of words.

This process is inherently non-deterministic. For instance, in *Bhagavadgītā*, the verse chunk: नासतोविद्यतेभावः gets interpreted by Śaṅkara as the list of words (*padapāṭha*): *na asataḥ vidyate bhāvaḥ* (the unreal has no existence), while Madhva takes it as: *na asataḥ vidyate abhāvaḥ* (the 'asat' is not inexistent).[2] This is actually a not infrequent situation, since, for instance, long *ā* may be obtained by sandhi from the four mutual combinations of *a* and *ā*. Since *a* is the negation prefix for adjectives and *ā* is the frequent preverb meaning "towards the locutor", one guesses that this leads to many solutions, with possible ambiguities between a notion and its negation as above. Often ambiguities will not be resolved before the verification of syntax and even semantics. And sometimes several interpretations are legitimate, like in the example above, which means that an extra level

[2]Madhav Deshpande, Indology Internet forum communication, 2002.

of interpretation of the text may be necessary to infer its unambiguous meaning. And of course poets took advantage of this feature to write intentional double-entendre works.

We solve the segmentation problem using finite-state technology. A finite state machine expresses the regular relation between streams of words and the streams of phonemes issued from the (external) sandhi rewriting at their mutual junction, and a transducer implements the inverse relation. The precise constructions are documented in (Huet 2005), where we show consistency, completeness and convergence of the algorithm. A general toolkit 'Zen' based on finite state machines described as decorated tries was abstracted from this effort and released as free software (Huet 2002; Huet 2003a; Huet 2003b).

What the completeness property of our algorithm says for our Sanskrit segmentation problem is that a sentence s admits a finite number of decompositions $s_1\ s_2\ ...\ s_n$ where all the words s_i belong to the (inflected forms) lexicon, and (external sandhi) rewrite rules are applied between consecutive words. A segmentation transducer is compiled from the inflected forms lexicon and sandhi rules array which produces exactly this set of solutions.

A crucial requirement is a *non-overlapping* condition, which demands that no word is such that it has a prefix λ participating on its left to sandhi with its predecessor and a suffix ρ participating on its right to sandhi with its successor, with a non-empty overlap of λ and ρ. This is true of external sandhi in the general case, since sandhi contexts are short, if one excludes from the list of words very short particles such as the emphatic u from Vedic Sanskrit, whose occurrence is constrained anyway by prosody (in Vedic hymns, u usually occurs at the end of shlokas).

This segmenter is theoretically complete for any phrase built from words which are inflected forms from the root stems in the dictionary. Note, in particular, that compounds are analysed into their subcomponents using the same segmentation automaton, since external sandhi applies there too, and the inflected form generator also produces the stems used as left components of compounds. This assumes that the inflected forms of a compound $A \cdot B$ are of the

form $A \cdot B'$, where B' is an inflected form of B. However, a difficulty appears for *bahuvrīhi* (exocentric) usage of compounds, since this assumption may be violated by compounds where B admits a specific gender, but $A \cdot B$ used as an adjective in some other gender may admit extra forms. For instance, *bīja* (seed) is the stem of a neutral substantive, but *raktabīja*, the monster "whose blood regenerates", admits as nominative form *raktabījaḥ*, not obtainable as compound of *rakta-* with an inflected form of *bīja*. This difficulty is genuine, since we have here a choice between admitting *bahuvrīhi* usage of every compound (and thus opening the way to serious overgeneration problems), or else listing explicitly in the lexicon every *bahuvrīhi* attested usage, a possibly daunting task.

We opted in this segmenter for the second solution, and chose to record in the lexicon such *bahuvrīhi* usage of compounds. Actually, a preprocessing phase on the lexicon identifies all compounds which are irregular, in the sense of not being obtainable by external sandhi of the iic. form of their left component with a regular stem of their right component. This analysis adds 180 'autonomous' compound forms. This list includes *dvandva* compounds with double dual forms, such as *mitrāvaruṇau*, compounds whose left component uses a feminine form, such as *durgāpūjā*, or an inflected form, such as *vasuṃdhara*, irregular external sandhi such as *pṛṣodara*, or internal sandhi such as *rāmāyaṇa*. Their inclusion as autonomous nouns allows the generation of their forms, since our automaton is not able to get their analysis (as compounds) from their components. Besides these exceptional cases, all compounds formed with stems from the lexicon are analysable, down to any nesting level, without the need to have them explicitly listed as lexicon entries.

A kind of reverse difficulty occurs for *avyayībhāva* compounds such as *yathāśakti*, which ought to be recognized as an invariable adverb, although its rightmost component admits inflexion. This is a minor cause of overgeneration.

The main problem with our segmenter is an overgeneration of spurious sandhi solutions with small particle words such as *āt*, *ām*, *upa*, etc. and enclitic agent formation suffixes such as *-ad*, *-ga*, *-da*, *-pa*, *-ya*. Also, a word such as the substantive *āya* clashes badly with

dative suffixes. This overgeneration will have to be dealt with either by morphology (so that compounds in say *-ga* are generated on their own, without *ga* appearing by itself) or by prosody considerations.

Here is a simple typical example of segmentation:

```
Chunk: tacchrutvaa
may be segmented as:

 Solution 1 :
[ tat <t|'s -> cch>]
[ 'srutvaa <>]
```

The system indicates that the final *t* of *tat* (this) transforms itself by external sandhi with the initial *ś* of *śrutvā* (having heard) to form the phoneme sequence *cch*, leading to the correct analysis of *tacchrutvā* (having heard this). In *devanāgarī*: तद् | श्रुत्वा = तच्छ्रुत्वा.

7. TAGGING

Since the segmenter is lexicon directed, it may be easily extended into a tagger. All that is needed is to keep a lemmatization table, recording the generative origin of inflected forms. An economical computer solution for this problem, taking advantage of the regularities of morphology for sharing maximally the inverse morphological map, is provided by the structure of *differential words* (Huet 2005).

Now we obtain a tagging transducer, with two levels of nondeterminism, since a phrase may be segmented into different subwords, and each segment word may be obtained by several morphological constructions. Here is a simple example:

```
Chunk: me.saanajaa.m'sca
may be segmented as:

 Solution 1 :
[  me.saan
   {acc. pl. m.}[me.sa] <>]
[  ajaan
   {acc. pl. m.}[aja#1]|{acc. pl. m.}[aja#2]
```

```
   <n|c -> .m'sc>]
[  ca
   {und.}[ca] <>]

 Solution 2 :
[ maa
  {und.}[maa#2]|{acc. sg. *}[aham] <aa|i -> e>]
[ i.saan
  {acc. pl. m.}[i.sa] <>]
[ ajaan
  {acc. pl. m.}[aja#1]|{acc. pl. m.}[aja#2]
  <n|c -> .m'sc>]
[ ca
  {und.}[ca] <>]
```

The first solution is the correct one (sheep and goats), whereas the second one is parasitic. The ambiguity arising from the homonyms aja_1 (goat) and aja_2 (unborn) is duly recorded, so that each segment stem points unambiguously to one of the lexicon entries.

For larger chunks, overgeneration may lead to literally thousands of solutions. This indicates that guidance from further layers (syntax and semantics) will be ultimately needed in order to reduce the set of solutions to manageable sizes.

8. SEGMENTATION OF VERB PHRASES

The next challenge was to analyse verb phrases. This involves several new difficulties. The first one was to build the morphology module for root declensions. In order to have deterministic internal sandhi, the phonemes *j* and *h* had to be partitioned into two variants each. Thus, for instance, *j* | *t* = *kt* (e.g. *yukta*) whereas *j'* | *t* = *ṣṭ* (e.g. *mārṣṭi*). Similarly *h* | *t* = *ḍh* (e.g. *leḍhi*) whereas *h'* | *t* = *gdh* (e.g. *dugdha*). These distinctions exhibit some of the Indo-European substrate of Sanskrit roots.

The second problem concerns the modelling of preverb prefixing. The natural idea would be to affix preverbs to conjugated verb forms, starting at roots, and to store the corresponding inflected forms along

with the declined nouns. But this is not the right model for Sanskrit verbal morphology, because preverbs associate to root forms with *external* and not *internal* sandhi (probably owing to their origin as postpositions). And putting preverbs in parallel with root forms and noun forms will not work either, because the non-overlapping condition mentioned above fails for the preverb *ā*. And this overlapping actually makes external sandhi non-associative. For instance, noting sandhi with the vertical bar, we get: (*iha* | *ā*) | *ihi* = *ihā* | *ihi* = *ihehi* (come here). Whereas: *iha* | (*ā* | *ihi*) = *iha* | *ehi* = **ihaihi*, incorrect. This seems to definitely doom the idea of storing conjugated forms such as *ehi*.

The proposed solution to this problem is to prepare special root forms prefixed by *ā* in the case where the root forms starts with *i* or *ī* or *u* or *ū* - cases where a non-associative behaviour of external sandhi obtains. But instead of applying the standard sandhi rule *ā* | *i* = *e* (and similarly for *ī*) we use *ā* | *i* = **e* where **e* is a *phantom* phoneme which obeys special sandhi rules such as: *a* | **e* = *e* and *ā* | **e* = *e*. Through the use of this phantom phoneme, overlapping sandhis with *ā* are dealt with correctly. Symmetrically we introduce another phantom phoneme **o*, obeying e.g. *ā* | *u* = **o* (and similarly for *ū*) and *a* | **o* = *ā* | **o* = *o*. This methodology is explained in (Huet 2003c).

The combination of three automata, one for nouns, one for preverb sequences, and one for root forms (plus the special forms with phantom phonemes) is designed to segment (and tag) simple verbal sentences, where a number of noun phrases is followed by a finite verb form.

It remains to explain what forms are to be entered in the preverbs automaton. We could, of course, just enter individual distinct preverbs, and allow looping in the preverbs phase. But this would be grossly over-generating. At the other extreme, we could record in the lexicon the preverb sequences used with a given root. But then, instead of one roots forms automaton, we would have to use many different automata (at least one for every equivalence class of the relation "admits the same preverb sequences"). We propose a middle way, where we have one preverbs automaton storing all the preverb

sequences used for at least one root. Namely: *ati*, *adhi*, *adhyava*, *anu*, *anuparā*, *anupra*, *anuvi*, *antaḥ*, *apa*, *apā*, *api*, *abhi*, *abhini*, *abhipra*, *abhivi*, *abhisam*, *abhyā*, *abhyud*, *abhyupa*, *ava*, *ā*, *ud*, *udā*, *upa*, *upani*, *upasam*, *upā*, *upādhi*, *ni*, *niḥ*, *nirava*, *parā*, *pari*, *parini*, *parisam*, *paryupa*, *pi*, *pra*, *prati*, *pratini*, *prativi*, *pratisam*, *pratyā*, *pratyud*, *prani*, *pravi*, *pravyā*, *prā*, *vi*, *vini*, *viniḥ*, *viparā*, *vipari*, *vipra*, *vyati*, *vyapa*, *vyava*, *vyā*, *vyud*, *sa*, *saṃni*, *saṃpra*, *saṃprati*, *saṃpravi*, *saṃvi*, *sam*, *samava*, *samā*, *samud*, *samudā*, *samudvi*, *samupa*.

We remark that preverb *ā* only occurs last in a sequence of preverbs, i.e. it can occur only next to the root form. Thus we do not have to augment the preverbs sequences with phantom phonemes.

Here is an example of tagging the tricky sentence discussed above:

```
Chunk: ihehi
may be segmented as:

 Solution 1 :
[ iha
  {und.}[iha] <a|aa|i -> e>]
[ aa|ihi
  {imp. a. sg. 2}[aa-i] <>]

 Solution 2 :
[ iha
  {und.}[iha] <a|i -> e>]
[ ihi
  {imp. a. sg. 2}[i] <>]
```

The first solution is the correct one (“come here”). The second one is a homophone without the *ā* preverb, corresponding to a non-correct “go here”. Also **ihaihi* is righly rejected as having no solution.

Remark 1. This exceptional treatment of the *ā* preverb corresponds to a special case in Pāṇini as well, who uses a similar device with a special mark after the preverb ā. This indicates that our approach is legitimate. The importance of our contribution is to show that this *generative* mechanism is also adequate for *analysis*, since it allows

us to regain the nonoverlapping condition needed for correct non-deterministic prediction.

Remark 2. It is easy to adapt the model to special cases and exceptions. For instance, preverb *adhi* combines with middle form *ye* of root *i* to form *adhīye* 'I study', although the usual sandhi rule between final *i* and initial *y* yields *iy*, predicting the wrong form **adhiye*. But our machinery allows to enter a specific sandhi rule *adhi*|*y*→ *adhīy*, and then *adhīye* may be correctly analysed:

```
Chunk: adhiiye
may be segmented as:

Solution 1 :
[ adhi <i|y -> iiy>]
[ ye
  {pr. m. sg. 1}[adhi-i] <>]
```

Many more special sandhi cases have to be properly dealt with in our sandhi platform, some of which being dependent on grammatical information (such as special sandhi for dual person forms) or specific to certain words (such as the 3rd person pronoun *sa*). Also, it is expected that the three-automata model will need to be revised with other components, for instance to recognize properly the use of auxiliary verb forms with substantive stems in *ī*, such as *nimittī kṛ* or *sajjī bhū*. Also a finer analysis of the use of preverbs with a given root may avoid generating spurious forms (often only one voice is used with a given preverb sequence for a given root).

Here is a non-pathological typical example of the analysis of a small sentence ("cat drinks milk"):

```
Chunk: maarjaarodugdha.mpibati
may be segmented as:

 Solution 1 :
[ maarjaaras
  {nom. sg. m.}[maarjaara] <as|d -> od>]
[ dugdham
  {acc. sg. m.|acc. sg. n.|nom. sg. n.
  |voc. sg. n.}[dugdha] <m|p -> .mp>]
```

```
[ pibati
  {pr. a. sg. 3}[paa#1] <>]
```

In the Web interface of the lexicon, the above tag sequence anchors hypertext pointers to the dictionary entries of the successive lexemes *mārjāra, dugdhā* and $pā_1$.

9. TUNING AND LEARNING

At the time of printing, morphology generation has been extended to participles (past passive and active, present active middle and passive, future active middle and passive, the last one having 3 possible forms), and to infinitives and absolutives (in *-tvā* and in *-ya*). From 525 roots, we generate an inflected forms database of 98265 root finite forms, and 204657 participial forms, which in addition to the 128057 noun forms obtained from the lexicon, give a total of about 431 000 forms. The segmentation process uses a total of 9 automata, modelling correctly compound formation, periphrastic forms (in -ī), affixing of preverbs, and forms which may appear only as right component of a compound.

Overgeneration is still problematic, and simple sentences may generate thousands of "solutions", most of them ungrammatical or nonsensical. It is clear that extra syntactic processing will be necessary to filter out most of these candidate solutions, as explained in the next section. Only then shall we be able to start processing a real corpus, as opposed to toy isolated examples. In order to trim spurious solutions, and rank the remaining ones in decreasing order of relevance, before parasitic ones, training of the stochastic automaton will have to take place.

Then a robust version of the segmenter, together with a lemmatizer, should provide a mechanism for lexicon acquisition. This way we may hope to tune our tagger to become a useful preprocessor for scholars, in such a way that fully tagged critical editions may be prepared with computer assistance. We should also allow the automated computation of concordance indexes, as well as various statistical analyses which are unfeasible at present.

10. PARSING

The next step in processing is the verification of government constraints. Verbal valencies (subcategorisation patterns) must be saturated by the available nominal cases of the current phrase (or rather their semantic rôle or *kāraka*). This constraint processing phase will trim out solutions which do not satisfy this requirement. For instance, in the example sentence above, we obtain from the fact that *pā* is a transitive verb the constraint that is requires a subject and an object. The voice being active, this means we need one nominative tag and one accusative tag. The unique solution verifying this constraint is to take *mārjāraḥ* as the subject and *dugdham* as the object.

In the same syntactic process, we shall group compounds, and attempt to unify segments by agreement, in order to refine chunks into a number of concording noun phrases. The interpretation of genitives will distinguish between object genitives which fulfill a genitive role in the verb valency, and between attributive genitives which operate as noun phrase complements.

A difficulty is expected from verbs expecting two accusatives, since the partition of the accusative chunks into the two *kāraka* roles – typically the learning and the learnt in verbs such as *paṭh* (to learn) – is likely to involve semantic features (such as animate versus inanimate). This will involve incorporating into the lexicon an ontology mapping, a standard apparatus in today's natural language treatment platforms. Traditions such as Indian semiotics (*navyanyāya*) may be applied to this semantic modeling.

A major difficulty will be to recognize dislocations and long-distance dependencies, with a penalty cost severe enough to defeat potentially exponential overgeneration. This parsing methodology, in constrast with traditional transformational frameworks, is close to the dependency grammars model. The link between dependency grammars and the Paninian *kāraka* theory was noticed by Bharati, Chaitanya and Sangal (Bharati et al. 1993; Bharati et al. 1995).

We may hope to tune our syntax analyser by using as training data a tree bank constructed over the years by Brendan Gillon (Gillon 1996), who used as corpus a list of typical sentences from Apte's

treatise on Sanskrit syntax (Apte 1885).

Further layers, dealing with discourse structure (such as anaphora resolution), rhetorics and semantics, are at this point rather remote and speculative, not only for Sanskrit, but for natural language in general.

11. CONCLUSION

What we presented in this paper is the state of the art of our Sanskrit reader; it is able to segment and tag simple sentences. The reader will have to be tested on real corpus, firstly on easy texts for which the current lexicon is complete. This will allow statistical training, aiming at listing the possible solutions in decreasing plausibility. A robust version will then be developed, in order to serve as a lexicon acquisition tool, for corpus for which the lexicon may not be complete.

At this point we should be ready to consider doing for Sanskrit what Perseus[3] offers for the classical Latin and Greek corpus.

REFERENCES

APTE, V. S. 1885. *The Student's Guide to Sanskrit Composition. A Treatise on Sanskrit Syntax for Use of Schools and Colleges*. Poona: Lokasamgraha Press.

BHARATI, Akshar & Vineet CHAITANYA & Rajiv SANGAL 1995. Natural Language Processing, a Paninian Perspective. Delhi: Prentice-Hall.

BHARATI, Akshar & Rajiv SANGAL 1993. Parsing Free Word Order Languages in the Paninian Framework. Proceedings, 31st conference of Association for Computational Linguistics, Columbus, Ohio, 1993: 105–111.

GILLON, Brendan S. 1996. Word Order in Classical Sanskrit. *Indian Linguistics* 57(1): 1–35.

[3] `http://www.perseus.tufts.edu/`

HUET, Gérard 2000. Structure of a Sanskrit Dictionary. Technical report, **INRIA**. `http://pauillac.inria.fr/~huet/PUBLIC/Dicostruct.ps`

—— 2001. From an Informal Textual Lexicon to a Well-structured Lexical Database: An Experiment in Data Reverse Engineering. In: *Working Conference on Reverse Engineering (WCRE'2001)*, **IEEE**: 127–135.

—— 2002. The Zen Computational Linguistics Toolkit. Technical report, **ESSLLI** Course Notes. `http://pauillac.inria.fr/~huet/ZEN/esslli.pdf`

—— 2003a. Zen and the Art of Symbolic Computing: Light and Fast Applicative Algorithms for Computational Linguistics. In: *Practical Aspects of Declarative Languages (PADL) symposium*. `http://pauillac.inria.fr/~huet/PUBLIC/padl.pdf`

—— 2003b. Automata Mista. In: N. Dershowitz (ed.), *Festschrift in Honor of Zohar Manna for his 64th anniversary*, Springer-Verlag LNCS vol. 2772: 359–372. `http://pauillac.inria.fr/~huet/PUBLIC/zohar.pdf`

—— 2003c. Towards Computational Processing of Sanskrit. In: *Proceedings, International Conference on Natural Language Processing (ICON-2003)* : 40–48. Mysore, Central Institute of Indian Languages.

—— 2005. Transducers as Lexicon Morphisms, Phonemic Segmentation by Euphony Analysis, Application to a Sanskrit Tagger. To appear in: *Journal of Functional Programming*. `http://pauillac.inria.fr/~huet/PUBLIC/tagger.pdf`.

INRIA = Institut National de Recherche en Informatique et en Automatique.
IEEE = Institute of Electrical and Electronics Engineers.
ESSLLI = European Summer School in Logic, Language and Information.