Towards Computational Processing of Sanskrit

Gérard Huet
INRIA-Rocquencourt,
BP 105, 78153 Le Chesnay CEDEX France

Abstract

We present in this paper recent progress in Sanskrit computational linguistics. We propose a solution to the tagging of verb phrases which correctly handles the non-associativity of external sandhi arising from the treatment of preverb *ā*. This involves a notion of *phantom phoneme*. We give exemples of execution of a computer tagger designed accordingly, which handles verb phrases as well as noun phrases.

Introduction

The corpus of Latin and Greek classics has been processed into an hypertext database of fully tagged documents, where every word may be looked up, with its morphological analysis, in such a way that the proper lexical entry is indexed. The resulting Digital Library Perseus, at site `http://www.perseus.tufts.edu/`, is a wonderful piece of scholarly expertise, presented with the latest technological tools. Unfortunately, there does not exist at present a similar facility for Sanskrit, partly because of the much larger size of the corpus, and partly because the proper computational linguistics tools have not been developed yet. The closest related effort is the Brown Sanskrit Library, at `http://sanskritlibrary.org/`, where a restricted corpus has been manually tagged by Pr. Peter Scharf [Scharf, 2003]. The present paper presents preliminary steps to fill this need.

1 Layers of linguistic modelling

We recall that linguistic models layer the representation of natural languages into submodels as follows

- 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)

This determines the overall architectural design of a computational linguistics platform. Of course the layers are not independent, for instance concord from syntax must be consistent with morphological features; also euphony makes 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 flexed forms lexicon) 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 interferences, 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 flexed words (internal sandhi) and at the level of shallow syntax for the glueing of words into sentences (external sandhi), leading conversely to a complex segmentation problem. This places the first serious difficulty in the mechanical processing of Sanskrit: either when processing a phoneme stream issued from a speech recognition module, or when processing a written sentence - the problems are basically the same - how to guess the sequence of words which by external sandhi rewriting would yield the given stream of phonemes. This sandhi decomposition is by no means unique, and actually only semantic considerations will ultimately curb down the overgeneration of this non-deterministic analysis.

2 Lexicon

This leads us to our first requirement:
Requirement 1. Segmentation is lexicon-directed.

This requirement needs further analysis. Is segmentation guided by a flexed forms lexicon (and then only external sandhi needs to be analysed) or is it guided just by a roots lexicon (in which case lemmatisation must be combined with segmentation, and then both external and internal sandhi must be inverted) ? We advocate the first method, which is simpler to implement, since flexed forms generation only needs internal sandhi synthesis, as opposed to analysis, the gain being that the process is basically deterministic. So we only need *in fine* a roots lexicon, the flexed forms lexicon being computed by morphology generation, with internal sandhi synthesis, whereas segmentation, directed by the flexed forms lexicon, only needs to invert external sandhi.

Now we understand clearly what is the first linguistic resource needed for Sanskrit processing: a

lexical data base of root words, with enough grammatical information to serve as specification for a morphology generator.

As usual in computational linguistics endeavours, the availability of linguistics resources ia a prerequisite to any real-scale processing. The first attempt at digitalizing an existing Sanskrit dictionary was realized in 1995 by Peter Schreiner from Universität Zürich with the optical scanning of Mylius' dictionary. Unfortunately, the publisher of this work, still protected by copyright, did not consent to its free dissemination, and so no further computer processing of this dictionary was possible. More recently, the indology team at Köln University completed the optical scanning of the Monier-Williams dictionary (170000 entries); furthermore, an HTML version issued from it was realized by R. B. Mahoney (`http://homepages.comnet.co.nz/~r-mahoney/mw_dict/mw_dict.html`), opening the way to a public-domain XML lexical database of significant coverage.

The author took another route, building upon a Sanskrit to French dictionary which he started from scratch in 1994. This dictionary reached the 10000 entries in 2000, and it then made sense to understand how to adapt it as a lexical database for linguistic processing. The source of the dictionary was reverse engineered into a structured lexical database [Huet, 2000; 2001], and the author started designing a set of tools for further computer processing as an experiment in the use of functional programming for a computational linguistics platform.

3 Morphology

The second linguistic resource needed at this point was a grammar for morphology derivations, basically declensions of substantives/adjectives, and conjugations of verbs. A morphology generator for nouns was realised, and presented at the XIth Sanskrit International Conference in Torino in April 2000. It used as subroutine an internal sandhi generator. These operations were specified as tables, defined by information gathered from various Western grammars [Whitney, 1924; Renou, 1984; Gonda, 1966].

An alternative would be of course to base the corresponding tables on the traditional approach of Pāṇini. But it should be noted that a proper modular design of the computational platform does not commit to the method of development of the linguistic data, and thus computer tools developed with grammar data issued from western work of modern linguists may be reused with grammar descriptions developed with the traditional approach. The generative grammar of Pāṇini is actually being compiled as a computer program by Dr Shivamurthy Swamiji [Swamiji, 2003].

Once the sandhi synthesis module and the declension paradigms module were finalized, a morphology generator for nouns could be obtained, and its iteration on the stem dictionary produced a lexicon of flexed forms. In 2001, from the 12000 entries of the dictionary (out of which 8000 are non-compound entries) we produced a list of 130000 flexed forms[1]. This completed requirement 1, at least for nominal phrases, and work on segmentation could proceed.

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

4 Segmentation of nominal phrases

External sandhi is easily describable in terms of string rewriting (over the alphabet of the 50 phonemes of Sanskrit). In language theory terms, it is a rational relation, and its analysis as well as its synthesis may be implemented in terms of finite state transducers [Koskenniemi, 1984; Kaplan and Kay, 1994; Roche and Schabes, 1997; Karttunen, 2000; Laporte, 1995]. An original method for implementing such transducers was designed, and a general toolkit Zen based on finite automata described as decorated tries was released as free software in 2002 [Huet, 2002b; 2002c; 2003b; 2003a].

In particular, a compiler of monoid rewrite rules into a segmentation transducer was produced, and its consistency, completeness and convergence were formally established [Huet, 2002a]. We extract from this work the statement of its main theorem:
Theorem. If the lexical system (L, R) is strict and weakly non-overlapping, s is an *(L,R)-sentence* iff the algorithm (*segment_all* s) returns a solution; conversely, the set of all such solutions exhibits all the proofs for s to be an *(L,R)-sentence*.

What this 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 (flexed forms) lexicon L, and (external sandhi) rewrite rules from the set R are applied between consecutive words. Furthermore a transducer is compiled from L and R which produces exactly this set of solutions.

A crucial requirement is the *non-overlapping* condition above, which demands that no word in L 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, 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 anyway is constrained by prosody (in Vedic hymns, u usually occurs at the end of shlokas).

On these bases, a segmentation program for nominal phrases was produced, and made available in 2002 through a common gateway interface (CGI) at site `http://pauillac.inria.fr/~huet/SKT/`.

This segmenter is theoretically complete for any nominal phrase built from words which are flexed forms form the root stems in the dictionary. Note in particular that compounds are analysed from their subcomponents using the same segmentation automaton, since external sandhi applies there too, and the flexed form generator produces the stems used as left components of compounds. This assumes that flexed forms of a compound $A\ B$ are of the form $A\ B'$, where B' is a flexed 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 definite gender, but $A\ B$ used as an adjective in some other gender may admit extra flexed forms. For instance, *bīja* (seed) is the stem of a neutral substantive, but *raktabīja*, the monster "whose blood is regenerated" admits as nominative form *raktabījaḥ*, not obtainable as compound of *rakta-* with a flexed 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 explictly in the lexicon every *bahuvrīhi* attested usage, a possibly daunting task.

We opted in his segmenter for the second solution, explicitated by the following requirement:

Requirement 2. The lexicon ought to record *bahuvrīhi* usage of compounds.

The main problem with this 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 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 exemple of the segmenter:

```
Chunk: tacchrutvaa
may be segmented as:

 Solution 1 :
[  tad with sandhi d|"s -> cch]
[  "srutvaa with no sandhi]
```

The system indicates that the final *d* of *tad* (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).

5 Tagging

Since the segmenter is lexicon directed, it may be easily extended into a tagger. All that is needed is to keep a stemming module, inverting the flexed forms lexicon. An economical 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, 2002a; 2002c].

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

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

 Solution 1 :
[  me.saan
 < { acc. pl. m. }[me.sa] >  with no sandhi]
[  ajaan
 < { acc. pl. m. }[aja#1] | { acc. pl. m. }[aja#2] >
   with sandhi n|c -> "m"sc]
[  ca
 < { und. }[ca] >  with no sandhi]

 Solution 2 :
[  maa
 < { und. }[maa#2] | { acc. sg. * }[aham] >
   with sandhi aa|i -> e]
```

```
[  i.saan
 < { acc. pl. m. }[i.sa] >  with no sandhi]
[  ajaan
 < { acc. pl. m. }[aja#1] | { acc. pl. m. }[aja#2] >
   with sandhi n|c -> "m"sc]
[  ca
 < { und. }[ca] >  with no sandhi]
```

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.

6 Segmentation of verb phrases

The next challenge was to analyse verb phrases. Here this involves several new difficulties. The first one is to build a morphology module for root declensions. This is a non-trivial task, since Sanskrit verbal morphology is rather complex. At present, a simplified prototype implements the present system of thematic roots.

The second problem is 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 flexed 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. And putting preverbs in parallel with root forms and noun forms will not work either, because the non-overlapping condition mentioned above fails for e.g. preverb $\bar{a}$. And this overlapping actually makes external sandhi non associative. For instance, noting sandhi with the vertical bar, we get: $(iha \mid \bar{a}) \mid ihi = ih\bar{a} \mid ihi = ihehi$ (come here). Whereas: $iha \mid (\bar{a} \mid ihi) = iha \mid ehi = {}^*ihaihi$, 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 $\bar{a}$-prefixed root forms in the case where the root forms starts with i or $\bar{\imath}$ or u or $\bar{u}$ - the cases where a non-associative behaviour of external sandhi obtains. But instead of applying the standard sandhi rule $\bar{a} \mid i = e$ (and similarly for $\bar{\imath}$) we use $\bar{a} \mid i = {}^*e$ where $^*e$ is a *phantom* phoneme which obeys special sandhi rules such as: $a \mid {}^*e = e$ and $\bar{a} \mid {}^*e = e$. Through the use of this phantom phoneme, overlapping sandhis with $\bar{a}$ are dealt with correctly. Similarly we introduce another phantom phoneme $^*o$, obeying e.g. $\bar{a} \mid u = {}^*o$ (and similarly for $\bar{u}$) and $a \mid {}^*o = \bar{a} \mid {}^*o = o$.

Now we propose to model the recognition of verb phrases built from a sequence of noun phrases, a sequence of preverbs, and a conjugated root form by a cascade of segmenting automata, with an automaton for nouns (the one demonstrated above), an automaton for (sequences of) preverbs, and an automaton for conjugated root forms augmented with phony forms (i.e. $\bar{a}$ prefixes using phantom phoneme sandhi). The sandhi prediction structure which controls the automaton is decomposed into

three phases, Nouns, Preverbs and Roots. When we are in phase Nouns, we proceed either to more Nouns, or to Preverbs, or to Roots, except if the predicted prefix is phony, in which case we proceed to phase Root. When we are in phase Preverbs, we proceed to Verbs, except if the predicted prefix is phony, in which case we backtrack (since preverb *ā* is accounted for in Preverbs). Finally, if we are in phase Roots we backtrack.

It remains to explain what forms to enter 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, nis, 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. This justifies not having to augment the Preverbs sequences with phantom phonemes.

Here is an exemple of tagging a verb phrase which exploits all the machinery:

```
Chunk: ihehi
may be segmented as:

 Solution 1 :
[  iha
 < { und. }[iha] >  with sandhi a|aa|i -> e]
[  aa|ihi
 < { imp. sg. 2 }[i#1] >  with no sandhi]

 Solution 2 :
[  iha
 < { und. }[iha] >  with sandhi a|i -> e]
[  ihi
 < { imp. sg. 2 }[i#1] >  with no sandhi]
```

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. 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. We profit of this occasion to point out that the *ā* preverb always occurs last in the preverbs sequence, an observation which to our knowledge is not made by Pāṇini.

This segmentation algorithm for verb phrases is the main contribution of this paper. Here is a non-pathological typical example of the analysis of a small sentence ("cat eats milk"):

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

 Solution 1 :
[  maarjaaras
 < { nom. sg. m. }[maarjaara] >  with sandhi as|d -> od]
[  dugdham
 < { acc. sg. m. | acc. sg. n. | nom. sg. n. | voc. sg. n. }
     [dugdha] > with sandhi m|p -> .mp]
[  pibati
 < { pr. sg. 3 }[paa#1] >  with no sandhi]
```

7 Tuning and Learning

As we already said, the full morphology of verb forms is yet to be completed. Then we shall 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 such a way that correct solutions appear before parasitic ones, some training of the automaton, using preference weights, will have to take place. One possibility would be to use for this purpose the manually tagged corpus of Peter Scharf's Sanskrit Library, and specially his *Rāmopākhyāna*.

Then a robust version of the segmenter, together with a lemmatizer, should provide a mechanism by which we would be able to bootstrap the core lexicon to fuller lexicons complete for a given corpus. 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. At this point we shall be ready to consider doing for Sanskrit what Perseus offers for the classical Greek corpus.

8 Parsing

Now we give more speculative thoughts in order to sketch plausible further developments. The next step will be to check the possible valencies (subcategorization patterns) of the verb in order to determine whether the nouns cases saturate it. This constraint processing phase will trim out solutions which do not satisfy this requirement (and remove from solutions which do satisfy it the spurious cases). In the same phase, we shall group compounds, and attempt to match segments by concord, 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 at the noun phrase level.

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 adding to the lexicon an ontology mapping, a quite standard apparatus in today's natural language treatment

platforms. However, we would like the ontology to be minimal in the sense of being *required* for such disambiguations, so to speak, as opposed to be generated out of introspection or other non-corpus consideration, such as theoretical models of conceptual structures. On the other hand, traditions such as indian semiotics (*navyanyāya*) may be put to use for this semantic modeling.

Then more complex sentences must be accounted for. For instance, the common pattern of successions of absolutives preceding a final finite verb form. We view such absolutives as *sentinels* in Pr. Narayana Murthy's linguistic model [Murthy, 1995]: they mark the transition between the linear structure and the hierarchical structure of a sentence. We may hope to tune our syntax analyser by using as training data a tree bank constructed over the years by Brendan Gillon [Gillon, 1995], who used as corpus 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.

9 Conclusion

We presented a notion of *phantom phonemes* which permits to restore associativity of juncture rewriting phonemic relations such as external sandhi. This allows us to use our predictive analyser for the segmentation of Sanskrit verb phrases, and thus to mechanically tag simple sentences. This opens the way to the complete analysis of the linear structure of a Sanskrit sentence.

References

[Apte, 1885] Vāman Shivarām Apte. *The Student's Guide to Sanskrit Composition. A Treatise on Sanskrit Syntax for Use of Schools and Colleges.* Lokasamgraha Press, Poona, India, 1885.

[Gillon, 1995] Brendan S. Gillon. Word order in classical sanskrit. Private communication, 1995.

[Gonda, 1966] Jan Gonda. *A concise elementary grammar of the sanskrit language (Tr. Gordon B. Ford Jr).* E. J. Brill, Leiden, 1966.

[Huet, 2000] Gérard Huet. Structure of a Sanskrit dictionary. Technical report, INRIA, 2000. `http://pauillac.inria.fr/~huet/PUBLIC/Dicostruct.ps`

[Huet, 2001] Gérard Huet. 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)*, pages 127–135. IEEE, 2001.

[Huet, 2002a] Gérard Huet. Transducers as lexicon morphisms, phonemic segmentation by euphony analysis, application to a sanskrit tagger. `http://pauillac.inria.fr/~huet/PUBLIC/tagger.pdf`, 2002.

[Huet, 2002b] Gérard Huet. The Zen computational linguistics toolkit. Technical report, ESSLLI Course Notes, 2002. `http://pauillac.inria.fr/~huet/ZEN/esslli.pdf`

[Huet, 2002c] Gérard Huet. The Zen computational linguistics toolkit: Lexicon structures and morphology computations using a modular functional programming language. In *Tutorial, Language Engineering Conference LEC'2002*, 2002.

[Huet, 2003a] Gérard Huet. Automata mista. In Nachum Dershowitz, editor, *Festschrift in Honor of Zohar Manna for his 64th anniversary*. Springer-Verlag LNCS vol. 2772, 2003. `http://pauillac.inria.fr/~huet/PUBLIC/zohar.pdf`

[Huet, 2003b] Gérard Huet. Zen and the art of symbolic computing: Light and fast applicative algorithms for computational linguistics. In *Practical Aspects of Declarative Languages (PADL) symposium*, 2003. `http://pauillac.inria.fr/~huet/PUBLIC/padl.pdf`

[Kaplan and Kay, 1994] Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20,3:331–378, 1994.

[Karttunen, 2000] Lauri Karttunen. Applications of finite-state transducers in natural language processing. In *Proceedings, CIAA-2000*, 2000.

[Koskenniemi, 1984] K. Koskenniemi. A general computational model for word-form recognition and production. In *10th International Conference on Computational Linguistics*, 1984.

[Laporte, 1995] Eric Laporte. Rational transductions for phonetic conversion and phonology. Technical report, IGM 96-14, Institut Gaspard Monge, Université de Marne-la-Vallée, 1995. Also in [Roche and Schabes, 1997].

[Murthy, 1995] K. Narayana Murthy. *Universal Clause Structure Grammar*. PhD thesis, Hyderabad University, 1995.

[Renou, 1984] Louis Renou. *Grammaire sanscrite*. Adrien Maisonneuve, Paris, 1984.

[Roche and Schabes, 1997] Emmanuel Roche and Yves Schabes. *Finite-State Language Processing*. MIT Press, 1997.

[Scharf, 2003] Peter Scharf. *Rāmopākhyāna - the Story of Rāma in the Mahābhārata. An independent-study Reader in Sanskrit*. RoutledgeCurzon, London, 2003.

[Swamiji, 2003] Shivamurthy Swamiji. *Ganakashtadhyayi*. `http://www.taralabalu.org/panini/shabdarupa/`, 2003.

[Whitney, 1924] William Dwight Whitney. *Sanskrit Grammar*. Leipzig, 1924. 5th edition.