Computational Linguistics for Sanskrit: a Software Engineering Approach

March 2001

Gérard Huet

INRIA-Rocquencourt

Introduction

This note sketches general principles for a software architecture model of natural language processing, illustrated by an experiment on building tools for the computer processing of sanskrit.

This experiment started from a sanskrit-to-french elementary lexicon developed by the author over several years. At some point the need was felt for the systematic design of more general tools, using modern computer technology. The original lexicon was reversed-engineered into a data base with a precise format, and it became the center of a variety of tools. We shall briefly describe here the various components of a proposal for generic linguistic tools, inspired from this initial experiment.

The model has basically three components. The first one is a lexicographic data base. The second one is a corpus data base. The third one is a text analysis interpreter. Various other modules act as service components: finite state transducers of various kinds, morphology processors, constraint satisfaction engines, proof search tools, etc. Details of control flow between the various components will depend according to the task aimed at: text annotation, discourse management, translation, etc.

The tools which are currently available, and which are all compiled mechanically from the lexical data base, comprise the following:

- a high-quality printable 320 pages sanskrit-to-french dictionary, with entries listed both in *devanāgarī* and romanisation with diacritics
- a web site giving an hypertext view of the same document

- various search tools of this web site, notably an index engine returning all entries (and the declensions thereof) matching a given prefix, a simplified index where diacritics may be omitted from the query, and a grammatical engine computing declensions of a given stem.

1 The lexicon

The lexical data base is the central component. For our sanskrit processor, the initial version of the data base was created by reverse engineering from a text lexicon developed over the years by the author. This was accomplished by writing a parser for the TEX input lexical conventions, and by turning TEX macros into abstract syntax operators. A detailed description of this abstract syntax is available as a research report[8], and will not be repeated here. We shall just point to salient points of this abstract structure when needed.

1.1 The grinder

The main tool used to extract information from this data-base is called the *grinder* (named after the corresponding core processor in the WordNet effort[5]). The grinder is a parsing process, which recognizes successive *items* in the data base, represents them as an abstract syntax value, and for each such item calls a *process* given as argument to the grinder. In other words, `grind` is a parametric module, or *functor* in ML terminology. Here is exactly the module type `grind.mli` in the syntax of our ML variant:

```
module Grind : functor(Process:Proc.Process_signature)->sig end;
```

with interface module `Proc` specifying the expected signature of a `Process`:

```
module type Process_signature = sig
value process_header : (Sanskrit.skt * Sanskrit.skt) -> unit;
value process_entry : Dictionary.entry -> unit;
value prelude : unit -> unit;
value postlude : unit -> unit;
end;
```

That is, there are two sorts of items in the data base, namely headers and entries. The grinder will start by calling the process `prelude`, will process every header with routine `process_header` and every entry with routine `process_entry`, and will conclude by calling the process `postlude`. Module interface `Dictionary` describes the dictionary data structures used for

representing entries (i.e. its abstract syntax as a set of ML datatypes), whereas module `Sanskrit` holds the private representation structures of sanskrit nouns (seen from `Dictionary` as an abstract type, insuring that only the sanskrit lexical analyser may construct values of type `skt`).

A typical process is the printing process `Print_dict`, itself a functor. Here is its interface:

```
module Print_dict : functor(Printer:Print.Printer_signature)
                          -> Proc.Process_signature;
```

It takes as argument a `Printer` module, which specifies low-level printing primitives for a given medium, and defines the printing of entries as a generic recursion over its abstract syntax. Thus we may define the typesetting primitives to generate a TEX source in module `Print_tex`, and obtain a TEX processor by a simple instanciation:

```
module Process_tex = Print_dict(Print_tex);
```

Similarly, we may define the primitives to generate the HTML source of a Web document, and obtain an HTML processor `Process_html` as:

```
module Process_html = Print_dict(Print_html);
```

It is very satisfying indeed to have such sharing in the tools that build two ultimately very different objects, a book with professional typographical quality on one hand, and a Web site fit for hypertext navigation and search on the other hand.

1.2 Structure of entries

Entries are of two kinds: cross-references and entries proper. Cross references are used to list alternative spellings of words and some irregular but commonly occurring flexed forms (typically pronoun declensions). Proper entries consist of three components : syntax, usage, and an optional list of cognate etymologies in other languages.

The syntax component consists itself of three sub-components: a heading, a list of variants, and an optional etymology. The heading spells the main stem (in our case, the so-called weak stem), together with a hierarchical level. At the top of the hierarchy, we find root verbs, non-compound nouns, suffixes, and occasional declined forms which do not reduce to just a cross reference, but carry some usage information. Then we have subwords, and subsubwords, which may be derived forms obtained by prefixing or suffixing their parent stem, or compound nouns.

Compound words are specially common in sanskrit, and it would be absurd to merge them all in one big alphabetic list. Thus they are usually grouped under the entry of their left component (which may itself be a compound, and so on, until we reach a root word). Sanskrit dictionaries may push more or less far this structural point of view, making their actual use more or less easy for beginners, since the alphabetic ordering of compounds may be further complicated by phonetic glueing ('sandhi') at the junction of the components. We shall not develop further this point here, but remark that sometimes the first component is merely a comment on the second component. This is true in particular of compounds starting with a numeral, indicating a classification of the second component into several subsorts. Thus the 'four noble truths' by which the tradition explains the teachings of Buddha should better be listed in the compound 'noble truth', where they constitute a detailed explanation of this notion, rather than under the numeral 'four', under which it is hopeless to list all such four-fold classifications. Thus we reserved a special kind of entry for such specific left-forming compounds, which are listed in the lexicon under their right component, where their explanation logically belongs.

Other entries which are subordinate to a more principal one are idiomatic expressions and citations. Thus we have a total of ten sorts of entries, classified into three hierarchical levels (to give a comparison, the much more exhaustive Monier-Williams sanskrit-to-english dictionary[18] has 4 hierarchical levels).

Let us now explain the structure of the usage component of our entries. We have actually three kinds of such usage structure, one corresponding to nouns (substantives and adjectives), another one corresponding to verbs, and still another one for idiomatic expressions. We shall now describe the substantives usage component, the verbs one being not very different in spirit, and the idioms one being a mere simplification of it.

The usage structure of a substantive entry is a list of *meanings*, where a meaning consists of a grammatical *role* and a *semantics* component. A role is itself the notation for a part-of-speech tag and an optional positional indication (such as 'enclitic' for postfix particles, or 'iic' [*in initio composi*] for prefix components). The part-of-speech tag is typically a gender (meaning substantive or adjective of this gender), or a pronominal or numeral classifier, or an undeclinable adverbial role, sometimes corresponding to a certain declension of the entry. This may sound a little bit hairy, but it actually corresponds to a fairly flexible concrete syntax at the disposal of the lexicographer, put into a rigid but rigorous structure for computational use by the data base processors.

The semantics component is itself a list of elementary semantic items, each possibly decorated by a list of spelling variants. Elementary semantic items consist in their turn of an *explanation* labeled with a *classifier*. The classifier is either 'Sem', in which case the explanation is to be interpreted as a substitutive definition, or else it is a field label in some encyclopedic classification, such as 'Myth' for mythological entries, 'Phil' for philosophical entries, etc, in which case the explanation is merely a gloss in natural language. In every case the explanation component has type *sentence*, meaning in our case french sentence, since it concerns a sanskrit-to-french bilingual dictionary, but here it is worth giving a few additional comments.

The first remark is that french is solely used as a semantic formalism, deep at the leaves of our entries. Thus there is a clear separation between a superstructure of the lexical database, which only depends on a generic dictionary structure and of the specific structure of the sanskrit language, and terminal semantic values, which in our case point to french sentences, but could as well point to english, german, hindi, etc representations within a multilingual context. Or more interestingly could be an intermediate semantic universal structure of the WordNet[5] kind (i. e. synsets). Thus all the structuring work which gave rise to the superstructure may be one day reused in a more generic setting.

The second comment is that the type 'sentence', seen as an abstract type in the dictionary module, may be itself refined to a finer analysis in the french manager module. Let us briefly describe what is the current state of this refinement. For the moment, we do not attempt to parse french sentences according to some grammatical notion of french syntax, we merely recognize punctuation symbols and a few specific notations. Thus a sentence is a list of utterances separated by semicolons; an utterance consists of a mood (affirmative, interrogative or exclamative) coloring a phrase, where a phrase is a list of subphrases separated by commas. Finally, a subphrase is a list of words, where words are classified as ordinary french words, strings denoting numbers, strings denoting sanskrit references, strings representing specific notations for dates, mathematical expressions, botanical or zoological species, etc. In other words, we have a precisely *tagged* french discourse. Some of these tags are useful for specific linguistic processing, such as antonyms or synonyms. Some tags are governance patterns, used to represent schematic phrases such as 'acheter qqc. <acc.> à qqn. <gen. abl.>'. Here the governance patterns provide a kind of grammatical valence to the (verbal) phrase, stating that this verb use needs two (noun) subphrases: one in the accusative case, of typology (buyable) object, and the other in the genitive or ablative case, of typology person. Such valence could be used at

parsing as a subtyping coercion, selecting the corresponding features from the tagging of the current verbal phrase, and at translation as a concrete syntax rewriting pattern.

The strings denoting sanskrit references are specially important, since they determine the hypertext links in the HTML version of the dictionary. There are two kinds of possible references, proper nouns starting with an upper case letter, and common nouns or other sanskrit words. For both categories, we distinguish *binding occurrences*, which construct HTML anchors, and *used occurrences*, which construct the corresponding references. In order to discuss more precisely these notions, we need to consider the general notion of scoping. But before discussing this important notion, we need a little digression about homonymy.

1.3 Homonyms

First of all, there is no distinction in sanskrit between homophons and homographs, since the written form reflects phonetics exactly (if one neglects the vedic accent). As in any language however, there are homonyms which are words of different origin and unrelated meanings, but which happen to have the same representation as a string of phonems. They may or may not have the same grammatical role. For such clearly unrelated words, we use the traditional solution of distinguishing the two entries by numbering them, in our case with a subscript integer. Thus we distinguish entry *aja*$_1$ 'he goat', derived from root *aj* 'to lead', from entry *aja*$_2$ 'unborn', derived from privative prefix *a* to root *jan* 'to be born'.

Actually, directly derived words, such as substantival root forms, are distinguished from the root itself, mostly for convenience reasons (the usage structure of verbs being superficially different from the one of substantives). Thus the root *diś*$_1$ 'to show' is distinguished from the substantive *diś*$_2$ 'direction', or *jñā*$_1$ 'to know' is distinct from the feminine substantive *jñā*$_2$ 'knowledge'.

Apart from this basic typing distinction between roots and atomic substantives, it might have been hoped that derivations leading to identity of forms would preserve the meaning, i.e. postulate *semantic confluence*. Alas, this is nor the case, for instance because of ambiguity between absolutives and gerundives for verbs derived by preverbs. Thus we have to distinguish between the gerundive (also called passive future participle) *pragṛhya*$_1$ 'to be taken' of verb *pragrah* 'take' and its absolutive (undeclinable past participle) *pragṛhya*$_2$ 'having taken'. Of course the second undeclinable may be taken as an adverbialisation of the adjectival gerundive, with a tense shift. This

identity modulo the time reference frame point of view is evident in the now obsolete naming of the absolutive as the 'gerund', and thus the duplication of such entries is debatable to say the least, since another option would be to group them and list their different roles. On the other end, it may be argued that they are two distinct formations, the origin of the absolutive being the instrumental of a verbal noun. Furthermore, confluence could obtain only for prefixed verbs, since roots admit an absolutive in *-tvā*.

Other non-confluence situations arise, though, between words sharing commun roots. For instance, the adjective *nirvācya*$_1$, derived by the negation prefix *nis* from the gerundive *vācya*, means 'blameless', as someone who should not be spoken (badly) of, whereas the gerundive *nirvācya*$_2$ of verb *nirvac* 'explain' means 'what should be explained'. Here confluence of the two formations would be natural, since *nirvac* is derived from root *vac* 'to speak' by preverb *nis* 'out of', whereas *vācya* 'to be said' is itself a gerundive form of *vac*. If we strive at precise semantic indexing, we have to distinguish the two entries. But if we understand that *nirvācya*$_1$ is issued from the semantic slipping of *vācya* from 'to be spoken of' into 'to be denounced' (with substantival form 'blame'), we see that confluence could be restored in this case through explicit indexing of word senses. Thus etymology could be explained on a more precise structure on 'notions' as word senses (sememes) as opposed to just on words (lexemes).

It remains that the frontier between homonymy and polysemy is thin indeed.

1.4 Scoping

There are two notions of scoping, one global, and the other one local. First, every reference ought to point to some binding occurrence, somewhere in the data base, so that a click on any used occurrence in the hypertext document ought to result in a successful context switching to the appropriate link (and not to some error message '404 not found' in your browser); ideally this link ought to be made to a unique binding occurrence. Such binding occurrences may be explicit in the document; typically, for proper nouns, this corresponds to a specific semantic item, which explains their denotation as the name of some human or mythological figure or geographical entity. For common nouns, the binding occurrence is usually implicit in the structure of the dictionary, corresponding either to the main stem of the entry, or to some auxiliary stem or flexed form listed as an orthographic variant. In this sense a binding occurrence has as scope the full dictionary, since it may be referred to from anywhere. In another sense it is itself within the scope of a

specific entry, the one in which it appears as a stem or flexed form or proper name definition, and this entry is itself physically represented within one HTML document, to be loaded and indexed when the reference is activated. In order to determine these, the grinder builds a trie data structure of all binding occurrences in the data base, kept in permanent storage. A second pass in a checking mode permits to verify that each used occurrence is bound somewhere. The precise page in which it occurs is determined by a prefix of its encoding string. This whole process is very similar to cross-reference analysis in a software package.

Actually, things are still a bit more elaborate, since each stem is not only bound lexicographically in some precise entry of the lexicon, but it is within the scope of some grammatical role which determines uniquely its declension pattern. This requirement is hard to explain without lengthy explanations about the precise abstract structure of entries, but is easy to explain by way of a representative example. Consider the following typical entry:

k mAr*kumāra* m. garçon, jeune homme; fils |prince; page; cavalier |myth. np. de Kumāra 'Prince', épith. de Skanda — n. or pur — f. *kumārī* adolescente, jeune fille, vierge.

Let us now look at the source entry in the data base:

```
\word{kumaara}
\sm
                    \sem{garçon, jeune homme; fils}
                \ou \sem{prince; page; cavalier}
                \ou \myth{np. de \npd{Kumaara} "Prince",
                          épith. de \np{Skanda}}
\cas \sn            \sem{or pur}
\cas \fem{kumaarii} \sem{adolescente, jeune fille, vierge}
\fin
```

There are actually four binding occurrences in this entry. The stem *kumāra* is bound initially with masculine gender for the meaning 'boy' (keyword `sm` standing for masculine substantive), and rebound with neuter gender (`sn`) for the meaning 'gold'. The stem *kumārī* is bound with feminine gender for the meaning 'girl'. Finally the proper name *Kumāra* is bound in the mythological subentry, the text of which contains an explicit reference to proper name Skanda (keyword `np` standing for proper name, and `npd` for proper name definition).

2 The grammatical engine

We are now ready to understand the second stage in our sanskrit linguistic tools, namely the grammatical engine. This engine allows the computation of flexed forms, such as declensions of substantives. It is based on the observation that in sanskrit, inflexion classes are determined by the ending of the stem and its grammatical gender. Since we just indicated that all defined occurrences of substantive stems occurring in the dictionary were in the scope of a gender declaration, this means that we can compute all flexed forms of the words in the lexicon by iterating a grammatical engine which knows how to decline a stem, given its gender (i. e. by computing a table of declined forms indexed by number and case).

2.1 Sandhi

Given a stem and its gender, standard grammar tables give for each number and case a suffix. Glueing the suffix to the stem is effected by a phonetic alliteration process known as *sandhi* (word meaning 'junction' in sanskrit). Actually there are two sandhi processes. One, called external sandhi, is a regular homomorphism operating on the two strings representing two contiguous words in the stream of speech. The end of the first string is modified according to the beginning of the second one, by a local alliteration process. Since sanskrit takes phonetics seriously, this alliteration occurs not just orally, but in writing as well. This *external sandhi* is relevant to contiguous words, and compound formation.

A more complex transformation, called *internal sandhi*, occurs for words derived by affixes and thus in particular for flexed forms in declension and conjugation. The two composed strings influence each other in a complex process which may influence non-local phonems. Thus prefixing *ni* (down) to root *sad* (to sit) makes verb *niṣad* (to sit down) by retroflexion of 's' after 'i', and further suffixing it with *na* for forming its past participle makes *niṣaṇṇa* (seated) by assimilation of 'd' with 'n' and further retroflexion of both occurrences of 'n'.

While this process remains deterministic (except for occasional cases where some phonetic rules are optional), and thus is easily programmable for the synthesis of flexed forms, the analysis of such derivations is non-deterministic in a more complex way than the simple external sandhi. This in our opinion determines where morphology analysis should split in sanskrit: morphology analysis of derived words and flexed forms ought to be done by table look-up in the full lexicon of flexed forms of simple words,

whereas morphology analysis of compounds ought to be done within syntactic analysis, where inversion of external sandhi has to be done anyway. This is consistent with the fact that derived forms of a given word are finite in number, whereas compounds may be formed at any depth by recursive embedding, yielding a potentially infinite generative lexicon.

We thus embarked on programming internal sandhi, a not-so-simple task in the view that existing western grammars differ in their listing of phonetic rules and respective priorities. In the absence of a systematic benchmark, a design process of trial and error has to be implemented, without a clear measure of correctness and completeness. In the last resort, a return to the systematic presentation in the manner of the grammatical treatises of the Paninean tradition may turn out to be necessary. However, see [11, 12] for a renewed phonetic theory of sandhi.

2.2 Declensions

Once internal sandhi is implemented, systematic declension tables may be written down to drive a declension engine. Here too the task is not trivial, given the large number of cases and exceptions. At present our grammatical engine, deemed sufficient for the corpus of classical sanskrit (that is, not attempting the treatment of complex vedic forms), operates with no less than 86 tables (each describing 24 combinations of 8 cases and 3 numbers). This engine may thus generate all declensions of substantives, adjectives, pronouns and numerals. It was found convenient to accrue the 3 grammatical genders with a fourth pseudo-gender 'Any', which is a pseudo gender attribute of words which inherit their gender from the context, such as the deictic pronouns *aham* (I) *tvad* (you) and *ātman* (self), and numerals greater than 4. Remark that this is different from adjectives, which admit the three genders, and for which concord with their qualified substantive is required.

It is to be remarked that this grammatical engine, available as a standalone CGI binary accessible through the Web page
`http://pauillac.inria.fr/~huet/SKT/decls.html`, is to a certain extent independent of the current state of the lexicon, and thus may be used to learn the declension of words belonging to a larger corpus. However, the only deemed correctness is that the words actually in the lexicon get their correct declension patterns, including exceptions. No warranty should be expected when submitting arbitrary stems to the engine, since we do not know good consistency checks for sanskrit stems, and exceptions have to be taken care of specifically anyway.

This grammatical engine is accessible online from the Web version of the

lexicon, since its abstract structure ensures us not only of the fact that every defined stem occurs within the range of a gender declaration, but conversely that every gender declaration is within the range of some defined stem. Thus we made the gender declarations (of non-compound entries) themselves mouse sensitive as linked to the proper instanciation of the grammatical CGI program. Thus one may navigate with a Web browser not only within the dictionary as an hypertext document (thus jumping in the example above from the definition of Kumāra to the entry where the name Skanda is defined, and conversely), but also from the dictionary to the grammar, obtaining all relevant flexed forms.

2.3 Flexed forms management

One special pass of the grinder generates the trie structure of all declensions of the stems appearing in the dictionary. This trie may be itself pretty-printed as a document describing all such flexed forms. At present this represents about 1700 pages of double-column fine print, for a total of around 200 000 forms of 8200 stems.

The task of generating the conjugation patterns of verbs is not yet started. This will doubtless be a rather serious endeavour, given the very big number of possible verbal forms, obvious from the following abstract structure of the verb structure, to be multiplied for finite verbal forms by 3 persons and 3 numbers:

```
type voice = [ Active | Reflexive ]
and mode = [ Indicative | Imperative | Causative
           | Intensive | Desiderative ]
and tense = [ Present of mode | Perfect | Imperfect | Aorist
            | Future ]
and nominal = [ Pp | Ppr of voice | Ppft | Ger | Infi | Peri ]
and verbal =
  [ Conjug of (tense * voice)
  | Passive
  | Optative of voice
  | Nominal of nominal
  | Absolutive | Conditional | Precative
  | Derived of (verbal * verbal)
  ];
```

Of course, we do not have to generate different tables for the 525 roots which are presently listed in the lexicon, since sanskrit verbs split into 10

conjugation classes, but it is to be expected that many exceptions will have to be separately treated, and that finer distinctions will have to be made. For instance in the above structure, the single tense 'aorist' hides the 7 different ways of forming aorist conjugation.

2.4 Index management

Another CGI auxiliary process is the index. It searches for a given string (in transliterated notation), first in the trie of defined stems, and if not found in the trie of all declined forms. It then proposes a dictionary entry, either the found stem (the closest stem the given string is an initial prefix of) or the stem (or stems) whose declension is the given string, or if both searches fail the closest entry in the lexicon in alphabetical order. This scheme is very effective, and the answer is given instantaneously. This shows that the trie datastructure is appropriate for this use, even the rather bulky declension one. After all, a trie may be thought of as the spanning tree of a finite automaton graph, and it would be hard to get more sharing from a recognizing graph, since our tries carry information at their leaves (i.e. they represent *maps* of strings into feature structures, not just *sets* of strings).

2.5 The unique source requirement

The present relative independence of the grammatical engine from the lexicon follows actually from pragmatic considerations. Many exceptions are recorded directly in its source code, as opposed to being fetched from the lexicon. Actually, in the lexicon abstract syntax, there are slots for listing special grammatical forms and other irregularities: the type for orthographic variants contains a constructor 'Declension' which takes as argument declension directives such as special stems. Some of these directives are used in the computation of flexed forms; for instance, the feminine stem of adjectives is indicated there. For completely irregular words, the full declension tables may actually be listed, but usually a few caracteristic cases are shown, the other being inferable by similarity. At present most of this information is ignored (besides being printed in the text version), because it is a non-trivial task to design a meta-language of grammatical annotations usable to describe minimal parameters to a grammatical engine.

For instance, in the Monier-Williams dictionary, an adjective whose masculine and neuter stem ends in *a* while its feminine stem ends in *ā* is listed as *mf(ā)n*, whereas it is listed as *mf(ī)n* if the feminine stems ends in *ī*. But a word such as *bālaka* (childish) is annotated *mf(ikā)n*, meaning that

its feminine stem is *bālikā*. And *śveta* (white) is listed as *mf(ā* or *śvenī)n*, a fine notation for humans, but a processing headache for machines. It is not clear where the line should be drawn between defining specific notation for irregular but frequently occurring cases, and rejecting a case as a plain specific exception.

The same holds true of listing compound words: when listed as a subentry of its left component, the right component ought to be enough to infer the compound, whose spelling may be computed as sandhi of its two components. But difficulties have to be expected of beginners, who may not easily spot say *tejoliṅga* as a compound of *tejas* and *liṅga*. And if the full form is written, it is not obvious to understand what is the second component of the compound, since sandhi analysis is ambiguous. For instance, *satyājñā* could be analysed as *satyā-jñā* (knowledge of truth), *satyā-ajñā* (ignorance of truth), or *satyā-ājñā* (authority of truth). In order to resolve such ambiguity, specially frequent since *ā* is a common preverb (indicating movement towards the locutor), while *a* is the privative prefix, Monier-Williams invented a special diacritic notation, where a long *ā* may be written with a circumflex notation *â*, where the two slopes of the circumflex accent may be independently put in plain or bold face. Needless to say, this subtelty demands sharp eyesight and high quality printing, and is out of reach for current optical scanning software. It should be recognised where it belongs, as specific notation for sandhi analysis, which does not easily extend to say the very frequent decomposition of *o* as *as-a*. The proper treatment of such abbreviations will have to wait for the design of unambiguous notation generating regular transducer operations, under which prefixing, but also hopefully pattern-matching modulo sandhi, will be treated.

Notwithstanding these difficulties, which demand compromise solutions with some redundancy in the lexicon, it should be recognised as long-term goal that the lexical database should hold all the morphological and grammatical information in a non-redundant minimal form. This is the well-known software engineering design requirement of unicity of source, which reduces the risk of inconsistency by independent updates, and optimises storage management. This requirement may be obtained by the usual abstraction tools of literate programming: good data structures, procedural encapsulation, modularity, etc., and by the proper algorithm libraries (finite state transducers notably). This does not preclude redundancy for proper ergonomy in the presentation of the linguistic material to a non-expert user to be actually computed out.

3 Syntactic analysis

The rest of this note is more speculative, since this stops to be a progress report on an implemented prototype which can be completely documented, measured, and generally experimented with, as it turns into a description of an extrapolated envisioned system, discussing various design requirements. This is justified by the thesis that such design requirements have a generic flavor across many languages, and by the hope that they will elicit comments from professional computational linguists.

3.1 Morphology analysis and tagging

Once verbal declensions are treated, and a non-deterministic external sandhi analyser written, we shall be in the position of attempting syntactic analysis, first in the form of a syntactic tagger recognizing parts of speech.

The morphology analyser will thus convert a stream of phonems into a stream of lexemes, by matching flexed forms against the input sequence, modulo rules of external sandhi. A backtrack mechanism will be necessary, in order to capture the non-determinism of sandhi analysis. Within the flexed forms will be considered the substantival initial compound form, so as to recognize compounds by the same process.

The proper descriptive formalism for sandhi analysis, and more generally analysis of phonetic processes, is well known. It is the realm of regular languages and relations, where (invertible) finite-state transducers may implement mechanisms such as Koskenniemi's two level parallel systems as regular relations[9, 10]. Thus a serious linguistic workbench must encompass a general package for such finite-state tools.

Every lexeme will be tagged with a *morphological tag*, which will declare it as an undeclinable particle, or a noun (substantive, adjective, pronoun or numeral) given with its gender, number, and case, or a verb (presented with a verbal information such as above, giving its mode, voice, and tense, plus its person and number for finite verbs). In all of these cases, the successive words will be presented as precise lexical entries, decorated with morphological attributes.

At this stage, we shall be able for instance to use such tagged texts for *concordance analysis*. To our knowledge, there is not at present an automated tool for sanskrit concordance analysis. All such concordance work has been done so far painstakingly by hand, such as the monumental vedic concordance of Bloomfield[3].

Anyone who tried to do a complete morphology analysis of real data, be it

ancient litterature or scientific articles or newspaper data, is confronted with problematic items: numerals, proper names, technical terms, abbreviations, acronyms, brand names, jargon, slang, foreign words, phonetic renditions of noises, smileys, etc. Then there are mistakes, typographical or other. Most of these items do not belong to the dictionary, but they must be accounted for and systematically classified. Even mistakes are important, if the purpose of the processing is to do statistics on the raw text, for instance. And of course even bona fide words may be missing in the dictionary, which is never a closed list, even for "dead" languages: many sumerian words are waiting for some archeologist to unearth them. And the Lewis Carrolls of the future will keep giving us delightful lexicographic inventions. So this quick discussion shows that there are at least 3 important phenomena which must be carefully analysed in the design of an electronic lexicon: the diachronic nature of the language, the intended coverage of the lexicon, and the ability to accommodate completely certain corpuses.

Actually, it is clear that computer storage is becoming so cheap that at some point virtually all digitally represented text, be it typed in, voiced in, or optically read, will be archived. We are all sitting in front of an inexhaustible blank sheet of paper, so to speak. And all those words will percolate through layers of lexicographic processors, from the spelling corrector in your text processor to the professional lexicographer who will make your neologisms visible to the community. The design of this percolating process, which will ultimately be both distributed and multilingual, is a fascinating research problem.

3.2 Acquisition and completeness

I shall stop here this digression by giving a sketch of the accretion process of my sanskrit dictionary. When I started this effort 6 years ago, I entered by hand words found across my readings in secondary sources. From the start, I was very careful about homogeneity and completeness, and I imposed myself a discipline of lexical transitive closure, as follows. First of all, entering a compound word forces the entering of its subparts. More generally, the lexicon ought to be closed by etymology, up to the root if it is known. Whenever some sanskrit notion appears in the gloss of some item, it should itself appear as an entry. As a consequence, whenever a word is part of some traditional enumeration, all its siblings will be listed with it.

At some point I became more systematic in the building up of a full coverage of frequent words, by entering systematically lexicons such as Bergaigne's[1]. After I designed the HTML processor, I could mechanically verify

my transitive invariants. Today I have a total coverage of 12000 entries, of which 7000 are non-compound words, issued from 525 verbal roots. It seems a good turning point to stop this elementary hand-made lexicon, and to use its computerised infrastructure to go after a more complete lexical database, accrued by corpus acquisition through the tagger described above. This will necessitate the completion of the verb flexion machinery, plus a reverse morphology analyser, which will propose, in a lexicographer assistant mode, candidate entries from failed taggings. These candidates may then be checked against the digitalised Monier-Williams provided by scholars from Köln University, to aim at a similar coverage, but in a structured lexical database fit for further text processing.

On this topic of completeness invariants, let us briefly mention synonyms and antonyms. In the current state of my lexicon, I try to maintain information about synonyms and antonyms, and such information is recognised as specific links that can be followed in navigation. The completeness invariant is here that whenever such links exist, the reverse links ought to exist as well, although no attempt at mechanical checking of this invariant is attempted. It is to be expected that at some future point compatibility with the WordNet structures such as *synsets* will be investigated. Let us mention however that special relationships hold for proper names, such as gods, which are named through epithets which designate aspects of these gods which are not equivalent, and for which we do not want to model this as an equivalence, but as an asymmetric relation. In the current version, there is a (more or less arbitrary) choice of one of the names as the canonical one, to which all other names point to. This 'solution' does not solve all problems of relating proper names. For instance, Prabhu (Lord) may be used to name a specific god as Supreme Being, but we do not want to identify all gods which could be called Prabhu. In the same way Kṛṣṇa has many names, which are not necessarily meaningful as names of other aspects or avatars of Viṣṇu. Other problems of a related nature are honorific titles, religious surnames for monks, and reigning names for kings. Then some human beings are deified (such as Buddha or Mañjughoṣa), some avatars of God are treated as historical figures (Rāma, Kṛṣṇa), and homonyms living centuries apart are treated as one (Patañjali grammarian and Patañjali theorician of yoga). The proper structure for proper name descriptions appears to be a complex affair.

3.3 Syntactic analysis proper

Let us now return to the problem of parsing sentences into some kind of phrasal or deep structure.

As is generally the case for langages with declensions, like latin or greek, the order of words in a sentence in not as important as for languages with prepositional phrases. This is specially true in indian languages, where word order is more or less free[2]. In terms of the underlying glue logic, we have a commutative form of linear logic, and record-like feature structures ('frames') tend to take the place of phrase structure.

Simple sentences, restricted to one finite verb form, will thus reduce to a pair (vector,verbal), where vector is a multiset of pairs (entry,attributes), where entry is a (possibly compound) noun stem and attributes is a triple (case,number,gender). The verbal form will be looked up in the lexicon for valence, a multiset of case attributes. A valence declaration will be deemed compatible with the current input if it is a subset of the case projection of the attributes. Some remaining entries will be accommodated as complements, either adjectival complements of other entries if their attributes agree, or genitive complements, or adresses in the case of vocatives. The rest of the remaining entries should then be considered as adverbial complements.

This basic scheme is very simplistic. We have not discussed undeclinables, for instance, which may be treated as adverbs, or as in the case of particles be treated as sentinels[19] breaking the sentence into sequential subphrases. For instance, absolutives often play this role of decomposing a sequence of actions into a succession of elementary actions affected by the same agent. Sentences built along this structure could be transformed into the proper sequence of elementary phrases, absolutives being transformed into finite verbal form of the proper tense, the common agent being turned into their subject.

When the sentence will be successfully decomposed along those lines, the multiset representation may be considered as some sort of deep structure of it. When a full automatic analysis will not be possible, at least the partial successes will make up a partial parts-of-speech tagging of the sentence. When several successful analyses will apply, several courses of action may be considered. First, a human user, presumably a sanskrit expert, will have the choice of choosing one parse. Second, some statistical information, compiled from a relevant corpus, can be used to disambiguate the sentence. Third, the various choices may be kept as analysis variants. This last possibility must be provided for anyway, since sanskrit poets have consciously played on the inherent ambiguity of processes such as sandhi to produce works with

double entendre.

A frequent case of ambiguity will arise from verbs accepting two accusative complements. Such verbs typically assume some taxonomy of concepts, making obvious the assignment of say one animate agent to one of its slots, and one edible object to the other. This points to the necessity of incorporating a taxonomy discrimination net within the lexicon, that will help to the proper choice of complements. It is to be expected of course that a simple static classification will not suffice, and that some sort of common sense reasoning will be involved at some point, so that for instance grass eaters become food themselves to their predators. The universal nature of these basic facts of life on our planet point to the multilingual framework in which they ought to be studied, although cultural differences will have to be accountable for, by overriding default postulates for instance. Again proper discipline will have to be enforced, in order to avoid the AI-completeness trap. Relevant literature comprises the WordNet effort[5], Pustejovky's generative lexicon[21], and Mel'čuk's lexical functions[16] and meaning-text theory[17].

4 Towards a full sanskrit resource manager

Now I will depart from describing existing work, or straightforward extensions of them for which task scheduling is well controled, in order to project in the future of where such an effort would lead.

4.1 Citations and Corpus

In the current structure of my lexicon, I have provided slots for occasional citations. This should be systematised, in the following way. First of all, some precise criterion must be coined to distinguish idiomatic expressions of generic enough scope from specific usage utterances which are best presented as actual citation from some well-defined corpus. This is not obvious, specially for languages such as sanskrit, whose recorded usage spans at least 30 centuries. What is a frequent vedic idiom may be unknown in classical sanskrit, and conversely. This is true in general of meanings, which must be discriminated by diachronic notations - *vidyut* is modern electricity as well as ageless lightning.

Concerning bona-fide citations, they must be attested in a mechanically verifiable manner. That is, not only the precise work (and its author when known) must be identified, but the precise reference must be given, not just in a verse number notation, but as an hypertext link in a concordance corpus. This sounds crazy at first: should we wait to have a given work

completely digitalised and tagged in order to make the first citation to it in our lexicon ? Should I wait for a full concordance of the critical edition of the *Mahābhārata* (90000 stanzas of 32 syllables) before giving any citation from the *Bhagavadgītā* ? Of course not, and actually there is an intermediate solution: it is simply to design a structure of *corpus skeleton* in which this indexation will take place. That is, within all sanskrit litterature, a section *epic* will contain as subsection the *Mahābhārata*, within which, as sixth of its eighteen major books, will be listed the *Bhīṣmaparvan*, within which, as its third episode, is found the *Bhagavadgītā*, within the 41 sections of which we shall find the relevant one, again a list of *ślokas*, which may all be non-instanciated, except the one of interest, along with its tagging, within which the proper hypertext anchor will be ready to be indexed from the lexicon citation index. All this is rather straightforward, since the appropriate technology and standards are now mature[34, 32].

Now this clearly defines new consistency/completeness invariants, in the mutual relationship between the lexicon, the corpus, and the grammatical tagging tool: the tagger ought to succeed on the *śloka*, thus all words within it ought to appear in the lexical database, where in some deep substructure the corresponding concordance indexes should be remembered. When the syntax tools will permit the analysis of this piece of text as a deep structure, indexed with a presupposition context given by a synthetic abstract (stating for instance who Arjuna and Kṛṣṇa are, consistently with the corresponding encyclopedic entry of the lexicon), then we shall be able to enrich the corpus with its first layer of interpretation.

Thus when the full critical edition of the *Mahābhārata* will be issued by the Bandharkar Institute, proper links to it will be propagated in the corpus skeleton structure. Thus the work of scholars the world over will progressively accrue this digital corpus, by proper linking mechanisms such as correspondance tables, finite-state transducers to account for different standards of translitteration, etc. Furthermore, the corpus itself will not be just a big set of sentences, but it should be full of cross-references, especially within sanskrit, where a strong tradition of commentaries exist. Of course this shall not happen instantly, but one may hope that standards such as the TEI[32] may make this dream come true one day.

Another envisioned feedback is that statistical computations on the digitalised corpus will help refine the frequency indexes within the lexicon, to help the adjustment of its diachnonic structure by unbiased computation. Scholarly questions such as "what is the frequency of the absolute genitive construction in the epic litterature" will be answerable by mechanised corpus crawlers, and so on.

Without getting carried away into too grandiose a scheme, let us simply state the underlying design rationale. The computational linguistic tools should be modular, with an open-ended structure, and their evolution should proceed in a breadth-first manner, encompassing all aspects from phonetics to morphology to syntax to semantics to pragmatics to corpus acquisition, with the lexical database as a core switching structure. And proper tools ought to be built, so that the analytic structure is confronted to the linguistic facts, and evolves through experimentally verifiable improvements.

4.2 Other consistency and completeness requirements

Another critical requirement concerns semantic entries. Their french contents ought to be analysed, and first of all tagged. This requires the obtention of a french lexicon. If this lexicon is at the Wordnet format, it will give us a first step toward a sanskrit WordNet dictionary (by transitivity, so to speak). The tagging of 'Sem' entries may even permit to compute by inversion a controlled french to sanskrit dictionary. It will also allow to check that french translations obey the grammatical rôle of the sanskrit subentry: substantives ought to correspond to substantives, adjectives to adjectives, adverbs to adverbs, verbs to verbs. For encyclopedic entries, it should be checked that the french gloss is a correct sentence, which requires a more complete parsing, but for rather regular and simple sentences - this "gloss sublanguage" of french may be an interesting challenge for french syntactic analysers. Finally, the total french vocabulary used may be analysed as a "control" vocabulary and its complexity (and semantic covering) may be investigated.

4.3 Semantic topology

The order of semantic entries in a dictionary may encompass several considerations. A historic development of the usage of a word may induce to put historically preceding usages before more modern ones, or to put concrete senses before more abstract or metaphoric usage, to go from general notions to more specific ones. But one may also prefer to list first usual senses before rarer connotations, according to some synchronic preference (like contemporary usage for languages used today). Thus there is a choice of semantic topology, and the serious lexicographer must do some reasoned choice.

For our sanskrit lexicon, which is so far of rather modest size (12000 entries) and which aims at covering frequent usage of classical sanskrit,

the choice was to start from senses close to the etymological origin, and to put figurative and metaphorical usage after more concrete senses. But as the lexicon will grow, this problem of classification of semantic entries will become more acute, and some more explicit rendering of the diachronic aspects of the language will have to be adopted. Distinctions between the vedic language, the epic one, the medieval one and the modern terms will be made, as well as specific buddhist usage, for instance. Design of a proper diachronism discriminant will ultimately rely on statistics gathered from corpus analysis (frequency counts of precise indexings by semantic meanings will be possible only after sense disambiguation, a difficult task which will require specialist guidance).

4.4 Authority

Perhaps the most unsettling question for a lexicographer is the justification of his authority. What is the ultimate legitimity of someone who should be a transparent witness to an elusive reality ? On what basis is one word entered and another one ignored, one sense recorded and another one ommitted, is there a quantitative attested use of a given word with an assumed meaning in a certain corpus ?

This goes for usual senses, especially for a non-native non-speaker and still-beginner of a language which was probably never the mother tongue of anyone, but which has an enormous tradition of erudition. But it is at least as acute for encyclopedic entries, the accuracy of which relies on an elusive encyclopedic knowledge. What should be written about a man-made-god such as Buddha ? Should the dictionary record his golden legend, since it is what constitutes the collective belief of the majority of Buddhists ? Or should we phrase defensive statements which state only the historically attested facts, or should we still be more prudent, in voicing the doubts that certain scholars manifest towards the very existence of any such historic person ?

I have no answer to such questions, but at some point this question of legitimity must be raised, and some record of sources leading to dubious entries must be collected and classified, leading to one more structural requirement.

5 Note on the technology used

All the programming of the various tools described here was done in Objective Caml[23], extended with the Camlp4 preprocessor[26]. The modularity

of the language was exploited in essential ways. The Grinder is a functor (parametric module), taking as argument a Process. The Print process is itself a functor, taking as argument a low level Printer. Thus all parsing and structure building is generic, as well as all printing, down to the low level of a printing driver; thus the production of the TeX paper document is maximally shared with the production of the hypertext Web site.

Most of the programs are completely applicative. The main data structure used for recording entries as well as flexed form is the trie, which insures sharing of initial substrings, implemented as an applicative zipper[7].

The full processing of the 1.6 Mb of text of the lexicon takes less than 2' on a 200 MHz PC with 64Mb of memory. The persistent data base of 12000 entries is 200Kb, its lookup and search is instantaneous. Even the larger persistent data base of 200 000 flexed forms is only 2.5Mb.

6 Conclusion

Sanskrit is actually an interesting case study for computational linguistics, because of a nice mix between grammatical and morphological complexity and lexical regularity. We have reported here on some experiments with lexical organisation, and interactions between a lexical data base and a grammatical engine. We have sketched the design of an ambitious sanskrit resources computerization which we believe is feasible with current technology, and scales up to management of large corpuses and extensive lexicons. This software engineering architectural model transposes to a certain extent to other languages, and may be considered as an initial proposal for a computational linguistics platform.

References

[1] Abel Bergaigne. "Manuel pour étudier la langue sanscrite." F. Vieweg, Paris, 1884.

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

[3] Bloomfield. "Vedic Concordance." Harvard University Oriental Series.

[4] Guy Cousineau and Michel Mauny. The Functional Approach to Programming. Cambridge University Press, 1998.

[5] Christiane Fellbaum, Ed. "Wordnet, an Electronic Lexical Database." MIT Press, 1998. See also:
`http://www.cogsci.princeton.edu/~wn`.

[6] Christopher J. Fynn. How to use Diacritics for Romanised Indic Text on the WWW. `http://www.user.dicon.co.uk/~cfynn/sanskrit.htm`.

[7] Gérard Huet. The Zipper. J. Functional Programming 7,5 (Sept. 1997), pp. 549–554.

[8] G. Huet "Structure of a sanskrit dictionary." INRIA Research Report to appear, oct. 2000.

[9] Ronald M. Kaplan and Martin Kay. "Regular Models of Phonological Rule Systems." Computational Linguistics (20,3), 1994.

[10] Lauri Karttunen. "Applications of Finite-State Transducers in Natural Language Processing." Proceedings of CIAA-2000.

[11] Brett Kessler. "External sandhi in Classical Sanskrit." M.S. Thesis, Nov. 1992.

[12] Brett Kessler. "Sandhi and Syllables in Classical Sanskrit." Proceedings, Twefth West Coast Conference on Formal Linguistics, E. Duncan, D. Farkas and P. Spaelty, (eds.), p. 35–50. CSLI, Stanford, Calif.

[13] Donald Knuth. The TEXbook. Addison-Wesley, 1984.

[14] Leslie Lamport. LATEX - A Document Preparation System. Addison-Wesley, 1986.

[15] Leslie Lamport et al. LATEX 2_ε - The macro package for TEX. Edition 1.6, Dec. 1994. Distributed as part of Version 19 of GNU Emacs.

[16] Igor Mel'čuk. "Lexical Functions in Lexicographic Description." BLS 8, pp. 427–444, 1973.

[17] Igor Mel'čuk. "Vers une linguistique Sens-Texte." Leçon inaugurale, Collège de France, 1997.

[18] Monier-Williams. "A Sanskrit-English dictionary, etymologically and philosophically arranged, with special reference to Greek, Latin, German, Anglo-Saxon, and other cognate Indo-European languages." Macmillan and Co. London and New York, 1872.

[19] K. Narayan Murthy. "Machine Assisted Translation." Proceedings of NLPRS-99 Fifth Natural Language Processing Pacific Rim Symposium, Beijing, China, November 5-7, 1999.

[20] Anshuman Pandey. Devanāgarī for TEX, version 1.6, Aug. 1998. Available from any CTAN archive site, such as `ftp://ftp.tex.ac.uk/`.

[21] James Pustejovsky. The Generative Lexicon. MIT Press, 1995.

[22] Dave Raggett, Arnaud Le Hors and Ian Jacobs, Eds. HTML 4.0 Specification. W3C Recommendation, April 24, 1998. `http://www.w3.org/TR/REC-html40`.

[23] Pierre Weis and Xavier Leroy. Le langage Caml. 2ème édition, Dunod, Paris, 1999.

[24] William D. Whitney. Roots, Verb-forms and Primary Derivatives of the Sanskrit Language. 1885. Repr. Motilal Banarsidass, Delhi, 1997.

[25] Acrobat Reader, an Adobe product, freely downloadable from: `http://www.adobe.com/products/acrobat/readstep.html`.

[26] The Camlp4 preprocessor. See: `http://caml.inria.fr/camlp4/`.

[27] The Common Gateway Interface. See: `http://www.w3.org/CGI/Overview.html`.

[28] Ghostscript. See `http://www.cs.wisc.edu/~ghost/index.html`.

[29] Objective Caml. See `http://caml.inria.fr/ocaml/index.html`.

[30] PDF, the portable document format, an Adobe product. See: `http://www.adobe.com/products/acrobat/adobepdf.html`.

[31] Postscript, an Adobe product. See: `http://www.adobe.com/print/postscript/main.html`.

[32] The Text Encoding Initiative. See: `http://www.uic.edu/orgs/tei/`.

[33] The Unicode standard 2.1. See: `http://charts.unicode.org/`.

[34] The XML norm, version 1.1, W3C consortium. See: `http://www.w3c.org/documents/XML/`.