Design of a Lexical Database for Sanskrit

Gérard Huet
INRIA-Rocquencourt
BP 105,
78153 Le Chesnay CEDEX
France
Gerard.Huet@inria.fr

Abstract

We present the architectural design rationale of a Sanskrit computational linguistics platform, where the lexical database has a central role. We explain the structuring requirements issued from the interlinking of grammatical tools through its hypertext rendition.

1 Introduction

Electronic dictionaries come into two distinct flavours. First, dictionaries and encyclopedia, meant for human usage, have converted to computer typesetting in the 80's, and to structured editing and various other kinds of computer interface facilities for lexicographers in the 90's. Major computer projects, using specific algorithm libraries, were endeavoured for the management of large dictionaries, such as the *New Oxford Dictionary*, or the *Trésor de la Langue Française*. Progressively, hypertext technology was systematically used, together with the progressive standardisation of the philology technology on the SGML norm. This allowed dictionaries and encyclopedia to develop hypertext versions for the end customer, with search software and other interface facilities. With the introduction of the Web technology in middle 90's, relying on SGML and its successor XML, more sophisticated hybridations between electronic books and the Internet could take place – sometimes endangering the very existence of traditional reference works.

In parallel, computational linguistics research developed the need for lexical databases, progressively storing grammatical information such as part of speech category or subcategorization valencies, and even more sophisticated data about syntax, with the general tendency to modularize the formal descriptions of syntax with lexicalized grammars. The next step was to store semantic information, a good example of which is the Wordnet structure. The progressive development of controlled treebanks and the availability of large digitalised corpuses opened the way to the systematic development of linguistic resources such as lexical databases, using automated acquisition technology. They reflect linguistic usage in a statistically relevant way, as opposed to the previous introspection based ad-hoc methods. The need to develop multilingual platforms pushes furthermore the integration of generic semantic knowledge into pivot formalisms.

It is somewhat surprising that the two lines of development of electronic dictionaries did not interact significantly. The different requirement needs for the two applications is often alluded to for making this separation unavoidable, but such arguments are not wholly convincing. Rather than technical divergences, it appears that it is sociological considerations (intellectual and commercial interests, cultural differences, etc) that prevent such cooperation. Indeed, information retrieval technology may usefully take advantage of information implicit in machine readable dictionaries, and conversely corpus analysis may reveal incompleteness or other other defects in lexicological structure (Krovetz, 1994; Krovetz and Croft, 1992). In the long run, linguistic technology is obviously of great potential application to the human information needs, and traditional dictionaries will need to take advantage of it to stay competitive. We shall argue, in this communication, that a lexical database may be used for both purposes, in a win-win cooperation. We base our opinions on a concrete experiment on the design of linguistics resources for the Sanskrit language.

2 From book form to web site

2.1 A Sanskrit-French paper dictionary

The author started from scratch a Sanskrit to French dictionary in 1994, first as a personal project in indology, then as a more structured attempt at covering Sanskrit elementary vocabulary. A systematic policy was inforced along a number of successive invariants. For instance, etymology, when known, was followed recursively in relevant entries. Any word could then be broken into morphological constituents, down to verbal roots when known. This "etymological" completeness requirement was at first rather tedious, since entering a new word may require the acquisition of many ancestors, due to complex compounding. But it appeared that the acquisition of new roots slowed down considerably after an initial "bootstrap" phase. When the number of entries approached 10000, with 520 roots, new roots acquisition became quite exceptional. This phenemenon is similar to the classical "lexical saturation" effect witnessed when one builds a lexicon covering a given corpus (Polguère, 2003). Progressively, a number of other consistency constraints were identified and systematically enforced, which proved invaluable in the long run.

At this point the source of the lexicon was a plain ASCII file in the LaTeX format. However, a strict policy of systematic use of macros, not only for the structure of the grammatical information, and for the polysemy representation, but also for internal quotations, ensured that the document had a strict logical structure, mechanically retrievable (and thus considerably easier to process without loss of information than an optically scanned paper dictionary (Ma et al., 2003)).

Indeed, around 2000, when the author got interested into adapting the data as a lexical database for linguistic processing, he was able without too much trouble to reverse engineer the dictionary into a structured lexical database (Huet, 2000; Huet, 2001). He then set to work to design a set of tools for further computer processing as an experiment in the use of functional programming for a computational linguistics platform.

The first design decision was to avoid standard databases, for several reasons. The first one is portability. Many database formats are proprietary or specific to a particular product. The second reason is that the functionalities of data base systems, such as query languages, are not well adapted to the management of lexical information, which is highly structured in a deep manner - in a nutshell, functional rather than predicative. Thirdly, it seemed best to keep the information in the concrete format in which it had been developed so far, with specific text editing tools, and various levels of annotation which could remain with the status of unanalysed comments, pending their possible later structuring. After all, ASCII is the most portable format, large text files is not an issue anymore, parsing technology is fast enough to render negligible compilation times, and the human ease of editing is the really crucial factor – any tool which the lexicographer has to fight to organise his data is counter-productive.

A detailed description of this abstract syntax is available as a research report (Huet, 2000), and will not be repeated here. We shall just point to salient points of this abstract structure when needed.

2.2 Grinding the abstract structure

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 (Miller, 1990; Fellbaum, 1998)). 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 pidgin ML:

```
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 sources of a Web site, 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, as well as grammatical informer, on the other hand[1].

2.3 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.

[1] http://pauillac.inria.fr/~huet/SKT/

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 fourfold 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 (*vocable*) are idiomatic locutions 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 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 locutions. 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 *sense* 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. The thematic role agent is also available as tag, typically for nouns which may be used in the masculine or the feminine, but not in the neuter. 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 sense component is itself a list of elementary semantic items (representing polysemy), 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 (Fellbaum, 1998) kind (i. e. synsets). Thus all the structuring work which gave rise to the superstructure may be one day reused in a multilingual setting – assuming some consensus on grammatical roles standard of course.

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.>'. Such a governance patterns provides a grammatical valence (subcategorization) 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.

2.4 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 phonemes. 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 nouns, 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.

2.5 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 is easy to explain by way of a representative example. Consider the following typical entry:

कुमार *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.

There are actually four binding occurrences in this entry. The stem *kumāra* is bound initially with masculine gender for the meaning 'boy', and rebound with neuter gender 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, bound in its own entry.

3 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 inflected forms, that is declensions of nouns and conjugations of finite forms of verbs. For nouns, we observe that in Sanskrit, declension paradigms 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 inflected 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).

Similary, for verbs, conjugation paradigms for the present system fall into 10 classes. Every root entry mentions explicitly its (possibly many) present classes.

3.1 Sandhi

Given a stem and its gender, standard grammar paradigm tables give for each number and case a suffix. Glueing the suffix to the stem is effected by a phonetic euphony 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 euphony process. Since Sanskrit takes phonetics seriously, this euphony 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 inflected forms in declension and conjugation. The two composed strings influence each other in a complex process which may influence non-local phonemes. 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 inflected 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 inflected forms ought to be done by table look-up in the full lexicon of inflected 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. The problem is complicated by the fact that some phonemes obey different rules according to the historical origin of certain words. We identified two situations in which a morpheme had to be split into two in order to have correct deterministic sandhi; thus j_1 combines with following t to become *kt* like in *bunakti*, whereas j_2 in the same situation yields *ṣṭ* like in *mārṣṭi*; similarly, phoneme h has two variants. 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, but even in this native tradition there are discrepancies. For external sandhi, a precise phonetic presentation exists (Kessler, 1992; Kessler, 1995), but for internal sandhi an exact definition in terms of finite state transducers is still wanting.

3.2 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 nominal 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.

It is to be remarked that this grammatical engine, available as a stand-alone executable, is to a certain extent independent of the current state of the lexicon, and thus may be used to tell 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 hypertext 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 inflected forms.

Similarly for roots, the present class indicator is mouse-sensitive, and yields on demand the corresponding conjugation tables. This underlines a general requirement for the grammatical tools: each such process ought to be callable from a concrete point in the text, corresponding unambiguously to a node in the abstract syntax of the corresponding entry, with a scoping structure of the lexicon such that from this node all the relevant parameters may be computed unambiguously.

In order to compute conjugated forms of non-root verbs, the list of its relevant preverbs is available, each preverb being a link to the appropriate entry (from which the etymological link provides the return pointer). Other derived stems (causative, intensive and desiderative forms) act also as morphology generators.

3.3 Inflected 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 inflected forms. At present this represents about 2000 pages of double-column fine print, for a total of around 200 000 forms of 8200 stems (133655 noun forms and 55568 root finite verbal forms).

Verbal morphology is specially complex. Here is the type structure of verbal forms in the lexicon:

```
type voice = [ Active | Middle ]
and mode =
  [ Indicative | Imperative
  | Optative | Imperfect ]
and tense = [ Present of mode
  | Perfect | Aorist | Future ]
and nominal =
  [ Past_part
  | Pres_part of voice
  | Prft_part
  | Futu_part
  | Gerundive
  | Absolutive
  | Infinitive
  | Periph_future
  | Periph_perfect
  | Stem
  ]
and secondary =
  [ Causative
  | Intensive
  | Desiderative ]
and verbal =
  [ Tense of tense and voice
  | Passive
  | Nominal of nominal
  | Derived of secondary and verbal
  ];
```

In the above structure, the single tense 'aorist' abstracts the 7 different ways of forming aorist conjugation.

3.4 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).

An auxiliary search engine searches Sanskrit words with a naive transcription, without diacritics. Thus a request for *panini* will return the proper link to *pāṇini*.

3.5 Lemmatization

The basic data structures and algorithms developed in this Sanskrit processor have actually been abstracted as a generic *Zen* toolkit, available as free software (Huet, 2002; Huet, 2003b; Huet, 2003d).

One important data structure is the *revmap*, which allows to store inflected forms as an invertible morphological map from stems, with minimal storage. The Sanskrit platform uses this format to store its inflected forms in a in such a way that it may directly be used as a lemmatizer. Each form is tagged with a list of pairs *(stem, features)*, where *features* gives all the morphological features used in the derivation of the form from root *stem*. A lemmatization procedure, available as a CGI executable, searches this structure. For instance, for form *devayos* it lists:

```
{ loc. du. m. | gen. du. m. |
  loc. du. n. | gen. du. n. }[deva]
```

where the stem *deva* is a hyperlink to the corresponding entry in the lexicon. Similarly for verbal forms. For *pibati* it lists:
`{ pr. a. sg. 3 }[paa_1]`, indicating that it is the 3rd person singular present form of root *pā*$_1$ in the active voice.

3.6 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 inflected 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.

We end this section by remarking that we did not attempt to automate derivational morphology, although some of it is fairly regular. Actually, compound formation is treated at the level of segmentation, since classical Sanskrit does not impose any bound on its recursion depth. Verb formation (which sequences of preverbs are allowed to prefix which root) is explicit in the dictionary structure, but it is also treated at the level of the segmentation algorithm, since this affix glueing obeys external sandhi and *not* internal sandhi, a peculiarity which may follow from the historical development of the language (preverbs derive from postpositions). At present, noun derivatives from verbal roots are explicit in the dictionary rather than being computed out, but we envision in some future edition to make systematic the derivation of participles, absolutives, infinitives, and periphrastic future and perfect, as well as the derived verbal stems of causative, intensive and desiderative.

4 Syntactic analysis

4.1 Segmentation and tagging

The segmenter takes a Sanskrit input as a stream of phonemes and returns a stream of solutions, where a solution is a list of (inflected) words and sandhi rules such that the input is obtainable by applying the sandhi rules to the successive pairs of words. It is presented, and its completeness is proved, in (Huet, 2004). Further details on Sanskrit segmentation are given in (Huet, 2003a; Huet, 2003c).

Combined with the lemmatizer, we thus obtain a (non-deterministic) tagger which returns all the (shallow) parses of an input sentence. Here is an easy example:

```
# process "maarjaarodugdha.mpibati";

 Solution 1 :
[  maarjaaras
 < { nom. sg. m. }[maarjaara] >
   with sandhi as|d -> od]
[  dugdham
 < { acc. sg. m. | acc. sg. n. |
     nom. sg. n. }[dugdha] >
   with sandhi m|p -> .mp]
[  pibati
 < { pr. a. sg. 3 }[paa#1] >
   with sandhi identity]
```

This explains that the sentence *mārjārodugdhaṃpibati* (a cat drinks milk) has one possible segmentation, where *maarjaras*, nominative singular masculine of *maarjara* (and here the stem is a hyperlink to the entry in the lexicon glosing it as *chat* i.e. cat) combines by external sandhi with the following word by rewriting into *maarjaro*, followed by *dugdham* which is the accusative singular masculine of *dugdha* (draught) or the accusative or nominative singular neuter of *dugdha* (milk - same vocable), which combines by external sandhi with the following word by rewriting into its nasalisation *dugdhaṃ*, followed by *pibati* ... (drinks).

4.2 Applications to philology

We are now at the stage which, after proper training of the tagger to curb down its overgeneration, we shall be able to use it for scanning simple corpus (i. e. corpus built over the root forms encompassed in the lexicon). The first level of interpretation of a Sanskrit text is its word-to-word segmentation, and our tagger will be able to assist a philology specialist to achieve complete morphological mark-up meaningfully. This will allow the development of concordance analysis tools recognizing morphological variants, a task which up to date had to be effected manually.

At some point in the future, one may hope to develop for Sanskrit the same kind of informative repository that the Perseus web site provides for Latin and Classical Greek[2]. Such resources are invaluable for the preservation of the cultural heritage of humanity. The considerable classical Sanskrit corpus, rich in philosophical texts but also in scientific, linguistic and medical knowledge, is an important challenge for computational linguistics.

Another kind of envisioned application is the mechanical preparation of students' readers analysing a text at various levels of information, in the manner of Peter Scharf's Sanskrit Reader[3].

4.3 Parsing

The next stage of analysis will group together tagged items, so as to fulfill constraints of subcategorization (accessible from the lexicon) and agreement. The result ought be a set of consistent dependency structures.

Many ambiguity problems are to be expected, such as the possible use of accusatives as adverbs, or the possible use of genitives as noun complements in noun phrases as well as verb complements for certain verbs. Verbs admitting double accusative slots will require some kind of semantic ontological classification. Sharing of arguments across clauses, a peculiarity of Sanskrit, will also raise non-linearity issues.

It is to be expected that these issues will need statistical training on a treebank. Such a resource is under construction, in cooperation with Brendan Gillon from Mac Gill University, a specialist in Sanskrit syntax (Gillon, 1996).

5 Corpus and lexicon interactions

5.1 Citations and Corpus

In the current structure of the lexicon, slots are provided for occasional citations. This should be systematised, in the following way. First of all, some precise criterion must be coined to distinguish idiomatic locutions 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 we 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

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

[3]`http://cgi-user.brown.edu/Departments/Classics/Faculty/Scharf/`

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 (XML, Unicode, TEI).

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.

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 may make this dream come true one day.

5.2 Lexicon acquisition

Another important interaction between the corpus and the lexicon is that progressively the lexicon ought to be completed in order to encompass the actual use of words in the corpus. This is not an easy requirement, since our current tagger is lexicon-directed. A robust version must be designed, which will recover gracefully from missing chunks. The analysis of the unknown chunks will require a stand-alone lemmatizer, a difficult task since internal sandhi is a complex non-local process. It is expected that approximate heuristic solutions will be sufficient to help Sanskrit scholars better than elusive exact solutions.

The proper encompassing of the corpus, in a language which spans 25 centuries over a subcontinent, will demand the design of an appropriate diachronic architecture. Statistical computations on the digitalised corpus will help adjust frequency indexes within the lexicon, leading to adjustment of its diachronic 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.

6 Conclusions

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. Proper tools have to be built, so that the analytic structure is confronted to the linguistic facts, and evolves through experimentally verifiable improvements. The interlinking of the lexicon, the grammatical tools and the marked-up corpus is essential to distill all linguistic information, so that it is explicit in the lexicon, while encoded in the minimal way which makes it non-redundant.

We have argued in this article that the design of a hypertext interface is useful to refine the structure of the lexicon in such a way as to enforce these requirements. However, such a linguistic platform must carefully distinguish between the external exchange formats (XML, Unicode) and the internal logical structure, where proper computational structures (inductive data types, parametric modules, powerful finite-state algorithms) may enforce the consistency invariants.

References

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database.* MIT Press.

Brendan S. Gillon. 1996. Word order in classical Sanskrit. *Indian Linguistics*, 57,1:1–35.

Gérard Huet. 2000. Structure of a Sanskrit dictionary. Technical report, IN-

RIA. `http://pauillac.inria.fr/~huet/PUBLIC/Dicostruct.ps`

Gérard Huet. 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.

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

Gérard Huet. 2003a. Lexicon-directed segmentation and tagging of Sanskrit. In *XIIth World Sanskrit Conference, Helsinki*.

Gérard Huet. 2003b. Linear contexts and the sharing functor: Techniques for symbolic computation. In Fairouz Kamareddine, editor, *Thirty Five Years of Automating Mathematics*. Kluwer.

Gérard Huet. 2003c. Towards computational processing of Sanskrit. In *International Conference on Natural Language Processing (ICON), Mysore, Karnataka*.

Gérard Huet. 2003d. 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`

Gérard Huet. 2004. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *Journal of Functional Programming*, to appear. `http://pauillac.inria.fr/~huet/PUBLIC/tagger.pdf`.

Brett Kessler. 1992. External sandhi in classical Sanskrit. Master's thesis, Stanford University.

Brett Kessler. 1995. Sandhi and syllables in classical Sanskrit. In D. Farkas E. Duncan and P. Spaelty, editors, *Twefth West Coast Conference on Formal Linguistics*. CSLI.

Robert Krovetz and W. Bruce Croft. 1992. Lexical ambiguity and information retrieval. *Information Systems*, 10(2):115–141.

R. Krovetz. 1994. Learning to augment a machine-readable dictionary. In *Proceedings of the EURALEX '94. Amsterdam, Holland*, pages 107–116.

Huanfeng Ma, Burcu Karagol-Ayan, David Doermann, Doug Oard, and Jianqiang Wang. 2003. Parsing and tagging of bilingual dictionaries. *Traitement Automatique des Langues*, 44,2:125–149.

G. A. Miller. 1990. Wordnet: a lexical database for English. *International Journal of Lexicography*, 3,4.

Alain Polguère. 2003. *Lexicologie et sémantique lexicale*. Presses de l'Université de Montréal.