Extracting dependency trees from Sanskrit texts

Oliver Hellwig

Institut für Sprachen und Kulturen Südasiens, Freie Universität Berlin, Germany

Abstract. In this paper, I describe a hybrid dependency tree parser for Sanskrit sentences improving on a purely lexical parsing approach through simple syntactic rules and grammatical information. The performance of the parser is demonstrated on a group of sentences from epic literature.
Keywords: Sanskrit syntax, Sanskrit word order, dependency tree

1 Introduction

Designing a reliable algorithm for the automatic syntactic analysis of Sanskrit phrases is an important, though still unsolved problem in computational linguistics. Some previous approaches to this problem are based on built-in rules that encode the syntax of regular Sanskrit phrases using, for instance, finite automata. How to find these syntactic rules is not often the central focus of interest, however. Many researchers use *kāraka* analysis (e.g., [6], claiming free word order for Sanskrit) or the syntax rules formulated in modern learner's manuals such as Apte's or Kale's grammars. Whether these rules describe the correct word order of classical Sanskrit texts remains open to discussion, however, because they may reflect either a pre- (Pāṇini's *kāraka* theory) or post-classical use of Sanskrit. Modern Indological research is no great help in finding the syntactic rules of the classical language. German Indologists such as Delbrück [4], Speyer [9], Canedo [2] and, more recently, Ickler [8] took great pains in analyzing large corpora in detail, but they concentrated on Vedic and pre-classical prose. Results of these stylistic studies of early Sanskrit are hardly applicable to classical texts, which are, in addition, written in verse in most cases (cmp. the critical remarks in [5] on the bias in selecting the texts). The same holds for Staal's frequently cited work [10], which does not care very much about the word order in real Sanskrit texts. In summary, the syntactic rules used in rule-based approaches are derived from theories about modern or pre-classical Sanskrit and then applied to the classical language. This procedure implies that Sanskrit syntax has remained unchanged over an interval of over 3000 years, a claim well suiting the tendency to deny any development in this language, but not founded on large-scale research.

At this point, we find ourselves in a chicken-and-egg dilemma. Before using a rule-based approach to analyze the syntax of classical texts, we need the syntactic rules for the classical language. To find these rules in significant numbers, we need a syntactic parser that is, as just described, rule-based in many cases. We may, therefore, simplify the starting problem: The aim is to design not a complete syntactic parser, but instead an

algorithm giving the most probable dependency tree of a lexically and morphologically analyzed sentence. Valid rules describing the word order of classical Sanskrit may be derived from a large number of manually corrected trees in a later step; however, this step is not discussed in this paper.
Among the numerous approaches to derive dependency trees from sentences in natural languages, a recently published thesis by Yuret is especially interesting because it combines an appealing basic idea with an unsupervised learning algorithm [12]. In this paper, I describe how the performance of Yuret's purely lexical parser can be improved through the addition of some simple, yet efficient, features such as information about grammar (2.2) and valences (2.2), smoothing probabilities (2.2) and a few fixed syntactic rules (2.2). Because test data for Sanskrit syntax are not available, the performance of the parser is demonstrated during the improvement steps by analyzing a few sample sentences from the RĀMĀYAṆA (2.3).

2 Building a lexicalized parser

2.1 Yuret's model

The starting point of the following experiments is the purely lexical dependency parser described by Yuret [12]. According to Yuret, syntactic information is captured by the mutual information between the lexical components of a sentence (cmp. [12, 26-31] and Footnote 1, below). Because mutual information does not depend on the order of two lexical items, the dependency structure constructed using Yuret's algorithm is undirected (but may, of course, be transformed into a directed graph as soon as the root item is identified). Furthermore, the dependency structure is acyclic and, therefore, a tree, which is equivalent to the claim that each word in a sentence should have only one governing word. Finally, Yuret claims that the dependency structure should be projective, i.e., its links should not cross. The author admits that this restriction holds only for many (not all!) sentences in natural languages; this warning is especially important for Sanskrit verses. However, projectivity simplifies the construction of the tree to such a degree that the occasional errors caused by this condition may be neglected. In summary, the application of Yuret's ideas to a sentence produces a dependency tree that reflects the syntactic structure of the sentence as given by its lexical components. The arrangement of such a tree is not necessarily identical to the structure found using, for instance, constituent analysis. However, items close to each other in the dependency tree should constitute syntactic units in regular constituent analysis.
Yuret's system consists of two parts. The *processor* builds dependency structures for a series of lexical units (= a sentence), while the *learner* represents the "knowlegde" or "brain" storing information gained from previously analyzed sentences. To parse a sentence, the processor searches for the most probable structure given the information stored in the

learner. Although there is a Viterbi style algorithm calculating the exact solution, Yuret proposes the following approximation that reduces computation time from $O(n^5)$ to $O(n^2)$ [12, 35ff.]:

1. Each sentence is read from left to right. For each word w_i to the left of the current word w_j, the conditional probability $p(w_j|w_i) = \frac{p(w_i,w_j)}{p(w_i)}$ is calculated.[1]
2. If $p(w_j|w_i) \neq 0$, the words w_i and w_j may be connected with a link. However, this link can only be created if (1) it does not intersect with an already existing link and (2) it does not create a cycle in the dependency structure. If condition (1) or (2) applies, the new link is only inserted if its value $p(w_j|w_i)$ is higher than the respective value of the existing link. In this case, the existing link is removed from the dependency tree.
3. When the sentence has been analyzed, the learner is updated with the linking information created in the second step. Therefore, only the cooccurrence frequencies of words being connected by a link are increased in the learner. During the first cycles of the algorithm, the "brain" of the learner is empty and no dependency structure can be created. In these cases, the learner is updated using pairs of adjacent words.

2.2 Improving Yuret's parser

After being trained on the current corpus of the `SanskritTagger` database (cmp. [7]), Yuret's parser is able to identify some syntactic substructures in unknown sentences. Nevertheless, when we analyze the sample phrases (cmp. 2.3), it becomes obvious that the parser never manages to identify the correct overall structure of any of the samples. This is probably

[1] Yuret uses the mutual information of ordered pairs of words $MI(w_i, w_j) = \log \frac{p(w_i,w_j)}{p(w_i,*)\cdot p(*,w_j)}$ instead of the conditional probability. If L denotes the dependency structure of a sentence S, W the set of all words w_i contained in S, and w_0 the head word of the structure, Yuret explains the use of mutual information as follows [12, 28/29]. The joint probability of the entire sentence is given by

$$p(S) = p(L)p(w_0) \prod_{(w_i,w_j)\in L} p(w_j|w_i) = p(L)p(w_0) \prod_{(w_i,w_j)\in L} \frac{p(w_i,w_j)}{p(w_i)}.$$

Because a projective tree constructed from a sentence S has $|S|-1$ connections after the head word has been identified, this expression can be rewritten as

$$p(S) = p(L) \prod_{w_i\in W} p(w_i) \prod_{(w_i,w_j)\in L} \frac{p(w_i,w_j)}{p(w_i,*)\cdot p(*,w_j)},$$

demonstrating, according to Yuret, that the syntactic information of a sentence can be expressed by the mutual information contained in the lexemes that constitute the sentence. Following Yuret, I set $p(S)$ to a constant factor. – For reasons of numerical accuracy, I use the logarithm of $p(w_j|w_i)$ instead of the raw value (multiplication → addition).

caused by the comparatively small number of data used for training the parser. While Yuret reports convincing results only for databases of more than 10 million words, the `SanskritTagger` database comprises about 2.5 million lexical units. This number is probably not large enough to arrive at reliable estimations of lexical cooccurrence. The following sections describe how to improve the performance of the parser without training it on more lexical data.

Grammatical information Every word stored in the `SanskritTagger` database is accompanied by grammatical information concerning, for instance, number, case or tense. Although Yuret deliberately excluded this kind of information from the parsing process, I have observed clearly superior analysis of the sample phrases when grammatical information was taken into account. To include grammar, the conditional probability of pairs of lexical items is multiplied by the conditional probability of the respective grammatical categories. These categories are a simplified version of those described in [7, 44] since only person and number are recorded for verbal forms.

Verbal valences The second improvement on Yuret's model concerns finite verbal forms and their preferred valences. In many cases, verbs show a strong preference for certain cases, which may be used to enforce links between verbs and their valences. For this sake, we have built from the training data a verb-valence dictionary that stores verbs and the cases that typically occur close to them. Before starting the main learning process, the part of the corpus used for training is scanned for verbal forms. If a finite verb is encountered we search for the next and previous two declined nouns that are included in the same sentence as the verb, and store the verb-case combination in a preliminary table. Next, we calculate the global relative frequencies of all cases and then find extraordinary strong verb-case combinations. A combination is considered strong if (1) it occurs with a frequency of more than 15% and the verb is referenced at least 100 times, or (2) the relative frequency of case c for verb i is significantly larger than the global relative frequency for this case. If a_c is the global average frequency of case c, f_{ci} is the frequency of case c given verb i, and n_i is the sum of all f_{ci} for verb i, we use a simple χ^2 test, with significance assigned at the 10% level ($\chi^2 \geq 1.64$, 1 df), to assess whether f_{ci} is significantly above the expected number of occurrences of case c:

$$\chi^2 = \frac{(f_{ci} - a_c \cdot n_i)^2}{a_c \cdot n_i} + \frac{((n_i - f_{ci}) - (1 - a_c) \cdot n_i)^2}{(1 - a_c) \cdot n_i}$$

If one of these two conditions is fulfilled, the verb and case are stored in a separate valence dictionary.
Whenever the combination of a verb and a declined noun is found during the processing of a new sentence, we search for the verb in the valence dictionary just described. If the case of the declined noun is among the favorite cases of the verb, the linking strength between the verb and

noun is enforced by a positive value (see page 6 for details). – Given the importance of valence information for parsing (see, for instance, [11]), the valence dictionary could be corrected manually in a future version of the program and even enriched by lexical information concerning preferred valences.

Smoothing cooccurrence frequencies One of the main problems in processing natural language using statistical methods is the sparseness of data, especially of n-grams of higher order. Among the many proposed solutions to this problem, we find simple strategies such as linear interpolation (see, e.g., [1]) or add-alpha smoothing. A more sophisticated strategy that makes use of lexical information gained from the training corpus was proposed by Dagan et al. [3], and it was successfully applied to the parsing of Sanskrit sentences. This strategy consists of two steps: a preprocessing step, during which similar words are retrieved from training data, and the actual smoothing step. During preprocessing, we calculate the Kullback-Leibler divergence D between all pairs of words w_i and w_j that are contained in the corpus C:

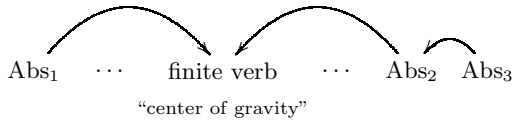
$$D_{ij} = \sum_{w_k \in C} p(w_k|w_i) \log \frac{p(w_k|w_i)}{p(w_k|w_j)}.$$

Based on D_{ij}, the n nearest words w_j are stored for each w_i. If an unknown bigram (w_1, w_2) is found during parsing, the conditional probability $p(w_2|w_1)$ is estimated using conditional probabilities of word pairs (w_1', w_2), where w_1' is a word similar to w_1 found in the preprocessing step. If S_u is the set consisting of the n words w_u nearest to w_1, the estimation $p^*(w_2|w_1)$ is calculated in the following way:

$$p^*(w_2|w_1) = \sum_{w_u \in S_u} p(w_2|w_u) \frac{D_{1u}}{\sum_{w_u \in S_u} D_{1u}}$$

Using simple syntactic rules The final and, in my opinion, most effective way to improve Yuret's model is the use of simple syntactic rules, which transform the lexicalized base model into a hybrid parsing approach. I distinguish between two types of syntactic rules: *Fixed* rules encode the syntactic structures of a phrase that are certain to occur (given that the phrase is complete). These rules create fixed links or prevent lexical links from being constructed. In addition to describing the basic syntactic structures of a sentence, these rules strongly reduce the number of possible links and thereby suppress improbable analyses. On the other hand, *enforcements* increase or decrease the linking strength, but they do not insert or remove links from the dependency tree.
Fixed rules can further be divided into positive and negative rules. *Positive* rules describe the basic structure of a sentence containing exactly one verb, which may be supplemented by a congruent subject and absolutives. To detect this basic structure, the sentence is repeatedly scanned from left to right. In the first scan, the finite verb is detected and linked to the beginning of the phrase (→ head-verb). Next, absolutives are found

and connected either to the head-verb or to other absolutives. The head verb serves as a "center of gravity" indicating the search direction from each absolutive contained in the sentence:



Finally, nominatives congruent with the head verb are connected to it. If several nouns can be connected with the verb, the most probable one given the lexical attraction between noun and verb is selected. *Negative* or restrictive rules prevent possible links from being inserted into the dependency tree. Currently, three negative rules are used:

1. Words contained in a composite must only be linked to the head of the composite or other words contained in the composite. – This rule describes the correct formation of composites, but it is sometimes neglected in real texts. To give just one example: Someone "whose body is pierced by arrows" should be a *śaraviddhaśarīraḥ*. However, expressions such as *śarair viddhaśarīraḥ* can be frequently found, for instance, in epic texts.
2. An indeclinable must not be linked with a noun or adjective. – Exceptions are indeclinables forming part of composites such as *su-* or *nānā-*.
3. Incongruent nouns, except for the combination *any case-genitive*, must not be linked. – The validity of this rule is not established for the nominal style of scientific Sanskrit; see, e.g., *prasiddhasādharmyāt sādhyasādhanam upamānam* (NyāSū, 1, 1, 6), where *-sādharmyāt* is dependent on *-sādhanam*.

Enforcements change the strength of a new link, but they do not influence its insertion directly. We have met the first type of enforcement in section 2.2: If a case is among the preferred valences of a finite verb or absolutive, the link between the noun and the verb is enforced. In addition, the following three syntactic enforcements are used:

1. The linking strength between grammatically congruent nouns is increased (e.g., *tena* $\overset{+}{\leftrightarrow}$ *balena*, *akṣayasya* $\overset{+}{\leftrightarrow}$ *ātmanaḥ*).
2. Links between nominatives not identified as subject (see above) and congruent finite verbs are enforced.
3. In the early phases of learning, indeclinables have a strong influence on the lexical information due to their high frequencies. Therefore, the strength of a link connecting an indeclinable with any other word (except for finite verbal forms) is weakened. This enforcement is only applied when negative rule 2 is not valid.

The parameter values for the four enforcements (including the combination of verb and valence from section 2.2) were estimated using a genetic algorithm. Running this algorithm repeatedly for 100 generations resulted in the following average parameter values: verb - valence: 3.2, congruent nouns or adjectives, nominative - verb: 3, indeclinables: 0.7.

2.3 Evaluation

In this paper, we cannot present a true evaluation because test data for Sanskrit syntax are not available. Therefore, we demonstrate the performance of the parser using two sentences from chapter RĀMĀYAṆA, BĀLAKĀṆḌA 9, which was excluded from the training set, and the popular benchmark sentence *pramāṇabhūta ācāryo darbhapavitrapāṇiḥ prāṅmukhaḥ śucau avakāśe upaviśya mahatā yatnena sūtraṃ praṇayati sma.* The two sentences from RĀMĀYAṆA are:
RĀM, Bā, 9, 6: *śrutvā tatheti rājā ca pratyuvāca purohitam.*
and
RĀM, Bā, 9, 32: *evaṃ sa nyavasat tatra sarvakāmaiḥ supūjitaḥ.*
Figure 1 shows the results of parsing the two sample phrases from the RĀMĀYAṆA using Yuret's basic model. As mentioned on page 3, the parser is not able to identify even the basic structures of the sentences, possibly due to the small number of training data. In addition, the strong influence of indeclinables on the dependency structure is clearly discernible. In Figure 2, the same two sentences are parsed with grammatical information (2.2), valences (2.2), smoothing (2.2) and syntactic *enforcements* (2.2) activated. Although the parser is still far from able to identify the correct structure of the sentences, it found some important substructures such as *pratyuvāca* ↔ *purohitam*, *tathā* ↔ *iti* (*iti* terminating a direct speech) and the complex made of two composites in the second sentence. As becomes apparent from intermediary stages of learning not reproduced in this paper, the detection of the last substructure was especially influenced by the valence dictionary. In the last test, whose results are displayed in Figure 3, all optimizations are activated. Now, each of the sentences contains only one error. The word *ca* should probably not be connected to the head verb in the first sentence. In the second sentence, *tatra* remains unconnected to the rest of the sentence (but could, of course, easily be associated with *nyavasat* after finishing the parsing process).
Parsing the "benchmark sentence" *pramāṇabhūtaḥ . . .* results in equally good analysis (cmp. Figure 4). After fixing *praṇayati* as the head verb, the algorithm connects the absolutive *upaviśya* to the verb and selects the composite ending in *-pāṇiḥ* as the subject of the sentence. Here, a human user would certainly select *ācāryaḥ*, and, in some runs of the learning process, this word is indeed marked as the subject of the sentence. These differences can be explained by the heuristic nature of the learning process and can perhaps be amended by running the process repeatedly with different initalizations and then averaging the results. Among the remaining substructures, attention should be paid to the adverbial expressions modifying the absolutive and the head verb. Both adverbial structures are connected to the right verb and are, in addition, sorted correctly (*yatnena* modifies *praṇayati* and is itself modified by *mahatā*, etc.). How the nominatives in the beginning of the sentence are connected remains open to discussion even for a human user. However, it should be noted that the parser correctly associates the directional adjective *prāṅmukhaḥ* with the absolutive *upaviśya* and not with the preceeding and congruent nominative *-pāṇiḥ*. On the whole, the few

restrictions introduced by the fixed syntactic rules clearly improve the analysis of the sentences.

3 Summary

In spite of the appealingly simple idea on which it is based, Yuret's parser is not able to correctly identify the syntactic structures of Sanskrit sentences. This behavior may be due to lack of training data. The performance of the parser can be improved when additional, non-lexical information about grammar and valences is included in the parsing process. The best performance is achieved when lexical and grammatical information is combined with a small number of fixed rules. These rules describe the basic components and structures of a complete sentence, but they are by far less detailed than the finite automata used by some researchers. Judging from the few samples that we have discussed in Section 2.3, such a hybrid approach can certainly be used as a starting point for building a database of the syntactic structures of classical Sanskrit. The strict projectivity of the dependency tree assumed in Yuret's original version of the algorithm remains an unsolved problem especially in the context of versified Sanskrit. In a future version of the parser, one may allow crossing links in the dependency structure if, for example, both links have a very high mutual information.

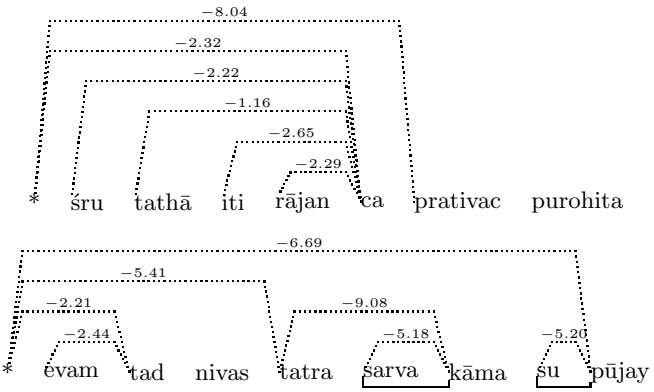


Fig. 1. Sample phrases parsed using Yuret's method – The numbers give the logarithm of the conditional lexical probability of two words.

References

1. T. Brants. TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference*, Seattle, 2000.

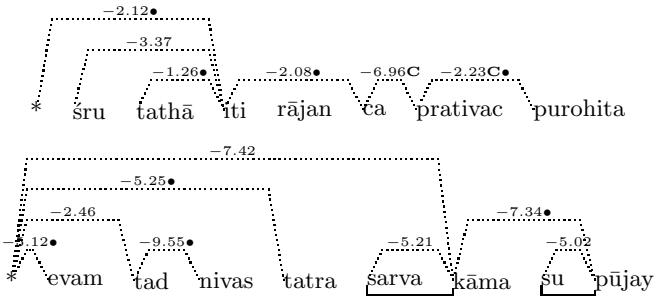


Fig. 2. Sample phrases parsed using all optimizations except for fixed syntactic rules – • = syntactic enforcement, **C** = probability estimated by smoothing

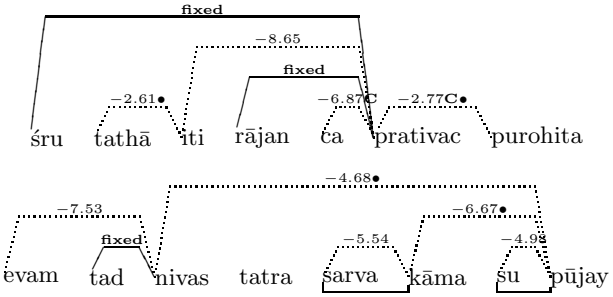


Fig. 3. Sample phrases parsed using all optimizations – Symbols are explained in the caption of Figure 2.

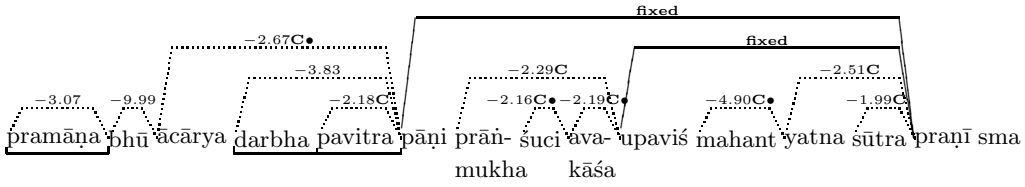


Fig. 4. "Benchmark sentence" parsed with all optimizations activated

2. José Canedo. *Zur Wort- und Satzstellung in der alt- und mittelindischen Prosa*. Vandenhoeck & Ruprecht, Göttingen, 1937.
3. Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69, 1999.
4. B. Delbrück. *Altindische Syntax*. Verlag der Buchhandlung des Waisenhauses, Halle, 1988.
5. Jan Gonda. *Old Indian*. Handbuch der Orientalistik, Zweite Abteilung, Erster Band, Erster Abschnitt. E.J. Brill, Leiden, 1971.
6. Pawan Goyal, Vipul Arora, and Laxmidhar Behera. Analysis of Sanskrit text: Parsing and semantic relation. In *Proceedings of the First International Sanskrit Computational Linguistics Symposium*, pages 23–36, 2007.
7. Oliver Hellwig. `SanskritTagger`, a stochastic lexical and POS tagger for Sanskrit. In *Proceedings of the First International Sanskrit Computational Linguistics Symposium*, pages 37–46, Rocquencourt, 2007.
8. Ingeborg Ickler. *Untersuchungen zur Wortstellung und Syntax der Chāndogyopaniṣad*. Göppinger Akademische Beiträge, 75. Verlag Alfred Kümmerle, Göppingen, 1973.
9. J.S. Speyer. *Vedische und Sanskrit-Syntax*. Grundriss der Indoarischen Philologie und Altertumskunde, III. Band, Heft A. Verlag von Karl J. Trübner, Strassburg, 1896.
10. J.F. Staal. *Word Order in Sanskrit and Universal Grammar*. Foundations of Language, Supplementary Series, Volume 5. D. Reidel Publishing Company, Dordrecht, 1967.
11. Oliver Wauschkuhn. *Automatische Extraktion von Verbvalenzen aus deutschen Textkorpora*. Shaker Verlag, Aachen, 1999.
12. Deniz Yuret. *Discovery of Linguistic Relations Using Lexical Attraction*. PhD thesis, Massachusetts Institute of Technology, 1998.