Anaphor Resolution in Sanskrit: Issues and Challenges

Madhav Gopal[1] and Girish Nath Jha[2]
[1]Centre for Linguistics, SLL & CS
[2]Special Centre for Sanskrit Studies
Jawaharlal Nehru University, New Delhi
{mgopalt, girishjha}@gmail.com

Abstract

This paper aims at presenting a study of anaphora and cataphora phenomena in Sanskrit based on Pañcatantra and their mechanical resolution. The scope of anaphors is limited to lexical anaphors (reflexives and reciprocals) in this study. The paper examines the structure of anaphor usage, their linguistic contribution and conditions when they are used. On the basis of this study we formulate an algorithm to resolve anaphors and cataphors automatically present in the text. The resolution would be done using POS tagged text of the language. The process of anaphora resolution would consist of three main steps: identification of the anaphors/cataphors, location of the candidates for their antecedents and matching the appropriate antecedent(s) from the set of candidates. The system will take as input tagged Sanskrit text and will produce an output with anaphors and their antecedents paired.

The style of Sanskrit texts is unique and this uniqueness has its own problems. The punctuation marking in Sanskrit texts is bizarre; they do not use any kind of reliable punctuation. Originally, Sanskrit had no punctuation. Apart from this, it is a highly synthetic language and the word boundaries in spoken as well as in written forms are often faded away due to intense concatenation. To identify word boundaries, especially in written form, is a challenge. Moreover, Sanskrit pronouns are sometimes compounded with nouns and participles. We have studied that roots of pronouns, possessive pronouns of all persons, and roots of lexical anaphors undergo compounding. In the process of compounding only root of the pronoun is left and the case, number, and gender features are dropped. And, this creates problems in identifying their antecedents, as the grammatical features are main clues for pairing anaphor and antecedents in our approach. A slightly uneasy problem in Sanskrit anaphora resolution is that Sanskrit has pronoun forms in abundance. Apart from the regular inflections of pronouns, the language has many strategies to encode pronominal information. All these forms have to be taken care of while designing a system. Also, some third person pronoun, relative and demonstrative forms are homophonous with some conjunctions in the language. These forms include ***tasmāt, tat, yat,*** and ***yena*** which have been used in the text in question. These words serve as linkers and they join the preceding sentence/clause to the following sentence. They must be disambiguated first while developing a resolution system. All these issues will be discussed in the paper at length.

The Sanskrit Anaphora Resolution System (SARS) Algorithm

The system currently considers only lexical anaphors (reflexives and reciprocals) and it identifies intra-sentential antecedents of these anaphors in the input text. The algorithm is based on our study of lexical anaphors available in the Pañcatantra. The input to the anaphora resolution system is the POS tagged (with MSRI Indic language tagset) text.

Algorithm for Reflexive Anaphora

1. Tokenize each sentence (S) of the input text.
2. Pick up the S in which PRF tag occurs.
3. Check whether the S has NP, NC, PPR, or PRL.
4. Consider all the NP, NC, PPR, and PRL in the S that precede the PRF.
5. Check whether NP, NC, PPR, or PRL has/have .nom tag.
6. If one of the NP, NC, PPR, or PRL has .nom tag, then that word is identified as the antecedent of the PRF.
7. If more than one of these have .nom tag then the nearest to PRF would be its antecedent.
8. If the conditions 6 or 7 are not met then the NP, NC, PPR, or PRL having .ins tag would be considered the antecedent of the PRF.

Algorithm for Reflexive Cataphora

9. If the S does not have any preceding NP, NC, PPR, or PRL with .nom tag then consider the following NP, NC, PPR, or PRL containing .nom tag.
10. If one of the following NP, NC, PPR, or PRL has .nom tag, then that word is identified as the antecedent.
11. If more than one of these have .nom tag, then the nearest to the PRF would be its antecedent.
12. If the conditions 9 or 10 are not met then the NP, NC, PPR, or PRL having .ins tag would be considered the antecedent of the PRF.

Algorithm for Reciprocal Anaphora

1. Tokenize each sentence (S) of the input text.
2. Pick up the S in which the PRC tag occurs.
3. Check whether the S has NP, NC, PPR, or PRL.

4. Consider all the NP, NC, PPR, and PRL in the S that precede the PRC.

5. Check whether NP, NC, PPR, or PRL has/have .du.nom or .pl.nom tag.

6. If one of the NP, NC, PPR, or PRL has .du.nom or .pl.nom tag, then that word is identified as the antecedent of the PRC.

7. If more than one of these have .du.nom or .pl.nom tag then the nearest to the PRC would be its antecedent.

8. If all of them are containing .sg.nom tag then all of them would be antecedents of the PRC.

9. If the conditions 6, 7 or 8 are not met then the NP, NC, PPR, or PRL having .du.ins or .pl.ins tag would be considered the antecedent of the PRC.

Algorithm for Reciprocal Cataphora

10. If the S does not have any preceding NP, NC, PPR, or PRL then consider the following NP, NC, PPR, or PRL containing .du.nom or .pl.nom tag.

11. If one of the following NP, NC, PPR, or PRL has .du.nom or .pl.nom tag, then that word is identified as the antecedent of the PRC.

12. If more than one of the following NP, NC, PPR, or PRL have .du.nom or .pl.nom tag, then the nearest to the PRF would be its antecedent.

13. If all of the following NP, NC, PPR, or PRL have .sg.nom tag, then all of them would be collectively identified as the antecedent of the PRC.

14. If the conditions 11, 12 or 13 are not met then the NP, NC, PPR, or PRL having .du.ins or .pl.ins tag would be considered the antecedent of the PRC.