Chapter 4

Compound Segmenter[1]

The task of a compound segmenter is to split a compound into its constituents. The segmentation of a compound requires the sandhi rules and a morphological analyser which can validate the splits. If we see Panini's Astadhyayi, we find the rules only for sandhi synthesizing not for splitting and sandhi is mandatory in compounds. The compound segmenter uses the same sandhi rules for splitting. Sandhi rules are in the form of triples (x,y,z) where x is the last letter of the first component and y is the first letter of second component. z contains the euphonic changes of x+y. For instance in the compound word सूर्योदयः (The rising of Sun), the word सूर्य is the first component and the word उदयः is the second component. The letter 'अ' is the last letter of the word सूर्य which is x and the letter 'उ' is the first letter of the word उदयः which is y. The z is x+y which is 'ओ'. To split a sandhi thus we need the sandhi rules in reverse form, viz given z, what are the possible values of x and y. The sandhi

[1]This work is jointly done with Vipul Mittal, an MS student of IIIT-Hyderabad.

rules are deterministic, i.e, given x and y, z is unique [with an exception of some विभाषा, where sandhi is optional, and thus x and y are unchanged in these cases]. These sandhi rules when inverted, lead to non-determinism. To illustrate, consider the following sandhi rules

a + a -> ā

a + ā -> ā

ā + a -> ā

ā + ā -> ā

When inverted, given 'ā', now there are 4 different ways of splitting it. All these possible splits should be constrained further to be morphologically valid. As an example following these rules, the string 'tatrāpi' can be split in 4 different ways as

tatra + api

tatra + āpi

tatrā + api

and tatrā + āpi

Of these only the first one will be validated by the morphological analyser and the other three answers wil be discarded. The above two words viz tatra and api are padas themselves and in case of tatrāpi mere sandhi has taken place. This is not an example of compound.

Issues in identifying the components of a compound :

When a compound is split, among the resulting components only the last one has a vibhakti and hence can be recognised by a morphological

analyser. But what about the components in the initial position? The compound रामालयः, suppose split as राम-आलयः, आलयः will be recognised by the morphological analyser. राम will also be analysed as a संबोधन form of राम. But as is obivious, in case of रामालयः, राम is not in the संबोधन form. Here राम is in its compound initial form (समासपूर्वपद). There are various similar issues related to identification of components of a compound, which are described below.

(a) **Identification of** समासपूर्वपद :- The component undergoes various morphological changes when it is used as a समासपूर्वपद. These changes are :

(i) Deletion of न् from प्रातिपदिक :- The 'न्' at the end of a प्रातिपदिक gets deleted when used as a पूर्वपद. For instance राजन् becomes राज in राजपुत्रः.

(ii) ह्रस्वविधान - In some cases the पूर्वपद changes to ह्रस्व. For instances the word इष्टका becomes इष्टक in इष्टकचितम्[2].

(iii) दीर्घविधान - In some cases the पूर्वपद gets दीर्घ. For instance the word भ्रु becomes भ्रू in भ्रूकुटिः[3]

(iv) आदेशविधान - In some cases the पूर्वपद gets substitute by another word. For instace the word अङ्गुलि becomes अङ्गुल in अङ्गुलाकर्णः, the word हृदय becomes हृद् in हृल्लेखः etc.

(b) **Identification of** समास-उत्तरपद :-

(i) **Bound morphemes** :- In case of उपपद-तत्पुरुष, we come across bound morphemes such as कारः, ज्ञः, जः etc as in कुम्भकारः, स-

[2]"इष्टकेषीकामालानां चिततूलभारिषु" (6-3-65)

[3]"इको ह्रस्वोऽङ्योः गालवस्य" (6-3-61) - सि० कौ०

वज्ञः, अग्रजः etc. The components कारः, ज्ञः, जः are not पदs, they do not have an independent exisence. But in case of a compound identification, these are to be recognised. कार, ज्ञ, ज being the प्रातिपदिकs, take all the possible vibhaktis and thus follow the regular paradigms.

(ii) **Change in gender** :- Two types of compounds viz the अव्ययीभाव and बहुव्रीहि may advocate gender changes in the final components. For example in उपगङ्गम्, the noun गङ्गा which is in feminine gets neuter gender due to the अव्ययीभाव compound formation. Similarly in case of पीताम्बरः, the second compound अम्बरः is in masculine gender whereas the gender of अम्बर is neuter.

(iii) **Change in number**:- Consider the word अनेकान्. This will be split by a compound splitter as अन्-एकान्. Here the पूर्वपद अन् should be recognised as a variation of न and एकान् should be recognised morphologically. Though एक is used many senses, in this context एक is used to indicate the संख्या(number) and hence can't have एकान् form which is in plural. But it needs to be recognised. So this is to be treated as a special case.

(iv) **Change in paradigms** :- In case of न अस्ति किञ्चन यस्य = अकिञ्चनः[4], we require a new paradigm to recognise the word किञ्चन.

To handle these phenomenon, a seperate module for morphological analysis is added. This module is invoked only on समासपूर्वपद and समासोत्तरपदs avoding the overgeneralisation.

[4]पा०सू० - ``मयूरव्यंसकादयश्च'' (2-1-72)

4.1 Segmenter

The task of a compound segmenter is to split a compound into its constituents. The segmentation of a compound requires the sandhi rules and a morphological analyser which can validate the splits. The segmenter uses reversed sandhi rules and first produces all the possible splits. These splits are then validated by morphological analyser and then the correct splits are selected. This filter produces multiple possibilities. These need to be ranked so that most probable answers are easily accessible. We describe below the GENerate-CONstrain-EVALuate cycle of the segmenter attributed to the optimality theory.

4.1.1 Scoring Matrix

A parallel corpus of Sanskrit text in sandhied and unsandhied form is being developed as a part of the Sanskrit Consortium project in India. The corpus contains texts from various fields ranging from children stories, dramas, purāṇas to Ayurveda texts. From around 100K words of such a parallel corpus, 25K words were found to be in sandhied forms. These 25K parallel instances of sandhied and unsandhied text were extracted and were used to get the frequency of occurence of various sandhi rules. If no instance of a sandhi rule is found in the corpus, for smoothing, we assign the frequency of 1 to this sandhi rule.

We define the estimated probability of the occurrence of a sandhi rule as follows :

Let R_i denote the i^{th} rule with f_{R_i} as the frequency of occurrence in the

manually split parallel text. The probability of rule R_i is :

$$P_{R_i} = \frac{f_{R_i}}{\sum_{j=1}^{n} f_{R_j}}$$

where n denotes the total number of sandhi rules found in the corpus.

Let a word be split into a candidate S_j with k constituents as $< c_1, c_2, ..., c_k >$ by applying $k-1$ sandhi rules $< R_1, R_2, ..., R_{k-1} >$ in between the constituents. It should be noted here that the rules $R_1, ..., R_{k-1}$ and the constituents $c_1, ..., c_k$ are interdependent since a different rule sequence will result in a different constituents sequence. The sequence of constituents are constrained by a language model whereas the rules provide a model for splitting. We define two measures each corresponding to the constituents and the rules to assign weights to the possible splits.

4.1.1.1 Language Model

Let the unigram probability of the sequence $< c_1, c_2, ..., c_k >$ be PL_{S_j} defined as :

$$PL_{S_j} = \prod_{x=1}^{k} (P_{c_x})$$

where P_{c_x} is the probability of occurrence of a word c_x in the corpus.

4.1.1.2 Split Model

Let the splitting model PS_{S_j} for the sandhi rules sequence $< R_1, R_2, ..., R_{k-1} >$ be defined as :

$$PS_{S_j} = \prod_{x=1}^{k-1}(P_{R_x})$$

where P_{R_x} is the probability of occurrence of a rule R_x in the corpus.

Therefore, the weight of the split S_j is defined as the product of the language and the split model as :

$$W_{S_j} = \frac{PL_{S_j} * PS_{S_j}}{k}$$

where the factor of k is introduced to give more preference to the split with less number of segments than the one with more segments.

4.1.2 Segmentation Algorithm

The approach followed is GENerate-CONstrain-EVALuate. In this approach, all the possible splits of a given string are first generated and the splits that are not validated by the morphological analyser are subsequently pruned out. Currently we apply only two constraints viz.

- C1 : All the constituents of a split must be valid morphs.
- C2 : All the segments except the last one should be valid compounding forms.

The system flow is presented in Figure 4.1.

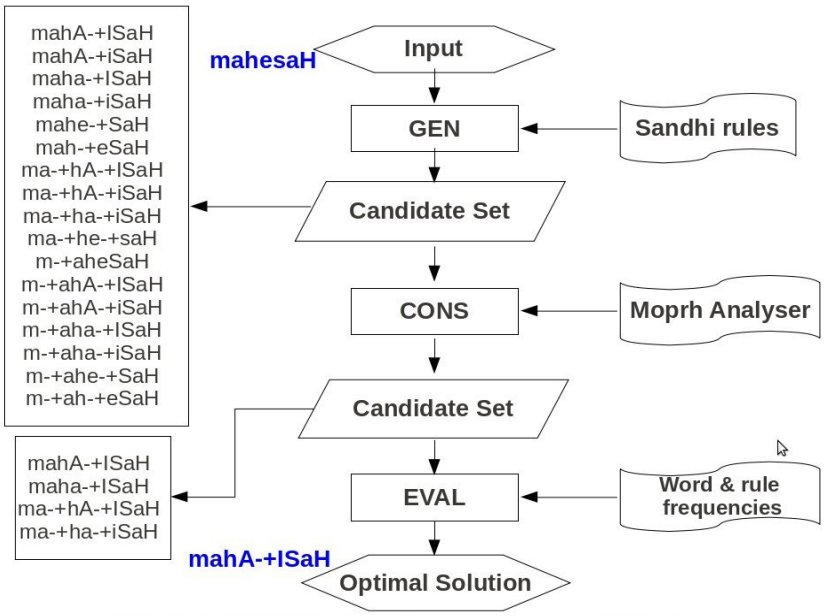

Figure 4.1 : Compound Splitter : System Data Flow.

The basic outline of the algorithm is :

1. Recursively break a word at every possible position applying a sandhi rule and generate all possible candidates for the input.
2. Pass the constituents of all the candidates through the morph analyser.
3. Declare the candidate as a valid candidate, if all its constituents are recognised by the morphological analyser, and all except the last segment are compounding forms.
4. Assign weights to the accepted candidates and sort them based on the weights as defined in the previous subsection.
5. The optimal solution will be the one with the highest weight.

4.1.2.1 Results

The current morphological analyser[5] can recognise around 140 million words. Using 2,650 rules and a test data of around 8,260[6] words parallel corpus for testing, we obtained the following results :

- Almost 92.5% of the times, the first segmentation is correct. And in almost 99.1% of the cases, the correct split was among the top 3 possible splits.
- The precision was about 92.46% (measured in terms of the number of words for which first answer is correct w.r.t. the total words for which correct segmentation was obtained).
- The system consumes around 0.04 seconds per string of 15 letters on an average.[7]

The complete rank wise distribution is given in Table 4.1.2.1.

Rank	% of words
1	92.4635
2	5.0492
3	1.6235
4	0.2979
5	0.1936
>5	0.3723

Rank-wise Distribution

[5]available at http://sanskrit.uohyd.ernet.in/scl/morph/index.html.
[6]The test data is extracted from manually split data of Mahābhāratam.
[7]Tested on a system with 2.93GHz Core 2 Duo processor and 2GB RAM.