Supplementary material for Sanskrit Sandhi Splitting using $seq2(seq)^2$

Rahul Aralikatte IBM Research
Neelamadhav Gantayat IBM Research
Naveen Panwar IBM Research
{rahul.a.r, neelamadhav, naveen.panwar}@in.ibm.com

Anush Sankaran IBM Research
anussank@in.ibm.com

Senthil Mani IBM Research
sentmani@in.ibm.com

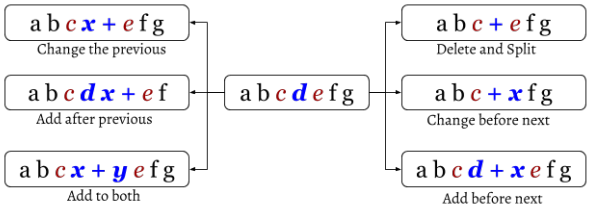

Figure 1: An example illustrating the different kinds of syntactical splits and the challenges for a sequence learning algorithm.

A Sandhi splitting challenges

Some of the major challenges faced by existing Sandhi splitting tools are briefly described below to motivate the hardness of the problem:

1. **Identifying multiple locations of split:** Identifying the location in a word where split has to be performed is the most challenging problem in performing splitting. As shown in Figure 1, transformation can happen in any location and in any form. Further, sandhi splitting involves identifying multiple potential locations, and validating them based on the previous locations.
2. **Cascading split effect:** There are some rules in which the effect of a split is not merely restricted to the immediate vicinity (neighboring characters). For example, in *uttarāyaṇa* → *uttara* + *ayana*, the *r* of *uttara* changes the *ṇ* of *ayana* to *n*.
3. **Samāsa:** The process of *Samāsa* is a process similar to Sandhi where words come together by discarding majority of their characters. A subset of the rules governing *Samāsa* overlaps with Sandhi. Thus, Sandhi splitters need to maintain two rule sets to correctly identify the constituent words. Existing systems require the user to explicitly pass the intermediate results back to it to perform the splitting correctly. For example, existing systems correctly split the word with a *Samāsa* *lakṣyasyārthatvavyavahārānurodhena* to form *lakṣyasya* + *arthatvavyavahāra* + *anurodhena*. However the second word, *arthatvavyavahāra*, contains a Sandhi and must be sent back into the system to get its constituent words.
4. **Incomplete rule set:** Though most of the splitting rules can easily be identified, there are many nuances which are often difficult to handle. There are also some rules which occur very rarely. For example, *sa yogī* → *saḥ* + *yogī*. Incomplete rule set during splitting will result in false negatives, such as, none of the existing splitters split (*a* + *chedyaḥ* → *acchedyaḥ*), correctly as *a* may not have the associated rule captured. Thus, heuristically defining all the splitting rules will be intractable, while learning them from examples is more generalizable.