Significance of an Accurate Sandhi-Splitter in Shallow Parsing of Dravidian Languages

Devadath V V **Dipti Misra Sharma**
Language Technology Research Centre
International Institute of Information Technology - Hyderabad, India.
devadathv.v@research.iiit.ac.in
dipti@iiit.ac.in

Abstract

This paper evaluates the challenges involved in shallow parsing of Dravidian languages which are highly agglutinative and morphologically rich. Text processing tasks in these languages are not trivial because multiple words concatenate to form a single string with morpho-phonemic changes at the point of concatenation. This phenomenon known as *Sandhi*, in turn complicates the individual word identification. Shallow parsing is the task of identification of correlated group of words given a raw sentence. The current work is an attempt to study the effect of *Sandhi* in building shallow parsers for Dravidian languages by evaluating its effect on Malayalam, one of the main languages from Dravidian family. We provide an in-depth analysis of effect of *Sandhi* in developing a robust shallow parser pipeline with experimental results emphasizing on how sensitive the individual components of shallow parser are, towards the accuracy of a sandhi splitter. Our work can serve as a guiding light for building robust text processing systems in Dravidian languages.

1 Introduction

Identification of individual words is crucial for the computational processing of a text. In Dravidian languages, word identification becomes complex because of *Sandhi*. *Sandhi* is a phenomenon of concatenation of multiple words or characters to form a single complex string, with morpho-phonemic changes at the points of concatenation (VV et al., 2014). The morphological units that can be concatenated in a *Sandhi* operation can belong to any linguistic class: a noun joins with a verb or a postposition or particle, a verb joins with other verbs, auxiliaries, connectives, adverbs combining with verbs and so on. The phenomenon is different from a noun compound multi-word expression in that the noun-compound concatenations are semantics-driven. Whereas *Sandhi* is not semantically driven but phonologically driven. This leads to misclassification of classes of words by pos-tagger which eventually affects parsing. Hence making shallow parsers for Dravidian languages is a challenging task.

Shallow parsing (Abney, 1992) is a task of automatic identification of correlated group of words (chunks) which reduces the computational effort at the level of full parsing by assigning partial structure to a sentence. To be precise, chunks are correlated group of words which contain only the head or content word and its modifiers. Shallow parser is not a single module but is a set of modules (Hammerton et al., 2002) with tokeniser, parts-of-speech tagger (pos-tagger) and chunker put in a pipeline. It has been experimentally proved that shallow parsers are useful in both text (Collins, 1996), (Buchholz and Daelemans, 2001) and speech processing domains (Wahlster, 2013).

The current work aims to give an in-depth analysis on the effect of *Sandhi* in shallow parsing of Dravidian languages with a focus on Malayalam, the most agglutinative language (Sankaran and Jawahar, 2013) in the Dravidian family. For the purpose of analysis, we chose to create our own pos-tagger and chunker trained on a new 70k words annotated corpus with word internal features of morpho-phonological nature particularly because *Sandhi* evolved out of morpho-phonological reasons.

In this paper, for the first time in the literature, we evaluate the impact of *Sandhi* and the resultant error propagation in shallow parser for Dravidian languages. In this work, we compare the

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics – Student Research Workshop, pages 37–42,
Berlin, Germany, August 7-12, 2016.

performances of pos-tagger, and chunker on a gold standard sandhi-split test data and how the error of a sandhi-splitting tool propagates to other components of shallow parsing pipeline. We have released the 70k annotated data and the trained models of pos-agger, chunker and shallow parser described in this paper[1].

2 Sandhi in Dravidian languages

Sandhi is a very common phenomenon in Sanskrit and Dravidian languages. Even though many languages exhibit agglutinative properties in morphemes, in these languages, this goes beyond the morphemes and agglutinates words with morpho-phonemic change. For example,

(1) **avanaareyaaN snEhikkunnath ?**
'avan_ aare aaN snEhikkunnath ?
'he whom is loving ?
'whom he is loving?

Example 1 is a valid sentence from Malayalam. There are two strings, *avanaareyaaN* and *snEhikkunnath*. Here second string is a single word but first string, *avanaareyaaN* is a combination of 3 sub-strings or words; *avan_*, *aare* and *aaN*. The last character "*n_*" of *avan_* is a pure consonant which can stand alone without the help of a vowel. When this word joins with the next word *aare*, "*n_*" of *avan_* becomes a normal consonant by joining with the first character "*aa*" of *aare*. When *aare* joins with the word *aaN*, an insertion of an additional character "*y*" happens, and together they form *avanaareyaaN*.

$$avan\_ + aare + aaN \rightarrow ava\mathbf{naa}re\mathbf{yaa}N$$

Sandhi happens in Dravidian languages at two levels. One is at morpheme level and other is at word level. In morpheme level, stem or root(s) join with the affixes to create a word along with morpho-phonemic changes as explained above. This is considered as *Internal Sandhi*. *Sandhi* between words as in example (1) is known as *External Sandhi*. For this work, *External Sandhi* is the matter of concern because this makes the individual word identification difficult.

[1]Data and Models created during this project can be found in this link `https://github.com/Devadath/Malayalam-Shallow-Parser`

3 Sandhi-Splitter

Sandhi-splitter is a tool which splits a string of conjoined words into a sequence of individual words, where each word in the sequence has the capacity to stand alone as a single word. To be precise, sandhi-splitter facilitates the task of individual word identification within such a string of conjoined words.

To the best of our knowledge, only 2 works have been published on Malayalam sandhi-splitter with proper empirical results (VV et al., 2014), (Kuncham et al., 2015). For all the experiments, we used the former sandhi splitter since the accuracy is better than the latter one. This method applies Bayesian methods at character level to find out the precise split points and used hand-crafted rules to induce morpho-phonemic changes. The remaining sections of this paper focus on an in-depth analysis of effect of *Sandhi* in shallow parsing in the language of Malayalam.

4 Effect of *Sandhi* in Shallow Parsing

4.1 An overview of Shallow Parser pipeline

A typical architecture of a shallow parser has three main modules namely sandhi-splitter, pos-tagger and chunker. Input to the shallow parser is a raw sentence and the output is a chunked text with its pos and chunk information of every word present in it. The diagram of the architecture is given in Figure 1.

Figure 1: Pipeline Architecture

4.2 Effect of *Sandhi* in POS-Tagging

As mentioned in Section 1, *Sandhi* happens between words from different grammatical categories.

(2) **addEhamoraddhyaapakanaaN**
'addEhaM oru addhyaapakan_ aaN
'He one teacher(male) is
'He is a teacher

Example 2 is a sentence in Malayalam with 4 words in it. Here first word *addEhaM* is a pronoun, second word *oru* is quantifier, third word *addhyaapakan_* is a noun and fourth word *aaN* is a copula. But these words are agglutinated

to become a string of words *addEhamoraddhyaapakanaaN*, making the system incapable of identifying individual words, consequently resulting in an erroneous POS tagging.

(3) **varumen_kil_**
'varuM en_kil_'
'come.FUT if'
'if comes'

In example 3, there are 2 different words from different grammatical classes. *varuM* is a finite verb whereas *en_kil_* is a conjunction and they are agglutinated together to form *varumen_kil_* which should not be tagged by a single pos-tag.

Because of the morphological richness, morphological features like root, prefix and suffix information are helpful features in identifying the grammatical category of a word. In order to evaluate the effect of *Sandhi* we decided to create a pos-tagger which uses both word-external (context) and word-internal features (morphological features) because a word is a result of *Sandhi* between morphemes like prefix, suffix, root, stem etc. For incorporating these features we used CRF (Lafferty et al., 2001) for building pos-tagger. We created a pos-annotated news corpus of 70k words by manual efforts along with bootstrapping. The pos information is incorporated after splitting and validating the *Sandhi*. The Tag-set we used to annotate is BIS tag-set[2], specially designed for Indian languages. Since morph analyzers are not available for Malayalam, in order to capture the prefix and suffix information, certain number of characters from the beginning and end of a word are used as features for POS Tagging. Table 1 shows the features used for pos-tagging.

Features
W_{-1}, W_0, W_1,
W[0], W[0:2], .., W[0:5],
W[-5], W[-5:-3], .., W[-5:EOW]

Table 1: Features for our POS Tagger. W is Word and W[1:m] refers to character at indexes. W_x refers to word at relative position of x with respect to current position. 'EOW' refers to End Of Word.

4.3 Effect of *Sandhi* in Chunking

Chunks are identified based on the pos-tags of words. Since a chunk is a group of a head word and its modifiers, they are meaningful subsets of a sentence. But if the individual words are not correctly identified, inappropriate pos-tags will be assigned and meaningless chunks will be created. There are 4 words and 3 chunks in example 2. [*addEhaM* PRP]NP, [*oru* QT *addhyaapakan_* NN]NP, and [*aaN* VM]VP. If the *Sandhi* is not identified and individual words are not extracted, system will fail to identify the meaningful sub-parts of a sentence like chunk/phrase/constituents. Similarly in example 3, the string *varumen_kil_* has two words and two chunks [*varuM* VM]VP and [*en_kil_* CC]CCP. Hence processing *Sandhi* in the first stage is extremely important in any NLP task for Dravidian languages.

For evaluating the effect of *Sandhi* in chunker, we decided to create a chunker. We incorporated chunk information using IIIT-tagset (Bharati, 2006) in the data annotated for pos-tagging, which is a corpus of 70k words. Table 2 shows the features we used for the chunker. Each feature is composed of a word and its corresponding POS tag.

Features
W_{-2}/POS_{-2},..,W_0/POS_0,..,W_2/POS_2

Table 2: Features for our Chunker. W is Word and POS refers to POS tag.

5 End to End Shallow parser

Shallow parser is a set of modules comprising of sandhi-splitter, pos-tagger and chunker in order. A raw text will be given as the input and the sandhi-splitter identifies individual words, pos-tagger assigns pos-tags to each word and chunker groups them to chunks and outputs the chunked sentence as shown in Figure 2.

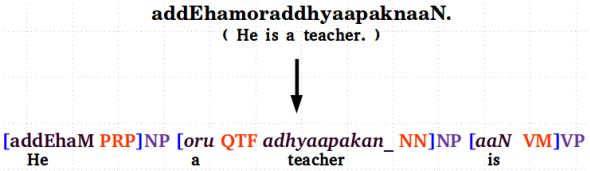

Figure 2: Example of a raw input and the subsequent chunked output

5.1 Data

We have used the manually created 70k pos and chunk annotated corpus which we have already

[2]http://tinyurl.com/hhllsky

mentioned in 4.2 and 4.3. Out of 70k data, 8k data has been taken as test data and remaining 62k as training data for pos-tagging and chunking. Since creating training data for sandhi-splitter is a laborious task, we used the sandhi-annotated training data of size 2k words used by (VV et al., 2014). Whereas test data will be the same 8k data which were employed for pos tagging and chunking.

5.2 Experiments

Two types of experiments have been conducted to evaluate the error propagation of the Malayalam shallow parser pipeline. In the first type of experiments, individual modules in the pipeline are considered as independent of the output of previous modules. In the second type of experiment individual modules are considered as dependent on the output of previous modules.

5.2.1 Experiment Type - 1

In this experiment, input to each module will not be affected by the performance of its previous modules. This experiment evaluates the performance of all the individual modules with respect to the current train and test data. Table 3 presents the results.

Module	P	R	F-1	A
Sandhi Splitter	91.77	62.95	74.68	88.46
POS tagger	90.45	90.49	90.47	90.45
Chunker	88.47	91.55	89.98	92.92

Table 3: Results of Experiment Type- 1 : Results of individual modules where each module will not be affected by the performance of its previous modules. Here, 'P' refers to Precision, 'R' to Recall, 'F-1' to F-Measure and 'A' to Accuracy

5.2.2 Experiment Type - 2

In these experiments, output of one module will be given as input to the next module, hence the performance of the previous module affects the next module. These experiments are to evaluate the error propagation from each module which eventually affects the final output. Here the evaluation of the pos-tagger is done based on the number of words which got correctly identified by the sandhi-splitter and then got the correct pos-tags by the pos-tagger. A chunk can be a word or a group of words. Hence a chunk is considered as correct only when there are exact number of words in the chunk where all the words in it should meet the criteria for the evaluation of pos-tagger. Shallow Parser pipeline evaluation scores are given in Table 4.

S	S+P	P+C	S+P+C
88.46	79.87	81.88	71.38

Table 4: Results of Experiment Type- 2: Pipeline accuracies where the performance of previous modules affect the subsequent modules. Here 'S' refers to sandhi splitter, 'P' to pos-tagger and 'C' to chunker. "+" indicates that the output of the previous module is given as the input to the next module.

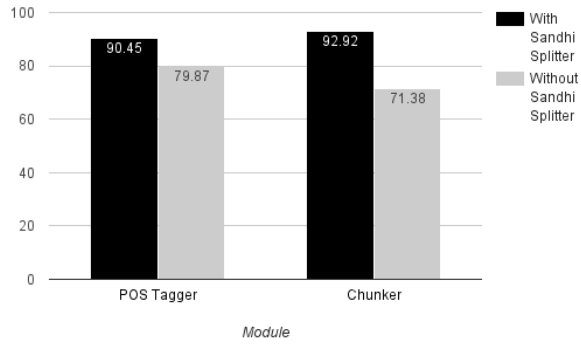

Figure 3: Comparison of accuracies of POS tagger and Chunker with and without Sandhi splitter

Error propagation due to the performance of sandhi-splitter is very high when compared to other modules. Accuracy of the pos-tagger, came down to 79% from 90% due to the errors caused by sandhi-splitter and further this brought down the accuracy of chunker to 71% from 92%.

6 Analysis of Experiment Type-2 Results In Various Modules

6.1 Sandhi-Splitter

We have 2 types of errors created by the sandhi splitter.

1. Not splitting a token which has to be split into words.
2. Splitting a token which should not have been split.

In this experiment, error 1 is more prevalent than error 2. For example, *aRivilla* (no knowledge)

should have been split into *aRiv* (knowledge) and *illa* (no). But the system failed to do so. The mentioned problem is due to the lack of diverse patterns in training data. When it comes to error 2, split occurs either between a root and its suffix or just splits in common sandhi split points like *ya, va* or *ma*. The word *aTiccamaRtti* (suppressed) got split into *aTiccaM* and *aRtti* where both the words are meaningless. This problem is also due to the lack of diverse patterns in training data. Another cause of errors are rules employed in Sandhi Splitter for inducing morpho-phonemic changes after split. Though the system correctly identified "n" as split point for *kaalinuLLa (which is for leg)*, but when the rules got applied, this became *kaalin_+uLLa*, where it should have been simply "*n*" which represents a dative case suffix. Whereas *kaalin_* which is meaningless in that context.

6.2 POS-Tagger

The errors in sandhi splitting will eventually affect the performance of pos-tagger in two ways along with sole errors created by pos tagger. Precisely the errors from sandhi splitter has been propagated to pos-tagger, along with errors from pos-tagger. Since it is in a pipeline, wrongly split tokens given to the pos-tagger will have unknown patterns which make the system unable to predict the tag accurately since the pos-tagger uses the morphological features defined based on characters. Subsequently, this will have an impact at the word level context as well. One such instance is where the string *raajaavaaN* (is king) got split into *raajaa* and *aaN*, where it should have been *raajaav* and *aaN*. Here *raajaa* (king) got tagged as adjective and *aaN* (is) ideally a verb but got tagged as a noun, since the previous word got tagged as an adjective.

6.3 Chunker

Errors from both sandhi-splitter and pos-tagger affect the performance of chunker. Errors together from sandhi-splitter and pos-tagger have been propagated to chunker. A chunk is tagged as incorrect when the words and number of words along with their respective pos-tags are not correct. Many instances have 2 or more words per chunk and the chunk-tag is decided based on pos-tags of words. Since it is in a pipeline, two types of errors can propagate,

- Errors due to unidentified or wrongly identified words from sandhi-splitter.
- Errors from pos-tagger, which was affected or unaffected by the errors from sandhi-splitter.

There are many instances where sandhi-splitter could not identify individual words from a token like *aRivilla* (no knowledge). Ideally *aRiv* and *illa*, where the first word is a noun and the other is a verb. Hence there should be a noun chunk (NP) and a verb chunk (VP). Since individual words are not available, pos-tags and chunk-tags will be wrongly identified. Similar would be the case of wrongly identified words.

7 Conclusion and Future Works

In this work we have discussed about experiments conducted to evaluate the significance of an accurate sandhi-splitter in shallow parsing of Dravidian languages, with a focus on Malayalam. We evaluated the performance of individual modules and pipeline with gold standard sandhi-split test data and how the error of a sandhi-splitting tool propagates to other components of shallow parsing pipeline. From the evaluation we found that *Sandhi* severely affects the performance of individual modules and hence the performance of shallow parser. This study validates the the need of a highly accurate sandhi-splitter for all Dravidian languages. As a future work, we propose to work in three main directions.

1. In order to reduce the error propagation in pipelined Shallow parser, joint modeling of a shallow parser is proposed.
2. Investigating further improvements in sandhi-splitting by formulating sandhi-splitting as a statistical machine translation task, where the raw text will be given as the source language and the target language will be the sentences with individual words identified.
3. Since manual creation of annotated data in huge amount is tedious, we plan to apply cross-lingual projection techniques to create Sandhi splitter for all Dravidian languages by exploiting their morphological similarity.

Acknowledgments

The authors are grateful to Pruthwik Mishra for the invaluable help and support during the work. We also would like to thank our colleagues for their critical and invaluable comments while writing the paper. This work has been done as the part of *Taruviithi*, a treebank creation project for Indian languages which is supported by the Department of Electronics and Information Technology (DEITY), India.

References

StevenP. Abney. 1992. Parsing by chunks. In RobertC. Berwick, StevenP. Abney, and Carol Tenny, editors, *Principle-Based Parsing*, volume 44 of *Studies in Linguistics and Philosophy*, pages 257–278. Springer Netherlands.

Akshar Bharati. 2006. Anncorra: Annotating corpora guidelines for pos and chunk annotation for indian languages.

S. Buchholz and W. Daelemans. 2001. Complex answers: A case study using a www question answering system. *Nat. Lang. Eng.*, 7(4):301–323, December.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 184–191. Association for Computational Linguistics.

James Hammerton, Miles Osborne, Susan Armstrong, and Walter Daelemans. 2002. Introduction to special issue on machine learning approaches to shallow parsing. *The Journal of Machine Learning Research*, 2:551–558.

Prathyusha Kuncham, Kovida Nelakuditi, Sneha Nallani, and Radhika Mamidi. 2015. Statistical sandhi splitter for agglutinative languages. In *Computational Linguistics and Intelligent Text Processing*, pages 164–172. Springer.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Naveen Sankaran and CV Jawahar. 2013. Error detection in highly inflectional languages. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1135–1139. IEEE.

Devadath VV, Litton J Kurisinkel, Dipti M Sarma, and Vasudeva Varma. 2014. Sandhi splitter for malayalam. In *Proceedings of International Conference On Natural Language Processing(ICON)*.

W. Wahlster. 2013. *Verbmobil: Foundations of Speech-to-Speech Translation*. Artificial Intelligence. Springer Berlin Heidelberg.