# Statistical Sandhi Splitter and its Effect on NLP Applications

Prathyusha Kuncham

**IIIT-Hyderabad** 

Kovida Nelakuditi Prathyusha.kuncham@gmail.com nelakuditi.kovida@research.iiit.ac.in radhika.mamidi@iiit.ac.in **IIIT-Hyderabad** 

Radhika Mamidi **IIIT-Hyderabad** 

#### Abstract

This paper revisits the work of (Kuncham et al., 2015) which developed a statistical sandhi splitter (SSS) for agglutinative languages that was tested for Telugu and Malayalam languages. Handling compound words is a major challenge for Natural Language Processing (NLP) applications for agglutinative languages. Hence, in this paper we concentrate on testing the effect of SSS on the NLP applications like Machine Translation, Dialogue System and Anaphora Resolution and show that the accuracy of these applications is consistently improved by using SSS. We shall also discuss in detail the performance of SSS on these applications.

#### Introduction 1

Sandhi which has its origin from Sanskrit 'samdhi' meaning "combination", it refers to a set of morphophonological changes i.e. fusion of final and initial sounds/characters at either morpheme or word boundaries. Sandhi is of two types; (i) Internal sandhi and (ii) External sandhi. "Macdonell, 1926"

Internal Sandhi: It refers to morphophonological changes that occur within a word i.e. across morpheme boundaries. For example, consider an English word "impatient" where /n/ in the negative morpheme "in-" has changed to /m/. This is seen for all the words starting with bilabial sounds that are prefixed with the "in-" morpheme.

External Sandhi: It refers to morphophonological changes that occur across word boundaries. When different words combine to form a compound word, we call it external sandhi. This type of sandhi occurs predominantly in Italian "Absalom et al., 2006" and Dravidian languages.

-> 'pUja'+'avyAkA' Ex<sup>1</sup>: 'pUjayyAkA'

after having finished the prayer prayer finished Here, we can observe the morphophonological change  $(a \rightarrow a + a)$  at the word boundaries.

"Kuncham et al., (2015) handled one type of external sandhi that is prominent in many agglutinative languages and poses many challenges to NLP applications as seen below."

### Machine Translation

akkaDikeLLu -> akkaDiki + veLLu there go there 20 When 'akkaDikeLLu' is given as input to Google "Telugu to English" Translate, it could not analyze the input and the English translation was 'Akkadikellu' which is wrong. But if the compound word is split and then given as input, Google Translate gave correct translation as 'Go

### Anaphora Resolution

there'.

nEnoccAnu -> nEnu + vaccAnu I came I came For the proper functioning of the Anaphora Resolution system, it is important to identify the pronoun 'nEnu' (I) in the input which is only possible by splitting the input.

### **Dialogue** System

raMgEmiTi -> raMgu + EmiTi what is the color color what If the compound word is given as input to a Dialogue system, it is very important to identify the question word 'EmiTi' (what) to give the correct answer.

From the above examples, we can see the importance of sandhi splitter in NLP applications. In this paper we show the performance and the effect of SSS on three NLP systems namely, (i) Telugu-English Machine Translation system, (ii) Dialogue System for 'Tourism' domain in Telugu and (iii) Anaphora Resolution system for Telugu.

<sup>&</sup>lt;sup>1</sup> All the examples are from Telugu language.

## 2 Related Work

Previous efforts on sandhi splitting primarily concentrated on building rule based systems to identify different words in the compound word. "Nair and Peter (2011) developed rules to identify all possible splits in any compound word i.e. both external and internal sandhi in Malayalam, an agglutinative language." "Joshi Shripad, (2012) implemented a rule based algorithm to split compound words into meaningful subwords in Marathi."

Apart from the traditional rule based systems, there are statistical systems for sandhi splitting as well. "Vempaty and Nagalla (2011) proposed a method using simple finite state automata for finding possible words in a given compound word." Finite state transducer (FST) is built from the syllables of base words and is used to identify possible candidates for a compound word. This approach fails for out-of-vocabulary (OOV) words i.e. if base word of any compound word doesn't exist in the FST. "Kuncham et al., (2015) built a statistical sandhi splitter (SSS) which identifies and generates meaningful words in a compound word using conditional random fields (CRFs)." "Natarajan and Charniak (2011) used statistical methods like Dirichlet Process and Gibbs Sampling for Sanskrit sandhi splitting."

In the recent years, the use of hybrid systems is increasing. Hybrid systems combine both statistical and rule based techniques. "Devadath, (2014) identifies split point statistically and uses character level rules specific to language to split the compound word accordingly."

"Popovi c et al., (2006) and Macherey et al., (2011) have discussed the challenges faced in machine translation due to compound words and handled compound words within the machine translation task." To the best of our knowledge, no one has shown the effect of sandhi splitting on various NLP applications.

In this paper, we discuss the effect of SSS (which gives better performance than the existing systems in Telugu language) on three different NLP applications i.e. Machine Translation, Anaphora Resolution and Dialogue System in Telugu. The results show that the performance of these systems is better after adopting SSS.

### 3 Statistical Sandhi Splitter (SSS)

In agglutinative languages, it is a common phenomenon to combine different words to form a compound word. So, sandhi splitting is an important step for any NLP application for these languages. SSS uses a statistical approach using CRF for the task of sandhi splitting. The approach consists of two stages namely, Segmentation and Word Generation.

### 3.1 Segmentation

In this stage, the boundaries between different words i.e. positions where morphophonological changes occur in a compound word are identified using CRF. The input for this task is a word and the output is the segments that show the boundary/split points in the input. The resulting segments may or may not be meaningful words which can be seen in the below example. Example:

Input: 'rAmuDoccADu' (*Ramudu came*) Output: 'rAmuD'-'occADu'

Here, 'rAmuDoccADu'->'rAmuDu'+'vaccADu'

Ramudu came Ramudu came In this example we can see that the segments 'rAmuD', 'occADu' are not meaningful words in Telugu language.

### 3.2 Word Generation

In this stage, meaningful words are generated from the segments obtained from the Segmentation stage. The input for this stage is the segments of a compound word and output is a meaningful word for each segment in the input. This stage consists of two components, (i) Class Label Assignment and (ii) Word Formation.

### 3.2.1 Class Label Assignment

The number of morphophonological changes occurring in any word is finite. The change can be either addition or deletion of characters at the end or at the starting of the segment. Each such change is taken as a separate class. Classes are extracted from the training data automatically. In this stage a class label is assigned to each segment using CRF.

Example:

In continuation to the example discussed in Segmentation stage, we have,

Class Label Assignment: 'rAmuD' \_u 'occADu' -o+va

In this example, we already know that the segments 'rAmuD' and 'occADu' are not meaningful. The first segment will be meaningful if "u" is added at its end and for the second segment to be meaningful, 'o' is removed and 'va' is added in its place. So these two segments fall into '\_u' and '-o+va' classes respectively.

### 3.2.2 Word Formation

This component generates meaningful words from the segments using class label information from Class Label Assignment stage.

The output of the Word Formation step for the example 'rAmuDoccADu' discussed in section 3.1 and 3.2.1 is as follows.

'rAmuDoccADu' -> 'rAmuDu' + 'vaccADu' Ramudu came Ramudu came

#### 4 Effect on NLP applications

"Kuncham et al., (2015) claim that compound words pose a problem for various NLP applications and that SSS is an attempt to reduce this effect." Here, we examine that claim by using SSS as a plugin before NLP applications like Machine Translation, Anaphora Resolution and Dialogue system. In this section we report our observations with respect to each of these applications.

#### 4.1 Machine Translation

We use Google Translate<sup>2</sup> for Telugu-English Translation because it is one of the state-of-theart commercial machine translation systems used today. Google Translate applies statistical learning techniques to build a translation model based on both monolingual text in the target language and aligned text consisting of examples of human translations between the languages.

We tested on 514 Telugu sentences which had 1890 words.



Fig 1: Cumulative N-gram BLEU scores on Telugu sentences in different experiment scenarios

BLEU score reported on manually sandhi divided data is 0.5003. This BLEU score would be the benchmark. BLEU score on sandhi combined data is 0.4506. We can observe the difference in the BLEU scores which tells us the importance of sandhi splitting in machine translation. As Telugu is a relatively morphologically rich language than English, it is very important that we split the compound words in Telugu when translating from Telugu to English. The BLEU score obtained by using SSS is 0.4810, which shows an improvement of **0.0304** over the sandhi combined data.

From the above reported BLEU scores, we can see that Google Translate fails to perform well in certain scenarios owing to the differences in the languages and mainly due to the high existence of compound words in Telugu. We will discuss through various examples how the differences in languages and compound words pose a challenge to machine translation. We further discuss the effect of SSS on Google Translate.

Different languages view the world with microscopes of different sensitivities. We may not find two languages with one to one mapping in their vocabulary and rules of the language. This is the very reason, Machine Translation is a challenging area of research. Following are some special constructions in Telugu that pose a problem for Machine Translation.

Examples

1. panIpATa cEsukuMTuMdi. -> At panipata *She is doing her work.* 

If we observe the above sentence from the source language (Telugu), the word 'panIpATa' is a compound word which has two words namely, 'panI', 'pATa', where the first word means work and the second word means 'things done after/during work' when used along with the former. This kind of word formation is unique in Indian languages and not found in English. Translating such kinds of words is problematic and not dealt by Google Translate which can be seen from its output 'At panipata'.

#### 2. eMduku mAneSAvu. -> Why quit why did you stop

Indian languages are pro drop languages whereas English is not. If we observe this example, the Telugu sentence has no word mapping to "you". The verbs in Telugu are inflected with gender, number and person information which helps to understand the meaning even if the subject is dropped. 'vu' in the verb 'mAneSAvu' (*stop*) gives 2nd person information, but the exact pronoun is dropped. This dropping is not possible in English language. In Telugu to English translation, accounting for the pro drop is a challenging

<sup>&</sup>lt;sup>2</sup> http://translate.google.com/translate\_t

task and we see that it is not properly handled by Google Translate.

3. rAmulu pAlu tAne pitukutADu . *Ramulu himself milks*.
-> Ramulu he milked milk.

Ellipsis poses a problem for translation in any language. In this example, the English translation for the Telugu sentence is "Ramulu himself milks". Here, we can observe that the object of verb "milk" is missing and it is only from the context we understand the sentence as "Ramulu milks the cows himself." Handling such cases require contextual knowledge. Recognizing ellipsis and bringing out the missing information in the target languages is a big challenge.

	Positive	Negative
True	119	1655
False	61	55

Table 1: Confusion matrix of SSS on Telugu sentences

Now we discuss the problems of compound words and the performance of SSS on machine translation. Table 1 gives the confusion matrix of SSS on 514 sentences which are discussed below in detail.

### **True positives**

This category includes all the compound words that should be split and are correctly split by SSS.

#### 1. Correct split, correct translation

**WS:**<sup>3</sup> tana iMTiki vacci koMceM annaM tecciMdi. (*She came to her house and brought some rice.*)

GT: To come to his house and brought some rice.

**WoS:** tana iMTikocci koMceM annaM tecciMdi. **GT:** Intikocci brought her a little rice.

'iMTikocci' (*came home*) is the compound word which is not recognized by Google Translate. But once it is correctly split into these two words 'iMTiki' (*home*), 'vacci' (*came*) the translator not only recognizes the words but also gives an answer close to the correct translation.

<sup>3</sup> WS –With SSS

GT - Google Translate Output

AS – Actual Split

WS: tulasi lEdu ani aDDaMgA tala UpiMdi. (*Tulasi shaked her head.*) GT: Shakes head across the basil.

WoS: tulasi lEdani aDDaMgA talUpiMdi . GT: The basil is not repeated horizontally.

In this example, we can observe that the compound word 'lEdani' (*not*) is recognized by Google Translate but is not translated correctly into English. When the compound word is split, the output of Google Translate is close to the correct translation.

#### 2. Correct split, wrong translation

**WS:** ippuDu tulasiki nayamu ayiMdi. (*Now Tu-lasi is healed.*)

GT: Now tulasiki was serious.

WoS: ippuDu tulasiki nayamayiMdi .

**GT:** Tulasiki healing now.

In this example, even though the compound word 'nayamayiMdi' (*healed*) is correctly split, the translation is incorrect to the extent that it gives an opposite sense.

#### **True negatives**

This category includes all the compound words that should not be split and are not split by SSS.

### WS: I gOlllu vEsuko. (Take these tablets)

**GT:** This can be marble.

This sentence does not contain any compound word, so no split is required.

#### **False positives**

This category includes words that should not be split but are split by SSS.

**WS:** sAyaM kAlaM rAmulu vaccADu. (*Ramulu came in the evening.*)

GT: Ramulu came to the aid of the season.

WoS: sAyaMkAlaM rAmulu vaccADu

**GT:** Ramulu returned in the evening. Here, 'sAyaMkAlaM' (*evening*) is the word that ideally should not be split, but SSS splits it.

#### **False negatives**

This category includes (a) compound words that should be split but not split by SSS and (b) compound words that are wrongly split.

WoS - Without SSS

(a) **WS:** raktaMtO idaMdutuMdi. (*This is supplied with blood.*)

**GT:** Idandutundi blood.

'idaMdutuMdi' (*this is supplied*) should be split into 'idi' (*this*) and 'aMdutuMdi' (*supplied*), but SSS doesn't split it resulting in non-identification of the word and thus incorrect translation by Google Translate.

(b) **WS:** vALLa mIda ottiDu ekkuva. (*More pressure on them*)

- **GT:** More pressure on them.
- **AS:** vALLa mIda ottiDi ekkuva
- **GT:** Another pressure

Here, there are two compound words 'vAL-LamIda' (on them) and 'ottiDekkuva' (more pressure). The first word is correctly split into 'vALLa' (them) and 'mIda' (on) whereas the latter is wrongly split. The correct split for the second one is 'ottiDi' (pressure), ekkuva' (more) which can be seen in **AS** (Actual Split). But strangely, Google Translate gives correct translation for the wrong split instead of the correct split.

#### **Some Special Cases:**

1(a). WS: kAni mUTa kanipiMca lEdu. (But the package is not seen.)

GT: But the package is not visible.WoS: kAni mUTa kanipiMcalEdu .GT: But the package did not.

1(b). **WS:** idi aMtA jariginA raMgaDu **IEva IEdu**. (*Rangadu did not get up even after all this*)

**GT:** Lev rangadu not it at all.

WoS: idaMwA jariginA raMgaDu lEvalEdu

**GT:** Rangadu risen at all this.

In the above sentences, SSS splits 'lEdu' (*not*) from words - 'kanipiMcalEdu' (*not visible*), 'levalEdu' (*did not get up*). Google Translate gives correct translation in sentence 1 (a) but not in sentence 1(b). The decision to split in this case is dependent on context, which SSS doesn't take into consideration.

2(a). WS: I mUTalanu mA tAtaki aMdiMcAli.(*Give these packages to my grandfather.*)GT: These kits provide our tataki.

Manual split: I mUTalanu mA tAtaku aMdiMcAli .

**GT:** These kits provide our grandfather.

In WS, 'tAtaki' (*to grandfather*) is not identified by the Translator as we can see, it is just transliterated in the English translation. But a variant of 'tAtaki', 'tAtaku' (*to grandfather*) (in manual split) is identified by the Google Translate. In general both these words are used alternatively in Telugu.

2(b). WS: civaraki oka cOTa pani dorikiMdi.

(Finally found a work at one place.)

**GT:** Finally found a place to work.

Manual split: civaraku oka cOTa pani doVrikiMdi

**GT:** Finally found a place to work.

In 2(a), the variants 'tAtaki' and 'tAtaku' are translated differently whereas in 2(b), the similar variants 'civaraki' (*finally*) and 'civaraku' are translated to same meaning in English.

#### 4.2 Anaphora resolution

Anaphora resolution is the problem of resolving references to earlier or later items in the discourse. These items are usually noun phrases representing objects in the real world called referents but can also be verb phrases, whole sentences or paragraphs.

An effort was made for building an Anaphora Resolution system for Telugu dialogues at IIIT-H. This system is a rule based system that handles nominal pronominal anaphora for human to human conversations. We examine the effect of SSS on this system and present our results in this section.

The corpus we used consists of 95 human conversations, each conversation may contain around 2-8 dialogues. Total pronouns in the corpus are 413. Most of the conversations in the corpus have been taken from the web.

	#pronouns correctly resolved	#pronouns wrongly resolved	Accuracy
Without	179	224	43.30
SSS			
With SSS	254	159	61.50

 Table 2: Accuracy of Anaphora Resolution system

 with and without using SSS

Here, we can see an improvement of 18.2% accuracy if SSS is used as a plugin before the Anaphora Resolution system. This improvement is because SSS could identify 53 more pronouns that were initially not identified by the Anaphora Resolution system as seen in Table 3.

Total pronouns	#pronouns iden- tified without SSS	#pronouns iden- tified with SSS
413	359	412

 Table 3: Pronouns identified with and without SSS by

 Anaphora Resolution system

Even though the number of pronouns identified by SSS is close to total pronouns in the corpus, there is a 5% error in splitting the compound words by SSS. The errors are of two types; (i) Wrong split and (ii) No split.

#### Wrong split:

ninnanE -> **ninnu** + anE

only yesterday you particle

In Telugu, 'ninnanE' has two senses, (a) 'only you' and (b) 'only yesterday'. If the word occurs with sense (a), it should be split and not in the case of (b) and the sense is decided only from the context. Here, 'ninnanE' should not be split but the split resulted into a pronoun 'ninnu' (*you*) which is wrong. From our analysis, this type of error was more than others i.e. 2.5% of total errors.

AvidelA -> Avida + elA

how she she how

This is the correct split for the compound word 'AvidelA' but the output from SSS is 'Avidu' and 'elA'. 'Avidu' is an unknown word in Telugu. This type of error constitutes of 1.6% of total errors.

### No split:

In this type of error, SSS could not split some compound words like the following example. This error constitutes of about 0.9% of total errors.

EMTadi -> EMTi + adi what is that what that Here, 'EMtadi' is not split by SSS.

### 4.3 Dialogue System:

A Dialogue System is a computer program that is designed to communicate with humans in a natural way in natural language. As mentioned in "Sravanthi et al., 2015", Sandhi is a challenge to Dialogue Systems and the effect of SSS on this system is discussed in this section. We prepared 281 questions on 'Tourist places in Hyderabad' domain in Telugu. Accuracy of the Dialogue System with and without using SSS is shown in Table 4.

	#Correctly An- swered questions	Accuracy
Without SSS	156	55.51
With SSS	175	62.27

 
 Table 4: Accuracy of Dialogue system with and without SSS

From this table we can see that there is an improvement in the overall accuracy of Dialogue system after using SSS but the increase in the accuracy is only 6.8%. This is because of the following reasons.

1. Borrowing of English words is common in Telugu language. If the compound words contain English words, it makes the split difficult for SSS. Moreover, occurrence of English words in 'Tourism' domain is high resulting in the increase in the percentage of errors.

This is the actual split for the compound word 'gOlkoMDekkaDuMdi' but SSS gives wrong split as 'gOlkoMDu', 'ekkaDa', 'uMdi'. Since 'gOlkoMDa' (Golconda) is not identified in the question, the Dialogue system gives wrong answer.

TaimiMgsEMTi -> TaimiMgs + EMTi what are the timings timings what the above is correct split The for 'TaimiMgsEMTi' but SSS couldn't split it. It fails to split if English words like 'timings', 'address', 'monuments' etc., occur in the compound words.

2. Presence of context dependent particles.

gOlkoMDanE Taimulo cUDaccu? What time can Golconda be visited?

The clitique 'E' is ambiguous and the split depends on the context as discussed in "Kuncham et al., 2015". Here, 'E' acts as a question marker which should be split to get the correct answer. But this type of context dependent cases is not handled by SSS.

3. Wrong splits by SSS.

cirunAmA (*address*) which should not be split but split by SSS as 'ciruni' and 'Ama' which have no sense in Telugu.

### 5 Conclusion

In conclusion, we can say that the presence of compound words degrade the performance of any NLP application for agglutinative languages which can be improved significantly by using SSS. We have presented our efforts in discussing the detail analysis of the performance and the effect of SSS on different NLP applications. As discussed in sections 4.2 and 4.3, splitting of some words depend on contextual information. SSS can be extended to handle these context dependent particles by considering whole sentences for training and learning features.

### Acknowledgement:

This work is partially supported by Information Technology Research Academy (ITRA), Government of India under ITRA-Mobile grant ITRA/15(62)/Mobile/VAMD/01

### References

- Kuncham, P., Nelakuditi, K., Nallani, S., and Mamidi, R. (2015). Statistical sandhi splitter for agglutinative languages. In Computational Linguistics and Intelligent Text Processing, pages 164–172. Springer.
- A. A. Macdonell, *A Sanskrit Grammar for students*. New Delhi, India: D.K. Printworld (P) Ltd., 1926.
- M. Absalom and J. Hajek, "Prosodic phonology and raddoppiamento sintattico: a re-evaluation," in Selected Papers from the 2005 Conference of the Australian Linguistic Society, Melbourne: Monash University. http://www. arts. monash. edu. au/ling/als, 2006.
- Nair, L. R. and Peter, S. D. (2011). Development of a rule based learning system for splitting compound words in malayalam language. In Recent Advances in Intelligent Computational Systems (RAICS), 2011 IEEE, pages 751–755. IEEE.
- Joshi Shripad, S. (2012). Sandhi splitting of Marathi compound words. Int. J. on Adv. Computer Theory and Engg, 2(2).
- Vempaty, P. C. and Nagalla, S. C. P. (2011). Automatic sandhi spliting method for telugu, an indian lan-guage. Procedia-Social and Behavioral Sciences, 27:218–225.

- Natarajan, A. and Charniak, E. (2011). S3-statistical sam. dhi splitting.
- Devadath V V, Litton J Kurisinkel, D. M. S. V. V. (2014). A sandhi splitter for malayalam.(accepted but yet to be published in proceedings of ICON 2014.)
- Popovi<sup>c</sup>, M., Stein, D., and Ney, H. (2006). Statistical machine translation of german compound words. In Advances in Natural Language Processing, pages 616–624. Springer.
- Macherey, K., Dai, A. M., Talbot, D., Popat, A. C., and Och, F. (2011). Language-independent compound splitting with morphological operations. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pages 1395– 1404. Association for Computational Linguistics.
- Sravanthi, M. C., Prathyusha, K., and Mamidi, R. (2015). A Dialogue System for Telugu, a resourcepoor language. In Computational Linguistics and Intelligent Text Processing, pages 364–374. Springer.