A Corpus-based Decompounding in Sanskrit

Siba Sankar Sahu[1] and Puneet Mamgain[2]

[1] Dept. of Computer Science and Engineering, IIT (BHU) Varanasi, India
[2] Sikkim Manipal Institute of Technology, India

Abstract. Unlike English, in highly inflected Indian languages like Bengali, Marathi, and Sanskrit, compound words are not multi-word expressions but created by combining two or more simple words without any orthographic separation. A compound word with unmarked word boundaries creates a problem for many computational tasks. Splitting compound words improves performances in Machine Translation, and Information Retrieval by reducing out-of-vocabulary words in the dictionary. So far, a number of decompounding techniques have been applied in European languages like German, Dutch, and Scandinavian. In this work, we apply a corpus-based decompounding technique in Sanskrit and improve splitting accuracy by applying various ranking methods. We evaluate the performance by different ranking methods against a gold standard in terms of Precision, Recall, and F-measure.

Keywords: Compound, Machine Translation, Information Retrieval

1 Introduction

Compounding is the process of combining two or more simple words to form a new complex word. Two or more words are combined if there is a semantic relation between them. In Sanskrit, combining of words could be direct concatenation or by using *sandhi* rules. In general, a compound word is divided into three categories: 'open compound', 'closed compound', and 'hyphenated compound'. In open compound, two or more words are combined to create a new meaningful word but use space between every candidate words. In the closed compound, two or more words are combined to create a new compound word but with no space between them. In the hyphenated compound, two or more words are combined with hyphens in between. In Indian languages like Bengali, Marathi, and Sanskrit, frequency of closed compound is very high compared to other types of compounds. A closed compound word reduces efficiency of Machine Translation, Speech recognition, and Information Retrieval systems.
Many decompounding techniques are proposed in European languages like German, Dutch and Czech which have shown to reduce OOV words in the dictionary, improve performance in Machine Translation and Hypernym Detection [4, 6, 1].

But, no such work has been proposed and evaluated in Sanskrit. In this work, we apply a corpus-based decompounding technique in Sanskrit and use various ranking methods to improve splitting accuracy. We built a gold standard and evaluate the performance of the compound splitting technique by combining various ranking methods in terms of Precision, Recall, and F-measure.
The paper is organised as follows. In Section 2 we review different kind of corpus-based decompounding techniques implemented and evaluated for different languages. Section 3 describes about corpus-based decompounding in Sanskrit and various ranking methods which used for improving splitting accuracy. Section 4 describes the dataset, experiment on decompounding, and shows improvements in splitting performance by using multiple ranking methods. Section 5 concludes with directions for future work.

2 Related Work

There exist a few decompounding techniques applied earlier in European languages. In general, they split the compound in three ways. i. rule-based ii. corpus-based iii. supervised method. Sometimes some hybrid approach are also used to improve the performance of compound splitting techniques. In this section, we present a brief overview of corpus-based decompounding methods implemented in different languages.
Koehn and Knight [4] proposed a compound splitting approach in German compound words to improve Statistical Machine Translation. The splitting point of compound words is determined by the highest geometric mean of word frequencies.

$$\arg\max S = \left(\prod_{e_i \in S} count(e_i) \right)^{\frac{1}{n}} \tag{1}$$

In this method, it is assumed that if a constituent part of a compound frequently occurs as an independent word in a corpus, it is likely to be part of the compound.

Marek [5] and Alfonseca [2]proposed a compound splitting technique based on a probability score. The splitting point score is determined by (sum of the negative logarithms of the) probabilities of the constituent elements, which is calculated by their frequency count(e) divided by the corpus size N.

$$\arg\min S = \sum_{e_i \in S} -\log \left(\frac{count(e)}{N} \right) \tag{2}$$

In this method it is assumed that, if the probability of constituent elements of a compound is high in the corpus then it has a positive effect on the prediction of compound constituent.
In the recent past, a corpus-based decompounding technique was implemented in an Indian language Bengali for Information Retrieval[3]. The characteristics of Bengali language is different from an European language and hence, the authors

did not apply the frequency based decompounding approach used in European languages. At first, they proposed a relaxed decompounding where a compound will not split to its constituent parts if all the constituent parts of the compound word are not individually valid words. Secondly, they perform a selective decompounding when a constituent of a compound co-occurs up to a certain level of the threshold with the compound word in the corpus.
In the light of recent developments, the motivation in the present article is to apply corpus-based decompounding in Sanskrit text where no previous work exists to the best of knowledge of the authors.

3 Data Set

We built a corpus by extracting the documents from Wikipedia, All India Radio Sanskrit news and Samprativartah news. Since the data will come from multiple sources having different forms and formats, extracting text involves data cleaning, removing formatting tags, handling images and advertisement etc. The data can be extremely noisy. Some of the standard techniques of text pre-processing like case-folding, removing punctuations, stop-word removal applied. Now, the raw corpus contains a total of 27,170,305 words in root form as well as inflected form. Moreover, we use a dictionary of 9568 words to generate candidate words.

4 Decompounding Approach

The algorithm implemented in three main steps

4.1 Candidate generation

At first, we scan the word from left to right and check sequence of word i.e. minimum length (L=3) is a valid word in the dictionary. If it is a valid word then generate a binary splits and repeat the process till end of the word. The splitter try to generate all possible constituent of a compound word. If a given word is not compound returns as it is.

4.2 Cleaning

During compound splitting there may be possibility of wrong candidate generation. To avoid wrong candidate generation we use following cleaning methods.

Suffix parts Sanskrit is a highly inflectional language and the root form of words more likely to be inflected. As a pre-processing of Sanskrit text we are not applied any kind of stemming technique for suffix removal hence, suffixes may be split-off during compound splitting. We merge the split of suffix length shorter than 4 characters. The length of suffix somehow arbitrary, and could be varied. Increasing in length of suffix parts discard the actual candidate splits and decreasing length of suffix parts less effect on cleaning method.

Fragment If the splitter generate one or two characters we can avoid through it.

4.3 Ranking

Now the cleaned list of words available for ranking. We use following ranking methods to rank the split.

Most known In this ranking method a score assigned to the split based on knowing the constituent parts. If all the constituent of a compound is known assign a higher value otherwise a lesser value.

Light and aggressive Light and aggressive assign a score based on number of splitting parts. light prefers the split with fewer parts whereas aggressive prefers the split with longer split parts.

Semantic Similarity The basic idea of word embedding is to represent the words in such a way that semantically similar words are close to each other whereas semantically dis-similar words are farther apart. We can implement word embedding in three ways. i.Word2vec model, ii.Fast-text model and iii.GloVe model. Here, we use Fast-text model to generate word embeddings in which skip gram predicts the surrounding context window of size 5 of given current word. We use the size of 300 dimensions with epoch size 25 and learning rate as 1. In the input layer words are broken into n-gram and fed to the neural network and output layer contain context words. We train the fast-text model by using above described dataset in Section 3. The pre-trained model used for ranking. We assumed that adjacent portions of splits are more similar then furtherapart splits. For Eg. Extremely difficult labour, Extremely and difficult are often found together, even with difficult and labour, but extremely and labour are not likely to be similar at all. We use Cosine Similarity measure to evaluate the distance between two word vectors.

5 Evaluation

We built a test dataset by extracting the document from Samprativarth News and choose 1190 randomly unique words. The gold standard of test data manually created by well known Sanskrit person. we evaluate the peformance of compound splitting technique by combining different ranking methods against a goldstandard interms of Precision, Recall, F-measure and Accuracy. We can evaluate the correctness of split by Koehn and Knight [4] methods.
correct split: words that should be split and were split correctly
correct non: words that should not be split and were not
wrong not: words that should be split but were not
wrong faulty split: words that should be split, were split, but wrongly
wrong split: words that should not be split, but were
precision: (correct split) / (correct split + wrong faulty split + wrong superfluous split)
recall: (correct split) / (correct split + wrong faulty split + wrong not split)
accuracy: (correct) / (correct + wrong)

Table 1. Evaluation of the methods compared against a manual annotated gold standard of splits: Using Shortest method gives the best accuracy (78.23%).

Method	Correct	Wrong			Metrics		
	Split	not	not	faulty	split	prec.	recall	acc.
Raw	0	802	388	0	0	-	0.0	67.3%
Most Known	65	761	172	69	123	25.29	21.2	69.4%
Semantic Similarity	81	765	94	116	134	24.4	27.8	71.09%
Aggressive	54	799	120	56	161	19.9	23.4	71.6%
Light	82	849	104	61	94	34.5	33.19	78.23%

6 Conclusions and Future Work

Decompounding is an important pre-processing step for compound languages like Bengali, Marathi, and Sanskrit. In this work, we investigate the effect of corpus-based decompounding in Sanskrit and improve splitting accuracy by different ranking methods. In Sanskrit, compound word occurs in two ways: one is direct concatenation of words and another is by applying sandhi rules. It has been observed that compound splitting technique splits the most of direct concatenation of compound words effectively but for sandhi-ed compound the performance of compound splitter is quite poor due to sandhi rule changed the first character of the second candidate appear in a modified form in the compound. Secondly, in sandhi-ed compounds the second candidate of compound word may not be word in dictionary. In different ranking methods shortest method gives highest splitting accuracy i.e. 78.23% as shown in above Table 1.
As part of future work, we plan to investigate the effect of decompounding as well as various ranking method in Sanskrit Information retrieval system. We would like to investigate the effect of decompounding in highly inflected Indian languages like Bengali and Marathi.

References

1. Adda-Decker, M.: A corpus-based decompounding algorithm for german lexical modeling in lvcsr. In: Eighth European Conference on Speech Communication and Technology (2003)
2. Alfonseca, E., Bilac, S., Pharies, S.: German decompounding in a difficult corpus. In: International Conference on Intelligent Text Processing and Computational Linguistics. pp. 128–139. Springer (2008)
3. Ganguly, D., Leveling, J., Jones, G.J.: A case study in decompounding for bengali information retrieval. In: International Conference of the Cross-Language Evaluation Forum for European Languages. pp. 108–119. Springer (2013)
4. Koehn, P., Knight, K.: Empirical methods for compound splitting. arXiv preprint cs/0302032 (2003)
5. Marek, T.: Analysis of german compounds using weighted finite state transducers. Bachelor thesis, University of Tübingen (2006)
6. Rigouts Terryn, A., Macken, L., Lefever, E.: Dutch hypernym detection: does decompounding help? In: Joint Second Workshop on Language and Ontology & Terminology and Knowledge Structures (LangOnto2+ TermiKS). pp. 74–78. European Language Resources Association (ELRA) (2016)