Coling 2010

23rd International Conference on Computational Linguistics

Proceedings of the

1st Workshop on South and southeast Asian Natural Language Processing

24 August 2010
Beijing International Convention Center

Produced by
Chinese Information Processing Society of China

To order the CD of Coling 2010 and its Workshop Proceedings, please contact:

Chinese Information Processing Society of China
No.4, Southern Fourth Street
Haidian District, Beijing, 100190
China
Tel: +86-010-62562916
Fax: +86-010-62562916
cips@iscas.ac.cn

Preface

Welcome to the Coling Workshop on *South and Southeast Asian Natural Language Processing (WSSANLP)*.
South Asia comprises of the countries- Afghanistan, Bangladesh, Bhutan, India, Maldives, Nepal, Pakistan and Sri Lanka. Southeast Asia, on the other hand, consists of Brunei, Burma, Cambodia, East Timor, Indonesia, Laos, Malaysia, Philippines, Singapore, Thailand and Vietnam.

There thousands of languages that belong to different language families like Indo-Aryan, Indo-Iranian, Dravidian, Sino-Tibetan, Austro-Asiatic, Kradai, Hmong-Mien, etc. In terms of population, South Asia and Southeast Asia represent 34.94 percent of the total population of the world. Some of the languages of these regions have a large number of native speakers: Hindi (5th largest according to number of its native speakers), Bengali (6th), Punjabi (12th), Tamil (18th), Urdu (20th), etc.

A characteristic of these languages is that they are under-resourced. But the words of these languages show rich variations in morphology. Moreover they are often heavily agglutinated and synthetic, making segmentation an important issue. The intellectual motivation for this workshop comes from the need to explore ways of harnessing the morphology of these Source (Lewis, 2009) languages for higher level processing. Table 1: Population and Number of Living Languages of The task of morphology, however, is South and Southeast Asia intimately linked with segmentation for these languages.

The goal of WSSANLP is:
Providing a platform to linguistic and NLP communities for sharing and discussing ideas and work on South and Southeast Asian languages and combining efforts.
Development of useful and high quality computational resources for under resourced South and Southeast Asian languages.

We are delighted to present you this volume of proceedings of 1st Workshop on South and Southeast Asian NLP. We have received total 18 long and short paper submissions. On the basis of our review process, we have competitively selected 13 papers, but unfortunately 6 papers were withdrawn from the workshop due to double submission and authors chose to present their paper in other events. We hope that we will be able to make this workshop so successful that people would like to present their papers in this workshop in future.

M. G. Abbas Malik, *Chair of Organizing Committee WSSANLP*

Workshop Chair:

Aravin K. Joshi, *Chair of the Worskhop*, University of Pennsylvania, (USA)

Organizers:

M. G. Abbas Malik, (*Chair of Organizing Committee*), GETALP-LIG, University of Grenoble (France)
Aasim Ali, CRULP, National University of Computer and Emerging Sciences (Pakistan)
Asif Ekbal, Dept. of Computational Linguistics, University of Heidelberg (Germany)
Dulip Herath, University of Colombo School of Computing (Sri Lanka)
Hong-Thi Nguyen, GETALP-LIG, University of Grenoble (France)
Muhammad Humayoun, LAMA, Universit de Savoie (France)
Menaka Sankaralingam AUKBC Research Centre, Chennai (India)
Monojit Choudhury, Researcher, Microsoft Research (India)
Sadaf Abdul Rauf, Universit du Maine (France)
Smriti Singh, Indian Institute of Technology Bombay (India)

Program Committee:

Ranaivo-Malanon Bali, Multimedia University (Malaysia)
Sivaji Bandyopadhyay, Jadavpur University (India)
Vincent Berment, GETALP-LIG / INALCO (France)
Laurent Besacier, GETALP-LIG, Universit de Grenoble (France)
Pushpak Bhattacharyya, IIT Bombay (India)
Christian Boitet, GETALP-LIG, Universit de Grenoble (France)
Nicola Cancedda, Xerox Research Center Europe (France)
Eric Castelli, International Research Center MICA (Vietnam)
Luong Chi Mai, Institute of IT, Vietnamese Academy of Science and Technology (Vietnam)
Laurence Danlos, University Paris 7 (France)
Georges Fafiotte, GETALP-LIG, Universit de Grenoble (France)
John A. Goldsmith, University of Chicago (USA)
Grard Huet, INRIA (France)
San San Hnin Tun, Cornell University (USA)
Sarmad Hussain, National University (Pakistan)
Abid Khan, University of Peshawar (Pakistan)
Wunna Ko Ko, Northern Illinois University (USA)
Bal Krishna Bal, University of Kathmandu (Nepal)
A. Kumaran, Microsoft Research (India)
Gurpreet Singh Lehel, Punjabi University Patiala (India)
Haizhou Li, Institute for Infocomm Research (Singapore)
Alec Marantz, New York University (USA)

Christian Monson, OHSU (USA)
Annie Montaut, INALCO Paris (France)
Hammam Riza, Agency for the Assessment and Application of Technology (Indonesia)
Rajeev Sangal, IIIT Hyderabad (India)
Anne Schiller, Xerox Research Center Europe (France)
L. Sobha, AU-KBC Research Centre (India)
Chan Somnoble, Royal University of Phnom Penh (Cambodia)
Virach Sornlertlamvanich, TCL, National Institute of Information and Communication Technology (Thailand)
Ruvan Weerasinghe, University of Colombo School of Computing (Sri Lanka)
Khaver Zia, Beacon House National University (Pakistan)

Invited Speaker:

Rajeev Sangal, IIIT Hyderabad (India)

Table of Contents

Conference Program

Tuesday, August 24, 2010

16:00–16:10 Opening Remarks

16:10–16:40 Invited Talk by Dr. Rajeev Sangal

16:40–17:00 *Boosting N-gram Coverage for Unsegmented Languages Using Multiple Text Segmentation Approach*
Solomon Teferra Abate, Laurent Besacier and Sopheap Seng

17:00–17:20 *Thai Sentence-Breaking for Large-Scale SMT*
Glenn Slayden, Mei-Yuh Hwang and Lee Schwartz

17:20–17:40 *Clause Identification and Classification in Bengali*
Aniruddha Ghosh, Amitava Das and Sivaji Bandyopadhyay

17:40–17:50 break

17:50–18:10 *A Paradigm-Based Finite State Morphological Analyzer for Marathi*
Mugdha Bapat, Harshada Gune and Pushpak Bhattacharyya

18:10–18:30 *Web Based Manipuri Corpus for Multiword NER and Reduplicated MWEs Identification using SVM*
Thoudam Doren Singh and Sivaji Bandyopadhyay

18:30–18:40 *A Word Segmentation System for Handling Space Omission Problem in Urdu Script*
Gurpreet Lehal

18:40–18:50 *Hybrid Stemmer for Gujarati*
Pratikkumar Patel, Kashyap Popat and Pushpak Bhattacharyya

18:50–19:00 Closing Remarks

Boosting N-gram Coverage for Unsegmented Languages Using Multiple Text Segmentation Approach

Solomon Teferra Abate
LIG Laboratory,
CNRS/UMR-5217
solomon.abate@imag.fr

Laurent Besacier
LIG Laboratory,
CNRS/UMR-5217
laurent.besacier@imag.fr

Sopheap Seng
LIG Laboratory,
CNRS/UMR-5217
MICA Center, CNRS/UMI-2954
sopheap.seng@imag.fr

Abstract

Automatic word segmentation errors, for languages having a writing system without word boundaries, negatively affect the performance of language models. As a solution, the use of multiple, instead of unique, segmentation has recently been proposed. This approach boosts N-gram counts and generates new N-grams. However, it also produces bad N-grams that affect the language models' performance. In this paper, we study more deeply the contribution of our multiple segmentation approach and experiment on an efficient solution to minimize the effect of adding bad N-grams.

1 Introduction

A language model is a probability assignment over all possible word sequences in a natural language. It assigns a relatively large probability to meaningful, grammatical, or frequent word sequences and a low probability or a zero probability to nonsensical, ungrammatical or rare ones. The statistical approach used in N-gram language modeling requires a large amount of text data in order to make an accurate estimation of probabilities. These data are not available in large quantities for under-resourced languages and the lack of text data has a direct impact on the performance of language models. While the word is usually a basic unit in statistical language modeling, word identification is not a simple task even for languages that separate words by a special character (a white space in general). For unsegmented languages, which have a writing system without obvious word delimiters, the N-grams of words are usually estimated from the text corpus segmented into words employing automatic methods. Automatic segmentation of text is not a trivial task and introduces errors due to the ambiguities in natural language and the presence of out of vocabulary words in the text.

While the lack of text resources has a negative impact on the performance of language models, the errors produced by the word segmentation make those data even less usable. The word N-grams not found in the training corpus could be due not only to the errors introduced by the automatic segmentation but also to the fact that a sequence of characters could have more than one correct segmentation.

In previous article (Seng et al., 2009), we have proposed a method to estimate an N-gram language model from the training corpus on which each sentence is segmented into multiple ways instead of a unique segmentation. The objective of multiple segmentation is to generate more N-grams from the training corpus to use in language modeling. It was possible to show that this approach generates more N-grams (compared to the classical dictionary-based unique segmentation method) that are potentially useful and relevant in language modeling. The application of multiple segmentation in language modeling for Khmer and Vietnamese showed improvement in terms of tri-gram hits and recognition error rate in Automatic Speech Recognition (ASR) systems.

This work is a continuation of our previous work on the use of multiple segmentation. It is conducted on Vietnamese only. A close analysis of N-gram counts shows that the approach has in fact two contributions: boosting the N-gram

Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), pages 1–7,
the 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010

counts that are generated with first best segmentation and generating new N-grams. We have also identified that there are N-grams that negatively affect the performance of the language models. In this paper, we study the contribution of boosting N-gram counts and of new N-grams to the performance of the language models and consequently to the recognition performance. We also present experiments where rare or bad N-grams are cut off in order to minimize their negative effect on the performance of the language models.

The paper is organized as follows: section 2 presents the theoretical background of our multiple segmentation approach; in section 3 we point out the set up of our experiment; in section 4 we present the results of our detailed statistical analysis of N-grams generated by multiple segmentation systems. Section 5 presents the evaluation results of our language models for ASR and finally, we give concluding remarks.

2 Multiple Text Segmentation

Text segmentation is a fundamental task in natural language processing (NLP). Many NLP applications require the input text segmented into words before making further progress because the word is considered the basic semantic unit in natural languages. For unsegmented languages segmenting text into words is not a trivial task. Because of ambiguities in human languages, a sequence of characters may be segmented in more than one way to produce a sequence of valid words. This is due to the fact that there are different segmentation conventions and the definition of word in a language is often ambiguous.

Text segmentation techniques generally use an algorithm which searches in the text the words corresponding to those in a dictionary. In case of ambiguity, the algorithm selects the one that optimizes a parameter dependent on the chosen strategy. The most common optimization strategies consist of maximizing the length of words ("longest matching") or minimizing the number of words in the entire sentence ("maximum matching"). These techniques rely heavily on the availability and the quality of the dictionaries and while it is possible to automatically generate a dictionary from an unsegmented text corpus using unsupervised methods, dictionaries are often created manually. The state-of-the-art methods generally use a combination of hand-crafted, dictionary and statistical techniques to obtain a better result. However, statistical methods need a large corpus segmented manually beforehand. Statistical methods and complex training methods are not appropriate in the context of under-resourced languages as the resources needed to implement these methods do not exist. For an under-resourced language, we seek segmentation methods that allow better exploitation of the limited resources. In our previous paper (Seng et al., 2009) we have indicated the problems of existing text segmentation approaches and introduced a weighted finite state transducer (WFST) based multiple text segmentation algorithm.

Our approach is implemented using the AT & T FSM Toolkit (Mohri et al., 1998). The algorithm is inspired with the work on the segmentation of Arabic words (Lee et al., 2003). The multiple segmentation of a sequence of characters is made using the composition of three controllers. Given a finite list of words we can build a finite state transducer M (or word transducer) that, once composed with an acceptor I of the input string that represent a single character with each arc, generates a lattice of the words that represent all of the possible segmentations. To handle out-of-vocabulary entries, we make a model of any string of characters by a star closure operation over all the possible characters. Thus, the unknown word WFST can parse any sequence of characters and generate a unique *unk* word symbol. The word transducer can, therefore, be described in terms of the WFST operations as $M = (WD \cup UNK)+$ where WD is a WFST that represents the dictionary and UNK represents the unknown word WFST. Here, $\cup$ and + are the union and Kleene "+" closure operations. A language model L is used to score the lattice of all possible segmentations obtained by the composition of our word transducer M and the input string I. A language model of any order can be represented by a WFST. In our case, it is important to note that only a simple uni-gram language model is used. The uni-gram model is estimated from a small training corpus segmented automatically into words using a dictionary based method. The composition of the sequence of input string I

with the word transducer M yields a transducer that represents all possible segmentations. This transducer is then composed with the language model L, resulting in a transducer that represents all possible segmentations for the input string I, scored according to L. The highest scoring paths of the compound transducer is the segmentation m that can be defined as:

$$P(m) = maxP(m_k)$$

The segmentation procedure can then be expressed formally as:

$$m = bestpath(I \circ M \circ L)$$

where ◦ is the composition operator. The N-best segmentations are obtained by decoding the final lattice to output the N-best highest scoring paths and will be used for the N-gram count.

3 Experimental Setup

3.1 Language Modeling

First, it is important to note that Vietnamese texts are naturally segmented into syllables (not words). Each syllable tends to have its own meaning and thus a strong identity. However, the Vietnamese monosyllable is not automatically a word as we would define a word in English. Often, two syllables go together to form a single word, which can be identified by the way it functions grammatically in a sentence. To have a word-based language model, word segmentation would, therefore, be a must in Vietnamese.

A Vietnamese training corpus that contains 3 millions sentences from broadcast news domain has been used in this experiment. A Vietnamese dictionary of 30k words has been used both for the segmentation and counting the N-grams. Therefore, in the experiments, the ASR vocabulary always remains the same and only the language model is changing. The segmentation of the corpus with dictionary based, "longest matching" unique segmentation method gives a corpus of 46 millions words. A development corpus of 1000 sentences, which has been segmented automatically to obtain 44k words, has been used to evaluate the tri-gram hits and the perplexity. The performance of each language model produced will be evaluated in terms of the tri-gram hits and perplexity on the development corpus and in terms of ASR performance on a separate speech test set (different from the development set).

First of all, a language model named lm_1 is trained using the SRILM toolkit (Stolcke 2002) from the first best segmentation (Segmul1), which has the highest scoring paths (based on the transducer explained in section 2) of each sentence in the whole corpus. Then, additional language models have been trained using the corpus segmented with N-best segmentation: the number of N-best segmentations to generate for each sentence is fixed to 2, 5, 10, 50, 100 and 1000. The resulting texts are named accordingly as Segmul2, Segmul5, Segmul10, Segmul50, Segmul100, Segmul1000. Using these as training data, we have developed different language models. Note that a tri-gram that appears several times in multiple segmentations of a single sentence has a count set to one.

3.2 ASR System

Our automatic speech recognition systems use the CMU's Sphinx3 decoder. The decoder uses Hidden Markov Models (HMM) with continuous output probability density functions. The model topology is a 3-state, left-to-right HMM with 16 Gaussian mixtures per state. The preprocessing of the system consists of extracting a 39 dimensional features vector of 13 MFCCs, the first and second derivatives. The CMU's SphinxTrain has been used to train the acoustic models used in our experiment.

The Vietnamese acoustic modeling training corpus is made up of 14 hours of transcribed read speech. More details on the automatic speech recognition system for Vietnamese language can be found in (Le et al., 2008). While the evaluation metric WER (Word Error Rate) is generally used to evaluate and compare the performance of the ASR systems, this metric does not fit well for unsegmented languages because the errors introduced during the segmentation of the references and the output hypothesis may prevent a fair comparison of different ASR system outputs. We, therefore, used the Syllable Error Rate (SER) as Vietnamese text is composed of syllables naturally separated by white space. The automatic speech recognition is done on a test corpus of 270 utterances (broadcast news domain).

4 Statistical Analysis of N-grams in Multiple Text Segmentation

The change in the N-gram count that results from multiple segmentation is two fold: first there is a boosting of the counts of the N-grams that are already found with the first best segmentation, and secondly new N-grams are added. As we have made a closed-vocabulary counting, there are no new uni-grams resulting from multiple segmentation. For the counting, the SRILM toolkit (Stolcke 2002) is used setting the -gtnmin option to zero so that all the N-gram counts can be considered.

Figure 1 shows the distribution of tri-gram counts for the unique and multiple segmentation of the training corpus. It can be seen that the majority of the tri-grams have counts in the range of one to three.

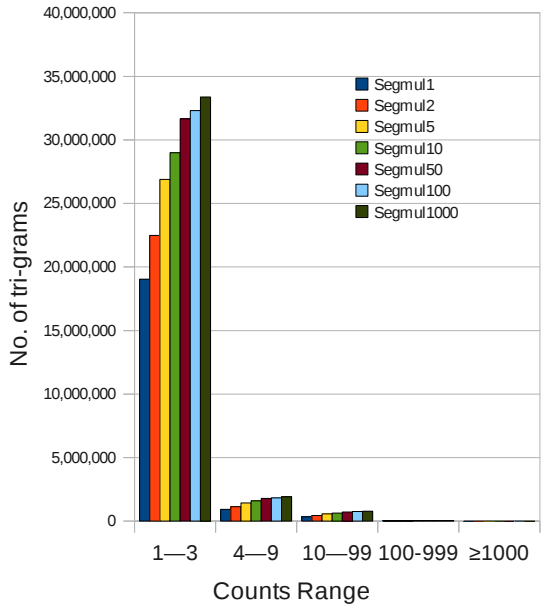

Figure 1: Distribution of tri-gram counts

The boosting (the counts of the tri-grams that are already found with the first best segmentation) effect of the multiple segmentation is indicated in table 1. We can see from the table that Segmul2, for example, reduced the number of rare tri-grams (count range 1-3) from 19.04 to 16.15 million. Consequently, the ratio of rare tri-grams to all tri-grams that are in Segmul1 is reduced from 94% (19.04/20.31*100) of Segmul1 only to 79% (15.96/20.31*100) by the boosting effect of Segmul1000, which increased the number of tri-grams with count range of 4-9 from 0.91M to 3.34M. This implies, in the context of under-resourced languages, that multiple segmentation is boosting the N-gram counts. However, one still has to verify if this boosting is relevant or not for ASR.

Multiple Seg.	Counts Range				
	1–3 (M)	4-9 (M)	10-99 (M)	100-999 (M)	≥1000 (M)
Segmul1	19.04	0.91	0.34	0.016	0.00054
Segmul2	16.15	3.23	0.89	0.043	0.0017
Segmul5	16.06	3.28	0.92	0.045	0.0017
Segmul10	16.03	3.30	0.93	0.045	0.0017
Segmul50	15.99	3.33	0.95	0.046	0.0017
Segmul100	15.98	3.33	0.95	0.046	0.0017
Segmul1000	15.96	3.34	0.96	0.046	0.0017

Table 1. boosting tri-gram counts

We have also analyzed the statistical behavior of the newly added tri-grams with regard to their count distribution (see figure 2). As we can see from the figure, the distribution of the new tri-grams is somehow similar to the distribution of the whole tri-grams that is indicated in figure 1.

As shown in table 2, the total number of newly added tri-grams is around 15 millions. We can see from the table that the rate of new tri-gram contribution of each segmentation increases as N increases in the N-best segmentation. However, as it is indicated in figure 2, the major contribution is in the area of rare tri-grams.

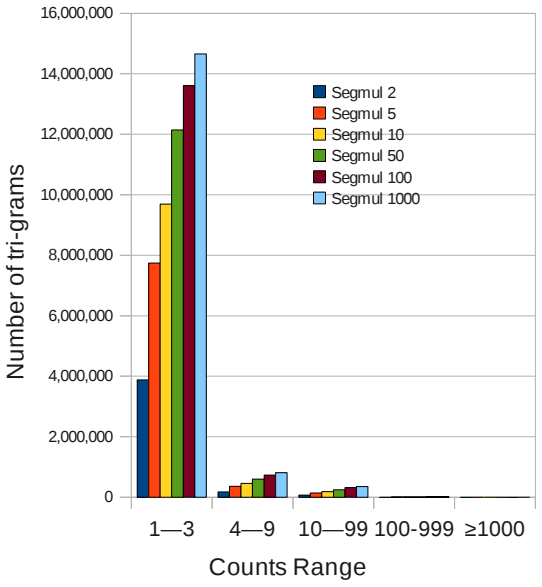

Figure 2: Distribution of new tri-gram counts

Mul. Segmentation	No.	%
Segmul2	4,125,881	26,05
Segmul5	8,249,684	52,09
Segmul10	10,355,433	65,39
Segmul50	13,002,700	82,11
Segmul100	14,672,827	92,65
Segmul1000	15,836,120	100,0

Table 2. tri-gram contribution of multiple segmentation

5 Experimental Results

In this section we present the various language models we have developed and their performance in terms of perplexity, tri-gram hits and ASR performance (syllable error rate).

We use the results obtained with the method presented in (Seng et al., 2009) as baseline. This method consists in re-estimating the N-gram counts using the multiple segmentation of the training data and add one to the count of a tri-gram that appears several times in multiple segmentations of a single sentence. These baseline results are presented in Table 3. The results show an increase of the tri-gram coverage and slight improvements of the ASR performance.

Language Models	3gs(M)	3g hit(%)	Ppl	SER
Lm_1	20.31	46.9	126.6	27
lm_2	24.06	48.6	118.1	**26.2**
Lm_5	28.92	49.2	125.9	27
Lm_10	32.82	49.4	129.0	26.5
Lm_50	34.20	49.7	133.4	26.7
lm_100	34.93	49.7	134.8	26.9
lm_1000	36.11	49.88	137.7	27.3

Table 3. Results of experiments using the baseline method presented in (Seng et al., 2009)

5.1 Separate effect of boosting tri-gram counts

To see the effect of boosting tri-gram counts only, we have updated the counts of the tri-grams obtained from the 1-best segmentation (baseline approach) by the tri-gram counts of different multiple segmentations. Note that no new tri-grams are added here, and we evaluate only the effect and, therefore, the tri-gram hit remains the same as that of lm_1.

We have then developed different language models using the uni-gram and bi-gram counts of the first best segmentation and the updated trigram counts after multiple segmentation. The performance of the language models have been evaluated in terms of perplexity and their contribution to the performance improvement of a speech recognition system. We have observed (detailed results are not reported here) that boosting only the tri-gram counts has not contributed any improvement in the performance of the language models. The reason is probably due to the fact that simply updating tri-gram counts without updating the uni-grams and the bi-grams lead to a biased and inefficient LM.

5.2 Separate effect of new tri-grams

To explore the contributions of only newly added tri-grams, we have added their counts to the N-gram counts of Segmul1. It is important to note that the model obtained in that case is different from the baseline model whose results are presented in Table 3 (the counts of the tri-grams already found in the unique segmentation are different between models). As it is presented in table 4, including only the newly added tri-grams consistently improved tri-gram hits, while the improvement in perplexity stopped at Segmul10. Moreover, the use of only new tri-grams do not reduce the speech recognition error rate.

Language Models	3gs (M)	3g hit(%)	ppl	SER
lm_1	20.3	46.9	126.6	27
lm_2_new	24.4	48.7	119.1	26.9
lm_5_new	28.6	49.0	122.5	27.8
lm_10_new	30.7	49.2	124.2	27.9
lm_50_new	33.3	49.4	126.8	27.8
lm_100_new	35	49.8	127.8	28
lm_1000_new	36.1	49.9	129.7	27.9

Table 4. Contributions of new tri-grams

5.3 Pooling unique and multiple segmentation models

We have developed language models by pooling unique and multiple segmentation models altogether. For instance, all the N-grams of lm_5 multiple segmentation are pooled with all N-grams of lm_1 unique segmentation before estimating the language model probabilities. In other words, ngram-count command is used with multiple count files. The results are presented in table 5.

As it can be noted from table 5, we have got a significant improvement in all the evaluation criteria as compared with the performance of lm_1 that has perplexity of 126.6, tri-gram hit

of 46.91% and SER of 27. The best result obtained (25.4) shows a 0.8 absolute SER reduction compared to the best result presented in (Seng et al., 2009).

Language Models	3gs (M)	3g hit(%)	ppl	SER
lm_1	20.31	46.9	126.6	27
lm_2+lm_1	24.4	48.7	120.9	**25.4**
lm_5+lm_1	29.12	49.2	123.2	26.2
lm_10+lm_1	31.4	49.4	124.2	26
lm_50+lm_1	34.3	49.7	126	26
lm_100+lm_1	35	49.8	126.5	26.2
lm_1000+lm_1	36.2	49.9	128	26.2

Table 5. Performance with pooling

5.4 Cutting off rare tri-grams

With the assumption that bad N-grams occur rarely, we cut off rare tri-grams from the counts in developing language models. We consider all tri-grams with a count of 1 to be rare. Our hope, here, is that using this cut off we will remove bad N-grams introduced by the multiple segmentation approach, while keeping correct new N-grams in the model. Table 6 shows the performance of the language models developed with or without tri-gram cut off for the baseline method (the results presented on the lines indicating All3gs are the same as the ones presented in Table 3) .

Language models		Evaluation Criteria			
		3gs (M)	3g hit (%)	ppl	SER
lm_1	All 3gs	20.31	46.91	126.6	27
	Cut off	4.17	38.09	129.3	26.6
lm_2	All 3gs	24.06	48.6	118.1	26.2
	Cut off	5.11	39.6	121.0	26.7
lm_5	All 3gs	28.92	49.2	125.9	27
	Cut off	6.4	40.11	129.2	26.6
lm_10	All 3gs	32.82	49.41	129.0	26.5
	Cut off	6.98	40.27	132.4	26.6
lm_50	All 3gs	34.20	49.68	133.4	26.7
	Cut off	7.8	40.51	136.9	26.9
lm_100	All 3gs	34.93	49.74	134.8	26.9
	Cut off	7.98	40.59	138.4	26.8
lm_1000	All 3gs	36.11	49.88	137.7	27.3
	Cut off	8.33	40.71	141.3	26.8

Table 6. Performance with cut off.

The result shows that cutting off reduced the number of tri-grams highly (4 tri-grams over 5 are removed in that case). It, therefore, reduces the size of the language models significantly. Although the results obtained are not conclusive, a reduction of recognition error rate has been observed in four out of the seven cases while the perplexity increased and the tri-gram hits decreased in all cases.

5.5 Hybrid of pooling and cutting off methods

As it has been already indicated, cutting off increased the perplexity of the language models and decreased the tri-gram hits. To reduce the negative effect of cutting off on tri-gram hits and perplexity, we have developed language models using both pooling and cut off methods. We then cut off tri-grams of count 1 from the pooled N-grams. The result, as presented in table 7, shows that we can gain significant reduction in recognition error rate and improvement in tri-gram hits as compared to lm_1 that is developed with cut off, even if no improvement in perplexity is observed.

The best result obtained (25.9) shows a 0.3 absolute SER reduction compared to the best system presented in (Seng et al., 2009).

Language Models	3gs (M)	3g hit (%)	ppl	SE R
lm_1 (no cutoff)	20.3	46.9	126.6	27
lm_1 (cutoff)	4.2	38.1	129.3	26.6
lm_2+lm_1 (cutoff)	5.2	39.7	126.4	26.8
lm_5+lm_1 (cutoff)	6.4	40.2	129.5	**25.9**
lm_10+lm_1 (cutoff)	7.0	40.3	131.1	26.3
lm_50+lm_1 (cutoff)	7.8	40.5	133.5	26.4
lm_100+lm_1 (cutoff)	8.0	40.6	134.3	26.4
lm_1000+lm_1 (cutoff)	8.3	40.7	161.5	26.7

Table 7. Performance with hybrid method

6 Conclusion

The two major contributions of multiple segmentation are generation of new N-grams and boosting N-gram counts of those found in first best segmentation. However, it also produces bad N-grams that affect the performance of language models. In this paper, we studied the contribution of multiple segmentation approach more deeply and conducted experiments on efficient solutions to minimize the effect of adding bad N-grams. Since only boosting the tri-gram counts of first best segmentation and adding only new tri-grams did not reduce recognition error rate, we have proposed to pool all N-grams of N-best segmentations to that of first best segmentation and got a significant improvement in perplexity and tri-gram hits from

which we obtained the maximum (0.8 absolute) reduction in recognition error rate.

To minimize the effect of adding bad N-grams, we have cut off rare tri-grams in language modeling and got reduction in recognition error rate. The significant reduction of tri-grams that resulted from the cut off revealed that the majority of tri-grams generated by multiple segmentation have counts 1. Cutting off such a big portion of the trigrams reduced tri-gram hits and as a solution, we proposed a hybrid of both pooling and cutting off tri-grams from which we obtained a significant reduction in recognition error rate.

It is possible to conclude that our methods make the multiple segmentation approach more useful by minimizing the effect of bad N-grams that it generates and utilizing the contribution of different multiple segmentations.

However, we still see rooms for improvement. A systematic selection of new tri-grams (for example, based on the probabilities of the N-grams and/or application of simple linguistic criteria to evaluate the usefulness of new tri-grams), with the aim of reducing bad tri-grams, might lead to performance improvement. Thus, we will do experiments in this line. We will also apply these methods to other languages, such as Khmer.

References

Lee, Young-Suk, Papineni, Kishore, Roukos, Salim Emam, Ossama and Hassan, Hany. 2003. *Language model based arabic word segmentation.* In Proceedings of the ACL'03, pp. 399–406.

Le, Viet-Bac, Besacier, Laurent, Seng, Sopheap, Bigi, Brigite and Do, Thi-Ngoc-Diep. 2008. *Recent advances in automatic speech recognition for vietnamese.* SLTU'08, Hanoi Vietnam.

Mohri, Mehryar, Fernando C. N. Pereira, and Michael Riley, "A rational design for a weighted finite-state transducer library," in Lecture Notes in Computer Science. Springer, 1998, pp. 144–158.

Seng, Sopheap, Besacier, Laurent, Bigi, Brigitte, Castelli, Eric. 2009. *Multiple Text Segmentation for Statistical Language Modeling.* InterSpeech, Brighton, UK,

Stolcke, Andreas. 2002. SRILM: an extensible language modeling toolkit. Proceedings of International Conference on Spoken Language Processing, volume II, 901–904 . 129.88.65.115

Thai Sentence-Breaking for Large-Scale SMT

Glenn Slayden
thai-language.com
glenn@thai-language.com

Mei-Yuh Hwang
Microsoft Research
mehwang@microsoft.com

Lee Schwartz
Microsoft Research
leesc@microsoft.com

Abstract

Thai language text presents challenges for integration into large-scale multi-language statistical machine translation (SMT) systems, largely stemming from the nominal lack of punctuation and inter-word space. For Thai sentence breaking, we describe a monolingual maximum entropy classifier with features that may be applicable to other languages such as Arabic, Khmer and Lao. We apply this sentence breaker to our large-vocabulary, general-purpose, bidirectional Thai-English SMT system, and achieve BLEU scores of around 0.20, reaching our threshold of releasing it as a free online service.

1 Introduction

NLP research has consolidated around the notion of the *sentence* as the fundamental unit of translation, a consensus which has fostered the development powerful statistical and analytical approaches which incorporate an assumption of deterministic sentence delineation. As such systems become more sophisticated, languages for which this assumption is challenged receive increased attention. Thai is one such language, since it uses space neither to distinguish syllables from words or affixes, nor to unambiguously signal sentence boundaries.

Written Thai has no sentence-end punctuation, but a space character is always present between sentences. There is generally no space between words, but a space character may appear within a sentence according to linguistic or prescriptive orthographic motivation (Wathabunditkul 2003), and these characteristics disqualify sentence-breaking (SB) methods used for other languages, such as Palmer and Hearst (1997). Thai SB has therefore been regarded as the task of classifying each space that appears in a Thai source text as either sentence-breaking (**sb**) or non-sentence-breaking (**nsb**).

Several researchers have investigated Thai SB. Along with a discussion of Thai word breaking (WB), Aroonmanakun (2007) examines the issue. With a human study, he establishes that sentence breaks elicited from Thai informants exhibit varying degrees of consensus. Mittrapiyanuruk and Sornlertlamvanich (2000) define part-of-speech (POS) tags for **sb** and **nsb** and train a trigram model over a POS-annotated corpus. At runtime, they use the Viterbi algorithm to select the POS sequence with the highest probability, from which the corresponding space type is read back. Charoenpornsawat and Sornlertlamvanich (2001) apply Winnow, a multiplicative trigger threshold classifier, to the problem. Their model has ten features: the number of words to the left and right, and the left-two and right-two POS tags and words.

We present a monolingual Thai SB based on a maximum entropy (ME) classifier (Ratnaparkhi 1996; Reynar and Ratnaparkhi, 1997) which is suitable for sentence-breaking SMT training data and runtime inputs. Our model uses a four token window of Thai lemmas, plus categorical features, to describe the proximal environment of the space token under consideration, allowing runtime classification of space tokens with possibly unseen contexts.

As our SB model relies on Thai WB, we review our approach to this problem, plus related preprocessing, in the next section. Section 2 also discusses the complementary operation to WB, namely, the re-spacing of Thai text generated by SMT output. Section 3 details our SB model and evaluates its performance. We describe the integration of this work with our large-scale SMT system in Section 4. We draw conclusions in Section 5.

Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), pages 8–16,
the 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010

2 Pre- and Post-processing

As will be shown in Section 3, our sentence breaker relies on Thai WB. In turn, with the aim of minimizing WB errors, we perform Unicode character sequence normalization prior to WB. As output byproducts, our WB analysis readily identifies certain types of named entities which we propagate into our THA-ENG SMT; in this section, we briefly summarize these preliminary processing steps, and we conclude the section with a discussion of Thai text re-spacing.

2.1 Character Sequence Normalization

Thai orthography uses an alphabet of 44 consonants and a number of vowel glyphs and tone marks. The four Thai tone marks and some Thai vowel characters are super- and/or sub-scripted with respect to a base character. For example, the อิ้ sequence consists of three code points: อ ◌ิ ◌้. When two or more of these combining marks are present on the same base character, the ordering of these code points in memory should be consistent so that orthographically identical entities are recognized as equivalent by computer systems. However, some computer word processors do not enforce the correct sequence or do not properly indicate incorrect sequences to the user visually. This often results in documents with invalid byte sequences.

Correcting these errors is desirable for SMT inputs. In order to normalize Thai input character sequences to a canonical Unicode form, we developed a finite state transducer (FST) which detects and repairs a number of sequencing errors which render Thai text either orthographically invalid, or not in a correct Unicode sequence.

For example, a superscripted Thai tone mark should follow a super- or sub-scripted Thai vowel when they both apply to the same consonant. When the input has the tone mark and the vowel glyph swapped, the input can be fully repaired:

อ า ◌่ น → อ ◌่ า น → อ่าน
อ ◌้ ◌ิ น → อ ◌ิ ◌้ น → อิ้น

Figure 1. Two unambiguous repairs

Other cases are ambiguous. The occurrence of multiple adjacent vowel glyphs is an error where the intention may not be clear. We retain the first-appearing glyph, unless it is a pre-posed vowel, in which case we retain the last-appearing instance. These two treatments are contrasted in Figure 2. Miscoding (Figure 3) is another variety of input error that is readily repaired.

จะะา → จะ
ใไป → ไป

Figure 2. Two ambiguous repairs

Within the Infoquest Thai newswire corpus, a low-noise corpus, about 0.05% of the lines exhibit at least one of the problems mentioned here. For some chunks of broad-range web scraped data, we observe rates as high as 4.1%. This measure is expected to under-represent the utility of the filter to WB, since Thai text streams, lacking intra-word spacing and permitting two unwritten vowels, have few re-alignment checkpoints, allowing tokenization state machines to linger in misaligned states.

อ ◌ำ า → อ ◌ำ → อำ
เ เ อ → แ อ → แอ

Figure 3. Two common mis-codings

2.2 Uniscribe Thai Tokenization

Thai text does not normally use the space character to separate words, except in certain specific contexts. Although Unicode offers the Zero-Width Space (ZWSP) as one solution for indicating word breaks in Thai, it is infrequently used. Programmatic tokenization has become a staple of Thai computational linguistics. The problem has been well studied, with precision and recall near 95% (Haruechaiyasak et al. 2008).

In our SMT application, both the sentence breaker and the SMT system itself require Thai WB, and we use the same word breaker for these tasks (although the system design currently prohibits directly passing tokens between these two components). Our method is to apply post-processing heuristics to the output of *Uniscribe* (Bishop et al. 2003), which is provided as part of the Microsoft® Windows™ operating system interface. Our heuristics fall into two categories: "re-gluing" words that *Uniscribe* broke too aggressively, and a smaller class of cases of further breaking of words that *Uniscribe* did not break.

Re-gluing is achieved by comparing *Uniscribe* output against a Thai lexicon in which desired breaks within a word are tagged. Underbreaking by *Uniscribe* is less common and is restricted to a number of common patterns which are repaired explicitly.

2.3 Person Name Entities

In written Thai, certain types of entities employ prescriptive whitespace patterns. By removing these recognized patterns from consideration, SB precision can be improved. Furthermore, because our re-gluing procedure requires a lookup of every syllable proposed by *Uniscribe*, it is efficient to consider, during WB, additional processing that can be informed by the same lookup. Accordingly, we briefly mention some of the entity types that our WB identifies, focusing on those that incorporate distinctive spacing patterns.

Person names in Thai adhere to a convention for the use of space characters. This helps Thai readers to identify the boundaries of multi-syllable surnames that they may not have seen before. The following grammar summarizes the prescriptive conventions for names appearing in Thai text:

```
<name-entity> ::= <honorific>  <full-name>
<full-name> ::= <first-name> [<last-name>]
<first-name> ::= <name-text> space
<last-name> ::= <name-text> space
<name-text> ::= <thai-alphabetic-char>+
<thai-alphabetic-char> ::= ก | ข | ฃ | ค | ...
```

Figure 4. Name entity recognition grammar

The re-glue lookup also determines if a syllable matches one of the following predefined special categories: name-introducing honorific (h), Thai or foreign given name (g), token which is likely to form part of a surname (s), or token which aborts the gathering of a name (i.e. is unlikely to form part of a name).

.../ว่า/นาย/จิ/ระ/นุช/ /วิ/นิจ/จกู/ล/ /ว่า/...											
ว่า	นาย	จิ	ระ	นุช		วิ	นิจ	จกู	ล		ว่า
	h	g0	g1	g2	sp0	s0	s1	s2	s3	sp1	
that	Mr.	<oov>	hit	beloved		<oov>	stable	<oov>	<oov>		said
...that	Mr.	Chiranut				Winichotkun					said...

Figure 5. Thai person-name entity recognition

Figure 5 shows a Thai name appearing within a text fragment, with *Uniscribe* detected token boundaries indicated by slashes. In the third row we have identified the special category, if any, for each token. The fourth line shows the English translation gloss, or <oov> if none. The bottom row is the desired translation output.

Our name identifier first notes the presence of an honorific {h} **นาย** followed by a pattern of tokens {g0-g*n*}, {s0-s*n*} and spaces {sp0, sp1} that is compatible with a person name and surname of sensible length.

Next, we determine which of those tokens in the ranges {g} and {s} following the honorific do not have a gloss translation (i.e., are not found in the lexicon). These tokens are indicated by <oov> in the gloss above. When the number of unknown tokens exceeds a threshold, we hypothesize that these tokens form a name. The lack of lexical morphology in Thai facilitates this method because token (or syllable) lookup generally equates with the lookup of a stemmed lemma.

2.4 Calendar Date Entities

Our WB also identifies Thai calendar dates, as these also exhibit a pattern which incorporates spaces. As a prerequisite to identifying dates, we map Thai orthographic digits {๐ ๑ ๒ ๓ ๔ ๕ ๖ ๗ ๘ ๙} to Arabic digits 0 through 9, respectively. For example, our system would interpret the input text **๒๕๔๐** as equivalent to "2540."

.../ใน/วัน/ที่/ /14/ มีนาคม/ /๒๕๔๐/ /และ/...									
ใน	วัน	ที่	sp	14	มีนาคม	sp	๒๕๔๐	sp	และ
on	day	which		14	March		2540		and
...on	March 14th, 1997								and...

Figure 6. Date entity recognition

Figure 6 shows a fragment of Thai text which contains a calendar date for which our system will emit a single token. As shown in the example, our system detects and adjusts for the use of Thai Buddhist year dates when necessary. Gathering of disparate and optional parts of the Thai date is summarized by the grammar in Figure 7.

```
<date-entity> ::= [<cardinal-words>] [space] <date>
<cardinal-words> ::= วันที่ | ที่
<date> ::= month-date [space] year
<year> ::= <tha-digit> <tha-digit> <tha-digit> <tha-digit>
<year> ::= <ara-digit> <ara-digit> <ara-digit> <ara-digit>
<month-date> ::= <day> [space] <month>
<day> ::= <thai-digit>+
<day> ::= <ara-digit>+
<month> ::= <month-full> | <month-abbr>
<month-full> ::= มกราคม | กุมภาพันธ์ | มีนาคม | ...
<month-abbr> ::= ม.ค. | ก.พ. | มี.ค. | ...
<tha-digit> ::= ๐ | ๑ | ๒ | ๓ | ๔ | ๕ | ๖ | ๗ | ๘ | ๙
<ara-digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Figure 7. Date recognition grammar

2.5 Thai Text Re-spacing

To conclude this section, we mention an operation complementary to Thai WB, whereby Thai words output by an SMT system must be re-spaced in accordance with Thai prescriptive convention. As will be mentioned in Section 4.2, for each input sentence, our English-Thai system has access to an English dependency parse tree, as well as links between this tree and a Thai transfer dependency tree. After using these links to transfer syntactic information to the Thai tree, we are able to apply prescriptive spacing rules (Wathabunditkul 2003) as closely as possible. Human evaluation showed satisfactory results for this process.

3 Maximum Entropy Sentence-Breaking

We now turn to a description of our statistical sentence-breaking model. We train an ME classifier on features which describe the proximal environment of the space token under consideration and use this model at runtime to classify space tokens with possibly unseen contexts.

3.1 Modeling

Under the ME framework, let B={**sb**, **nsb**} represent the set of possible classes we are interested in predicting for each space token in the input stream. Let C={linguistic contexts} represent the set of possible contexts that we can observe, which must be encoded by binary features, $f_j(b,c), 1 \le j \le k$, such as:

$$f_1(b,c) = \begin{cases} 1 \text{ if the previous word is English } \textit{and } b = \textbf{nsb}. \\ 0 \text{ otherwise.} \end{cases}$$

This feature helps us learn that the space after an English word is usually not a sentence boundary.

$$f_2(b,c) = \begin{cases} 1 \text{ if the distance to the previous honorific} \\ \text{is less than 15 tokens } \textit{and } b = \textbf{nsb} \\ 0 \text{ otherwise.} \end{cases}$$

This feature enables us to learn that spaces which follow an honorific are less likely to mark sentence boundaries. Assume the joint probability *p(b,c)* is modeled by

$$p(b,c) = Z \prod_{j=1}^{k} \alpha_j^{f_j(b,c)}$$

where we have k free parameters $\{\alpha_j\}$ to estimate and Z is a normalization factor to make $\sum_{b,c} p(b,c) = 1$. The ME learning algorithm finds a solution $\{\alpha_j\}$ representing the most uncertain commitment

$$\max \; H(p) = -\sum p(b,c) \log p(b,c)$$

that satisfies the observed distribution $\hat{p}(b,c)$ of the training data

$$\sum p(b,c) f_j(b,c) = \sum \hat{p}(b,c) f_j(b,c), \; 1 \le j \le k \, .$$

This is solved via the *Generalized Iterative Scaling* algorithm (Darroch and Ratcliff 1972). At run-time, a space token is considered an **sb**, if and only if p(**sb**|c) > 0.5, where

$$p(\textbf{sb}|c) = \frac{p(\textbf{sb},c)}{p(\textbf{sb},c) + p(\textbf{nsb},c)} \, .$$

3.2 Feature Selection

The core context of our model, {*w*, *x*, *y*, *z*}, is a window spanning two tokens to the left (positions *w* and *x*) and two tokens to the right (positions *y* and *z*) of a classification candidate space token.

c	**token characteristic**
yk	Yamok (syllable reduplication) symbol ๆ
sp	space
๐๙	Thai numeric digits
num	Arabic numeric digits
ABC	Sequence of all capital ASCII characters
c*nn*	single character (derived from hex)
c*kkmmnn*	single character (derived from UTF8 hex)
ascii	any amount of non-Thai text
(Thai text)	Thai word (derived from lemma)

Table 1. Categorical and derived feature names

The possible values of each of the window positions {*w*, *x*, *y*, *z*} are shown in Table 1, where the first match to the token at the designated position is assigned as the feature value for that position. Foreign-text tokens plus any intervening space are merged, so a single "ascii" feature may represent an arbitrary amount of non-Thai script with interior space.

Figure 8 shows an example sentence that has been tokenized. Token boundaries are indicated by slashes. Although there are three space tokens in the original input, we extract four contexts. The shaded boxes in the source text—and the shaded line in the figure—indicate the single **sb** context that is synthesized by wrapping, to be described in Section 3.4.

For each context, in addition to the {*w, x, y, z*} features, we extract two more features indicated by {*l* ,*r*} in Figure 8. They are the number of

tokens between the previous space token (wrapping as necessary) and the current one, and the number of tokens between the current space token and the next space token (wrapping as necessary). These features do not distinguish whether the bounding space token is **sb** or **nsb**. This is because, processing left-to-right, it is permissible to use a feature such as "number of tokens since last **sb**," but not "number of tokens until next **sb**," which would be available during training but not at runtime.

ลักษณะการอ้างอิงแบบ R1C1 ถูกแปลงไปเป็นลักษณะการอ้างอิงแบบ A1

"R1C1 reference style was converted to A1 reference style."

/ลักษณะ/การ/อ้าง/อิง/แบบ/ /R1C1/ /ถูก/แปลง/ไป/เป็น/ลักษณะ/การ/อ้าง/อิง/แบบ/ /A1/

b	c=*w*	*c=x*	*c=y*	*c=z*	*c=l*	*c=r*
nsb	อิง	แบบ	ABC	sp	5	1
nsb	sp	ABC	ถูก	แปลง	1	9
nsb	อิง	แบบ	ABC	sp	9	1
sb	sp	ABC	ลักษณะ	การ	1	5

Figure 8. A Thai sentence and the training contexts extracted. Highlighting shows the context for **sb**.

In addition to the above core features, our model emits certain extra features only if they appear:

- An individual feature for each English punctuation mark, since these are sometimes used in Thai. For example, there is one feature for the sentence end period (i.e. full-stop);
- The current nest depth for paired glyphs with directional variation, such as brackets, braces, and parentheses;
- The current parity value for paired glyphs without directional distinction such as "straight" quotation marks.

The following example illustrates paired directional glyphs (in this case, parentheses):

.../ยูนิลิเวอร์/ /(/ประเทศ/ /ไทย/)/ /จำกัด/ /เปิดเผย/ว่า/...
...Unilever (Thailand) Ltd. disclosed that...

b	c=*w*	*c=x*	*c=y*	*c=z*	*c=pn*
nsb	(	**ประเทศ**	**ไทย**	)	1

Figure 9. Text fragment illustrating paired directional glyphs and the context for the highlighted space

In Figure 9, the space between **ประเทศ** "country" and **ไทย** "Thai," generates an **nsb** context which includes the features shown, where "pn" is an extra feature which indicates the parenthesis nesting level. This feature helps the model learn that spaces which occur within parentheses are likely to be **nsb**.

Parity features for the non-directional paired glyphs, which do nest, are true binary features. Since these features have only two possible values (*inside* or *outside*), they are only emitted when their value is "inside," that is, when the space under consideration occurs between such a pair.

3.3 Sentence Breaker Training Corpus

Thai corpora which are marked with sentence breaks are required for training. We assembled a corpus of 361,802 probable sentences. This corpus includes purchased, publicly available, and web-crawled content. In total it contains 911,075 spaces, a figure which includes one inter-sentence space per sentence, generated as described below.

3.4 Out-of-context Sentences

For SB training, paragraphs are first tokenized into words as described in Section 2.2. This process does not introduce new spaces between tokens; only original spaces in the text are classified as **sb**/**nsb** and used for the context features described below. To keep this distinction clear, token boundaries are indicated by a slash rather than space in the examples shown in this paper.

For 91% of our training sentences, the paragraphs from which they originate are inaccessible. In feature extraction for each of these sentences, we wrap the sentence's head around to its tail to obtain its **sb** context. In other words, for a sentence of tokens t_0-t_{n-1}, the context of **sb** (the last space) is given by

$$\{ w=t_{n-2}, x=t_{n-1}, y=t_0, z=t_1 \}.$$

This process was illustrated in Figure 8. Although not an ideal substitute for sentences in context, this ensures that we extract at least one **sb** context per sentence. The number of **nsb** contexts extracted per sentence is equal to the number of interior space tokens in the original sentence. Sentence wrapping is not needed when training with sentence-delimited paragraph sources. Contexts **sb** and **nsb** are extracted from the token stream of the entire paragraph and wrapping is used only to generate one additional **sb** for the entire paragraph.

3.5 Sentence Breaker Evaluation

Although evaluation against a single-domain corpus does not measure important design requirements of our system, namely resilience to broad-domain input texts, we evaluated against the ORCHID corpus (Charoenporn et al. 1997) for the purpose of comparison with the existing literature. Following the methodology of the studies cited below, we use 10-fold ×10% averaged testing against the ORCHID corpus.

Our results are consistent with recent work using the Winnow algorithm, which itself compares favorably with the probabilistic POS trigram approach. Both of these studies use evaluation metrics, attributed to Black and Taylor (1997), which aim to more usefully measure sentence-breaker utility. Accordingly, the following definitions are used in Table 2:

$$\text{space-correct} = \frac{(\#\text{correct sb}+\#\text{correct nsb})}{\text{total \# of space tokens}}$$

$$\text{false break} = \frac{\#\text{sb false positives}}{\text{total \# of space tokens}}$$

It was generally possible to reconstruct precision and recall figures from these published results[1] and we present a comprehensive table of results. Reconstructed values are marked with a dagger and the optimal result in each category is marked in boldface.

	Mittrapiyanuruk et al.	Charoenpornsawat et al.	Our result
method	**POS Trigram**	**Winnow**	**MaxEnt**
#sb in reference	10528	1086†	2133
#space tokens	33141	3801	7227
nsb-precision	90.27†	91.48†	**93.18**
nsb-recall	87.18†	**97.56**†	94.41
sb-precision	74.35†	**92.69**†	86.21
sb-recall	79.82	77.27	**83.50**
"space-correct"	85.26	89.13	**91.19**
"false-break"	8.75	**1.74**	3.94

Table 2. Evaluation of Thai Sentence Breakers against ORCHID

Finally, we would be remiss in not acknowledging the general hazard of assigning sentence breaks in a language such as Thai, where source text authors may intentionally include or omit spaces in order to create syntactic or semantic ambiguity. We defer to Mittrapiyanuruk and Sornlertlamvanich (2000) and Aroonmanakun (2007) for informed commentary on this topic.

[1] Full results for Charoenpornsawat et al. are reconstructed based on remarks in their text, including that "the ratio of the number of [**nsb** to **sb**] is about 5:2."

4 SMT System and Integration

The primary application for which we developed the Thai sentence breaker described in this work is the Microsoft® BING™ general-domain machine translation service. In this section, we provide a brief overview of this large-scale SMT system, focusing on Thai-specific integration issues.

4.1 Overview

Like many multilingual SMT systems, our system is based on hybrid generative/discriminative models. Given a sequence of foreign words, $\boldsymbol{f}$, its best translation is the sequence of target words, $\boldsymbol{e}$, that maximizes

$$\boldsymbol{e}^* = \text{argmax}_{\boldsymbol{e}}\, p(\boldsymbol{e}|\boldsymbol{f}) = \text{argmax}_{\boldsymbol{e}}\, p(\boldsymbol{f}|\boldsymbol{e})p(\boldsymbol{e})$$

$$= \text{argmax}_{\boldsymbol{e}}\, \{\log p(\boldsymbol{f}|\boldsymbol{e}) + \log p(\boldsymbol{e})\}$$

where the translation model $p(\boldsymbol{f}|\boldsymbol{e})$ is computed on dozens to hundreds of features. The target language model (LM), $p(\boldsymbol{e})$, is represented by a smoothed n-grams (Chen 1996) and sometimes more than one LM is adopted in practice. To achieve the best performance, the log likelihoods evaluated by these features/models are linearly combined. After $p(\boldsymbol{f}|\boldsymbol{e})$ and $p(\boldsymbol{e})$ are trained, the combination weights λ_i are tuned on a held-out dataset to optimize an objective function, which we set to be the BLEU score (Papineni et al. 2002):

$$\{\lambda_i^*\} = \max_{\{\lambda i\}} \text{BLEU}(\{e^*\},\{r\})$$

$$\boldsymbol{e}^* = \text{argmax}_{\boldsymbol{e}}\, \{\sum_i \lambda_i \log\, p_i(\boldsymbol{f}|\boldsymbol{e}) + \sum_j \lambda_j \log p_j(\boldsymbol{e})\}$$

where $\{r\}$ is the set of gold translations for the given input source sentences. To learn λ_i we use the algorithm described by Och (2003), where the decoder output at any point is approximated using n-best lists, allowing an optimal line search to be employed.

4.2 Phrasal and Treelet Translation

Since we have a high-quality real-time rule-based English parser available, we base our Eng-

lish-to-Thai translation (ENG-THA) on the "treelet" concept suggested in Menezes and Quirk (2008). This approach parses the source language into a dependency tree which includes part-of-speech labels.

Lacking a Thai parser, we use a purely statistical phrasal translator after Pharaoh (Koehn 2004) for THA-ENG translation, where we adopt the name and date translation described in Sections 2.3 and 2.4.

We also experimented with phrasal ENG-THA translation. Though we actually achieved a slightly better BLEU score than treelet for this translation direction, qualitative human evaluation by native speaker informants was mixed. We adopted the treelet ENG-THA in the final system, for its better re-spacing (Section 2.5).

4.3 Training, Development and Test Data

Naturally, our system relies on parallel text corpora to learn the mapping between two languages. The parallel corpus contains sentence pairs, corresponding to translations of each other. For Thai, quality corpora are generally not available in sufficient quality for training a general-domain SMT system. For the ENG-THA pair, we resort to Internet crawls as a source of text. We first identify paired documents, break each document into sentences, and align sentences in one document against those in its parallel document. Bad alignments are discarded. Only sentence pairs with high alignment confidence are kept in our parallel corpus. Our sentence alignment algorithm is based on Moore (2002).

For our ENG-THA translation system, we assembled three resources: a parallel training corpus, a development bitext (also called the lambda set) for training the feature combination weights $\{\lambda_i\}$, and a test corpus for BLEU and human evaluation. Both the lambda and the test sets have single reference translations per sentence.

Data Set	#Sentences
(ENG||THA) training	725K
(ENG,THA) lambda	2K
(ENG,THA) test	5K
THA LM text	10.3M
ENG LM text	45.6M

Table 3. Corpus size of parallel and monolingual data

Although it is well known that language translation pairs are not symmetric, we use these same resources to build our THA-ENG translation system due to the lack of additional corpora.

Our parallel MT corpus consists of approximately 725,000 English-Thai sentence pairs from various sources. Additionally we have 9.6 million Thai sentences, which are used to train a Thai 4-gram LM for ENG-THA translation, together with the Thai sentences in the parallel corpus. Trigrams and 4-grams that occur only once are pruned, and n-gram backoff weights are re-normalized after pruning, with the surviving KN smoothed probabilities intact (Kneser and Ney 1995). Similarly, a 4-gram ENG LM is trained for THA-ENG translation, on a total of 45.6M English sentences.

For both the lambda and test sets, THA LM incurs higher out-of-vocabulary (OOV) rates (1.6%) than ENG LM (0.7%), due to its smaller training set and thus smaller lexicon. Both translation directions define the maximum phrase/treelet length to be 4 and the maximum re-ordering jump to be 4 as well.

4.4 BLEU Scores

To evaluate our end-to-end performance, we compute case insensitive 4-gram BLEU scores. Translation outputs are WB first according to the Thai/English tokenizer, before BLEU scores are computed. The BLEU scores on the test sets are shown in Table 4. We are not aware of any previously published BLEU results for either direction of this language pair.

	BLEU
THA-ENG	0.233
ENG-THA	0.194

Table 4. Four-gram case-insensitive BLEU scores.

Figures 10 and 11 illustrate sample outputs for the each translation direction, with reference translations.

INPUT: ในประเทศไทยมีกล้วยไม้ประมาณ ๑๗๕ ชนิด ถ้าสูญพันธุ์ไปจากประเทศไทย ก็หมายถึงสูญพันธุ์ไปจากโลก
OUTPUT: In Thailand a Orchid approximately 175 type if extinct from Thailand. It means extinct from the world.
REF: In Thailand, there are about 175 species of Orchid. If they disappear from Thailand, they will be gone from the world.

Figure 10. THA-ENG Sample Translation Output

INPUT: In our nation the problems and barriers we face are just problems and barriers of law not selection or development.
OUTPUT: ในประเทศชาติของเรา ปัญหาและอุปสรรคที่เราเผชิญอยู่เพียงปัญหาและอุปสรรคของกฎหมายไม่เลือกหรือพัฒนา
REF: ในประเทศของเราปัญหาและอุปสรรค ก็เป็นปัญหาอุปสรรคทางด้านกฎหมาย แต่ไม่เป็นปัญหาอุปสรรคในการคัดเลือกและพัฒนาพันธุ์

Figure 11. ENG-THA Sample Translation Output

Although the translation quality is far from being perfect, SMT is making good process on building useful applications.

5 Conclusion and Future Work

Our maximum entropy model for Thai sentence-breaking achieves results which are consistent with contemporary work in this task, allowing us to overcome this obstacle to Thai SMT integration. This general approach can be applied to other South-East Asian languages in which space does not deterministically delimit sentence boundaries.

In Arabic writing, commas are often used to separate sentences until the end of a paragraph when a period is finally used. In this case, the comma character is similar to the space token in Thai where its usage is ambiguous. We can use the same approach (perhaps with different linguistic features) to identify which commas are sentence-breaking and which are not.

Our overall system incorporates a range of independent solutions to problems in Thai text processing, including character sequence normalization, tokenization, name and date identification, sentence-breaking, and Thai text respacing. We successfully integrated each solution into an existing large-scale SMT framework, obtaining sufficient quality to release the Thai-English language pair in a high-volume, general-domain, free public online service.

There remains much room for improvement. We need to find or create true Thai-English directional corpora to train the lambdas and to test our models. The size of our parallel corpus for Thai should increase by at least an order of magnitude, without loss of bitext quality. With a larger corpus, we can consider longer phrase length, higher-order n-grams, and longer reordering distance.

References

W. Aroonmanakun. 2007. Thoughts on Word and Sentence Segmentation in Thai. In *Proceedings of the Seventh International Symposium on Natural Language Processing, Pattaya, Thailand, 85-90*.

F. Avery Bishop, David C. Brown and David M. Meltzer. 2003. Supporting Multilanguage Text Layout and Complex Scripts with Windows 2000. *http://www.microsoft.com/typography/developers/uniscribe/intro.htm*

A. W. Black and P. Taylor. 1997. Assigning Phrase Breaks from Part-of-Speech Sequences. *Computer Speech and Language*, 12:99-117.

Thatsanee Charoenporn, Virach Sornlertlamvanich, and Hitoshi Isahara. 1997. *Building A Thai Part-Of-Speech Tagged Corpus (ORCHID)*.

Paisarn Charoenpornsawat and Virach Sornlertlamvanich. 2001. Automatic sentence break disambiguation for Thai. In *International Conference on Computer Processing of Oriental Languages (ICCPOL)*, 231-235.

S. F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, 310-318. Morristown, NJ: ACL.

J. N. Darroch and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5): 1470-1480.

Choochart Haruechaiyasak, Sarawoot Kongyoung, and Matthew N. Dailey. 2008. A Comparative Study on Thai Word Segmentation Approaches. In *Proceedings of ECTICON 2008*. Pathumthani, Thailand: ECTI.

Reinhard Kneser and Hermann Ney. 1995. Improved Backing-Off for M-Gram Language Modeling. In *Proceedings of International Conference on Acoustics, Speech and Signal Procesing (ICASSP)*, 1:181-184.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of the As-*

sociation of Machine Translation in the Americas (AMTA-2004).

Arul Menezes, and Chris Quirk. 2008. Syntactic Models for Structural Word Insertion and Deletion during Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

P. Mittrapiyanuruk and V. Sornlertlamvanich. 2000. The Automatic Thai Sentence Extraction. In *Proceedings of the Fourth Symposium on Natural Language Processing*, 23-28.

Robert C. Moore. 2002. Fast and Accurate Sentence Alignment of Bilingual Corpora. In *Machine Translation: From Research to Real Users (Proceedings, 5th Conference of the Association for Machine Translation in the Americas, Tiburon, California)*, Springer-Verlag, Heidelberg, Germany, 135-244

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics*. Stroudsburg, PA: ACL.

David D. Palmer and Marti A. Hearst. 1997. Adaptive Multilingual Sentence Boundary Disambiguation. *Computational Linguistics*, 23:241-267.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual meeting of the Association for Computational Linguistics*, 311–318. Stroudsburg, PA: ACL.

Adwait Ratnaparkhi, 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 133-142.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries, In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 16-19.

Suphawut Wathabunditkul. 2003. Spacing in the Thai Language. *http://www.thailanguage.com/ref/spacing*

Clause Identification and Classification in Bengali

Aniruddha Ghosh[1] **Amitava Das**[2] **Sivaji Bandyopadhyay**[3]
Department of Computer Science and Engineering
Jadavpur University
arghyaonline@gmail.com[1] amitava.santu@gmail.com[2] sivaji_cse_ju@yahoo.com[3]

Abstract

This paper reports about the development of clause identification and classification techniques for Bengali language. A syntactic rule based model has been used to identify the clause boundary. For clause type identification a Conditional random Field (CRF) based statistical model has been used. The clause identification system and clause classification system demonstrated 73% and 78% precision values respectively.

1 Introduction

The clause identification is one of the shallow semantic parsing tasks, which is important in various NLP applications such as Machine Translation, parallel corpora alignment, Information Extraction and speech applications. Grammatically a clause is a group of words having a subject and a predicate of its own, but forming part of a sentence. Clause boundary identification of natural language sentences poses considerable difficulties due to the ambiguous nature of natural languages. Clause classification is a convoluted task as natural language is generally syntactically rich in formation of sentences or clauses.

By the classical theory of Panini (Paul and Staal, 1969) a clause is the surface level basic syntactic element which holds the basic dependent semantics (i.e. lexical semantic have no dependency) to represent the overall meaning of any sentence. This syntactic to semantic derivation proceeds through two intermediate stages: the level of *karaka* relations, which are comparable to the thematic role types and the level of inflectional or derivational morphosyntax.

Fillmore's Case Grammar (Fillmore et. al, 2003), and much subsequent work, revived the Panini's proposals in a modern setting. A main objective of Case Grammar was to identify syntactic positions of semantic arguments that may have different realizations in syntax.

In the year of 1996 Bharati et al. (1996) defines the idea of Chunk or local word group for Indian languages. After the successful implementation of Shakti[1] , the first publicly available English-Hindi machine translation system the idea of chunk became the most acceptable syntactic/semantic representation format for Indian languages, known as Shakti Standard Format (SSF).

In 2009 Bali et al. (2009) redefines the idea of chunk and establishes that the idea of chunking varies with prosodic structure of a language. Boundary of chunk level is very ambiguous itself and can differ by writer or speaker according to their thrust on semantic.

Therefore it is evident that automatic clause identification for Indian languages needs more research efforts. In the present task, clause boundary identification is attempted using the classical theory of Panini and the Case Grammar approach of Fillmore on the shallow parsed output in SSF structure. It may be worth mentioning that several basic linguistic tools in Indian languages such as part of speech tagger, chunker, and shallow parser follow SSF[2] as a standard.

Previous research on clause identification was done mostly on the English language (Sang and Dejean, 2001). There have been limited efforts on clause identification for Indian languages. One such effort is proposed in Ram and Devi,

[1] http://shakti.iiit.ac.in/
[2] http://ltrc.iiit.ac.in/MachineTrans/research/tb/shakti-analy-ssf.pdf

Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), pages 17–25,
the 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010

(2008) with statistical method. The idea of generative grammar based on rule-based descriptions of syntactic structures introduced by Chomsky (Chomsky, 1956) points out that every language has its own peculiarities that cannot be described by standard grammar. Therefore a new concept of generative grammar has been proposed by Chomsky. Generative grammar can be identified by statistical methods. In the present task, conditional random field (CRF) [3] -based machine learning method has been used in clause type classification. According to the best of our knowledge this is the first effort to identify and classify clauses in Bengali.

The present system is divided into two parts. First, the clause identification task aims to identify the start and the end boundaries of the clauses in a sentence. Second, Clause classification system identifies the clause types.

Analysis of corpus and standard grammar of Bengali revealed that clause boundary identification depends mostly on syntactic dependency. For this reason, the present clause boundary identification system is rule based in nature. Classification of clause is a semantic task and depends on semantic properties of Bengali language. Hence we follow the theory of Chomsky's generative grammar to disambiguate among possible clause types. The present classification system of clause is a statistics-based approach. A conditional random field (CRF) based machine learning method has been used in the clause classification task. The output of the rule based identification system is forwarded to the machine learning model as input.

The rest of the paper is organized as follows. In section 2 we elaborate the rule based clause boundary identification. The next section 3 describes the implementation detail with all identified features for the clause classification problem. Result section 4 reports about the accuracy of the hybrid system. In error analysis section we reported the limitations of the present system. The conclusion is drawn in section 5 along with the future task direction.

2 Resource Acquisition

Bengali belongs to Indo-Aryan language family. A characteristic of Bengali is that it is under-resourced. Language research for Bengali got attention recently. Resources like annotated corpus and linguistics tools for Bengali are very rarely available in the public domain.

2.1 Corpus

We used the NLP TOOLS CONTEST: ICON 2009[4] dependency relation marked training dataset of 980 sentences for training of the present system. The data has been further annotated at the clause level. According to the standard grammar there are two basic clause types such as Principal clause and Subordinate clause. Subordinate clauses have three variations as Noun clause, Adjective clause and Adverbial clause. The tagset defined for the present task consists of four tags as Principal clause (PC), Noun clause (NC), Adjective clause (AC) and Adverbial clause (RC). The annotation tool used for the present task is Sanchay[5]. The detailed statistics of the corpus are reported in Table 1.

	Train	Dev	Test
No of Sentences	980	150	100

Table 1: Statistics of Bengali Corpus

2.1.1 Annotation Agreement

Two annotators (Mr. X and Mr. Y) participated in the present task. Annotators were asked to identify the clause boundaries as well as the type of the identified clause. The agreement of annotations among two annotators has been evaluated. The agreements of tag values at clause boundary level and clause type levels are listed in Table 2.

	Boundary	Type
Percentage	76.54%	89.65%

Table 2: Agreement of annotators at clause boundary and type level

It is observed from the Table 2 that clause boundary identification task has lower agreement value. A further analysis reveals that there are almost 9% of cases where clause boundary has nested syntactic structure. These types of clause boundaries are difficult to identify. One of such cases is Inquisitive semantic (Groenendijk, 2009) cases, ambiguous for human annota-

[3] http://crf.sourceforge.net/

[4] http://ltrc.iiit.ac.in/nlptools2009/

[5] http://ltrc.iiit.ac.in/nlpai_contest07/Sanchay/

tors too. It is better to illustrate with some specific example.

```
If John goes to the party,
will Mary go as well?
```

In an inquisitive semantics for a language of propositional logic the interpretation of disjunction is the source of inquisitiveness. Indicative conditionals and conditional questions are treated both syntactically and semantically. The semantics comes with a new logical-pragmatically notion that judges and compares the compliance of responses to an initiative in inquisitive dialogue (Groenendijk, 2009). Hence it is evident that these types of special cases need special research attention.

2.2 Shallow Parser

Shallow parser[6] for Indian languages, developed under a Government of India funded consortium project named Indian Language to Indian Language Machine Translation System (IL-ILMT), are now publicly available. It is a well developed linguistic tool and produce good credible analysis. For the present task the linguistic analysis is done by the tool and it gives output as pruned morphological analysis at each word level, part of speech at each word level, chunk boundary with type-casted chunk label, vibhakti computation and chunk head identification.

2.3 Dependency parser

A dependency parser for Bengali has been used as described in Ghosh et al. (2009). The dependency parser follows the tagset[7] identified for Indian languages as a part of NLP TOOLS CONTEST 2009 as a part of ICON 2009.

3 Rule-based Clause Boundary Identification

Analysis of a Bengali corpus and standard grammar reveals that clause boundaries are directly related to syntactic relations at sentence level. The present system first identifies the number of verbs present in a sentence and subsequently finds out dependant chunks to each verb. The set of identified chunks that have relation with a particular verb is considered as a clause. But some clauses have nested syntactic formation, known as inquisitive semantic. These clauses are difficult to identify by using only syntactic relations. The present system has limitations on those inquisitive types of clauses.

Bengali is a verb final language. Most of the Bengali sentences follow a Subject-Object-Verb (SOV) pattern. In Bengali, subject can be missing in a clause formation. Missing subjects and missing keywords lead to ambiguities in clause boundary identification. In sentences which do not follow the SOV pattern, chunks that appear after the finite verb are not considered with that clause. For example:

```
wAra AyZawana o parimANa
xeKe buJawe asubiXA hayZa ei
paWa hAwi geCe.

After seeing the size and
effect, it is hard to under-
stand that an elephant went
through this way.
```

In the above example, there is hardly any clue to find beginning of subordinate clause. To solve this type of problem, capturing only the tree structure of a particular sentence has been treated as the key factor to the goal of disambiguation. One way to capture the regularity of chunks over different sentences is to learn a generative grammar that explains the structure of the chunks one finds. These types of language properties make the clause identification problem difficult.

3.1 Karaka relation

Dependency parsing generates the inter chunk relation and generates the tree structure. The dependency parser as described in Section 2.3 used as a supportive tool for the present problem.

In the output of the dependency parsing systems, most of the chunks have a dependency relation with the verb chunk. These relations are called as *karaka* relation. Using dependency relations, the chunks having dependency relation i.e. *karaka* relation with same verb chunk are grouped. The set of chunks are the members of a clause. Using this technique, identification of chunk members of a certain clause becomes independent of SOV patterns of sentences. An example is shown in Figure 1.

[6] http://ltrc.iiit.ac.in/analyzer/bengali/
[7] http://ltrc.iiit.ac.in/nlptools2009/CR/intro-husain.pdf

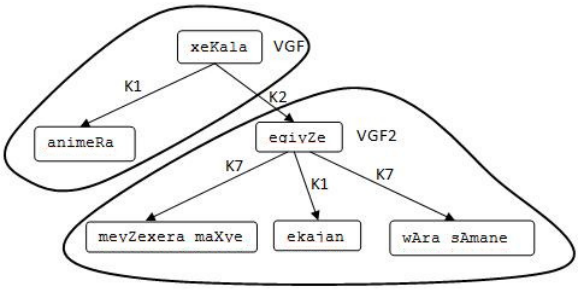

Figure 1: Karaka Relations

3.2 Compound verbs

In Bengali language a noun chunk with an infinite verb chunk or a finite verb chunk can form a compound verb. An example is shown in Figure 2.

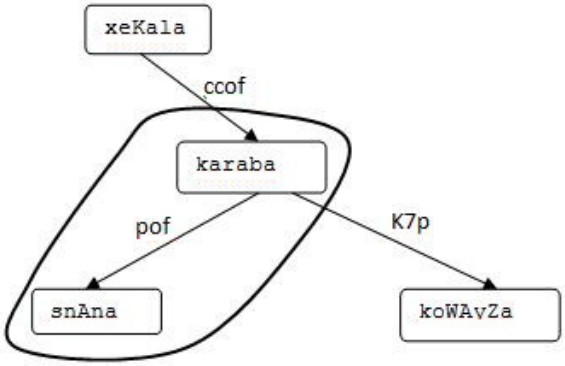

Figure 2: Compound Verb

In the above example, the noun chunk and the VGF chunk form a compound verb. These two consecutive noun and verb chunks appearing in a sentence are merged to form a compound verb. These chunks are connected with a part-of relation in Dependency Parsing. The set of related chunks with these noun and verb chunks are merged.

3.3 Shasthi Relation (r6)

In dependency parsing the genitive relation are marked with *shasthi* (r6) relation. The chunk with *shasthi* (r6) (see the tagset of NLP Tool Contest: ICON 2009) relation always has a relation with the succeeding chunk. An example is shown in Figure 3.

In the example as mentioned in Figure 3, the word "*wadera*"(their) has a genitive relation with the word in the next chunk "*manera*"(of mind). These chunks are placed in a set. It forms a set of two chunks members. The system generates two different types of set. In one forms a set of members having relation with verb chunks. Another set contains two noun chunks with genitive relation. Now the sets containing only noun chunks with genitive relation does not form a clause. Those sets are merged with the set containing verb chunk and having dependency relation with the noun chunks. An example is shown in Figure 3.

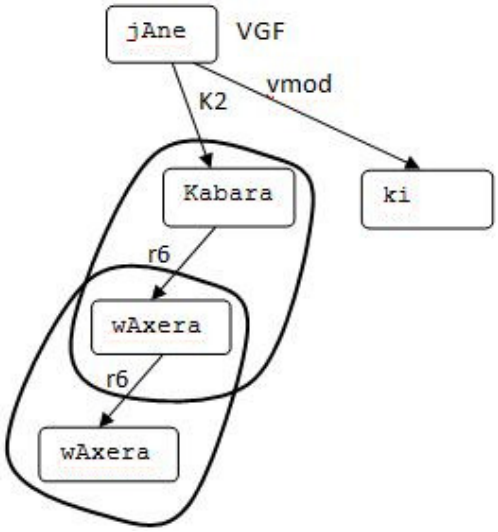

Figure 3: Shasthi Relation

Consider ω is set of all sets containing two chunk members connected with genitive marker. Consider β is a set of all sets consisting of related chunks with a verb chunk. λ is a element of ω. α is a element of β. Now, If a set λ which can have common chunks from a α set then λ set is associated with the proper α set. So, $\lambda \cap \alpha \neq$ Null then $\alpha = \alpha \cup \lambda$. If a set λ which can have common chunks from two α sets which leads to ambiguity of associability of the λ set with the proper α set. If $\lambda \cap \alpha$ = verb chunk, then λ set will be associated with α set containing the verb chunk. From the related set of chunk of verb chunks, system has identified the clauses in the sentence. Afterwards, the clauses are marked with the B-I-E (Beginning-Intermediate-End) notation.

4 Case Grammar-Identification of Karaka relations

The classical Sanskrit grammar Astadhyayi [8] ('Eight Books'), written by the Indian gramma-

[8] http://en.wikipedia.org/wiki/P%C4%81%E1%B9%87ini

rian Panini sometime during 600 or 300 B.C. (Robins, 1979), includes a sophisticated theory of thematic structure that remains influential till today. Panini's Sanskrit grammar is a system of rules for converting semantic representations of sentences into phonetic representations (Kiparsky, 1969). This derivation proceeds through two intermediate stages: the level of *karaka* relations, which are comparable to the thematic role types described above; and the level of morpho-syntax.

Fillmore's Case Grammar (Fillmore, 1968), and much subsequent work, revived the Panini's proposals in a modern setting. A main objective of Case Grammar was to identify semantic argument positions that may have different realizations in syntax. Fillmore hypothesized 'a set of universal, presumably innate, concepts which identify certain types of judgments human beings are capable of making about the events that are going on around them'. He posited the following preliminary list of cases, noting however that 'additional cases will surely be needed'.

- **Agent:** The typically animate perceived instigator of the action.
- **Instrument:** Inanimate force or object causally involved in the action or state.
- **Dative:** The animate being affected by the state or action.
- **Factitive:** The object or being resulting from the action or state.
- **Locative:** The location or time-spatial orientation of the state or action.
- **Objective:** The semantically most neutral case, the concept should be limited to things which are affected by the action or state.

The SSF specification handles this syntactic dependency by a coarse-grain tagset of Nominitive, Accusative, Genitive and Locative case markers. Bengali shallow parser identifies the chunk heads as part of the chunk level analysis. Dependency parsing followed by a rule based module has been developed to analyze the inter-chunk relationships depending upon each verb present in a sentence. Described theoretical aspect can well define the problem definition of clause boundary identification but during practical implementation of the solution we found some difficulties. Bengali has explicit case markers and thus long distant chunk relations are possible as valid grammatical formation. As an example:

```
bAjAre yAoyZAra samayZa xeKA
kare gela rAma.

bAjAre yAoyZAra samayZa rAma
xeKA kare gela.

rAma bAjAre yAoyZAra samayZa
xeKA kare gela.

Rama came to meet when he
was going to market.
```

In the above example *rAma* could be placed anywhere and still all the three syntactic formation are correct. For these feature of Bengali many dependency relation could be missed out located at far distance from the verb chunk in a sentence. Searching for uncountable numbers of chunks have dependency relation with a particular verb may have good idea theoretically but we prefer a checklist strategy to resolve the problem in practice. At this level we decided to check all semantic probable constituents by the definition of universal, presumably innate, concepts list. We found this is a nice fall back strategy to identify the clause boundary. Separately rules are written as described below.

4.1 Agent

Bengali is a verb final language. Most of the Bengali sentences follow a Subject-Object-Verb (SOV) pattern. In Bengali, subject can be missing in a clause formation. Missing subjects and missing keywords lead to ambiguities in clause boundary identification.

দরজাটা বন্ধ করো।
Close the door.

In the previous case system marks "*দরজাটা/door*" as an "*Agent*" whereas the "*Agent*" is "*you*" (2nd person singular number), silent here.

We developed rules using case marker, Gender-Number-Person (GNP), morphological feature and modality features to disambiguate these

types of phenomena. These rules help to stop false hits by identifying no 2nd person phrase was there in the example type sentences and empower to identify proper phrases by locating proper verb modality matching with the right chunk.

4.2 Instrument

Instrument identification is ambiguous for the same type of case marker (nominative) taken by agent and instrument. There is no animate/inanimate information is available at syntactic level.

শ্যামের বাঁশির সুর মন্ত্রমুগ্ধকর।

The music of Shyam's messmerized me.

সুমির ছাতা।

The umbrella of Sumi.

Bengali sentences follow a Subject-Object-Verb (SOV) pattern. Positional information is helpful to disambiguate between agent and instrument roles.

4.3 Dative

	Bengali		English Gloss
General	সকাল/সন্ধ্যে/রাত/ভোর...		Morning/evening/night/dawn…
	_টার সময়/সময়/ঘটিকায়/মিনিট/সেকেন্...		O clock/time/hour/minute/second…
	সোমবার/মঙ্গলবার/রবিবার...		Monday/Tuesday/Sunday …
	বৈশাখ/জৈষ্ঠ/...		Bengali months…
	জানুয়ারী/ফেব্রুয়ারী		January/February…
	দিন/মাস/বছর...		Day/month/year…
	কাল/ক্ষন/পল...		Long time/moment…
Relative	আগে/পরে...		Before/After…
	সামনে/পেছনে...		Upcoming/
	Special Cases	উঠলে/থামলে...	When rise/When stop…

Table 3: Categories of Time Expressions

Time expression identification has a different aspect in NLP applications. People generally studied time expression to track event or any other kind of IR task. Time expressions could be categorized in two types as General and Relative.

In order to apply rule-based process we developed a manually augmented list with pre defined categories as described in Table 3. Still there are many difficulties to identify special cases of relative time expressions. As an example:

চাঁদ উঠলে আমরা রওনা হবো।

When moon rise we will start our journey.

In the previous example the relative time expression is "*উঠলে/when rise*" is tagged as infinite verb (for Bengali tag level is VGNF). Statistics reveals that these special types of cases approximately are only 1.8-2% in overall corpus.

These types of special cases are not handled by the present system.

4.4 Factitive

The particular role assignment is the most challenging task as it separately known as argument identification. To resolve this problem we need a relatively large corpus to learn fruitful feature similarities among argument structures.

A manually generated list of causative postpositional words and pair wise conjuncts as reported in Table 4 has been prepared to identify argument phrases in sentences.

	Bengali	English Gloss
General	জন্য/কারনে/হেতু...	Hence/Reason/Reason
Relative	যদি_তবে	If_else
	যদিও_তবুও	If_else

Table 4: Categories of Causative Expressions

4.5 Locative

Rules have been written using a manually edited list as described in Table 5. Morphological locative case marker feature have been successfully used in identification of locative marker. There is an ambiguity among Agent, Dative and Locative case marker as they orthographically generates same type of surface form (using common

suffixes as: ে, ের etc). There is less differences we noticed among their syntactic dependency structure throughout the corpus. Positional information helps in many cases to disambiguate these cases.

দেশে কাজ নেই বাবু।

There is unemployment in country side.

A different type of problem we found where verb plays locative role. As an example:

লোকে যেখানে কাজ করে সেখানে।

Where people works there.

Here *"যেখানে কাজ করে/Where people works"* should be identified as locative marker. But this is a verb chunk and leads difficulty. Corpus statistics reveals that this type of syntactic formation is approximately 0.8-1.0% only and not been handled by the present system.

	Bengali	English Gloss
General	মাঠে/ঘাটে/রাস্তায়	Morning/evening/night/dawn…
Relative	আগে/পরে...	Before/After…
	সামনে/পেছনে...	Front/Behind

Table 5: Categories of Locative Expressions

4.6 Objective

The concept of objectivity is limited to things or human which are affected by the action or state. Statistical parser is a best way out for the present problem. The *karma karaka* (k2) identified by the dependency parser is simply the objective constituent of any clause.

5 Identification the Type of Clauses

After marking of the clause boundaries, clause types are identified. According to the clause structure and functions in a sentence, clauses are classified in to four types as principal clause, noun clause, adverbial clause and adjective clause. To identify the clause types, a CRF based statistical approach has been adopted.

5.1 Generative Grammar

In theoretical linguistics, generative grammar refers to a particular approach to the study of syntax. A generative grammar of a language attempts to give a set of rules that will correctly predict which combinations of words will form grammatical sentences. Chomsky has argued that many of the properties of a generative grammar arise from an "innate" universal grammar. Proponents of generative grammar have argued that most grammar is not the result of communicative function and is not simply learned from the environment. Strongly motivated by Chomsky's generative grammar we adopt the CRF based machine learning to learn the properties of a language and apply the knowledge to typecast clause classification as well.

5.2 Conditional Random Fields (CRF)

CRFs are undirected graphical models which define a conditional distribution over a label sequence given an observation sequence. CRF usually trained based on input features. Maximum likelihood is being calculated on chosen features for training.

5.2.1 Features

The vitality of using any machine learning approach is in identification of proper feature set. Conditional Random Field (CRF) works on a conditional distribution over a label sequence given an observation sequence. Hence CRF used here to statistically capture the prosodic structure of the language. The features experimentally found useful are chosen as listed below.

5.2.2 Chunk Label

An *n*-gram chunk label window has been fixed to capture internal arrangement of any particular clause type.

5.2.3 Chunk Heads

Chunk head pattern is the vital clue to identify the any clause pattern.

5.2.4 Word

In the clause type identification task words play a crucial part as word carries the information of the clause type.

From the input file in the SSF format, all the morphological information like root word, chunk heads are retrieved. The clause type identification depends on the morphological information along with the position in the sentences and also the surrounding chunk labels. Therefore the CRF based statistical tool calculates the probability of

the morphological information along with the dependency relations of the previous three and next three chunks. For the present task a quad-gram technique is used as most of the sentences have around 10 chunks.

The input file in the SSF format includes Chunk labels and word. The clause information in the input files are in B-I-E format so that the begin (B) / inside (I) / end (E) information for a clause are associated as a feature. The chunk heads, words are identified from the training file and noted as an input feature in the CRF based system. Each sentence is represented as a feature vector for the CRF machine learning task. The input features associated with each word in the training set are the word, clause boundary tags, chunk tag and clause type tags.

6 Error Analysis

During the development stage of the system we had studied the various clause boundary labeling errors committed by the system. In the above examples, the system faces ambiguity to derive the rules for the identification of the clause members when a noun chunk acts as a noun modifier of a clause. In complex sentences, the verb chunk of the subordinate clause may have noun modifier relation with the principal clause. As System forms the groups the chunks with dependency relation, system merges the subordinate clause with principal clause. An example is shown in Figure 4.

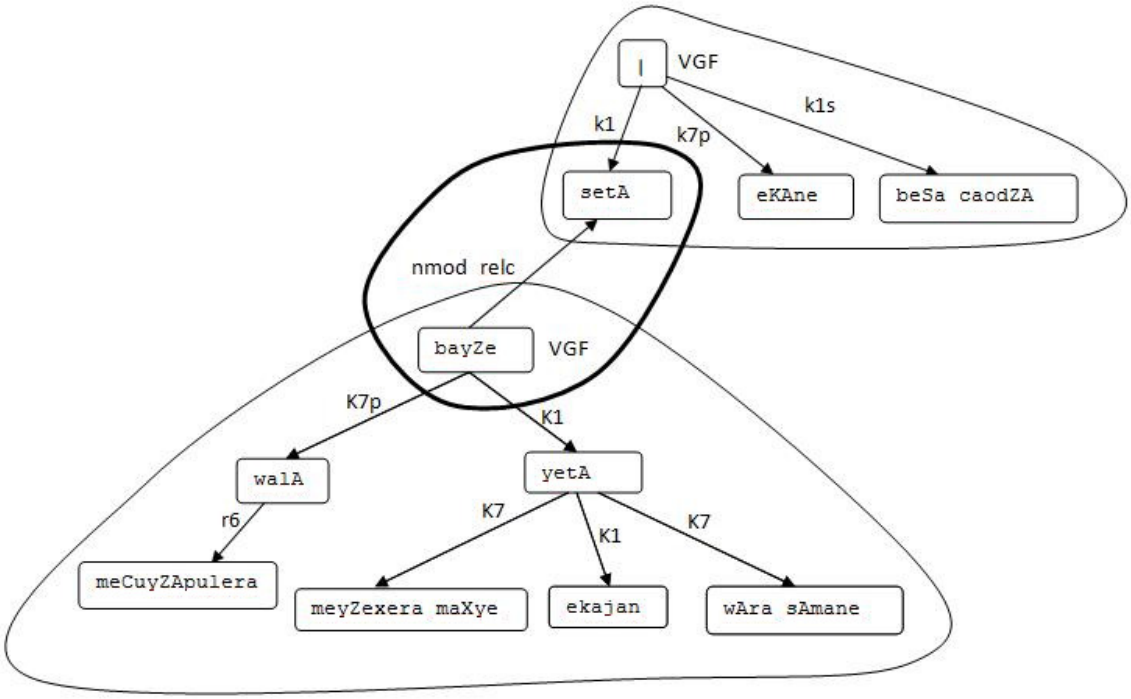

Figure 4: Shasthi Relation

7 Experimental results

System	Precision	Recall
Boundary	73.12%	75.34%
Classification	78.07%	78.92%

Table 6: Performance of present System

The accuracy of the rule-based clause boundary identification system is 73.12% and 78.07% is the accuracy clause type classification system as reported in Table 6.

8 Conclusion

This paper reports about our works on clause identification and classification in Bengali language. We have used the rule based system to identify clause boundary and a statistical CRF based model is used to decide the type of a clause.

In future we would like to study different semantic relations which can regulate clause type and boundary.

References

A. Ghosh, A. Das, P. Bhaskar, S. Bandyopadhyay. Dependency Parser for Bengali: the JU System at ICON 2009, In NLP Tool Contest ICON 2009, December 14th-17th, 2009, Hyderabad.

Akshar Bharati, Vineet Chaitanya , Rajeev Sangal. Natural Language Processing A Paninian Perspective. Prentice Hall of India (1995).

Charles J. Fillmore, Christopher R. Johnson, and Miriam R. L. Petruck. 2003. Background to FrameNet. International Journal of Lexicography, 16:235–250.

Chomsky, Noam (1956). "Three models for the description of language". IRE Transactions on Information Theory 2: 113–124.

Erik F. Tjong kim sang and Herve Dejean Introduction to CoNLL-2001 shared task: clause identification.

Groenendijk, J.: (2009), 'Inquisitive Semantics: Two Possibilities for Disjunction'. In Lecture Notes in Computer Science. ISBN- 978-3-642-00664-7. Volume- 5422/2009. Berlin, Heidelberg. Pages-80-94.

Kalika Bali, Monojit Choudhury, Diptesh Chatterjee, Arpit Maheswari, Sankalan Prasad. Correlates between Performance, Prosodic and Phrase Structures in Bangla and Hindi: Insights from a Psycholinguistic Experiment. In Proceeding of ICON 2009. Hyderabad. India.

Kiparsky, Paul and J. F. Staal (1969). 'Syntactic and semantic relations in Panini.' Foundations of Language 5, 83-117.

Robins, R. H. (1979). A Short History of Linguistics (2nd Edition). London: Longman.

Vijay Sundar Ram. R and Sobha Lalitha Devi, 2008 Clause Boundary Identification Using Conditional Random Fields.

A Paradigm-Based Finite State Morphological Analyzer for Marathi

Mugdha Bapat **Harshada Gune** **Pushpak Bhattacharyya**

Department of Computer Science and Engineering,
Indian Institute of Technology Bombay

{harshadag,mbapat,pb}@cse.iitb.ac.in

Abstract

A morphological analyzer forms the foundation for many NLP applications of Indian Languages. In this paper, we propose and evaluate the morphological analyzer for Marathi, an inflectional language. The morphological analyzer exploits the efficiency and flexibility offered by finite state machines in modeling the morphotactics while using the well devised system of paradigms to handle the stem alternations intelligently by exploiting the regularity in inflectional forms. We plug the morphological analyzer with statistical pos tagger and chunker to see its impact on their performance so as to confirm its usability as a foundation for NLP applications.

1 Motivation and Problem Definition

A highly inflectional language has the capability of generating hundreds of words from a single root. Hence, morphological analysis is vital for high level applications to understand various words in the language. Morphological analyzer forms the foundation for applications like information retrieval, POS tagging, chunking and ultimately the machine translation. Morphological analyzers for various languages have been studied and developed for years. But, this research is dominated by the morphological analyzers for agglutinative languages or for the languages like English that show low degree of inflection. Though agglutinative languages show high morpheme per word ratio and have complex morphotactic structures, the absence of fusion at morpheme boundaries makes the task of segmentation fluent once the model for implementation of morphotactics is ready. On this background, a morphological analyzer for highly inflectional language like Marathi which has the tendency to overlay the morphemes in a way that aggravates the task of segmentation presents an interesting case study.

Eryiğit and Adalı (2004) propose a suffix stripping approach for Turkish. The rule based and agglutinative nature of Turkish allows the language to be modeled using FSMs and does not need a lexicon. The morphological analyzer does not face the problem of the changes taking place at morpheme boundaries which is not the case with inflectional languages. Hence, although apprehensible this model is not sufficient for handling the morphology of Marathi.

Many morphological analyzers have been developed using the two-level morphological model (Koskenniemi, 1983) for morphological analysis. **(**Oflazer, 1993; Kim et al., 1994) have been developed using PC-Kimmo (Antworth, 1991), a morphological parser based on the two-level model. Conceptually, the model segments the word in its constituent parts, and accounts for phonological and orthographical changes within a word. While, the model proves to be very useful for developing the morphological analyzers for agglutinative languages or the languages with very less degree of inflection, it fails to explicitly capture the regularities within and between paradigms present in the inflectional languages. Marathi has a well defined paradigm-based system of inflection. Hence, we decided to develop our own model which works on the similar lines of PC-Kimmo (Antworth, 1991) but exploits the

Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), pages 26–34,
the 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010

usefulness of paradigm-based inflectional system.

Bharati et al. (2004) propose a paradigm based algorithm for morphological analysis of Hindi, an inflecting language. In Hindi, the inflected forms of roots do not allow further attachment of any other suffixes. In contrast, in Marathi once the root is transformed into its inflected form it is followed by suffixes to show its agreement with the other words in the sentence. Some postpositions derive new words which themselves may undergo inflection and allow attachment of other suffixes. This makes the simple paradigm-based model proposed in this work unfit for Marathi morphological analysis.

Dixit et al. (2006) developed a morphological analyzer with a purpose of using it for spell checking. Though their analyzer successfully analyzes the words with a single suffix, its scope is restricted to the handling of only first level suffixes.

1.1 Our Approach

In this paper, we present the morphological analyzer for Marathi which is official language of the state of Maharashtra (India). With 90 million fluent speakers worldwide, Marathi ranks as the 4th most spoken language in India and the 15th most in the world. The methodology is based on the use of paradigm-based inflectional system combined with finite state machines (FSMs) for modeling the morphotactics. To the best of our knowledge, such an approach has never been tried out for Marathi. The crux of the system lies in the detailed study of morphosyntactic phenomena, the morphotactic structure of the language and the use of paradigm-based inflectional system.

The approach can be used for other inflectional languages by developing the resources like language specific inflection rules and the FSM that models the morphotactics for the language.

1.2 Marathi Morphology

Marathi is a morphologically rich language. It is highly inflectional and also shows derivation to a high degree. Like other synthetic languages, Marathi morphological analysis faces some well-known challenges. Words contain multiple morphemes fused together in such a way that, it becomes difficult to segment them. A single morpheme contains a bunch of grammatical attributes associated with it which creates a challenge for morphological parsing. A single root is capable of generating hundreds of words by combining with the other morphemes.

The complexity involved in the formation of a polymorphemic word can be better illustrated using an example. Consider the word देवासारख्याला {devaasaarakhyaalaa} (to the one like the god). The nominal root 'देव' {deva} (god) gets inflected to the oblique case, singular form 'देवा' {devaa} which is then followed by the adjectival suffix 'सारखा' {saarakhaa} (alike). This derives the adjective 'देवासारखा' {devaasaarakhaa} (the one like the god) which then starts behaving like a noun. This noun on getting inflected to the oblique case, singular form देवासारख्या {devasaarakhyaa} is followed by the case marker ला {laa} (to). This gives the word देवासारख्याला {devaasaarakhyaalaa} (to the one like the god). Equation 1 illustrates this process.

$$(deva \text{ "}\rightarrow\text{" } devaa) + saarakhaa = devaasarakhaa$$
$$(devasaarakhaa \rightarrow devaasarakhyaa) + laa = devaasaarakhyaalaa$$

Equation 1. Formation of देवासारख्याला {devaasaarakhyaalaa} (to the one like the god)

This suggests that the process of formation of polymorphemic words is recursive in nature with inflection taking place at every level of recursion.

Section 2 discusses the design of the morphological analyzer which tries to overcome the problems discussed above with respect to Marathi language. Sections 3 and 4 discuss the linguistic resources and the processing of words belonging to various categories respectively. Sections 5 and discuss the classification of suffixes and development of automata based on this classification respectively. Section 7 briefs on the experimental setup and the results.

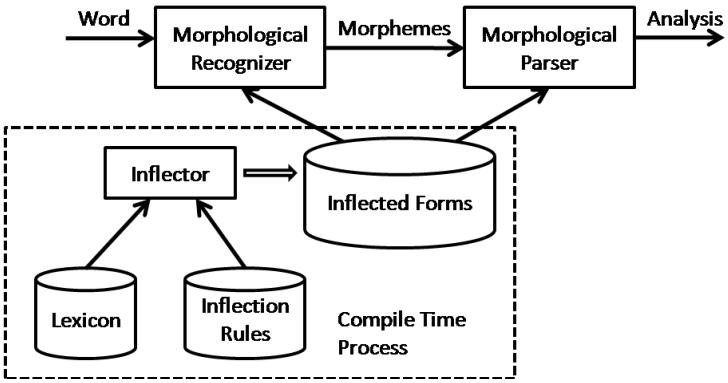

Figure 1. Architecture of Marathi Morphological Analyzer

2 Morphological Analyzer for Marathi

The formation of polymorphemic words leads to complexities which need to be handled during the analysis process. FSMs prove to be elegant and computationally efficient tools for modeling the suffix ordering in such words. However, the recursive process of word formation in Marathi involves inflection at the time of attachment of every new suffix. The FSMs need to be capable of handling them. Koskenniemi (1983) suggests the use of separate FSMs to model the orthographic changes. But, Marathi has a well devised system of paradigms to handle them. One of our observations led us to a solution that combines paradigm-based inflectional system with FSM for modeling. The observation was that, during the i^{th} recursion only $(i-1)^{th}$ morpheme changes its form which can be handled by suitably modifying the FSM. The formation of the same word devaasaarakhyaalaa described above can be viewed as illustrated in Equation 2**.**

$$(deva \rightarrow devaa) + saarakhaa = devaasarakhaa$$
$$(deva \rightarrow devaa) + (saarakhaa \rightarrow sarakhyaa) + laa = devaasaarakhyaalaa$$

Equation 2. Simulating the formation of देवासारख्याला {devaasaarakhyaalaa} (to the one like the god)

Generalizing the word formation process we arrived at the formulation specified by Equation 3.

$$Polymorphemic\ word = (inflected\_morpheme1) + inflected\_morpheme2 + ..$$

Equation 3. Formulation of Polymorphemic Word Formation

This requires a morphotactic FSM which is aware of the inflected forms of morphemes in addition to the actual morphemes to handle the above recursive process of word formation. We use the paradigm-based system to generate the inflected form of the morphemes and feed them to the FSM. Figure 1 shows the architecture of the morphological analyzer based on this philosophy.

Inflector inflects all morphemes in the lexicon using the inflection rules associated with the paradigms to which they belong.

Given a word, Morphological Recognizer recognizes the constituent morphemes in their inflected forms using finite state machine that models the morphotactics. For example, the output of the Morphological Recognizer for the word devaasaarakhyaalaa is devaa + saarakhyaa + laa. Morphological Parser outputs per morpheme analysis of the word using the morphemes recognized by the Morphological Recognizer.

3 Linguistic Resources

The linguistic resources required by the morphological analyzer include a lexicon and inflection rules for all paradigms.

3.1 Lexicon

An entry in lexicon consists of a tuple <*root, paradigm, category*>. The *category* specifies the grammatical category of the *root* and the *paradigm* helps in retrieving the inflection rules associated with it. Our lexicon contains in all 24035 roots belonging to different categories.

3.2 Inflection Rules

Inflection rules specify the inflectional suffixes to be inserted (or deleted) to (or from) different positions in the root to get its inflected form. An inflectional rule has the format: <*inflectional suffixes, morphosyntactic features, label*>. The element *morphosyntactic features* specifies the set of morphosyntactic features associated with the inflectional form obtained by applying the given inflection rule. Following is the exhaustive list of morphosyntactic features to which different morphemes get inflected:

1) Case: Direct, Oblique
2) Gender: Masculine, Feminine, Neuter, Non-specific
3) Number: Singular, Plural, Non-specific
4) Person: 1^{st}, $2^{nd,}$ 3^{rd}
5) Tense: Past, Present, Future
6) Aspect: Perfective, Completive, Frequentative, Habitual, Durative, Inceptive, Stative
7) Mood: Imperative, Probabilitive, Subjunctive, Conditional, Deontic, Abiltive, Permissive

The *label* specifies the morphotactic class to which the inflected form (generated by applying the inflection rule) belongs. It is used by the Morphological Recognizer.

4 Category Wise Morphological Formulation

The grammatical categories observed in Marathi include nouns, pronouns, verbs, adjectives, adverbs, conjunctions, interjections and postpositions. The morphemes belonging to different categories undergo different treatment.

4.1 Noun Morphology

Marathi nouns inflect for number and case. Postpositions get attached to the oblique forms of the nouns (known as stems) to show their relationship with other words in the sentence. A single stem is used for the attachment of all postpositions which makes nominal morphology absolute economic in nature. For example various forms of the word दार {daara} (door) are दारास {daaraasa} (to the door), दाराने {daaraane} (by the door), दाराशेजारी {daaraashejarii} (besides the door). Please note that the same stem दारा {daaraa} is used for the attachment of various postpositions.

Depending upon their ending, gender and the inflectional patterns, the nouns in Marathi can be classified into various paradigms. A paradigm is a complete set of related inflectional forms associated with a given root. All words that share the similar inflectional forms fall in the same paradigm. Table 1 presents the paradigm दार {daara} (door).

		Case	
		Direct	Oblique
Number	Singular	दार {daara}	दारा {daaraa}
	Plural	दारे {daare}	दारां {daaraaN}

Table 1. Paradigm Table for दार {daara} (door)

कापड {kaapaDa} (cloth), पान {paana} (leaf), पुस्तक {pustaka} (book), कपाट {kapaaTa} (cupboard) are the few nouns that fall into this paradigm.

Every paradigm has a set of inflection rules associated with it one corresponding to every inflectional form of the word. A noun has four inflectional forms each one corresponding to a case-number pair. Hence, every paradigm has four inflectional rules associated with it.

An inflectional rule for Marathi consists of a tuple specifying the inflectional suffixes that should be inserted and deleted from ultimate and penultimate position of the root. Table 2 lists the inflectional suffixes that collectively form an inflectional rule.

The procedure to obtain the inflected form of the given root R belonging to paradigm P by applying the inflectional rule I <UD, UI, PUD, PUI> is as follows:

i. R =R - PUD
ii. R = R + PUI
iii. R = R – UD
iv. R = R + UI

Suffix	**Description**
Ultimate Deletion	Suffix to be deleted from the ultimate position of the root

(UD)	
Ultimate Insertion (UI)	Suffix to be inserted at the ultimate position of the root
Penultimate Deletion (PUD)	Suffix to be deleted from the penultimate position of the root
Penultimate Insertion (PUI)	Suffix to be inserted at the ultimate position of the root

Table 2. Suffixes in an Inflectional Rule

For a given word, even if a single rule out of the four is different from the set of available paradigms, a new paradigm needs to be created. Table 3 shows the paradigm भक्त {bhakta} (devotee). Note that, the only difference between the two paradigm tables is in the direct case plural form.

		Case	
		Direct	Oblique
Number	Singular	भक्त {bhakta}	भक्ता {bhaktaa}
	Plural	भक्त {bhakta}	भक्तां {bhaktaaN}

Table 3. Paradigm Table for भक्त {bhakta} (devotee)

In this way, our lexicon contains 16448 nouns categorized into 76 paradigms. Out of the 76 paradigms, 30 correspond to feminine gender, 29 to masculine and 17 to neuter gender. This set of paradigms includes three null paradigms, one corresponding to each gender. In modern Marathi, the stem of the proper nouns or foreign words transliterated in Marathi is same as the root. In short, postpositions can be directly attached to these roots without any modification. Such nouns belong to the null paradigm.

4.2 Postposition Morphology

Postpositions follow the stems of nouns and pronouns. Postpositions in Marathi can be broadly classified into case markers and shabdayogi avyayas. Shabdayogi avyayas show the relationship of nouns and pronouns with the other words in the sentence while deriving the adjectives or adverbs in most of the cases. Depending upon the category of the word derived by them they are classified as adjectival and adverbial suffixes respectively. We have 142 postpositions listed in our lexicon.

4.3 Classification of Postpositions

The first step towards defining the morphotactics of a language is the classification of various suffixes into classes depending upon the morphemes they can follow and the morphemes that can follow them. Given the list of 142 postpositions, we carefully examined each one to understand its morphotactic behavior and came up with the classification of Marathi postpositions as presented in the Table 4.

Class	**Ordering Rules**	**Example**
Case markers	They can follow any oblique form. No other suffixes can follow them.	ने {ne} (by)
Adjectival Suffixes	They can follow an oblique form of a root. Since they derive an adjective, they can be followed by any other suffixes.	सारखा {saarakhaa} (alike
Possessive case marker	It can follow any oblique form. It can be followed by any other suffixes.	चा {chaa} (the one belonging to something)
Closing suffixes	They can follow any oblique form. No other suffixes can follow them.	पेक्षा {pekshaa} (instead of)
चा {chaa} adjectival suffix	It can follow Peculiar NSTs and Nearly closing postpositions. It can be followed by case markers.	चा {chaa} (the one)
Peculiar	They can follow any ob-	जवळ

NSTs	lique form. They can be followed only by Exclusive postpositions and चा {chaa} adjectival suffix.	{jawaLa} (near)
Exclusive postpositions	They can follow peculiar NSTs. They close the word.	ई {ii} (inside)
Nearly closing postpositions	They can follow oblique forms of nouns and pronouns. They can be followed by चा {chaa} adjectival suffix.	पर्यंत {paryanta} (uptil)
Shuddha-shabdayogi avyayas	They can follow almost any morpheme except oblique forms of nouns. They can be followed by some postpositions. But, this behavior is quite irregular and needs more investigation. In most of the cases, these suffixes close the word. Hence, we consider them to be occurring only at the end of the word.	च {cha} (only)

Table 4. Classification of Postpositions

4.4 Verbs

The verbs inflect for gender, number and person of the subject and the direct object in a sentence. They also inflect for tense and aspect of the action as well as mood of the speaker in an illocutionary act. They may even undergo derivation. Further discussion on verbal morphology will be based on Aakhyaata theory (inflection) and Krudanta theory (derivation) (Damale, 1970). Our lexicon contains 1160 verb roots classified into 22 paradigms.

Aakhyaata Theory forms the basis of verbal inflection in Marathi. *Aakhyaata* (*आख्यात*) refers to tense, aspect, and mood. Aakhyaata is realized through an aakhyaata suffix which is a closing suffix, attached to the verb root. There are 8 types of aakhyaatas named after the phonemic shape of the aakhyaata suffix. Associated with every aakhyaata are various aakhyaata-arthas which indicate the features: tense, aspect and mood. An aakhyaata may or may not agree with gender. There are around 80 Aakhyaata suffixes in Marathi.

Krudanta Theory forms the basis of verbal derivation in Marathi. Krudanta refers to the word ending in a krut-pratyaya (a suffix which refers to an action). Krut-pratyayas are attached at the end of verbs to form non-infinitive verb forms. These forms usually belong to one of the categories: noun, adverb or adjective. They contribute to the aspect of the verb from which they are derived. We cover only the krudanta forms which are regular in behavior.

Irregular Verbs: Some verbs in Marathi have different behavior as compared to the other verbs (regular verbs). These verbs are present in some inflected forms for which no definite stem exists.

4.5 Adjectives

Marathi adjectives can be classified into two categories: ones that do not inflect and others that inflect for gender, number and case where such an inflection agrees with the gender and number of the noun modified by them. The inflectional forms of the adjectives are generated using similar procedure as that of nouns.

4.6 Pronouns

There are nine types of pronouns in Marathi. Pronouns possess very irregular behavior resulting into a large number of suppletive forms. In addition to these forms every pronoun has a specific oblique form (one each for singular and plural) to which shabdayogi avyayas can be attached.

4.7 Indeclinable Words

Adverbs, conjunctions and interjections are the indeclinable words. Some adverbs can be followed by a subset of postpositions.

5 Morphotactics and Automata

Along with the postpositions mentioned in the Table 4 the complete set of morphemes in Marathi includes the roots and their inflectional forms. Every morpheme is labeled according to the class it belongs to. These labels are used to define the 'Morphotactic FSM' that models Ma-

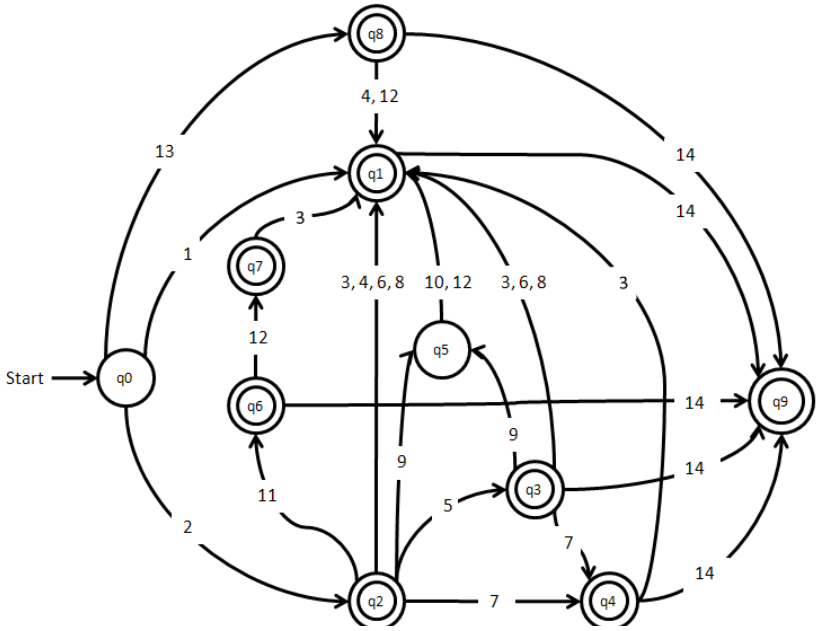

Figure 2. Morphotactic FSM

rathi language. Table 5 enlists various labels used in the Morphotactic FSM.

Type of Suffix		**Label**
Nouns, pronouns, nominal or adjectival krudantas	DF	1
	OF	2
Case markers		3
Adjectival postpositions	DF	4
	OF	5
Possessive case marker	DF	6
	OF	7
Closing postpositions		8
Peculiar NSTs		9
Exclusive postpositions		10
Nearly closing postpositions		11
चा {chaa} adjectival suffix		12
Adjective		1
Aakhyaatas		1
Adverbial krudantas		1
Adverbs-1		1
Adverbs-2		13
Shuddhashabdayogi avyayas		14

Table 5. Morphotactic Labels of Morphemes
DF: Direct form of a root or a suffix
OF: Oblique form of a root or a suffix
Adverb-1: The adverbs those cannot be followed by any postpositions
Adverb-2: The adverbs those can be followed by some postpositions

Note that, the *label* field mentioned in the inflection rules refers to the corresponding labels of the morphemes mentioned in Table 5.

Figure 2 shows the FSM for morphological recognition of Marathi. The input symbols are the labels of the morphemes as mentioned in the Table 5. The classification of the suffixes as specified in Table 5 explains the construction of FSM. We use SFST [1] (Stuttgart Finite State Transducer) for implementing the FSM.

6 Experiments

Morphological analysis caters to the needs of variety of application like machine translation, information retrieval, spell-checking. Different applications are interested in different bit of information provided by the analyzer like the stem, the root, the suffixes or the morphosyntactic features. Hence, the performance evaluation of a morphological analyzer has to be observed in terms of its impact on the performance of the applications that use it. Hence, we carry out the evaluation in two parts: In **direct evaluation** we directly measure the accuracy of morphological analyzer on the given data. In **indirect evaluation**, we observe the improvement in the performances of statistical pos tagger and chunker

[1] http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/SFST.html

by using the morphological analyzer to generate the morphological features that help in boosting their accuracies. We used the corpora in TOURISM and NEWS domain for all our experiments.

6.1 Direct Evaluation

We used Marathi Morphological Analyzer for the analysis of 21096 unique words. We manually measured the accuracy of the morphological analyzer by counting the number of correctly analyzed words out of the total number of words. In the cases where a word has multiple analyses, the word was counted as correctly analyzed only when all of the correct analyses are present. Note that, in order to emphasize more on the usefulness of our *approach* towards morphological analysis of Marathi, we added most of the roots used in the corpus to the lexicon before starting the experiments. For a language like Marathi, it is required to build a very rich lexicon which can be done over a larger period of time.

Out of the 21096 unique words, 20503 (97.18%) were found to be correctly analyzed. Of the remaining 593 words, 394 words could not be recognized by Morphological Recognizer and 199 words were assigned the incorrect or insufficient analyses.

By taking a closer look at the 394 words which were not recognized (segmented) we could come up with the causes of recognition failure as listed in Table 6.

Cause	**Number of Words**
Lexicon Coverage	82 (20.81%)
Absence of Rules	69 (17.51%)
Acronyms	66 (16.75%)
Compound words	55 (13.96%)
Irregular forms needing further investigation	47 (11.92%)
Transliterated words which are uncommon	25 (6.34%)
Unidentified words	20 (5.08%)
Dialect words/ words used in spoken language	20 (5.08%)
Use of common nouns as proper nouns	5 (1.27%)
Missing Paradigm	3 (0.76%)
Fusion (Sandhii)	2 (0.51%)

Table 6. Causes of Recognition Failure

6.2 Indirect Evaluation

CRF based sequence labelers (pos tagger and chunker) were trained using morphological features and the other elementary features like (contextual words and bigram tags). The morphological features include ambiguity scheme (set of all possible categories of a word) and the suffixes for the pos tagger whereas just the suffixes in case of chunker.

To throw the light of role played by morphological analyzer in improving the accuracies of the sequence labelers, we performed the experiments using two sets of features: The Learning Based (LB) labeler was trained using only elementary features whereas Morphologically Driven Learning Based (MDLB) labeler used the morphological features along with the elementary features. The results were obtained by performing 4-fold cross validation over the corpora. The average accuracy of MDLB Pos tagger turned out to be 95.03 as compared to 85% of LB. The average accuracy of MDLB chunker was found to be 97.87% whereas that of LB was found to be 96.91%. .

7 Conclusion and Future Work

We presented a high accuracy morphological analyzer for Marathi that exploits the regularity in the inflectional paradigms while employing the Finite State Systems for modeling the language in an elegant way. The accuracy figures as high as 97.18% in direct evaluation and the performance improvement in shallow parsing speak about the performance of the morphological analyzer. We gave detailed description of the morphological phenomena present in Marathi. The classification of postpositions and the development of morphotactic FSA is one of the important contributions since Marathi has complex morphotactics. As a next step the morphological analyzer can be further extended to handle the derivation morphology and compound words.

References

Antworth, E. L. 1990. *PC-KIMMO: A Two-level Processor for Morphological Analysis.* Occasional Publications in Academic Computing. Summer Institute of Linguistics, Dallas, Texas.

Bharati, Akshar, Vineet Chaitanya, and Rajeev Sanghal 1995. *Natural Language Processing: A Paninian Perspective*. Prentice Hall, India.

Damale, M. K. 1970. *Shastriya Marathii Vyaakarana*. Deshmukh and Company, Pune, India.

Dixit, Veena, Satish Dethe, and Rushikesh K. Joshi. 2006. *Design and Implementation of a Morphology-based Spellchecker for Marathi, an Indian Language*. In Special issue on Human Language Technologies as a challenge for Computer Science and Linguistics. Part I. 15, pages 309–316. Archives of Control Sciences.

Eryiğit, Gülşen and Adalı Eşref. 2004. *An Affix Stripping Morphological Analyzer for Turkish*. In IASTED International Multi-Conference on Artificial Intelligence and Applications. Innsbruck, Austria, pages 299–304.

Kim, Deok-Bong., Sung-Jin Lee, Key-Sun Choi, and Gil-Chang Kim (1994). *A two-level Morphological Analysis of Korean*. In Conference on Computational Linguistics (COLING), pages 535–539.

Koskenniemi, Kimmo 1983. *Two-level Morphology: a general computational model for word-form recognition and production*. University of Helsinki, Helsinki.

Oflazer, Kemal 1993. *Two-level Description of Turkish Morphology.* In The European Chapter of the ACL (EACL).

Web Based Manipuri Corpus for Multiword NER and Reduplicated MWEs Identification using SVM

Thoudam Doren Singh
Department of Computer Science and Engineering
Jadavpur University
thoudam.doren@gmail.com

Sivaji Bandyopadhyay
Department of Computer Science and Engineering
Jadavpur University
sivaji_cse_ju@yahoo.com

Abstract

A web based Manipuri corpus is developed for identification of reduplicated multiword expression (MWE) and multiword named entity recognition (NER). Manipuri is one of the rarely investigated language and its resources for natural language processing are not available in the required measure. The web content of Manipuri is also very poor. News corpus from a popular Manipuri news website is collected. Approximately four and a half million Manipuri wordforms have been collected from the web. The mode of corpus collection and the identification of reduplicated MWEs and multiword NE based on support vector machine (SVM) learning technique are reported. The SVM based reduplicated MWE system is evaluated with recall, precision and F-Score values of 94.62%, 93.53% and 94.07% respectively outperforming the rule based approach. The recall, precision and F-Score for multiword NE are evaluated as 94.82%, 93.12% and 93.96% respectively.

1 Introduction

The NER and MWE identification are important tasks for natural language applications that include machine translation and information retrieval. The present work reports the NER and reduplicated MWE identification of Manipuri on web based news corpus. The use of web as a corpus for teaching and research on languages has been proposed several times (Rundell, 2000; Fletcher, 2001; Robb, 2003; Fletcher 2004). A special issue of the Computational Linguistics journal on web as corpus (Kilgarriff and Grefenstette, 2003) was published. Several studies have used different methods to mine web data. The web walked into the ACL meetings starting in 1999. The special interest group of ACL on web as corpus is promoting interest in the use of the web as a source of linguistic data, and as an object of study in its own right. India is a multilingual country with a lot of cultural diversity. Bharati et al. (2001) reports an effort to create lexical resources such as transfer lexicon and grammar from English to several Indian languages and dependency Treebank of annotated corpora for several Indian languages. In Indian context, a web based Bengali corpus development work from web is reported in Ekbal and Bandyopadhyay (2008) and Manipuri-English semi automatic parallel corpora extraction by Singh et. al., (2010). Newspaper is a huge source of readily available documents. In the present work, the Manipuri monolingual corpus has been developed from web for NLP and related tasks.

Manipuri is a scheduled Indian language spoken approximately by three million people mainly in the state of Manipur in India and in the neighboring countries namely Bangladesh and Myanmar. It is a Tibeto-Burman language and highly agglutinative in nature, influenced and enriched by the Indo-Aryan languages of Sanskrit origin and English. The affixes play the most important role in the structure of the language. In Manipuri, words are formed in three processes called affixation, derivation and compounding. The majority of the roots found in the

Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), pages 35–42,
the 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010

language are bound and the affixes are the determining factor of the class of the words in the language. Annotated corpus, bilingual dictionaries, name dictionaries, WordNet, morphological analyzers etc. are not yet available in Manipuri in the required measure.

In the present work, the tasks of identification of Manipuri multiword named entity (MNE) and reduplicated multiword expression (RMWE) identification using support vector machine (SVM) learning technique on the corpus collected from web is reported.

Works on multiword expressions (MWEs) have started with a momentum in different languages. In the Indian context, some of the works can be seen in (Dandapat et. al., 2006; Kunchukuttan and Damani, 2008; Kishorjit et. al., 2010). The identification of MWEs in several languages concentrate on compound nouns, noun-verb combination, some on idioms and phrases and so on but not much on RMWEs. The reason may be that the reduplicated words are either rare or easy to identify for these languages since only complete duplication and some amount of partial reduplication may be present in these languages. On the other hand, reduplicated MWEs are quite large in number in Manipuri and there are wide varieties of reduplicated MWEs in Manipuri.

2 Manipuri News Corpus and Statistics

The content of Manipuri language on the web is very poor. One of the sources is the daily news publications. Again, there is no repository. Thus, the possibility of deploying web crawler and mining the web corpus is not possible. The Manipuri news corpus is collected from http://www.thesangaiexpress.com/ covering the period from May 2008 to May 2010 on daily basis. The Manipuri news is available in PDF format. A tool has been developed to convert contents from PDF documents to Unicode format. There are 15-20 articles in each day. Considering the Manipuri corpus covering the period from May 2008 to May 2010, there are 4649016 wordforms collected[1].

2.1 Conversion from PDF to UTF-8

The general Manipuri news collected is in PDF format. A tool has been developed to convert Manipuri news PDF articles to Bengali Unicode. The Bengali Unicode characters are used to represent Manipuri as well. The conversion of PDF format into Unicode involves the conversion to ASCII and then into Unicode using mapping tables between the ASCII characters and corresponding Bengali Unicode. The mapping tables have been prepared at different levels with separate tables for single characters and conjuncts with two or more than two characters. The single character mapping table contains 72 entries and the conjunct characters mapping table consists of 738 entries. There are conjuncts of 2, 3 and 4 characters. Sub-tables for each of the conjuncts are prepared. English words are present on the Manipuri side of the news and they are filtered to avoid unknown character features.

2.2 Use of language resources

The Manipuri web corpus collected from the web is cleaned by removing the unknown characters. After the cleaning process, the running texts are picked up followed by spelling correction. The web based news corpus is POS tagged using the 26 tagset[2] defined for the Indian languages based on the work of (Singh et. al., 2008). The Manipuri news corpus developed in this work has been used to identify MNE and RMWEs identification.

3 Support Vector Machine

The SVM (Vapnik, 1995) is based on discriminative approach and makes use of both positive and negative examples to learn the distinction between the two classes. The SVMs are known to robustly handle large feature sets and to develop models that maximize their generalizability. Suppose we have a set of training data for a two-class problem: $\{(x_1, y_1), ...(x_N, y_N)\}$, where $x_i \varepsilon$ RD is a feature vector of the i^{th} sample in the training data and $y_i \varepsilon \{+1, -1\}$ is the class to which x_i belongs. The goal is to find a decision function that accurately predicts class y

[1] There are no publications on some occasions.

[2] http://shiva.iiit.ac.in/SPSAL2007/iiit_tagset_guidelines.pdf

for an input vector x. A non-linear SVM classifier gives a decision function $f(x)$= sign $(g(x))$ for an input vector where, $g(x) = \sum_{i=1}^{m} w_i K(x, z_i) + b$ Here, $f(x)$=+1 means x is a member of a certain class and $f(x)$=-1 means x is not a member. The support vector is represented by z_i and stands for the training examples; m is the number of support vectors Therefore, the computational complexity of $g(x)$ is proportional to m. Support vectors and other constants are determined by solving a certain quadratic programming problem. $K(x, z_i)$ is a kernel that implicitly maps vectors into a higher dimensional space. Typical kernels use dot products: $K(x, z_i) = k(x.z)$ .A polynomial kernel of degree d is given by $K(x, z_i)$ = (1+x)d. We can use various kernels, and the design of an appropriate kernel for a particular application is an important research issue.

The MNE/RMWE tagging system includes two main phases: training and classification. The training process has been carried out by YamCha[3] toolkit, an SVM based tool for detecting classes in documents and formulating the MNE/RMWE tagging task as a sequence labeling problem. Here, both one vs rest and pairwise multi-class decision methods have been used. Different experiments with the various degrees of the polynomial kernel function have been carried out. In one vs rest strategy, K binary SVM classifiers may be created where each classifier is trained to distinguish one class from the remaining K-1 classes. In pairwise classification, we constructed K (K-1)/2 classifiers considering all pairs of classes, and the final decision is given by their weighted voting. For classification, the TinySVM-0.07[4] classifier has been used that seems to be the best optimized among publicly available SVM toolkits.

4 Multiword Named Entity Recognition

Named Entity Recognition for Manipuri is reported in (Singh et. al., 2009). The present work focuses and reports on the recognition of multiword NEs. In order to identify the MNEs, 28,629 wordforms from Manipuri news corpus has been manually annotated and used as training data with the major named entity (NE) tags, namely person name, location name, organization name and miscellaneous name to apply Support Vector Machine (SVM) based machine learning technique. Miscellaneous name includes the festival name, name of objects, name of building, date, time, measurement expression and percentage expression etc. The SVM based system makes use of the different contextual information of the words along with the variety of word-level orthographic features that are helpful in predicting the MNE classes.

MNE identification in Indian languages as well as in Manipuri is difficult and challenging as:

- Unlike English and most of the European languages, Manipuri lacks capitalization information, which plays a very important role in identifying MNEs.
- A lot of MNEs in Manipuri can appear in the dictionary with some other specific meanings.
- Manipuri is a highly inflectional language providing one of the richest and most challenging sets of linguistic and statistical features resulting in long and complex wordforms.
- Manipuri is a relatively free word order language. Thus MNEs can appear in subject and object positions making the NER task more difficult compared to others.
- Manipuri is a resource-constrained language. Annotated corpus, name dictionaries, sophisticated morphological analyzers, POS taggers etc. are not yet available.

MNE Tag	Meaning	MNE Examples
B-LOC	Beginning, Internal or the End of a multiword location name	থাঙ্গা (Thanga)
I-LOC		মোইরাংথেম (Moirangthem)
E-LOC		লৈকায় (Leikai)
B-PER	Beginning, Internal or the End of a multiword person name	ওইনাম (Oinam)
I-PER		ইবোবি (Ibobi)
E-PER		মীতৈ (Meetei)

Table 1. Named entity examples

[3]http://chasen-org/~taku/software/yamcha/
[4]http://cl.aist-nara.ac.jp/~taku-ku/software/TinySVM

In the present work, the NE tagset used have been further subdivided into the detailed categories in order to denote the boundaries of MNEs properly. Table 1 shows examples.

5 Reduplicated MWEs Identification

Manipuri is very rich in RMWEs like other Tibeto-Burman languages. The work of (Singh, 2000) describes the linguistic rules for identifying reduplicated words. A rule based Manipuri RMWE identification is reported in (Kishorjit and Bandyopadhyay, 2010). The process of reduplication (Singh, 2000) is defined as: 'reduplication is that repetition, the result of which constitutes a unit word'. These single unit words are the MWEs. The RMWEs in Manipuri are classified as: 1) Complete RMWEs, 2) Partial RMWEs, 3) Echo RMWEs and 4) Mimic RMWEs. Apart from these four types of RMWEs, there are also cases of a) Double RMWEs and b) Semantic RMWEs.

Complete RMWEs: In the complete RMWEs the single word or clause is repeated once forming a single unit regardless of phonological or morphological variations.

মরিক মরিক ('*marik marik*') which means 'drop by drop'.

অটেক অটেকপা ('*atek atek-pa*') which means 'fresh'

করি করি ('*kari kari*') means 'what/which'.

Partial RMWEs: In case of partial reduplication the second word carries some part of the first word as an affix to the second word, either as a suffix or a prefix.

For example, চথোক চৎসিন ('chat-thok chat-sin') means 'to go to and fro'; শামী লানমী ('*saa-mi laan-mi*') means 'army'.

Mimic RMWEs: In the mimic reduplication the words are complete reduplication but the morphemes are onomatopoetic, usually emotional or natural sounds. For example, করক করক ('*krak krak*') means 'cracking sound of earth in drought'.

Echo RMWEs: The second word does not have a dictionary meaning and is basically an echo word of the first word. For example, থকসি খাসি ('*thak-si kha-si*') means 'good manner'.

Double RMWEs: Such type of reduplication generally consists of three words where the prefix or suffix of the first two words is reduplicated but in the third word the prefix or suffix is absent. An example of double prefix reduplication is ইমুন ইমুন মুনবা ('*i-mun i-mun mun-ba*') which means, 'completely ripe'.

Semantic RMWEs: Both the reduplication words have the same meaning and so also is the MWE. Such types of MWEs are very special to the Manipuri language. For example, পামবা কৈ *('paamba kei')* means 'tiger' and each of the component words means 'tiger'.

5.1 Role of suffix and prefix

Apart from the above cases meaningful prefixes or suffixes are used with RMWEs otherwise they are ungrammatical.

Suffixes/ wh- duplicating words	Part of Speech
দা (*–da*), গি (*–gi*) and কি (*–ki*)	Noun
বা (*–ba*) and পা (*–pa*)	Adjective
না (*–na*)	Adverb
করি করি ('*kari kari*'), কনা কনা ('*kanaa kanaa*'), কদায় কদায় ('*kadaay kadaay*') and করম করম ('*karam karam*')	Wh- question type

Table 2. Suffixes/wh- duplicating words list used in Complete Reduplication and POS tagging

Prefix: With such prefixes the semantic shapes are different and sometimes even the same prefix carries a different meaning. By these prefixation, the root is reduplicated as given below:

{[ই(*i*)-/পঙ(*pang*)-/খঙ(*khang*)-/ত(*ta*)-/পুম(*pum*)-/শুক(*suk*)] + Root }

→

{[ই(*i*)-/পঙ(*pang*)-/খঙ(*khang*)-/ত(*ta*)-/পুম(*pum*)-/শুক(*suk*)] + Root + Root}

মহাক্না ইৱাঙ ৱাঙই
mahaak-na i-waang waang-ngi
he/she-nom –tall tall-asp
He/She is the tallest

Suffix: There are some suffixes that carry certain meaning when used with RMWEs. Commonly used suffixes are, ত্রিক (-trik) / দ্রিক (-drik), থ্রোক (-throk), দ্রোং (-drong), শুক (-suk), শঙ (-sang), দ্রিং (-dring), শিত্ (-sit), শিন (-sin), দ্রেং (-

dreng), শোক (-sroke) etc. Generally these suffixes indicate a superlative degree or emphatic meaning.

Some examples are as follows,

মুনত্রিক মুনবা

mun-trik mun-ba

ripe ripe

'very ripe'

Role of affix in Partial Reduplication: Character-wise comparisons are done with not less than two characters either from front or rear for both the words since the second word is not a complete repetition.

Also the last few characters of the first word and the same number of first characters of the second word are compared to check the partial reduplication. The prefixes or suffixes are verified with a list of accepted suffixes and prefixes (see table 2) to validated the reduplication.

Role of affix in Echo Reduplication: Identification of echo reduplication is done by comparing the equality of suffixes of consecutive two words w1 and w2.

6 Best Feature Selection for SVM

The use of prefix/suffix information works well for the highly inflected languages like the Indian languages. Different combinations from the following set for identifying the best feature set for MNE/RMWE are experimented: F={ $w_{i-m,..,}w_{i-1}, w_i, w_{i+1,....,}w_{i+n}$, |prefix|<=n, |suffix|<=n, MNE/RMWE tag(s) of previous word(s), POS tag(s) of the current and/or the surrounding word(s), First word, Length of the word, Digit information, Infrequent word}, where w_i is the current word; w_{i-m} is the previous m^{th} word and w_{i+n} is the next n^{th} word. Following are the details of the features:

1 Context word feature: Preceding and following words of a particular word since the surrounding words carry effective information for the identification of MNE/RMWEs.

2 Word suffix: Word suffix information is helpful to identify MNE/RMWEs. This is based on the observation that the MNE/RMWEs share some common suffixes. The fixed length (say, n) word suffix of the current and/or the surrounding word(s) can be treated as the feature. If the length of the corresponding word is less than or equal to n − 1 then the feature values are not defined and are denoted by 'ND'. The feature value is also not defined (ND) if the token itself is a punctuation symbol or contains any special symbol or digit. Word suffixes are the effective features and work well for the highly inflective Indian languages like Manipuri.

3 Word prefix: Word prefixes are also helpful to identify MNE/RMWEs. It is based on the observation that MNE/RMWEs share some common prefix strings. This feature has been defined in a similar way as that of the fixed length suffixes.

4 MNE and RMWE Information: The MNE/RMWE tag(s) of the previous word(s) have been used as the only dynamic feature in the experiment. The output tag of the previous word is very informative in deciding the MNE/RMWE tag of the current word.

5 Digit features: Several binary valued digit features have been defined depending upon the

(i). Presence and/or the exact number of digits in a token.

(a). CntDgtCma: Token consists of digits and comma

(b). CntDgtPrd: Token consists of digits and periods

(ii). Combination of digits and symbols. For example,

(a). CntDgtSlsh: Token consists of digit and slash

(b). CntDgtHph: Token consists of digits and hyphen

(c). CntDgtPrctg: Token consists of digits and percentages

(iii). Combination of digit and special symbols. For example,

(a). CntDgtSpl: Token consists of digit and special symbol such as $, # etc.

These binary valued digit features are helpful in recognizing miscellaneous NEs such as measurement expression and percentage expression.

6 Infrequent word: The frequencies of the words in the training corpus have been calculated. A cut off frequency has been chosen in order to consider the words that occur with less than the cut off frequency in the training corpus. A binary valued feature 'Infrequent' is defined to check whether the current word appears in this infrequent word list or not. This is based on the observation that the infrequent words are most probably MNE/RMWEs.

7 Length of a word: This binary valued feature is used to check whether the length of the current word is less than three or not. We have observed that very short words are most probably not the MNE/RMWEs.

8 Part of Speech (POS) information: We have used an SVM-based POS tagger (Singh et. al., 2008) that was originally developed with 26 POS tags, defined for the Indian languages. The POS information of the current and/or the surrounding words can be effective for MNE/RMWE identification.

The Table 3 gives the statistics of training, development and test sets. The various notations used in the experiments are presented in Table 4. The Table 5 shows the recall (R), precision (P) and F-Score (FS) in percentage in the development set.

	Training	**Development**	**Test**
# of sentences	1235	732	189
#of wordforms	28,629	15,000	4,763
# of distinct wordforms	8671	4,212	2,207

Table 3. Statistics of the training, development and test sets

Notation	**Meaning**
W[-i,+j]	Words spanning from the i^{th} left position to the j^{th} right position
POS[-i, +j]	POS tags of the words spanning from the i^{th} left to the j^{th} right positions
Pre	Prefix of the word
Suf	Suffix of the word
NE [-i, -j]	NE tags of the words spanning from the i^{th} left to the j^{th} left positions

Table 4. Meaning of the notations

Feature	**R %**	**P %**	**FS %**
Static: W[-2,+2], POS[-2,+2], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-2,-1]	**94.26**	**96.72**	**95.47**
Static: W[-3,+3], POS[-3,+3], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-3,-1]	88.23	94.82	91.40
Static: W[-3,+2], POS[-3,+2], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-3,-1]	90.32	93.18	91.72
Static: W[-4,+3], POS[-4,+3], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-2,-1]	88.15	92.62	90.32
Static: W[-4,+3], POS[-4,+3], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-3,-1]	86.24	92.31	89.17
Static: W[-2,+2], POS[-	88.	91.	90.

2,+2], |Pre|<=4, |Suf|<=4, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-2,-1]	70	49	07
Static: W[-3,+3], POS[-3,+3], |Pre|<=4, |Suf|<=4, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-3,-1]	85.05	90.09	87.49
Static: W[-4,+3], POS[-4,+2], |Pre|<=4, |Suf|<=4, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-2,-1]	78.55	89.54	83.68
Static: W[-4,+4], POS[-4,+4], |Pre|<=4, |Suf|<=4, Length, Infrequent, FirstWord, Digit **Dynamic:** MNE/RMWE[-3,-1]	73.71	89.44	80.81

Table 5. Results on the development set

7 Results on the Test Set

The best feature set (F) of Manipuri MNER and RMWE is identified as F=[prefixes and suffixes of length upto three characters of the current word, dynamic NE tags of the previous two words, POS tags of the previous two and next two words, digit information, length of the word]. After the selection of the best feature set, the SVM based system for MNE and RMWEs is tested on the test set of 4,763 wordforms.

Reduplicated MWE type	**Recall %**	**Precision %**	**F-Score %**
Complete and mimic	96.21	95.12	95.66
Partial	88.32	85.03	86.64
Echo	97.76	96.45	97.10
Double	93.23	94.23	93.72
Semantic	74.45	81.56	77.84

Table 6. Result on RMWE test set

In this work, SVM that parses from left to right is considered. The break-up of the RMWEs and the scores are given in Table 6. The handling of semantic RMWEs requires further investigation to improve the performance. The rule based RMWE identification (Kishorjit and Bandyopadhyay, 2010) shows a recall, precision and F-Score of 94.24%, 82.27% and 87.68% respectively.

Multiword NE	**Recall %**	**Precision %**	**F-Score %**
Person	94.21	95.12	94.66
Location	94.32	95.03	94.67
Organization	95.76	93.45	94.59
Miscellaneous	92.23	91.23	91.72

Table 7. Result on MNE test set

It is observed that the SVM based system outperforms the rule based system. Table 7 shows the break-up scores of different types of MNEs and Table 8 shows the overall scores of MNE and RMWE.

	Recall %	**Precision %**	**F-Score %**
MNE	94.82	93.12	93.96
RMWE	94.62	93.53	94.07

Table 8. Overall recall, precision and F-Scores on test set

8 Conclusion

In this paper, the development of RMWEs identification and recognition of MNE for a resource-constrained language using web based corpus of Manipuri is reported. This training data of 28,629 is then manually annotated with a coarse-grained tagset of four NE tags and six RMWEs in order to apply SVM and tested with 4,763 wordforms. The SVM classifier makes use of the different contextual information of the words along with the various orthographic word-level features. A number of experiments have been carried out to find out the best set of features for MWE in Manipuri. The SVM based RMWE system outperforms the rule based approach. The SVM based RMWE shows recall, precision and F-Score of 94.62%, 93.53% and 94.07% respectively. The rule based RMWE

identification shows a recall, precision and F-Score of 94.24%, 82.27% and 87.68% respectively. The overall performance of the system shows reasonable output for both MNE and RMWE.

References

Bharati, A., Sharma, D. M., Chaitanya, V., Kulkarni, A. P., & Sangal, R., 2001. LERIL: Collaborative effort for creating lexical resources. *In Proceedings of the 6th NLP Pacific Rim Symposium Post-Conference Workshop*, Japan.

Dandapat, S., Mitra, P., and Sarkar, S., 2006. Statistical investigation of Bengali noun-verb (N-V) collocations as multi-word-expressions, *In Proceedings of Modeling and Shallow Parsing of Indian Languages (MSPIL)*, Mumbai, pp 230-233

Ekbal, A., and Bandyopadhyay, S., 2008. A web based Bengali news corpus for Named Entity Recognition, *Lang Resources & Evaluation* (2008) 42:173–182, Springer

Fletcher, W. H., 2001. Concordancing the web with KWiCFinder. *In Proceedings of the Third North American Symposium on Corpus Linguistics and Language Teaching*, Boston, MA, 23–25 March 2001.

Fletcher, W. H., 2004. Making the web more use-ful as source for linguists corpora. In U. Conor & T. A. Upton (Eds.), *Applied corpus linguists: A multidimensional perspective* (pp. 191–205). Amsterdam: Rodopi.

Kilgarriff, A., and Grefenstette, G., 2003. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3), 333–347.

Kishorjit, N., and Bandyopadhyay, S., 2010. Identification of Reduplicated MWEs in Manipuri: A Rule Based Approch, *In proceedings of 23rd International Conference on the Computer Processing of Oriental Languages (ICCPOL 2010) - New Generation in Asian Information Processing* , Redmond City, CA

Kunchukuttan, A., and Damani, O. P., 2008. A System for Compound Nouns Multiword Expression Extraction for Hindi, *In Proceedings of 6th International conference on Natural Language Processing (ICON 2008)*, Pune, India

Robb, T., 2003. Google as a corpus tool? *ETJ Journal*, 4(1), Spring.

Rundell, M., 2000. The biggest corpus of all. Humanising Language Teaching, 2(3)

Singh. Chungkham Y., 2000. Manipuri Grammar, *Rajesh Publications*, Delhi, pp 190-204

Singh, Thoudam D., Ekbal, A., Bandyopadhyay, S. 2008. Manipuri POS tagging using CRF and SVM: A language independent approach, *In proceeding of 6th International conference on Natural Language Processing (ICON -2008)*, Pune, India, pp 240-245

Singh, Thoudam D., Kishorjit, N., Ekbal, A., Bandyopadhyay, S., 2009. Named Entity Recognition for Manipuri using Support Vector Machine, *In proceedings of 23rd Pacific Asia Conference on Language, Information and Computation (PACLIC 23)*, Hong Kong, pp 811-818

Singh, Thoudam D., Singh, Yengkhom R. and Bandyopadhyay, S., 2010. Manipuri-English Semi Automatic Parallel Corpora Extraction from Web, *In proceedings of 23rd International Conference on the Computer Processing of Oriental Languages (ICCPOL 2010) - New Generation in Asian Information Processing* , Redmond City, CA

Vapnik, Vladimir N. 1995: The nature of Statistical learning theory. Springer

A Word Segmentation System for Handling Space Omission Problem in Urdu Script

Gurpreet Singh Lehal
Department of Computer Science
Punjabi University, Patiala
gslehal@gmail.com

Abstract

Word Segmentation is the foremost obligatory task in almost all the NLP applications, where the initial phase requires tokenization of input into words. Like other Asian languages such as Chinese, Thai and Myanmar, Urdu also faces word segmentation challenges. Though the Urdu word segmentation problem is not as severe as the other Asian language, since space is used for word delimitation, but the space is not consistently used, which gives rise to both space omission and space insertion errors in Urdu. In this paper we present a word segmentation system for handling space omission problem in Urdu script with application to Urdu-Devnagri Transliteration system. Instead of using manually segmented monolingual corpora to train segmenters, we make use of bilingual corpora and statistical word disambiguation techniques. Though our approach is adapted for the specific transliteration task at hand by taking the corresponding target (Hindi) language into account, the techniques suggested can be adapted to independently solve the space omission Urdu word segmentation problems. The two major components of our system are : identification of merged words for segmentation and proper segmentation of the merged words. The system was tested on 1.61 million word Urdu test data. The *recall* and *precision* for the merged word recognition component were found to be 99.29% and 99.38% respectively. The words are correctly segmented with 99.15% accuracy.

1 Introduction

Word segmentation is the foremost obligatory task in all NLP application, where the initial phase requires tokenization of input into words. For languages like English, French and Spanish etc. tokenization is considered trivial because the white space or punctuation marks between words is a good approximation of where a word boundary is. Whilst in various Asian languages such as Chinese, Thai and Myanmar, white spaces is rarely or never used to determine the word boundaries, so one must resort to higher levels of information such as: information of morphology, syntax and statistical analysis to reconstruct the word boundary information (Papageorgiou, 1994; Nie et al, 1995; Wang et al, 2000; Xu et al, 2005).

Though the Urdu word segmentation problem is not as severe as some of the other Asian language, since space is used for word delimitation, but the space is not consistently used, which gives rise to both space omission and space insertion errors in Urdu. Durrani(2007) and Durrani and Hussain(2010) have discussed in detail the various Urdu word segmentation issues while Jawaid and Ahmed(2009) and Abbas et al(2009) have discussed the Hindi-Urdu transliteration issues. A word segmentation system for handling space insertion problem in Urdu script has been presented by Lehal(2009).

Hindi and Urdu are variants of the same language characterized by extreme digraphia: Hindi is written in the Devanagari script from left to right, Urdu in a script derived from a Persian modification of Arabic script written from right to left. Hindi and Urdu share grammar, morphology, vocabulary, history, classical literature *etc.* Because of their identical grammar and nearly identical core vocabularies,

Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), pages 43–50,
the 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010

most linguists do not distinguish between Urdu and Hindi as separate languages. The difference in the two scripts has created a script wedge as majority of Urdu speaking people in Pakistan cannot read Devnagri, and similarly the majority of Hindi speaking people in India cannot comprehend Urdu script. To break this script barrier an Urdu-Devnagri transliteration system has been developed. The transliteration system faced many problems related to word segmentation of Urdu script as discussed above.

In this paper we present a word segmentation system for handling space omission problem in Urdu script with application to Urdu-Devnagri Transliteration system. Instead of using manually segmented monolingual corpora to train segmenters, we make use of bilingual corpora and statistical word disambiguation techniques. Though our approach is adapted for the specific transliteration task at hand by taking the corresponding target (Hindi) language into account, the techniques suggested can be adapted to independently solve the space omission Urdu word segmentation problems.

2 Urdu script: a brief overview

Urdu is a Central Indo-Aryan language of the Indo-Iranian branch, belonging to the Indo-European family of languages. It is the national language of Pakistan. It is also one of the 22 scheduled languages of India and is an official language of five Indian states.

Urdu script has 35 simple consonants, 15 aspirated consonants, one character for nasal sound and 15 diacritical marks. Urdu characters change their shapes depending upon neighboring context. But generally they acquire one of these four shapes, namely isolated, initial, medial and final. Urdu characters can be divided into two groups, non-joiners and joiners. The non-joiners can acquire only isolated and final shape and do not join with the next character. On contrary joiners can acquire all the four shapes and get merged with the following character. A group of joiners and/or non-joiner joined together form a ligature. A word in Urdu is a collection of one or more ligatures. The isolated form of joiners and non-joiners is shown in figures 1-2.

آ ا د ڈ ذ ر ڑ ز ژ و ے

Figure 1. Non-Joiners in Urdu

ب پ ت ٹ ث ج چ ح خ س ش ص ض ط ظ ع غ ف ق ک گ ل م ن ہ ی ہ

Figure 2. Joiners in Urdu

The space character is used in Urdu both to generate correct shaping and also to separate words. Though for words ending with non-joiners correct shaping is generated even when space is not typed and thus, many times a user omits the space. The sequence of Urdu words written together without space is still readable because of the character joining property in Urdu. As for example, consider the word cluster انکارکردیاہے , which is composed of four words انکار, کر , دیا and ہے. The Urdu readers can very easily segment and read the four words separately, but the computer will read them as a single word since there is no space in between. Similarly, the word cluster پرزوردیاگیاہے is composed of five words(پر, زور, دیا, گیا and ہے), which can be easily read as five separate words by Urdu readers but will be considered as a single word by the computer.

Another unique feature of Urdu is that the Urdu words are usually written without short vowels or diacritic symbols. Any machine transliteration or text to speech synthesis system has to automatically guess and insert these missing symbols. This is a non-trivial problem and requires an in-depth statistical analysis.

An Urdu word is a combination of ligatures (characters which join together) and isolated characters. For example انکار is composed of isolated characters ا and ر and ligature نکا . A ligature or isolated character will be called as Urdu character cluster (UCC) in this paper. A Urdu word is thus a combination of UCCs . As for example, the word انکار is composed of three UCCs ا , نکا and ر . We borrow the term, Orthographic Word used by Durrani and Hussain(2010) to define our segmentation process. An Orthographic Word (OW) is a combination of UCCs separated by spaces or punctuation marks. An OW may contain single or multiple Urdu words. Our task is to identify if an OW contains multiple words and in that case properly segment the words.

As for example, consider the sentence:

میزبان ٹیم کی جانب سے رام نریش نے ہیروکاکرداراداکیا

It contains nine OWs

1. میزبان
2. ٹیم
3. کی
4. جانب
5. سے
6. رام
7. نریش
8. نے
9. ہیروکاکرداراداکیا

The first eight OWs contain single Urdu words, while the last OW contains 5 Urdu words(ہیرو, کا, کردار, ادا and کیا)

3 Segmentation Model for Urdu

There are three major issues in the automatic Urdu word segmentation. The first problem is to decide if the orthographic word represents a single word or a multiple word cluster. The second is the ambiguity issue. Since a word cluster can be segmented into words in multiple ways, the correct word boundary detection becomes a challenge. As for example the OW گیاناسے can be segmented as گیان + اسے or نا + گیا + سے. The third problem is the segmentation of unknown word. Unknown word refers to word that does not exist in the dictionary or corpus. Unknown words can be categorized into the different types such as error words, abbreviation, proper names, derived words, foreign words, compounds etc. The unknown word causes segmentation error since the word does not exist in the dictionary, it could be incorrectly segmented into shorter words. For example, the word, ڈرمیٹالوجی , which is a foreign word, gets segmented into four words (ڈر , میٹا , لو and جی) after dictionary look-up as the word ڈرمیٹالوجی is not present in the corpus.

The input is an Urdu Orthographic Word and the system first makes the decision if the OW contains single or multiple Urdu words. In case the OW contains multiple words, the individual Urdu words are extracted from the OW. These different stages are discussed in detail in following sections. As can be seen from the figure, at each stage we make use of lexical resources both from Urdu and Hindi languages. The details of the resources used are in Table 1.

The system architecture is shown in Fig. 3.

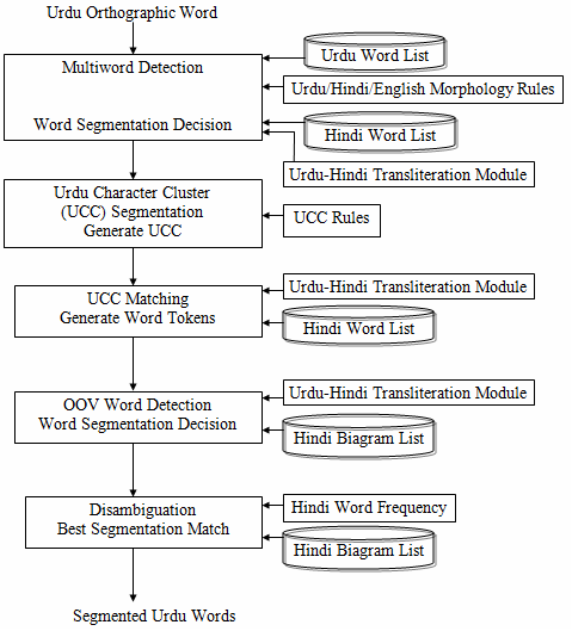

Figure 3. System Architecture

Table 1. Lexical resources used in system

Resource	**Count**
Urdu Word Frequency List	121,367 words
Hindi Word Frequency List	159,426 words
Hindi Word Bigram List	2,382,511 bigrams

4 Decision Stage

In the decision stage, the system decides if the OW contains single or multiple Urdu words. It could so happen that the OW contains single word only and we may break up into smaller words. The decision is based on Urdu and Hindi, word frequency lists analysis as well as Urdu/English/Hindi Morphological rules. To decide if the word cluster is containing multiple words, we first search for OW in the Urdu word list. If it is found then it means that the OW is a valid Urdu word and does not need any further segmentation and quit over there.

It could happen that the OW could be an inflection, whose root form maybe present in the

Urdu word list. Even though the Urdu word list contains inflected forms, but for many words all the inflections may not be present. This problem is more pronounced for English terms, which have become part of Urdu language. For such words, the inflections could follow both rules of English and Urdu. For example plural of یونیورسٹی (university) could be both *universitiyon* وںیورسٹیوںی as well as *universities* یونیورسٹیز. The first form follows the Urdu infection rules while the second form follows the English inflection rules. Similarly we found both the Urdu and English inflections for the English word secretary in Urdu text (سیکرٹریوں and سیکرٹریز) . Thus if the OW is not found in the Urdu word list, we use both Urdu and English morphological rules to generate its root form and search for the root form in the Urdu word list. If the root form is found, we assume the word to be a valid Urdu word and quit there.

It is widely reported in word segmentation papers, that the greatest barrier to accurate word Segmentation is in recognizing words that are not in the lexicon of the segmenter. Thus if a word or its root form is not present in the Urdu word list it will be wrongly presumed to be a multi word cluster. To alleviate this problem, the Urdu corpus has been supplemented with Hindi corpus, which has helped in increasing the word segmentation as well as multi-word recognition accuracy. It was found many times that the Urdu word may be a proper noun, foreign word or some valid out of vocabulary word, which is not present in Urdu corpus but present in the Hindi word list. Another advantage of checking in the Hindi corpus is that many of the Hindi words, which are written as single word are usually written as two words in Urdu. For example, کرےگا (करेगा), کھیلتے (खेलते), ایمانداری (ईमानदारी), چئیرمین (चेयरमैन) etc. These Urdu words are many times written as a single word and in that case if passed to Hindi word list would still report as correct. For checking the OW in Hindi word list, we first transliterate it to Hindi and then search for it in the Hindi wordlist. If the transliterated word is found, then the OW is not considered for segmentation. Like Urdu, it may also happen that the root word of OW may be present in the Hindi word list. So like Urdu, we use both Urdu and English morphological rules to generate its root form and search for the root form in the Hindi word list. If the root form is found, we assume the word to be a valid Urdu word and quit there. If the OW passes all the above stages, then it is considered a candidate for segmentation.

The steps in brief are :

- Search for OW in Urdu List. If OW is present in the list then quit. example : مطابق
- Determine the root form of OW using Urdu Morphological rules and search for the root form in Urdu List. If found then quit. example : سیکرٹریوں
- Determine the root form of OW using English Morphological rules and search for the root form in Urdu List. If found then quit. example : ٹورنامنٹس
- Let HW = Transliteration of OW in Hindi. Search for HW in the Hindi Word List. If HW is present in the list then quit. example : ایمانداری
- Determine the root form of HW using Hindi Morphological rules and search for the root form in the Hindi List. If found then quit. example : چیئرمینوں
- Determine the root form of HW using English Morphological rules and search for the root form in the Hindi List. If found then quit. example : ہولڈرز
- Go to the segmentation stage. example : تھااس

5 Segmenting the Orthographic Word

The Urdu orthographic word is next broken into Urdu Character Combinations (UCC) using Urdu orthographic rules. Unlike word segmentation that is a difficult task, segmenting a text into UCCs is easily achieved by applying the set of rules. These adjacent UCCs are then combined to form a sequence of Urdu words. We need to list all possible segmentations and design a strategy to select the most probable correct segmentation from them.

As for example, consider the OW توجواب: It is segmented into four UCCs : تو, جو. ا and ب . The adjacent clusters can be combined to form 6 word segmentations:

- جواب + تو
- اب + توجو

- ب + توجوا
- اب + جو + تو
- ب + ا + توجو
- ب + ا + جو + تو

5.1 Longest Matching

The method scans an input sentence from left to right, and select the longest match with a dictionary entry at each point. In case that the selected match cannot lead the algorithm to find the rest of the words in the sentence, the algorithm will backtrack to find the next longest one and continue finding the rest and so on. This algorithm fails to find the correct segmentation in many cases because of its greedy characteristic.

5.2 Maximum Matching

This method first generates all possible segmentations for a sentence and then selects the one that contain the fewest words, which can be done efficiently by using dynamic programming technique. When the alternatives have the same number of words, the algorithm cannot determine the best candidate and some other heuristics have to be applied.

We tried both longest matching and maximum matching and the smallest unit taken for combining is UCC. But we found shortcomings in both the matchings. For example the OW کرارہاہے gets segmented as کرار +ہا +ہے using longest matching, while it should be کرا+رہا+ہے . Similarly the OW بروزاتوارکودن gets segmented as بروز+اتوار+کودن using maximum matching while it should be بروز+اتوار+کو+دن.

Thus we see that both longest string match and smallest words fail sometimes. If these algorithms are supplemented by statistical information such as frequency analysis and n-grams then these failures can be avoided. So in our present work, we apply maximal matching algorithm along with these statistics. Initially we used unigram frequency of occurrence for deciding the best word combination. Each Urdu word in the combination is formed by joining adjacent UCCs. In each of the combination, we first convert each of the Urdu word to Hindi. The combination with highest combined product of the unigram frequency of occurrences is finally selected. Thus in the above example, the OW توجواب: will be segmented as تو + جواب, as shown in Table 2.

Table 2. Product of Frequency of Occurrence

Urdu Combination	Hindi Combination (Frequency of occurrence)	Frequency Product
تو جواب	तो (0.005161) जवाब (0.00026)	1.34221E-06
توجو اب	तोजो (4.16E-07) अब (0.001623)	6.75557E-10
توجوا ب	तोजवा (0) ब (4.48E-05)	0
تو جو اب	तो (0.005161) जो (0.002602) अब (0.001623)	2.18028E-08
توجو ا ب	तोजो (4.16E-07) अ (3.6E-05) ब (4.48E-05)	6.69866E-16
تو جو ا ب	तो (0.005161) जो (0.002602) अ (3.6E-05) ब (4.48E-05)	2.16191E-14

It is interesting to see that for segmentation of Urdu words, we used Hindi language statistical analysis instead of Urdu language statistical analysis. Since the current system is part of

Urdu-Hindi transliteration system, we prefer the output to be segmented according to Hindi rules. There are many words which are otherwise joined in Hindi but written as separate words in Urdu. So if we use the Urdu language modeling for segmentation, the word gets broken. Some of the examples are:
اغواکار is written as combination of two words اغوا+ کار in Urdu but its equivalent Hindi word अग़वाकार is written as a single word. Similarly, in Hindi text the verbs are concatenated with the future auxiliaries "gaa", "gii" and "ge", while they are written separately in Urdu. Thus کرین +گے are written separately, but their equivalent Hindi form करेंगे is written as single word. So the advantage of using Hindi training data is that the words get segmented according to the desired Hindi rules. Another problem with Urdu training data was that the Urdu training itself contains merged words. So the words had to be manually separated, though fortunately the Urdu corpus compiled by CRULP (www.crulp.org) has been quite clean, but many words were missing particularly English ones. Another problem is that the words are broken even in the cleaned Urdu corpus. On the other hand when we used the Hindi training data for word segmentation, the problems of merged or broken words in the training text were not encountered. Also the Hindi corpus compiled by us had much larger vocabulary coverage, while the Urdu corpus we used for training purpose had many common words such as اوباما, خطرے , گاندھی, جیکسن etc. missing. Thus the word segmentation algorithm which used the Hindi training set had much better segmentation accuracy as compared to the Urdu training set.

We observed that though the above scheme worked fine in majority of the cases, but in a few cases it failed to segment properly as it did not take care of the context or adjacent words. As for example consider the OW : مردیاعورت. It contains six CCs: مر, د, یا, عو, ر and ت. The word combination selected by above methodology is :

مر + دیا + عورت , though the correct combination is مرد + یا + عورت. It was observed that as we did not take care about adjacent words, thus wrong combination was selected. If the bigram information is added, then such problems were reduced.

We thus use both unigram and bigram frequency analysis for deciding the best word combination. Each Urdu word in the combination is formed by joining adjacent UCCs. In each of the combination, we first convert each of the Urdu word to Hindi. Next we find the unigram and bigram frequency of occurrence of each Hindi word and Hindi word pair in the combination. The bigram frequencies are normalized to avoid multiplication by zero. The combination with highest combined product of the unigram and bigram frequencies of occurrences is finally selected. Using this methodology we were able to generate the sequence combination is مرد + یا + عورت in above example.

As we are using Hindi training data, it was observed that sometimes we had merged words which did not had equivalent transliterated words in our Hindi frequency list. As example, the OW ترازابلیس had to be segmented as تراز + ابلیس, but the equivalent transliterated Hindi terms of تراز and ابلیس, were not found in the Hindi frequency list. As a result, the OW is not segmented. To take care of such situations, if we cannot segment using the Hindi frequency list, our system then goes for maximal matching using the Urdu training data. Thus in above example, after search fails in Hindi training set, the system searches for the minimum word combination and on finding the above two words in the Urdu training set segments the OW into these words.

6 Over Segmentation

For wrongly spelled or OOV (out of vocabulary) Urdu words, the system may forcibly break the word into smaller words. As for example, our system forcibly broke the OW گردوہر into گر + دو +ہر . This problem proved difficult to tackle, though we were able to partially solve it. It was found that usually the OOV words were broken into small unrelated words. So we put the condition on the system to accept only those word segments which contained at least one word of length greater than three or at least one bigram pair was present in the Hindi bigram list. The presence of at least one bigram pair ensured that all the words were not unrelated. Thus in the

above example, the OW gets split into three words, all of length two. These words when transliterated to Hindi get converted to गिर + दो + हर. On searching the bigram list, it was found that neither of the bigram pair < गिर, दो > and < दो , हर > was present and thus this word segmentation was rejected.

7 Experiments

We tested our system on a test data of 1,613,991 Urdu words. In the decision stage, it was found that 116,078 words, which make 7.19% of original text were not found in the Urdu corpus and were considered candidates for segmentation. After morphological analysis of these words, 2851 Urdu words were found to be valid Uru words and were removed from the segmentation candidate list. After converting the remaining Urdu words to Hindi and checking them in Hindi corpus, only 35,226 words were left which were not present in Hindi corpus. Therefore from original 16,13,991 only 35,226 (2.19%) were passed onto segmentation stage for checking for merged words.

In the segmentation stage it was found that out of 35,226 words, 24,001 words (68.13%) had merged words. The number of merged words varied from 2 to 6. Table 3 show the frequency of number of merged words found in word clusters. As can be seen from the table 96.71% of merged word clusters had two merged words.

Table 3. Frequency of Merged Words

Number of merged words	Frequency Percentage
2	96.71%
3	2.99%
4	0.25%
5	0.037%
6	0.004%

The *recall* and *precision* for the decision stage, which decides if the OW needs to be segmented, were found to be 99.29% and 99.38% respectively.

The word segmentation algorithm was able to correctly segment the words with 99.15% accuracy.

8 Conclusions

In this paper, we have presented a system for solving the space omission problem in Urdu text. This system is part of the larger system designed for transliteration of Urdu text to Hindi. We have combined statistical language modeling of both Urdu and Hindi languages in development of the system. We have presented a new scheme of using Hindi for segmenting Urdu text after transliteration, because Hindi uses spaces consistently versus Urdu which has both space omission and insertion problems. This is the first time such a segmentation scheme for handling Urdu space omission problem has been presented. The word segmentation algorithm was able to correctly segment the words with 99.15% accuracy.

Acknowledgements

The author will like to acknowledge the support provided by ISIF grants for carrying out this research.

References

Durrani N. 2007. Typology of Word and Automatic Word Segmentation in Urdu Text Corpus. *National University of Computer and Emerging Sciences*, Lahore, Pakistan.

Durrani N. and Hussain Sarmad. 2010. Urdu Word Segmentation.http://www.crulp.org/Publication/papers/2010/Urdu_Word_Segmentation_NAACL.pdf (accessed on 5th July 2010).

Jawaid Bushra and Ahmed Tafseer. 2009. Hindi to Urdu Conversion: Beyond Simple Transliteration. *Proceedings of the Conference on Language & Technology*, Lahore,.Pakistan, 24-31.

Lehal G. S. 2009. A Two Stage Word Segmentation System For Handling Space Insertion Problem In Urdu Script. *Proceedings of World Academy of Science, Engineering and Technology,* Bangkok, Thailand, 60: 321-324.

Malik Abbas, Besacier Laurent, Boitet Christian and Bhattacharyya Pushpak. 2009. A hybrid Model for Urdu Hindi Transliteration. *Proceedings of the*

2009 Named Entities Workshop, ACL-IJCNLP 2009, Singapore, 177-185.

Nie, J.Y., Hannan, M.L. & Jin, W. 1995. Combining dictionary, rules and statistical information in segmentation of Chinese. *Computer Processing of Chinese and Oriental Languages*, 9(2): 125-143.

Papageorgiou Constantine P. 1994. Japanese word segmentation by hidden Markov model. *Proc. of the HLT Workshop*, 283–288.

Wang Xiaolong, , Fu Guohong, Yeung Danial S., Liu James N.K., and Luk Robert. 2000. Models and algorithms of Chinese word segmentation. *Proceedings of the International Conference on Artificial Intelligence (IC-AI'2000)*, Las Vegas, Nevada, USA, 1279-1284.

Xu Jia, Matusov Evgeny, Zens Richard, and Ney. 2005. Hermann.Integrated Chinese word segmentation in statistical machine translation. *Proceedings of the International Workshop on Spoken Language Translation*, Pittsburgh, PA, 141-147.

Hybrid Stemmer for Gujarati

Pratikkumar Patel **Kashyap Popat**
Department of Computer Engineering
Dharmsinh Desai University
pratikpat88@gmail.com
kan.pop@gmail.com

Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
pb@cse.iitb.ac.in

Abstract

In this paper we present a lightweight stemmer for Gujarati using a hybrid approach. Instead of using a completely unsupervised approach, we have harnessed linguistic knowledge in the form of a hand-crafted Gujarati suffix list in order to improve the quality of the stems and suffixes learnt during the training phase. We used the EMILLE corpus for training and evaluating the stemmer's performance. The use of hand-crafted suffixes boosted the accuracy of our stemmer by about 17% and helped us achieve an accuracy of 67.86 %.

1 Introduction

Stemming is the process of conflating related words to a common stem by chopping off the inflectional and derivational endings. Stemming plays an important role in Information Retrieval (IR) systems by reducing the index size and increasing the recall by retrieving results containing any of the various possible forms of a word present in the query. This is especially true in case of a morphologically rich language like Gujarati, where a single word may take many forms. The aim is to ensure that related words map to common stem, irrespective of whether or not the stem is a meaningful word in the vocabulary of the language.

Current state of the art approaches to stemming can be classified into three categories, viz., rule based, unsupervised and hybrid. Building a rule based stemmer for a morphologically rich language is an uphill task considering the different inflectional and morphological variations possible. Purely unsupervised approaches on the other hand fail to take advantage of some language phenomenon which can be easily expressed by simple rules. We thus follow a hybrid approach by enhancing an unsupervised system with a list of hand-crafted Gujarati suffixes.

The remainder of this paper is organized as follows. We describe related work in section 2. Section 3 explains the morphological structure of Gujarati. We describe our approach in section 4. The experiments and results are described in section 5. Section 6 concludes the paper highlighting the future work.

2 Background and Related Work

The earliest English stemmer was developed by Julie Beth Lovins in 1968. The Porter stemming algorithm (Martin Porter, 1980), which was published later, is perhaps the most widely used algorithm for English stemming. Both of these stemmers are rule based and are best suited for less inflectional languages like English.

A lot of work has been done in the field of unsupervised learning of morphology. Goldsmith (2001, 2006) proposed an unsupervised algorithm for learning the morphology of a language based on the minimum description length (MDL) framework which focuses on representing the data in as compact manner as possible. Creutz (2005, 2007) uses probabilistic maximum a posteriori (MAP) formulation for unsupervised morpheme segmentation.

Not much work has been reported for stemming for Indian languages compared to English and other European languages. The earliest work reported by Ramanathan and Rao (2003) used a hand crafted suffix list and performed longest match stripping for building a Hindi stemmer. Majumder et al. (2007) developed YASS: Yet Another Suffix Stripper which uses a clustering based approach based on string dis-

Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), pages 51–55,
the 23rd International Conference on Computational Linguistics (COLING), Beijing, August 2010

tance measures and requires no linguistic knowledge. They concluded that stemming improves recall of IR systems for Indian languages like Bengali. Dasgupta and Ng (2007) worked on unsupervised morphological parsing for Bengali. Pandey and Siddiqui (2008) proposed an unsupervised stemming algorithm for Hindi based on Goldsmith's (2001) approach.

Unlike previous approaches for Indian languages which are either rule based or completely unsupervised, we propose a hybrid approach which harnesses linguistic knowledge in the form of a hand-crafted suffix list.

3 Gujarati Morphology

Gujarati has three genders (masculine, neuter and feminine), two numbers (singular and plural) and three cases (nominative, oblique/vocative and locative) for nouns. The gender of a noun is determined either by its meaning or by its termination. The nouns get inflected on the basis of the word ending, number and case.
The Gujarati adjectives are of two types – declinable and indeclinable. The declinable adjectives have the termination *-ũ* (ુ઼ં) in neuter absolute. The masculine absolute of these adjectives ends in *-o* (ો) and the feminine absolute in *-ī* (ી). For example, the adjective સારું (*sārũ* - good) takes the form સારું (*sārũ*), સારો (*sāro*) and સારી (*sārī*) when used for a neuter, masculine and feminine object respectively. These adjectives agree with the noun they qualify in gender, number and case. The adjectives that do not end in *-ũ* in neuter absolute singular are classified as indeclinable and remain unaltered when affixed to a noun.

The Gujarati verbs are inflected based upon a combination of gender, number, person, aspect, tense and mood.

There are several postpositions in Gujarati which get bound to the nouns or verbs which they postposition. e.g. *-nũ* (નું : genitive marker), *-mã̃* (માં : in), *-e* (ે : ergative marker), etc. These postpositions get agglutinated to the nouns or verbs and not merely follow them.

We created a list of hand crafted Gujarati suffixes which contains the postpositions and the inflectional suffixes for nouns, adjectives and verbs for use in our approach.

4 Our Approach

Our approach is based on Goldsmith's (2001) take-all-splits method. Goldsmith's method was purely unsupervised, but we have used a list of hand crafted Gujarati suffixes in our approach to learn a better set of stems and suffixes during the training phase. In our approach, we make use of a list of Gujarati words extracted from EMILLE corpus for the purpose of learning the probable stems and suffixes for Gujarati during the training phase. This set of stems and suffixes will be used for stemming any word provided to the stemmer. We have described the details of our approach below.

4.1 Training Phase

During the training phase, we try to obtain the optimal split position for each word present in the Gujarati word list provided for training. We obtain the optimal split for any word by taking all possible splits of the word (see Figure 1) and choosing the split which maximizes the function given in Eqn 1 as the optimal split position. The suffix corresponding to the optimal split position is verified against the list of 59 Gujarati suffixes created by us. If it cannot be generated by agglutination of the hand crafted suffixes, then the length of the word is chosen as the optimal split position. i.e. the entire word is treated as a stem with no suffix.

$\{stem_1+suffix_1, stem_2+suffix_2, \ldots, stem_L+suffix_L\}$
ઘરના= {ઘ + રના, ઘર + ના, ઘરન + ા,ઘરના + NULL}

Figure 1. All Possible Word Segmentations

$$f(i) = i*\log(freq(stem_i)) + (L-i)*\log(freq(suffix_i)) \quad \text{(Eqn 1)}$$

i: split position (varies from 1 to L)
L: Length of the word

The function used for finding the optimal split position reflects the probability of a particular split since the probability of any split is determined by the frequencies of the stem and suffix generated by that split. The frequency of shorter stems and suffixes is very high when compared to the slightly longer ones. Thus the multipliers *i* (length of $stem_i$) and *L-i* (length of $suffix_i$) have been introduced in the function in order to compensate for this disparity.

Once we obtain the optimal split of any word, we update the frequencies of the stem and suffix generated by that split. We iterate over the word list and re-compute the optimal split position until the optimal split positions of all the words remain unchanged. The training phase was observed to take three iterations typically.

4.2 Signatures

After the training phase, we have a list of stems and suffixes along with their frequencies. We use this list to create signatures. As shown in Figure 2, each signature contains a list of stems and a list of suffixes appearing with these stems.

Stems	Suffixes
પશુ (*pashu* - animal)	ના (*nā*)
જંગ (*jang* - war)	નો (*no*)
	ને (*ne*)
	નું (*nũ*)
	ની (*nī*)

Figure 2. Sample Signature

The signatures which contain very few stems or very few suffixes may not be useful in stemming of unknown words, thus we eliminate the signatures containing at most one stem or at most one suffix. The stems and suffixes in the remaining signatures will be used to stem new words. An overview of the training algorithm is shown in Figure 3.

Step 1: Obtain the optimal split position for each word in the word list provided for training using Eqn 1 and the list of hand crafted suffixes

Step 2: Repeat Step 1 until the optimal split positions of all the words remain unchanged

Step 3: Generate signatures using the stems and suffixes generated from the training phase

Step 4: Discard the signatures which contain either only one stem or only one suffix

Figure 3. Overview of training algorithm

4.3 Stemming of any unknown word

For stemming of any word given to the stemmer, we evaluate the function in Eqn 1 for each possible split using the frequencies of stems and suffixes obtained from the training process. The word is stemmed at the position for which the value of the function is maximum.

5 Experiments and Result

We performed various experiments to evaluate the performance of the stemmer using EMILLE Corpus for Gujarati. We extracted around ten million words from the corpus. These words also contained Gujarati transliterations of English words. We tried to filter out these words by using a Gujarati to English transliteration engine and an English dictionary. We obtained 8,525,649 words after this filtering process.

We have used five-fold cross validation for evaluating the performance. We divided the extracted words into five equal parts of which four were used for training and one for testing. In order to create gold standard data, we extracted thousand words from the corpus randomly and tagged the ideal stem for these words manually.

For each of the five test sets, we measured the accuracy of stemming the words which are present in the test set as well as gold standard data. Accuracy is defined as the percentage of words stemmed correctly.

The experiments were aimed at studying the impact of (i) using a hand-crafted suffix list, (ii) fixing the minimum permissible stem size and (iii) provide unequal weightage to the stem and suffix for deciding the optimal split position. Various results based on these experiments are described in the following subsections.

5.1 Varying Minimum Stem Size

We varied the minimum stem size from one to six and observed its impact on the system performance. We performed the experiment with and without using the hand-crafted suffix list. The results of this experiment are shown in Table 1 and Figure 4.

The results of this experiment clearly indicate that there is a large improvement in the performance of the stemmer with the use of hand-crafted suffixes and the performance degrades if we keep a restriction on the minimum stem size. For higher values of minimum stem size, all the valid stems which are shorter than the minimum stem size do not get generated leading to a decline in accuracy.

Min Stem Size	Accuracy	
	With hand-crafted suffixes	Without hand-crafted suffix-es
1	**67.86 %**	50.04 %
2	67.70 %	49.80 %
3	66.43 %	49.60 %
4	59.46 %	46.35 %
5	51.65 %	41.22 %
6	43.81 %	36.89 %

Table 1. Effect of use of hand-crafted suffixes and fixing min. stem size on stemmer's performance

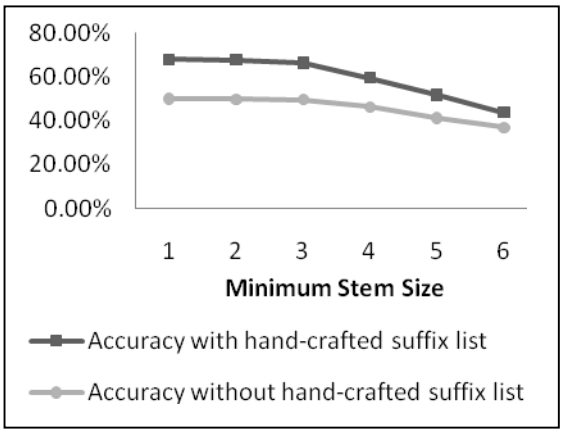

Figure 4. Variation stemmer's accuracy with the variation in min. stem size

There are several spurious suffixes which get generated during the training phase and degrade the performance of the stemmer when we don't use the hand-crafted suffix list. e.g. 'ક' is not a valid inflectional Gujarati suffix but it does get generated if we don't use the hand-crafted suffix list due to words such as 'અનેક' (*anek* - many) and 'અને' (*ane* - and). A simple validation of the suffixes generated during training against the hand-crafted suffix list leads to learning of better suffixes and in turn better stems during the training phase thereby improving the system's performance.

Thus we decided to make use of the hand-crafted suffix list during training phase and not to put any restriction on the minimum stem size.

5.2 Providing unequal weightage to stem and suffix

We have provided equal weightage to stem and suffix in Eqn 1 which is responsible for determining the optimal split position of any word. We obtained Eqn 2 from Eqn 1 by introducing a parameter 'α' in order to provide unequal weightage to the stem and suffix and observe its effect on system performance. We used Eqn 2 instead of Eqn 1 and varied α from 0.1 to 0.9 in this experiment. The results of this experiment are shown in Table 2.

$$f(i) = \alpha * i * \log(freq(stem_i)) + (1-\alpha) * (L-i) * \log(freq(suffix_i)) \quad \text{(Eqn 2)}$$

α	Accuracy
0.1	53.52 %
0.2	61.71 %
0.3	65.43 %
0.4	67.30 %
0.5	67.86 %
0.6	67.48 %
0.7	67.49 %
0.8	67.72 %
0.9	66.45 %

Table 2. Effect of α on the stemmer's performance

The accuracy was found to be maximum when value of α was fixed to 0.5 i.e. stem and suffix were given equal weightage for determining the optimal split of any word.

6 Conclusion and Future Work

We developed a lightweight stemmer for Gujarati using a hybrid approach which has an accuracy of 67.86 %. We observed that use of a hand-crafted Gujarati suffix list boosts the accuracy by about 17 %. We also found that fixing the minimum stem size and providing unequal weightage to stem and suffix degrades the performance of the system.

Our stemmer is lightweight and removes only the inflectional endings as we have developed it for use in IR system. The list of hand-crafted suffixes can be extended to include derivational suffixes for performing full fledged stemming which may be required in applications such as displaying words in a user interface.

We have measured the performance of the stemmer in terms of accuracy as of now. We plan to evaluate the stemmer in terms of the index compression achieved and the impact on precision and recall of Gujarati IR system.

References

Creutz, Mathis, and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

Creutz, Mathis, and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *Association for Computing Machinery Transactions on Speech and Language Processing,* 4(1):1-34.

Dasgupta, Sajib, and Vincent Ng. 2006. Unsupervised Morphological Parsing of Bengali. *Language Resources and Evaluation*, 40(3-4):311-330.

Goldsmith, John A. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153-198

Goldsmith, John A. 2006. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(4):353-371

Jurafsky, Daniel, and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. 2nd edition. Prentice-Hall, Englewood Cliffs, NJ.

Lovins, Julie B. 1968. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22-31

Majumder, Prasenjit, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. 2007. YASS: Yet another suffix stripper. *Association for Computing Machinery Transactions on Information Systems*, 25(4):18-38.

Pandey, Amaresh K., and Tanveer J. Siddiqui. 2008. An unsupervised Hindi stemmer with heuristic improvements. In *Proceedings of the Second Workshop on Analytics For Noisy Unstructured Text Data*, 303:99-105.

Porter, Martin F. 1980. An algorithm for suffix stripping. *Program*, 14(3):130-137.

Ramanathan, Ananthakrishnan, and Durgesh D. Rao, A Lightweight Stemmer for Hindi, *Workshop on Computational Linguistics for South-Asian Languages*, EACL, 2003.

Tisdall, William St. Clair. 1892. *A simplified grammar of the Gujarati language : together with A short reading book and vocabulary.* D. B. Taraporevala Sons & Company, Bombay.

The EMILLE Corpus, http://www.lancs.ac.uk/fass/projects/corpus/emille/

Author Index