A Framework for Learning Morphology using Suffix Association Matrix

Shilpa Desai
Department of Computer Science and Technology
Goa University, Goa, India
sndesai@gmail.com

Jyoti Pawar
Department of Computer Science and Technology
Goa University, Goa, India
jyotidpawar@gmail.com

Pushpak Bhattacharyya
Department of Computer Science and Engineering
IIT, Powai, Mumbai India
pb@cse.iitb.ac.in

Abstract

Unsupervised learning of morphology is used for automatic affix identification, morphological segmentation of words and generating paradigms which give a list of all affixes that can be combined with a list of stems. Various unsupervised approaches are used to segment words into stem and suffix. Most unsupervised methods used to learn morphology assume that suffixes occur frequently in a corpus. We have observed that for morphologically rich Indian Languages like Konkani, 31 percent of suffixes are not frequent. In this paper we report our framework for Unsupervised Morphology Learner which works for less frequent suffixes. Less frequent suffixes can be identified using p-similar technique which has been used for suffix identification, but cannot be used for segmentation of short stem words. Using proposed Suffix Association Matrix, our Unsupervised Morphology Learner can also do segmentation of short stem words correctly. We tested our framework to learn derivational morphology for English and two Indian languages, namely Hindi and Konkani. Compared to other similar techniques used for segmentation, there was an improvement in the precision and recall.

1 Introduction

Learning morphology by a machine is crucial for tasks like stemming, machine translation etc. Rule based affix stripping approach, semi-supervised, unsupervised learning of morphology and finite state approach as some of the well known methods used to learn morphology by a machine. Rule based affix stripping approaches (Lovins, 1968; Porter, 1980; Paice, 1990; Loftsson, 2008; Maung et. al, 2008) depend heavily on linguistic input and require a lot of human effort, especially for morphologically rich languages. Pure unsupervised approaches learn morphology from a corpus (Freitag, 2005; Goldsmith, 2001; Hammarström, 2011). The accuracy of pure unsupervised methods is relatively low. Semi-supervised approaches use minimal linguistic input and unsupervised methods to automate morphology learning process (Forsberg, 2007; Lindén, 2008; Chan, 2008; Dreyer, 2011). Semi-supervised approaches perform better than pure unsupervised approaches. Finite state approaches (Koskenniemi, 1983; Beesley & Kartunnen, 2003) represent morphology using finite state machines. Finite state approaches require linguistic input in the form of paradigm identification. Unsupervised and semi-supervised methods can provide input to build finite state based morphology systems reducing the time taken to build such systems.

In this paper we report the framework for an Unsupervised Morphology Learner. Most unsupervised segmentation techniques (Freitag, 2005; Goldsmith, 2001; Hammarström, 2011) which learn morphology from a corpus assume that suffixes are frequent in a corpus. We observed that for morphologically rich Indian languages like Hindi and Konkani, the assumption that suffixes are frequent does not hold true. These languages are morphologically rich and 31 percent of verb suffixes are not frequent in the corpus. Thus, we choose not to make any such assumption about the frequency of suffix occurrence in our unsupervised learning of morphology. One promising methodology for unsupervised segmentation which does not make any suffix frequency assumptions is p-similar

Proceedings of the 5th Workshop on South and Southeast Asian NLP, 25th International Conference on Computational Linguistics, pages 28–36, Dublin, Ireland, August 23-29 2014.

technique for morpheme segmentation first proposed by Gaussier (1999). Researchers have used this method for suffix identification and not for segmentation (Gaussier, 1999; Sharma, 2006). We extended this less studied technique to segment words by introducing the concept of suffix association matrix, thus giving us an unsupervised method which correctly identifies suffixes irrespective of their frequency of occurrence in the corpus and also segments short stem words. To the best of our knowledge, most reported work which uses p-similar technique for suffix identification (Gaussier, 1999; Sharma, 2006) enforce a restriction on stem-length that it should be at least five. This restriction works well for suffix identification but not for segmentation. For Indian languages like Hindi and Konkani, we observed that the restriction leads to an inability to segment many words with short stem-length. Especially many verb stems in Indian languages have stem-length less than five. To overcome this shortcoming, we have proposed an Unsupervised Morphology Learner (UML) framework.

We implemented UML framework for derivational morphology and tested our method for English language and two Indian languages namely Konkani and Hindi. The rest of the paper is organized as follows; section 2 is on related work. Section 3 provides the terminology used in the paper. The motivation for this work is presented in section 4. Unsupervised Morphology Learner (UML) framework is presented in section 5. Experimental results are discussed in section 6 and finally we conclude the paper in section 7.

2 Related Work

Unsupervised learning of morphology is done at different levels, namely, affix list identification, segmenting word into stem and affix, and generating a list of paradigms i.e. a list of all stems with information of the suffixes that each stem combines with (Hammarström, 2011). In his survey paper, Hammarström (2011) summarizes work related to unsupervised morphology. Most recent work in morphology learning is semi-supervised. Such methods use a small set of example paradigms as input to train the system and classify unseen words into paradigms or learn new paradigms (Lindén, 2009; Dreyer, 2011).

A popular pure unsupervised morphology technique was first proposed by Goldsmith (2001) which does not assume any linguistic input. Goldsmith (2001) introduced a set of heuristics that develops a probabilistic morphological grammar, and used Minimum Description Length (MDL) as a tool to evaluate it. The technique used for affix and paradigm identification was based on affix occurrence frequency. Several different authors have appreciated MDL as the motivation for segmentation. Some authors (Gelbukh et. al., 2004; Bacchin, 2005) have used random segmentation and picked the best segmentation to minimize size or find splits where constituent morphemes occur in multiple splits.

Our work is inspired by a less studied p-similar technique proposed by Gaussier (1999). p-similar techniques have been used for suffix identification rather than segmentation in most related unsupervised morphology learners (Sharma, 2006). Here the restriction on stem-length first proposed by Gaussier is upheld. Sharma's (2006) work deals with neutral suffix only and does not capture non-neutral suffixes. These studies are limited to suffix identification and do not generate paradigms.

3 Terminology Used

Let L be a language with alphabet set $\sum$.

$W= \{w| w \subset \sum^*\}$ be set of valid words in language L.

Let d: $W \rightarrow W$ denote a derivation function where $d(w_x)=w_y$ iff words w_x and w_y are derivationally related to each other in L.

Let w_xs_y denote concatenation of strings w_x and s_y where $w_x, s_y \in \sum^*$.

Let S_N be set of neutral derivational suffixes.

$S_N = \{s|w_2=w_1s$ and $w_2,w_1 \in W$ and $d(w_1)=w_2$ and $s \in \sum^*\}$

For example, when s=er, w_1=farm and w_2=farmer

Let S_B be set of non-neutral derivational suffixes.

$S_B = \{s_x,s_y|ws_x=ws_y$ and $d(ws_x)=ws_y$ and $w, s_x, s_y \in \sum^*$ and $w \notin W\}$

For example, when s_x=ify, s_y=ity and w=quant suffixes *ify, ity* are non neutral suffixes.

4 Motivation

Primarily, frequency based suffix identification techniques (Goldsmith, 2001; Hammarström, 2011) commonly used in recent times, fail to identify suffixes with low frequency. We explored suffix identification techniques which could identify suffixes irrespective of frequency of occurrence in the corpus. We chose one such method p-similar technique. However p-similar technique (Gaussier, 1999) cannot be used directly for segmentation as it results in a high number of false positives. Hence we proposed a suffix association matrix to avoid the false positives. According to p-similar technique, given two words x, y ∈ W, if ∃ b_1 such that $x=b_1s_1$ and $y=b_1s_2$ where $b_1, s_1, s_2 \in \sum^+$, then b_1 is a stem and s_1, s_2 are suffixes, provided they satisfy the following conditions:

a. A suffix is valid only when it occurs with at least two different stems
b. A stem is valid when it occurs with at least two identified suffixes
c. Stem length should be five or more

The third condition on stem length was introduced to improve the precision of the suffix list generated. However the aim was to only generate a suffix list and not segment word into stem + suffix. We probed the possibility of applying this effective p-similar technique to segment words. We faced the following issues when trying to use p-similar technique for segmentation:

- The technique failed for short-stem length words because of the restriction placed on stem-length. Example words with stem like *walk, talk* are not segmented.
- When words like addiction, addictive, aggression and aggressive are part of the input, suffixes identified are *"on"* and *"ve"* in place of *"ion"* and *"ive"*. This problem is called over-stemming.
- When words like *cannon, cannot, America, American, agent, agency* are part of the input, *"n"* and *"t"* are identified as suffix. Although *"n"* and *"t"* are valid suffix for some words, *cannon=canno+n* and *cannot=canno+t* are wrong segmentation.

We realize that the candidate stem-suffix pair b_i+s_i identified using the p-similar technique falls under one of the following cases:

Case 1: b_i is a valid stem and s_i is a valid suffix for stem b_i. For example, *mistake+NULL*, *mistake+n* are valid. Suffixes *NULL* and *n* are valid for stem *mistake*.

Case 2: b_i is an invalid stem and s_i is a invalid suffix. Example *addicti+on* and *addicti+ve* and *aggressi+on* and *aggressi+ve* are invalid; *addict+ion* and *addict+ive* and *aggress+ion* and *aggress+ive* are valid.

Case 3: b_i is a valid stem and s_i is a invalid suffix for stem b_i. For example *year+n* is invalid. Suffix *n* is invalid for stem *year* while suffix *NULL* and *ly* are valid for stem year.

Case 4: b_i is an invalid stem for any suffix and s_i is valid for some other stem. Example *canno+n* and *canno+t* are invalid pairs; *absen-ce* and *absen-t* and valid; *mistake+NULL* and *mistake+n* are valid.

To overcome the problems faced in cases 2, 3 and 4 we have proposed the following framework

5 Unsupervised Morphology Learner Framework

UML can be used to learn derivational morphology or inflectional morphology. When the input given is a lexicon, the framework will learn derivational morphology. If a corpus is used as input it will learn both derivational and inflectional morphology and not distinguish between the two. We have tested our framework with lexicon as input to learn derivational morphology. The framework for the proposed UML is shown below in Figure 1. UML has five modules. It uses a lexicon resource or a corpus as input. It generates three final resources and two intermediate resources which are enhanced into the final resources.

The resource used as input could be:

- Lexicon L: It is list of dictionary words found in the language. This resource is generated from a WordNet of a language used to learn derivational morphology or
- Corpus C: A collection of un-annotated text used to learn both inflectional and derivational morphology.

The intermediate resource generated:

- *Candidate Stem-Suffix List*: It is the initial list of stems and suffixes identified for an input language using the p-similar technique. It consists of two sets namely set of suffix S_{suffix} and set of stem S_{stem}. Sample entries in these set for English language are S_{suffix} = *{ er, ic, ly, ness, ment, …}* and S_{stem}= *{adorn, attack,….}*
- *Initial Paradigms:* This is a list of all stems with information of which suffixes combine with which stems in the input lexicon L or Corpus. Sample entry in *Initial Paradigms List* is *ic.y= academ + allerg + geometr + homeopath + horrif + letharg + majest + prehistor + specif + strateg* where *"ic"* and *"y"* are suffixes which combine with the stems like *adadem*.

The final resources generated:

- *Stem-Suffix List:* This resource is generated from the *Candidate Stem-Suffix List* resource by pruning invalid suffixes. It is a useful resource as it gives the stems of words from a lexicon which could later be used for identifying stems in a corpus for stemming inflectional words.
- *Suffix-Association Matrix:* This resource helps us identify for how many instances a suffix s_1 has occurred with a suffix s_2 in the Lexicon/Corpus. It is a crucial resource in eliminating the shortcoming of p-similar technique to morphologically segment words with short stem length as well as overcome chance association of suffix found.
- *Morphology Paradigms:* This resource contains paradigms extracted from the words found in the input lexicon/corpus. It is a refined version of *Initial Paradigm* resource.

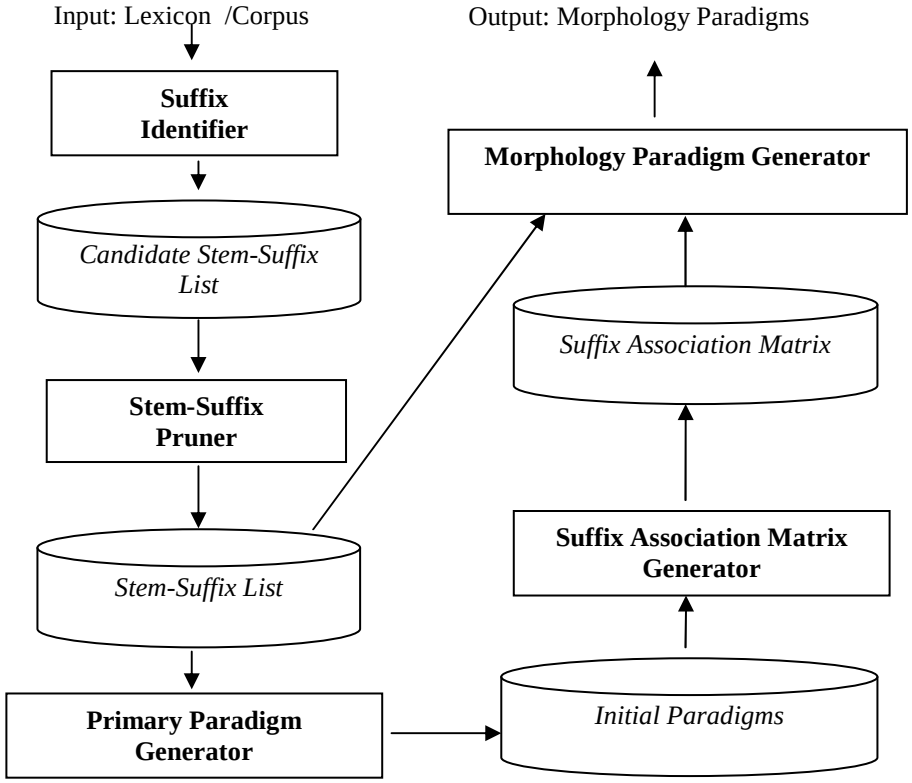

Figure 1: Unsupervised Morphology Learner (UML) Framework

UML comprises of five main modules, a brief description and algorithm for each of the module is given below:

Module 1 - Suffix Identifier

Description: Identifies the Candidate suffixes using p-similar technique. It generates a temporary resource namely *Candidate Stem-Suffix List*. For every word in the corpus, it checks if there is another word with a common stem, adds common stem to stem list and rest to suffix list, provided that a stem occurs with more than one suffix and a suffix occurs with more than one stem.

Input: Lexicon of the language L (or raw un-annotated corpus for inflectional morphology C)
Output: *Candidate Stem-Suffix List* resource
Algorithm:

For each input word $p \in L$,
find q, r, $s \in L$, such that $\exists\ b_1, b_2, b_3$
where $p=b_1s_1$, $q=b_1s_2$, $r=b_2s_1$, $s= b_2s_3$ where $b_1, b_2, b_3, s_1, s_2, s_3 \in \sum^*$.
Add b_1 to set of stems S_{stem},
Add s_1 to set of suffixes S_{suffix},
EndFor

Module 2 - Stem-suffix pruner:
Description: This module applies heuristic $\mathbf{H_1}$ stated below. $\mathbf{H_1}$ is framed to correct the stem-suffix list to fix the problem of over-stemming.
H_1: Given suffix s_i for stem b_i if $\exists\ a \in \sum^*$ such that $as_i \in S_{suffix}$ and $b_ja=b_i$ and $b_j \in S_{stem}$ where S_{stem} is set of stems and S_{suffix} is set of suffixes then replace b_i by b_j and s_i by as_i
Input: *Candidate Stem-Suffix List* resource
Output: *Stem-Suffix List* resource
Algorithm:

For each suffix s_1 from suffix list,
If $\exists\ a \in \sum^*$ such that $as_1 \in S_{suffix}$ and $b_2a=b_1$ and $b_1, b_2 \in S_{stem}$ then
replace b_1 by b_2 and s_1 by as_1.
EndIf
EndFor

Module 3 - Primary Paradigm Generator:
Description: Using *Stem-Suffix List* this module generates the *Initial Paradigms* list. A paradigm is composed of suffixes that go together for a list of stems in the input lexicon/corpus.
Input: *Stem-Suffix List* resource
Output: *Initial Paradigms* resource
Algorithm:

For each input word $p \in L$, if $p=b_1s_1$ where $b_1 \in S_{stem}$ and $s_1 \in S_{suffix}$.
Set paradigm-string= s_1.
For every $q \in L$ such that $q= b_1s_2$ where $b_1 \in S_{stem}$ and $s_2 \in S_{suffix}$,
Set paradigm-string = paradigm-string.s_2.
Add paradigm-string to $S_{paradigm}$, set of paradigm.
EndFor
EndFor
For each paradigm-string $p_1 \in S_{paradigm}$ where $p_1 =$"$s_{x1}.s_{x2} \ldots s_{xn}=b_1$"
and $s_{x1}, s_{x2}, \ldots, s_{xn} \in S_{suffix}$ and $b_1 \in S_{stem}$
Set collapse-paradigm-string = $s_{x1}.s_{x2} \ldots s_{xn}=b_1$
If $\exists$ paradigm-string $p_2 \in S_{paradigm}$ such that $p_2 =$" $s_{x1}.s_{x2} \ldots s_{xn} =b_2$" and $b_2 \in S_{stem}$
Set collapse-paradigm-string = collapse-paradigm-string + b_2
Add collapse-paradigm-string to $S_{initial\text{-}paradigm}$, set of Initial Paradigms
EndIf
EndFor

Module 4- Suffix Association Matrix Generator:
Description: From the *Initial Paradigms*, this module computes the *Suffix Association Matrix* resource. Suffix association matrix is a square matrix where each row and column corresponds to a suffix in suffix list. An entry in this matrix gives how many times a particular suffix occurs with another suffix in the *Initial Paradigms* resource.
Input: *Initial Paradigms* resource
Output: *Suffix Association Matrix* resource

Algorithm:

Let M be suffix association matrix which is $|S_{suffix}| * |S_{suffix}|$. If $S_{suffix} = \{s_1, s_2, \ldots..s_p\}$ M has dimension p X p.
Initialize M=0;
For each paradigm-string $p_1 \in S_{initial\text{-}paradigm}$ where $p_1 = "s_{x1}.s_{x2} \ldots s_{xn}=b_1+ b_2+ b_3+\ldots+ b_m"$
    For i= 1 to n
        For j= i+1 to n
            $M[s_{xi}][s_{xj}]= M[s_{xi}][s_{xj}] + m$; where $s_{xi} = s_q$ and $s_{xi} = s_r$ and $1<= q, r <=p$
        EndFor
    EndFor
EndFor

Module 5 - Morphology Paradigm Generator:

Description: Using *Stem-Suffix List* and Suffix Association Matrix this module generates *Morphology Paradigms List* resource. It is a pruned version of *Initial Paradigms* resource which uses Suffix Association matrix to remove less likely suffix combination in *Initial Paradigms*

Input: *Stem-Suffix List* resource

Output: *Initial Paradigms* resource

Algorithm:

For each input word $p \in L$, if $p=b_1s_1$ where $b_1 \in S_{stem}$ and $s_1 \in S_{suffix}$.
    Set paradigm-string= s_1.
    For every $q \in L$ such that $q= b_1s_2$ where $b_1 \in S_{stem}$ and $s_2 \in S_{suffix}$,
        If $M[s_1][s_2]$ > threshold value
            Set paradigm-string = paradigm-string.s_2.
            Add paradigm-string to $S_{paradigm}$, set of paradigm.
        EndIf
    EndFor
EndFor
For each paradigm-string $p_1 \in S_{paradigm}$ where $p_1 = "s_{x1}.s_{x2} \ldots s_{xn}=b_1"$
                and $s_{x1}, s_{x2}, \ldots, s_{xn} \in S_{suffix}$ and $b_1 \in S_{stem}$
    Set collapse-paradigm-string = $s_{x1}.s_{x2} \ldots s_{xn}=b_1$
    If $\exists$ paradigm-string $p_2 \in S_{paradigm}$ such that $p_2 =" s_{x1}.s_{x2} \ldots s_{xn} =b_2"$ and $b_2 \in S_{stem}$
        Set collapse-paradigm-string = collapse-paradigm-string + b_2
        Add collapse-paradigm-string to $S_{initial\text{-}paradigm}$, set of Initial_Paradigms
    EndIf
EndFor

5.1 Significance of Suffix Association Matrix

Suffix association matrix is a measure of how many times a particular suffix is associated with another suffix in the input resource. It is an important contribution as it provides us an alternate way to prune invalid stem-suffix pairs identified, rather than a restriction on the stem-length. Suffixes which are associated with each other more frequently are more likely to provide a correct paradigm than those where we find only a few chance instances of suffix associations.
Figure 2 illustrates an instance of suffix association matrix for the English language

	NULL	er	ing	ly
NULL	-	46	225	129
er	46	-	22	15
ing	225	22	-	0
ly	129	15	0	-

Figure 2: Instance of Suffix Association Matrix

This matrix helps handle valid stem with invalid suffix case. For instance wrong segmentation of the word *"bother"* as *"both+er"*. From the Suffix Association Matrix we check with which

suffixes *er* is commonly associated. We then make a list of words with stem *"both"* and other suffix which commonly associate with suffix *"er"* like suffix *"ing"* We search a corpus for existence of such words like *"bothing"*. Thus rejecting the segmentation *bother=both+er*. This matrix also provides a solution to invalid stem with valid suffix. For instance *canno+n* and *canno+t* are invalid segmentations although the suffix *"n"* and *"t"* are valid in some other context. In such a rare association of a suffix *"n"* and *"t"* the corresponding entry in the suffix association matrix is found to be very low. We ran our algorithm for various values of threshold and found five as an optimal value. Any suffix association below five were pruned as chance associations.

5.2 Significance of heuristic H_1

This heuristic is used to handle the problem of over-stemming that occurs in p-similar technique. For example the p-similar technique identifies both *"ion"* and *"on"* as suffix. While segmenting a word like *"addiction"* we need to decide if *"addicti+on"* or *"addict+ion"* is correct. H_1 helps us in correctly segmenting the word as *"addict+ion"*.

5.3 Limitations of UML

UML is restricted to identify concatenative morphology paradigms only. Presently it identifies suffixes only and does not support irregular morphology wherein the stem undergoes a change before suffixation.

6 Experimental Results

The implementation of UML is done in Java. After applying our method, the paradigms obtained were compared to the paradigms obtained using p-similar method with minimum stem-size five. The precision was computed as ratio of number of words correctly segmented to total number of words segmented. Recall is computed as ratio of number of words correctly segmented to number of words in given input which could be segmented. The results have been tabulated in Table 1 below.

Method	**Number of Paradigms**	**Recall**	**Precision**	**F-Score**
Language : *English*				
Data Set: English lexicon with 21813 entries was obtained from the English WordNet[1]				
p-similar with stems size >5	**1163**	**0.85**	**0.93**	**0.89**
UML for derivational morphology	**413**	**0.92**	**0.93**	**0.92**
Language : *Hindi*				
Data Set: Hindi lexicon with 23807 entries was extracted from the Hindi WordNet[2]				
p-similar with stems size >5	**1127**	**0.83**	**0.87**	**0.85**
UML for derivational morphology	**332**	**0.87**	**0.94**	**0.90**
Language : *Konkani*				
Data Set: Konkani lexicon with 25838 entries was extracted from the Konkani WordNet[3]				
p-similar with stems size >5	**1088**	**0.75**	**0.77**	**0.75**
UML for derivational morphology	**274**	**0.87**	**0.87**	**0.87**

Table 1: Results for English, Hindi and Konkani Language

[1] http://wordnet.princeton.edu/wordnet/download/
[2] http://www.cfilt.iitb.ac.in/wordnet/webhwn/
[3] http://konkaniwordnet.unigoa.ac.in

6.1 Effect of stem length on recall

We list below in Table 2, a few examples of how recall is reduced as words with short stem length are not segmented, when the minimum stem size is five.

Language	Suffix for which word not segmented	Number of words not segmented	Few examples of words not segmented
English	er	9	eater, farmer, owner...
Hindi	ी[4] (I;;Hindi suffix)	35	अरबी (arabic; arab; name of a country), आलसी (aalas; lazy;), आसानी (aasani; easiness;)
Konkani	ी (I;;Konkani suffix)	43	आनंदी (anandi; being happy;), आरोपी (aaropi; accused;)

Table 2: Effect of stem length

We observe that number of words not segmented in English is relatively very less as compared to the Indian languages Hindi and Konkani. Thus the restriction on stem-length works efficiently for English as compared to the Indian languages Hindi and Konkani.

6.2 Effect of stem length on precision

When we restrict the stem-length to five we observe that some wrong segmentation of words are pruned. Listed below in Table 3, are some examples

Language	Suffix for which word not segmented	Number of words not segmented	Few examples of words not segmented (wrongly)
English	er	32	bother, boxer, cater, sober …
Hindi	ी (I;;Hindi suffix)	8	चाँदी (chandi; silver;), चोटी (choti; peak;)
Konkani	ी (I;;Konkani suffix)	6	आजी (Aaji; grandmother;), काळी (kaalli; black;)

Table 3: Effect of stem-length on precision

We observe that for English, many word segmentations with stems-length less than five, identified by p-similar technique are correctly pruned by applying the restriction. We observe that wrong segmentations in case of Indian languages Hindi and Konkani are less when compared to English.

7 Conclusion

Unsupervised Morphology Learner framework thus can be effectively used to generate paradigms for Indian languages which have low frequency suffixes and words with short stem lengths. Suffix Association Matrix and heuristics H_1 is advantageous over p-similar technique with stem length restriction for languages like Konkani and Hindi which have many short length valid stems. The derivational suffixes obtained from UML with Lexicon as input can be used to distinguish from inflectional morphology suffixes when the framework is used with a corpus as input.

[4] A word in Indian language is followed by transliteration in Roman Script, translation in English and gloss in brackets

Reference

Bacchin, M., Ferro, N., and Melucci, M. (2005). A probabilistic model for stemmer generation. Information Processing and Management, 41(1):121–137.

Beesley K & Karttunen Lauri. 2003. Finite State Morphology. Stanford, CA: CSLI Publications.

Chan, E. 2008. Structures and Distributions in Morphology Learning. Ph.D thesis, University of Pennsylvania.

Dreyer, M. 2011. A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings. Ph.D thesis, The Johns Hopkins University, Baltimore, Maryland

Freitag, D. 2005. Morphology induction from term clusters. In Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), pages 128–135, Ann Arbor, Michigan. Association for Computational Linguistics.

Gaussier Eric. 1999. *Unsupervised learning of derivational morphology from inflectional lexicons.* In ACL'99 Workshop Proceedings: Unsupervised Learning in Natural Language Processing : 24–30 ACL

Gelbukh, A. F., Alexandrov, M., and Han, S.-Y. (2004). Detecting inflection patterns in natural language by minimization of morphological model. In Sanfeliu, A., Trinidad, J. F. M., and Carrasco-Ochoa, J. A., editors, Proceedings of Progress in Pattern Recognition, Image Analysis and Applications, 9th Iberoamerican Congress on Pattern Recognition, CIARP '04, volume 3287 of Lecture Notes in Computer Science, pages 432–438. Springer-Verlag, Berlin.

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

Hammarstrom Harald and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, (2):309–350.

Koskenniemi, K. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. Helsinki, Department of General Linguistics, University of Helsinki.

Koskenniemi, K. 1996. *Finite-state morphology and information retrieval*. Proceedings of the ECAI-96 Workshop on Extended Finite State Models of Language ECAI, Budapest, Hungary : 42-56

Lindén, K. 2008. A probabilistic model for guessing base forms of new words by analogy. In Proceedings of CICLing-2008: 9th International Conference on Intelligent Text Processing and Computational Linguistics, volume 4919 of Lecture Notes in Computer Science, pages 106–116. Springer.

Lindén, K. and Tuovila, J. 2009 Corpus-based Paradigm Selection for Morphological Entries. In Proceedings of NODALIDA 2009, Odense, Denmark, May 2009

Loftsson, H. 2008. Tagging Icelandic text: A linguistic rule-based approach. Nordic Journal of Linguistics 31(1). 47–72.

Lovins J. B. 1968. *Development of a stemming algorithm*. Mechanical Translation and Computer Linguistic., vol.11, no.1/2: 22-31.

Maung, Zin Maung & Yoshiki Mikami. 2008. A rule-based syllable segmentation of myanmar text. In Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, 51–58. Hyderabad, India: Asian Federation of Natural Language Processing.

Paice, C.D. 1990. *Another stemmer*. SIGIR Forum, 24: 56-61

Porter, M. F. 1980. *An algorithm for suffix stripping.* Program 14 : 130-7.

Sharma U, (2006). Unsupervised Learning of Morphology of a Highly Inflectional Language, Ph.D. thesis, Tezpur University, Assam, India