Optimal Stem Identification in Presence of Suffix List

Vasudevan N and Pushpak Bhattacharyya

Computer Science and Engg Department
IIT Bombay, Mumbai
vasudevan@cse.iitb.ac.in, pb@cse.iitb.ac.in

Abstract Stemming is considered crucial in many NLP and IR applications. In the absence of any linguistic information, stemming is a challenging task. Stemming of words using suffixes of a language as linguistic information is in comparison an easier problem. In this work we considered stemming as a process of obtaining minimum number of lexicon from an unannotated corpus by using a suffix set. We proved that the exact lexicon reduction problem is NP-hard and came up with a polynomial time approximation. One probabilistic model that minimizes the stem distributional entropy is also proposed for stemming. Performances of these models are analyzed using an unannotated corpus and a suffix set of Malayalam, a morphologically rich language of India belonging to the Dravidian family.

1 Introduction

Stemming is a crucial component in most of the NLP applications. Since the stemming identifies the same stem for all inflectional variants of a lexeme, it will improve the performance of information retrieval systems. Inflectional suffix in a word carries its morphosyntactic information and paradigm information. For example, by stemming the word **boys**, we will get the suffix **s** from that word. This suffix carries the information about its plurality. Such information is essential for many NLP applications like machine translation. This indicates the importance of building good stemmer for languages.

Building a morphology analyser or a stemmer is always a challenging task for all languages. This is more challenging for inflectional, agglutinative and isolated languages. Lot of linguistic expertise is needed for building such tools. This is again difficult for those languages which have no linguistic tradition. The linguistic expertise needed for stemming may not be available for such languages. So they have to rely on word forms in a corpus and their processing. Therefore building a low cost automatic stemmer from a corpus for such languages has a great significance.

Building a stemmer using only an unannotated corpus is the most inexpensive feasible approach. Since there is no availability of direct linguistic information and supervision, such a system with good performance is very difficult to build.

Knowledge of inflectional suffixes in a language will reduce the level of difficulty. I.e. the performance of stemming using only an unannotated corpus can be improved by using a set of inflectional suffixes. Usually a language has a small closed set of inflectional suffixes. Identification of this suffix set of a language is a relatively easier job. So we consider a semi-supervised scenario where the suffix list is given. In this work we are trying to build a stemmer using an unannotated corpus with sufficiently large number of distinct words and a set of inflectional suffixes. Sample sets of inflectional suffixes in English and Malayalam are shown below (Example 1,2)

Example 1. English: { **s**, **es**, **ed**, **ing**, . . . }

Example 2. Malayalam: { കൾ {kal} (Plural), കളെ {kale} (Plural + Accusative), കളുടെ {kalude} (Plural + Genitive), മാർ {mar} (Plural), മാരെ {mare} (Plural + Accusative), മാരുടെ {marude} (Plural + Genitive), ുക {uka} (Present), ക്കുക {kkuka} (Present), ി {i} (Past), ും {um} (Future), . . . }

Inputs of this stemming problem are an unannotated corpus (*Corpus*) and a suffix set ($Suffixset$), and the output is the stem of each word. A small input and its output is shown in Example 3.

Example 3. Input: *Corpus*= {**mosses**, **moss**, **boys**, **boy**}, $Suffixset$= {**es**, **s**, ϕ}, where ϕ is the null suffix.

Output: {**moss** (**mosses**), **moss** (**moss**), **boy** (**boys**), **boy** (**boy**)}

We make an assumption that $Suffixset$ includes all orthographic variations of all valid suffixes in the language. കൾ {kal}, ക്കൾ {kkal} and ങ്ങൾ {ngngal} are the variants of Malayalam[1] plural marker കൾ {kal}. So a Malayalam $Suffixset$ contains all these variants. Some languages have only one possible suffix in a word, while others have more. In such languages a word has the structure $stem.suffix_1.suffix_2 \ldots suffix_x$. For example, a Malayalam word കുട്ടികളുടെ {kuttikalude} (child+Plural+Genitive) contains the stem കുട്ടി {kutti} (child), a plural marker കൾ {kal} and a genitive case marker ുടെ {ude}. Concatenated sequence of suffixes, $suffix_1.suffix_2 \ldots suffix_x$ is considered as a suffix in such words. So $Suffixset$ of such a language contains some concatenations of suffixes also. In the previous example of Malayalam word കുട്ടികളുടെ {kuttikalude}, കളുടെ {kalude} is actually a concatenation of suffixes കൾ {kal} and ുടെ {ude}. But we consider this as a single suffix. We take another assumption that the *Corpus* contains sufficiently large number of words.

To define the stemming problem with suffix set, let us introduce a term *possible_stem_set*. *possible_stem_set* of a word w be the set of all prefixes of w such that w can be expressed as the concatenation of that prefix and a suffix from $Suffixset$.

$possible\_stem\_set(w) = \{st : \exists x \in Suffixset \text{ such that } w = st.x \}$

Consider the input shown in Example 3. *possible_stem_set* of each word in the *Corpus* is shown in the Table 1. Now formally we can say, stemming is the

Table 1: *possible_stem_set* and their Correct Stem

word	***possible_stem_set***	**correct stem**
mosses	{*mosses*, *mosse*, *moss*}	*moss*
moss	{*moss*, *mos*}	*moss*
boys	{*boy*, *boys*}	*boy*
boy	{*boy*}	*boy*

process of identifying $stem(w)$ optimally for all word w, where $stem(w)$ is an element of $possible\_stem\_set(w)$.

The stemming process is the selection of correct entry from *possible_stem_set* of each word in $Corpus$. In the previous example, the stemmer needs to select the stems **moss** from *possible_stem_set(**mosses**)*, **moss** from *possible_stem_set(**moss**)*, **boy** from *possible_stem_set(**boys**)* and **boy** from *possible_stem_set(**boy**)*.

The roadmap of the paper is as follows. Related work of the stemming problem is briefly summarized in section 2. We propose two models for stemming problem in this work. In section 3, we propose a deterministic model that reduces the number of distinct stems. In section 4, we propose a probabilistic model that learns the distribution by reducing the entropy. A case study of these models in one of the morphologically rich language Malayalam is included in section 5. Performances of these models are measured in this section by using a wordlist and a suffix set.

2 Related work

Morphology learning is one of the widely attempted problem in the literature. A recent survey paper by Harald Hammarstrom [1] gives an overall view on unsupervised morphology learning. There are lots of probabilistic approaches for morphology learning. Linguistica model [2], maximum a posteriori model [3], stochastic transducer based model [4] and generative probabilistic model [5] are the relevant probabilistic models for stemming that we found in the current literature. A Markov Random Field by Dreyer [6] is also a useful work related to unsupervised morphology.

Graph based model [7], lazy learning based model [8], clustering based same stem identification model [9, 10] ParaMor system for paradigm learning [11] are also relevant works in the same area. Full morpheme segmentation and automatic induction of orthographic rules by Sajib Dasgupta [12, 13] is also a relevant work.

We never found any model which use the information from suffix list. To the best of our knowledge, this is the first attempt for stemming in presence of suffix list. We found that Linguistica model is the closely related approach to ours. Frequency of stem and suffix candidates plays a crucial role in Linguistica model. Linguistica model is an optimal stem identification model by reducing

[1] A morphologically rich language of India belonging to the Dravidian family.

Table 2: Examples of *valid_mapping*

valid_mapping **(f:word → stem)**	**range(f)**	**|range(f)|**
{(*mosses* → *mosses*), (*moss* → *moss*), (*boys* → *boys*), (*boy* → *boy*)}	{*mosses*, *moss*, *boys*, *boy*}	4
{(*mosses* → *mosses*), (*moss* → *moss*), (*boys* → *boy*), (*boy* → *boy*)}	{*mosses*, *moss*, *boy*}	3
{(*mosses* → *mosses*), (*moss* → *mos*), (*boys* → *boys*), (*boy* → *boy*)}	{*mosses*, *mos*, *boys*, *boy*}	4
{(*mosses* → *mosses*), (*moss* → *mos*), (*boys* → *boy*), (*boy* → *boy*)}	{*mosses*, *mos*, *boy*}	3
{(*mosses* → *mosse*), (*moss* → *moss*), (*boys* → *boys*), (*boy* → *boy*)}	{*mosse*, *moss*, *boys*, *boy*}	4
{(*mosses* → *mosse*), (*moss* → *moss*), (*boys* → *boy*), (*boy* → *boy*)}	{*mosse*, *moss*, *boy*}	3
{(*mosses* → *mosse*), (*moss* → *mos*), (*boys* → *boys*), (*boy* → *boy*)}	{*mosse*, *mos*, *boys*, *boy*}	4
{(*mosses* → *mosse*), (*moss* → *mos*), (*boys* → *boy*), (*boy* → *boy*)}	{*mosse*, *mos*, *boy*}	3
{(*mosses* → *moss*), (*moss* → *moss*), (*boys* → *boys*), (*boy* → *boy*)}	{*moss*, *boys*, *boy*}	3
{(*mosses* → *moss*), (*moss* → *moss*), (*boys* → *boy*), (*boy* → *boy*)}	{*moss*, *boy*}	**2**
{(*mosses* → *moss*), (*moss* → *mos*), (*boys* → *boys*), (*boy* → *boy*)}	{*moss*, *mos*, *boys*, *boy*}	4
{(*mosses* → *moss*), (*moss* → *mos*), (*boys* → *boy*), (*boy* → *boy*)}	{*moss*, *mos*, *boy*}	3

the description length. In this work, we minimizes the number of distinct stems and entropy of stem distribution.

3 Minimum Stem Range (MSR) model

Given the suffix set, stemming can be viewed as a process of reduction of lexicon entry. Minimum Stem Range (MSR) model is a direct and intuitive translation of this aspect of stemming problem to a well-defined computational model. MSR model finds out a mapping from each word to one of the string in its *possible_stem_set*. If suffix set is complete then actual stem of each word should be in *possible_stem_set* of that word. So, if *possible_stem_set* of a word contains exactly one entry then any mapping identifies the actual stem of that word. In the Table 1, *possible_stem_set* of **boy** contains only one element **boy**. So any mapping to *possible_stem_set* choose the correct stem **boy** from *possible_stem_set(**boy**)*. Otherwise the actual stem needs to be identified by using the information from other words.

3.1 Model

MSR model is a computational model for the stemming problem. This model finds a mapping from each word in input corpus to its stem (a starting substring of the word). A mapping function f from each word in input corpus to its stem (a starting substring of the word) is called as a *valid_mapping* if and only if each word in input corpus is gets mapped to one of the stems in its *possible_stem_set*. All *valid_mappings* from Example 3 are shown in Table 2 (*possible_stem_sets* of this sample corpus is shown in Table 1). A mapping (**mosse**, **mos**, **boys**, **boy**) means, first word **mosses** is gets mapped to **mosse**, second word **moss** is gets mapped to **mos**, **boys** is gets mapped to **boys** and **boy** is gets mapped to **boy**.

MSR model will find out a *valid_mapping* with minimum range. Lets define the MSR model formally.

Input: *Corpus* (a sufficiently large list of plain words) and *Suffixset* (set of all suffixes)

Output: A *valid_mapping*, f^* with minimum cardinality of range,

$$f^* = \underset{f \in valid\_mapping}{argmin} \left\{ |range(f)| \right\}$$

Out of these 12 *valid_mappings* shown in Table 2, (*moss*, *moss*, *boy*, *boy*) have the minimum range. So the MSR model will identify this mapping.

Theorem 1. *If a language does not have any morphological ambiguity and MSR model identifies a valid stem for at least one word from all group of words with same stem then MSR model will identify the correct stem from all words.*

Proof. Lets assume there is no morphological ambiguity. So there will be exactly one split that generates a valid stem and a valid suffix. If one of the valid stem is in *possible_stem_set* of a word then that will be the correct stem of that word. Otherwise there will be more than one way to split that word and that will be a morphological ambiguity. So if MSR problem identifies a valid stem from a word then that will be the correct stem of that word.

Suppose there are m groups of words with the same stem. There exist a *valid_mapping* with range of size exactly m by correctly identifying all correct stems. So the range of output of MSR model will be less than or equal to m.

Let f be the output of MSR model that identifies correct stem from at least one word from all m groups. So, all valid stems from all words in input corpus will be in the range of f. Since there exist exactly one valid stem in each *possible_stem_set*, any incorrect *valid_mapping* produces an invalid stem. So if there is any incorrect mapping, then the range of f will be greater than m. If there is any incorrect mapping in f then f will not be the output of MSR model. Therefore the MSR model will identify the correct stem from all words.

Theorem 2. *Problem of computing MSR model (MSR problem) is NP-Hard*

Proof. To prove the MSR problem is NP-hard, we want to find a polynomial time reduction from an existing NP-hard problem to MSR problem. Let us take minimum vertex cover problem, a known NP-hard problem for the reduction. Input of minimum vertex cover problem is an undirected graph $G = (V, E)$ and output is a set of vertices (vertex cover) VC with minimum cardinality, where $VC \subseteq V$ and for every edge e_{ij} in E, either $v_i \in VC$ or $v_j \in VC$. Given an instance of vertex cover problem $(G = (V, E))$, we construct an instance to MSR problem $(Corpus, Suffixset)$ as follows.

Suppose $G = (V, E)$, $V = \{v_i\}$, $E = \{e_{ij}\}$ $1 \le i, j \le n$ be the input of vertex cover problem. For every edge e_{ij}, without loosing generality we can say $i \le j$. For each edge e_{ij} add a word $c_1.c_2.\ldots.c_j.m_{ij}$ in to the *Corpus*. Here $c_1, c_2, \ldots c_j$ and m_{ij} can be any distinct characters. For each edge e_{ij} add $c_{i+1} \ldots c_j.m_{ij}$ and m_{ij} in to the *Suffixset* (if $i = j$ then add only m_{ij} in to the *Suffixset*).

Consider a small graph shown below. Corresponds to four edges (e_{12},e_{13},e_{23},e_{34}) add four words to the *Corpus*. Similarly corresponds to four edges add eight suffixes to the *Suffixset*. These words and suffixes are shown in the second and third columns of Table 3.

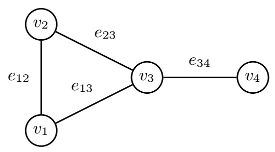

Figure 1: Example Graph to Elucidate Reduction Procedure

Table 3: Reduction Example of the Graph

	word list	suffix list	possible_stem_set	f	VC
e_{12}	$c_1c_2m_{12}$	c_2m_{12}, m_{12}	$\{c_1, c_1c_2\}$	c_1	v_1
e_{13}	$c_1c_2c_3m_{13}$	$c_2c_3m_{13}$, m_{13}	$\{c_1,c_1c_2c_3\}$	c_1	v_1
e_{23}	$c_1c_2c_3m_{23}$	c_3m_{23}, m_{23}	$\{ c_1c_2, c_1c_2c_3\}$	$c_1c_2c_3$	v_3
e_{34}	$c_1c_2c_3c_4m_{34}$	c_4m_{34}, m_{34}	$\{ c_1c_2c_3, c_1c_2c_3c_4\}$	$c_1c_2c_3$	v_3

Let f be the output of MSR problem and R is the range of f. Since all c_i and m_{ij} are distinct, *possible_stem_set*$(c_1 \ldots c_j.m_{ij})$ will be $\{c_1 \ldots c_j, c_1 \ldots c_i\}$. See the *possible_stem_set* of each word in the previous example shown in Table 3. So each string in R will be in the form of $c_1 \ldots c_k$ for some k. For each such string, add the vertex v_k in to VC. Here the cardinality of R and VC will be same. In the example the corresponding vertex cover (VC) of f function is a minimal vertex cover ($\{v_1, v_3\}$).

To prove the correctness of reduction we want to show the solution of MSR problem will be same as the solution of minimum vertex cover problem.

Let say f^* is the solution of MSR problem and VC^* is the corresponding output instance of minimum vertex cover problem. For each word $c_1 \ldots c_j.m_{ij}$ either $c_1 \ldots c_j$ or $c_1 \ldots c_i$ will be in the range of any *valid_mapping*. Therefore VC^* will be a valid vertex cover and that can be the solution of minimum vertex cover problem. Suppose VC^* is not the solution of minimum vertex cover problem, then there exist another valid vertex cover VC', such that $|VC'| < |VC^*|$. Then for VC' we can define a *valid_mapping*, f',

$$f'(c_1c_2 \ldots c_jm_{ij}) = \begin{cases} c_1c_2 \ldots c_i \text{ if } v_i \text{ is in } VC' \\ c_1c_2 \ldots c_j \text{ otherwise} \end{cases}$$

Here we can see $|range(f')| \leq |VC'|$

$\Rightarrow |range(f')| < |VC^*|$

$\Rightarrow |range(f')| < |range(f^*)|$

By the definition of f^* this is a contradiction. Therefore VC^* always will be the solution of minimum vertex cover problem.

Let VC^* is the solution of minimum vertex cover problem. Now we can define a corresponding *valid_mapping* f^* for VC^*,

$$f^*(c_1c_2 \ldots c_jm_{ij}) = \begin{cases} c_1c_2 \ldots c_i \text{ if } v_i \text{ is in } VC^* \\ c_1c_2 \ldots c_j \text{ otherwise} \end{cases}$$

Here $|VC^*| \geq |range(f^*)|$. If $|VC^*| > |range(f^*)|$ then there exist another vertex cover with less cardinality. But VC^* is the minimum vertex cover. So $|range(f^*)|$ will be same as $|VC^*|$.

Suppose f^* is not the solution of MSR problem, then there will be another *valid_mapping* f' such that $|range(f')| < |range(f^*)|$. Then there will be a valid vertex cover VC' corresponds to f', s.t. $|range(f')| = |VC'|$

$\Rightarrow |VC'| < |range(f^*)|$

$\Rightarrow |VC'| < |VC^*|$

But VC^* is the minimum vertex cover. So such a valid vertex cover does not exist. Therefore f^* will be a solution of MSR problem.

3.2 Approximation

Since the MSR problem is NP-hard, it is difficult to find stems from a large data. In order to solve the stemming problem computationally, an approximation of the above model is required. Similarity of this model to set cover problem can be utilized to find an approximation algorithm. MSR problem can be reduced to set cover problem. Input of set cover problem is a set U and a set of subsets (S) of U such that, $\bigcup_{s \in S} s = U$. Output of set cover problem is a subset of S, say C, such that $\bigcup_{s \in C} s = U$.

Let St be the set of all possible stems, i.e. $St = \bigcup_{w \in Corpus} possible\_stem\_set(w)$. Let $W(st)$ is the set of words in *Corpus* where *st* is an element of its *possible_stem_set*, i.e. $W(st) = \{w : st \in possible\_stem\_set(w)\}$. Take the sample input corpus {**mosses**, **moss**, **boys**, **boy**} and input suffix set {**es**, **s**, ϕ}. Set of all possible stems and its corresponding $W(st)$ are shown in the Table 4. Now consider the *Corpus* as U and the set $\{W(st)\}$ as S of set cover problem. Here $\bigcup_{st \in St} W(st) = Corpus$. The output of this set cover problem is a set of stems St' such that, for any word in *Corpus*, there exist at least one possible stem of that word in St'. By choosing any one possible stem of each word from St', we can find a solution of MSR problem.

Table 4: *Possible Stems and their W() Set*

Stem	*mosses*	*mosse*	*moss*	*mos*	*boys*	*boy*
W(stem) **(S)**	{*mosses*}	{*mosses, moss*}	{*moss*}	{*moss*}	{*boys*}	{*boys, boy*}

Any approximation algorithm of set cover problem can be directly used for this stemming problem. Best-known approximation of set cover problem is the greedy algorithm. The greedy algorithm choose subsets one by one from S that have maximum number of uncovered elements in U, and cover those uncovered elements. The complexity of approximation algorithm is $O(NM)$ and approximation factor is $log(N)$, where N is the number of words in corpus and M is the number of suffixes in suffix set.

4 Minimum Stem Entropy (MSE) model

The MSR model and its approximation are deterministic approaches for the stemming problem. Greedy approximation of the above model may not find optimum mapping. Here we propose a new model, Minimum Stem Entropy (MSE) model, which is a probabilistic approach for the stemming problem.

4.1 Model

The probabilistic model assumes an uncertainty to choose the correct stem from *possible_stem_set* of each word. Input of MSE model is same as that of MSR model. Output of this model is a conditional probability distribution of stem given word ($Pr(st|w)$). $Pr(st|w)$ is the probability that string st (from possible stem set of w, $PSS(w)$) is the stem of word w. From the $Pr(st|w)$ we can select the maximum probable stem candidate as the correct stem of that word, i.e. stem of a word w, $Stem(w) = argmax_{st}\{Pr(st|w)\}$. Two basic conditions for $Pr(st|w)$ are,

$Pr(st|w) \geq 0$ for all st in possible stem set of w

$$\sum_{st \in PSS(w)} \{Pr(st|w)\} = 1$$

Many such probability distributions are possible. Most suitable distribution based on input corpus needs to be learned. The MSE model selects the distribution that minimizes the entropy of stem distribution. A stem distribution $Pr(st)$ is the probability of st is to be identified as correct stem of a randomly chosen word. We can write the stem distribution in terms of $Pr(st|w)$ as,

$$Pr(st) = \sum_{w \in corpus} \{Pr(w) \times Pr(st|w)\}$$

The MSE problem can be written as an optimization problem. I.e.

$$Pr(st|w) = \underset{Pr(st|w)}{argmin}\Big\{ Entropy(Pr(st)) \Big\}$$

$$= \underset{Pr(st|w)}{argmin}\Big\{ Entropy\Big(\sum_{w \in corpus} \{Pr(w) \times Pr(st|w)\}\Big)\Big\}$$

$$= \underset{Pr(st|w)}{argmin}\Big\{ -\sum_{st} \Big\{ \sum_{w}\{Pr(st|w) \times Pr(w)\} \times log\Big(\sum_{w}\{Pr(st|w) \times Pr(w)\}\Big)\Big\}\Big\}$$

MSE learns the probability of possible stem of each word that minimize the entropy of complete stem distribution. A stem distribution with low entropy has a tendency for a small stem set. So this model has a similarity with MSR model. With minimum stem distributional entropy, probability of a string is to be a correct stem in a randomly chosen word is either low or high. I.e. a string can easily classify in to a valid stem class or invalid stem class. So the process tries to learn the information about stems. The learning process identify the stem distribution such that, maximum information about stem is available from the input corpus. The formal definition of this model is shown below.

Input: *Corpus* (a sufficiently large list of plain words) and *Suffixset* (set of all suffixes)

Output: $Pr(st|w)$ for all w in *Corpus* and *st* in *possible_stem_set(w)*

Consider a small corpus $\{$**mosses**, **moss**$\}$ and a suffix set $\{$**es**, **s**, $\phi\}$, where ϕ is the null suffix. The *possible_stem_set* of **mosses** is $\{$**mosses**, **mosse**, **moss**$\}$ and *possible_stem_set* of **moss** is $\{$**moss**, **mos**$\}$. Here the word probability can be taken as uniform distribution. So,

$Pr(\mathbf{mosses}) = \frac{1}{2} \times Pr(\mathbf{mosses}|\mathbf{mosses})$

$Pr(\mathbf{mosse}) = \frac{1}{2} \times Pr(\mathbf{mosse}|\mathbf{mosses})$

$Pr(\mathbf{moss}) = \frac{1}{2} \times (Pr(\mathbf{moss}|\mathbf{mosses}) + Pr(\mathbf{moss}|\mathbf{moss}))$

$Pr(\mathbf{mos}) = \frac{1}{2} \times Pr(\mathbf{mos}|\mathbf{moss})$

In this case, for $Pr(\mathbf{moss}|\mathbf{mosses}) = 1$ and $Pr(\mathbf{moss}|\mathbf{moss}) = 1$, will get a zero stem distributional entropy. So the MSE will converge to this distribution, and the string **moss** will be selected as the stem of both **mosses** and **moss**.

4.2 Methodology

To learn the model from a corpus and a suffix set, an optimization problem needs to be solved. Since the objective function is not convex, an iterative hill climbing like approach is used. We used the Frank-Wolfe algorithm [14] to solve the optimization.

In each iteration of Frank-Wolfe algorithm, we solved a linear program. By converging to local optima, the algorithm will find out the best probability distribution given the input corpus and suffix set. Most probable stem from each word in corpus is then identified.

5 Case study - Malayalam

Malayalam is a Dravidian language spoken by 32 million people primarily in Kerala, a state in southern India [15]. Malayalam is highly agglutinative and inflectionally rich with a free word order. This language has a strong postpositional inflection. A Malayalam noun can be inflected for case, number, person and gender, e.g. **വേലക്കാരന്മാരുടെ** {velakkaaranmaarude} (worker+Masculine+Plural+Genitive). Verb can be inflected by suffix for mood, aspect and tense, e.g. **പറഞ്ഞു** {paranju} (told), **പറയാം** {parayam} (may tell), **പറഞ്ഞിരിക്കും** {paranjirikkum} (will tell).

Approximated MSR model and MSE model are evaluated on Malayalam. We used a Malayalam unannotated corpus of size around 20000 words and a suffix set of around 200 Malayalam suffixes. We are extracted these words from the web. Malayalam is used by less than 0.1% of all the websites. Our assumption is that, the wordlist is sufficiently large so that it contains all morphological variants of words. In practical case this assumption may not hold. To reduce the gap between the ideal case and the real case, we make sure that, the training wordlist contains at least one more inflectional variant for each word. So we manually added such inflectional variants to wordlist if necessary.

We used the Morfessor [3] system to get a ballpark accuracy. We trained the Morfessor 1.0 model using their script downloaded from www.cis.hut.fi/projects/morpho/morfessor1.0.perl with default arguments. The Morfessor score is around 17% only. Since Morfessor is not using any information from suffix set, a direct comparison between the Morfessor score and our scores is meaningless. Since we are unable to find any other comparable approaches for this problem from the literature, we want to constructed baseline models to check the significance of these proposed models.

The problem is about selecting correct entry from possible stem set of each word. We considered different trivial strategies to select the stem from possible stem set. One basic information is that a stem is a prefix of the word by stripping some valid suffix. One trivial approach that uses only this basic information is a random selection from possible stem set. We considered this as one of the baselines. Length of the suffix (or stem) is another easily available information that can use for stemming. Based on this information we can build two simple strategies, smallest stem or largest stem. We want to choose one of these approaches. Since we are doing experimentation on data, we decided to consider both approaches for experimentation and decide based on the scores. So we considered these two approaches as second and third baselines. Performances of two new proposed models and these three baseline models are evaluated using around 1500 Malayalam Unicode words from wordlist. The accuracies are shown in the Table 5.

Table 5: Stemming Accuracies and Comparison with Baseline;

Model	Baseline-1 (Min Stem length)	Baseline-2 (Max Stem length)	Baseline-3 (Random selection)	Approx. MSR	MSE
Accuracies (20K words)	84.58 %	20.09 %	44.20 %	91.32 %	**93.91** %

The results shown in the above table indicates the correctness and significance of proposed models for Malayalam. It also shows that MSE model slightly outperforms the Approx-MSR model. From the scores it is clear that, choosing the stem with minimum length is more suitable for Malayalam. If a word in a morphologically rich language ends with a valid suffix, most likely that will be its suffix. In agglutinative languages, more than one suffix can attach to a stem. So a word may ends with more number of valid suffixes. In this case the actual stem is the prefix by stripping largest suffix from the word. Since Malayalam is a morphologically rich and agglutinative language, the high accuracy of smallest stem baseline compared to largest stem baseline is very intuitive. Also note that, this may not be true for morphologically poor languages.

To get more insight about the shortcomings of these experimentations and models, we did an error analysis of two newly proposed systems. Each and every wrong splits are analyzed. Errors in both models are common and it follows same distribution. The errors in the models are mainly three types. These errors are as follows.

1. Because of the incomplete suffix set, the models may unable to split any other inflections of some words. So the *possible_stem_set* of such words will be totally independent from *possible_stem_set* of other words. Then our stemming models choose one stem from *possible_stem_set* of such words randomly and that may leads to wrong stemming.
2. Some orthographically similar words with different stem affect the stemming process. For example, നിക്കോളെ {nikkole} and നിക്കോള {nikkola} are two different proper nouns, but it looks very similar. By removing one of the accusative case markers {e} from നിക്കോളെ {nikkole} we will get നിക്കോ ള {nikkola}, but നിക്കോള+Accusative {nikkola+Accusative} is നിക്കോളയെ {nikkolaye}. In this case our stemming models wrongly selects the word നിക്കോള {nikkola} as the stem of നിക്കോളെ {nikkole}.
3. There may exist multiple solutions that minimize the number of distinct stems or stem distributional entropy. In such cases, our models randomly choose one of the solutions. That solution may have some wrong stems.

Some wrong stems identified by our models and its error type (3 types mentioned above) are shown in the Table 6.

Table 6: Erroneous Stems Samples Found During Error Analysis

Word	**Stem (Identified)**	**Stem (Correct)**	**Error Type**
ചെനാറിൻ {chenaarin} (Chenar(Proper noun)+Gen)	ചെനാറിൻ {chenaarin}	ചെനാ {chenaa}	1
തൊൽക്കാപ്പിയരും {tholkkappiyarum} (Tholkkappiyar(Proper noun)+Conj)	തൊൽക്കാപ്പിയരു {tholkkappiyaru}	തൊൽക്കാപ്പിയര {tholkkappiyara}	1
വേണ്ടി {vendi} (for that)	വേണ്ട {venda}	വേണ്ടി {vendi}	1
നിക്കോളെ {nikkole} (Nikkole(Proper noun))	നിക്കോള {nikkola}	നിക്കോളെ {nikkole}	2
മില്ലൻ {millan} (Millan(Proper noun))	മില്ല {milla}	മില്ലന {millana}	3
അമലുകളുടെ {amalukalude} (Amal(Proper noun)+Pl+Gen)	അമലുകള {amalukala}	അമല {amala}	1

One of the reason for the remaining errors are the incomplete suffix set. If the suffix set is not complete then stems of some words in the input corpus will not be identified. Stem information from such words may useful in the process of stem identification of other words.

Consider the word വേണ്ടി {vendi} in the Table 6. Since ിയും {iyum} and ിയോ {iyo} are not in the suffix set, the system cannot split വേണ്ടിയോ {vendiyo} and വേണ്ടിയും {vendiyum} (other inflectional variants of വേണ്ടി {vendi}). In this case, effectively there is no other word in the corpus for the splitting of the word വേണ്ടി {vendi}. So a string from *possible_stem_set* of വേണ്ടി {vendi} selects randomly.

Majority of remaining errors are because of this problem. So completion of the suffix set is considered for further improvement.

6 Conclusion and Future Work

To solve the stemming problem by using an unannotated corpus and a suffix set, two models are proposed. Problem of computing first model that directly reduces the number of lexicon entries is NP-hard. A greedy approximation for this model is also proposed. Second model is a probabilistic model and it reduces the entropy of stem distribution. Approximated version of first model and second model are evaluated on Malayalam corpus. We got the best accuracy of 93% by using the MSE model. Improvement in suffix set is proposed for future work. Analysing the performances of these proposed models in other languages is also considered for future work.

References

1. Hammarström, H., Borin, L.: Unsupervised learning of morphology. CL (2011) 309–350
2. Goldsmith, J.A.: Unsupervised learning of the morphology of a natural language. CL (2001) 153–198
3. Creutz, M., Lagus, K.: Unsupervised models for morpheme segmentation and morphology learning. TSLP **4** (2007)
4. Clark, A.: Partially supervised learning of morphology with stochastic transducers. In: NLPRS. (2001) 341–348
5. Snover, M.G., Jarosz, G.E., Brent, M.R.: Unsupervised learning of morphology using a novel directed search algorithm: taking the first step. In: Proc. of ACL-WMPL-02. (2002) 11–20
6. Dreyer, M., Eisner, J.: Graphical models over multiple strings. In: Proc. of EMNLP-09. (2009) 101–110
7. Johnson, H., Martin, J.: Unsupervised learning of morphology for english and inuktitut. In: Proc. of NAACL-HLT-03. (2003) 43–45
8. van den Bosch, A., Daelemans, W.: Memory-based morphological analysis. In: Proc. of ACL-99. (1999)
9. Hammarström, H.: A naive theory of affixation and an algorithm for extraction. In: Proc. of HLT-NAACL-06. (2006) 79–88
10. Hammarström, H.: Poor man's stemming: Unsupervised recognition of same-stem words. In: AIRS. (2006) 323–337
11. Monson, C., Carbonell, J.G., Lavie, A., Levin, L.S.: Paramor and morpho challenge 2008. In: CLEF. (2008) 967–974
12. Dasgupta, S., Ng, V.: High-performance, language-independent morphological segmentation. In: HLT-NAACL. (2007) 155–163
13. Dasgupta, S., Ng, V.: Unsupervised morphological parsing of bengali. Language Resources and Evaluation (2006) 311–330
14. Lawphongpanich, S.: Frank-wolfe algorithm. In: Encyclopedia of Optimization. (2009) 1094–1097
15. David, S.M.I.P.S.: A morphological processor for malayalam language. Technical report, South Asia Research (2007)