On Application of Conditional Random Field in Stemming of Bengali Natural Language Text

Sandipan Sarkar
IBM
sandipansarkar@gmail.com,
sandipan.sarkar@in.ibm.com

Sivaji Bandyopadhyay
Computer Science & Engineering Department,
Jadavpur University
sbandyopadhyay@cse.jdvu.ac.in,
sivaji_cse_ju@yahoo.com

Abstract

While stochastic route has been explored in solving the stemming problem, Conditional Random Field (CRF), a conditional probability based statistical model, has not been applied yet. We applied CRF to train a set of stemmers for Bengali natural language text. Care had been taken to design it language neutral so that same approach can be applied for other languages. The experiments yielded more than 86% accuracy.

1 Introduction

Applications of stochastic methods in solving the stemming problem is not new. Along the rule-based approaches, this approach had been used for since 1994. The obvious advantage of stochastic stemmers is their language neutrality. Unlike rule-based stemmers, statistical stemmers usually do not require any language specific knowledge. Thus this type of stemmers is generic and can be applied in any language.

Several supervised and unsupervised statistical methods were applied before to address the problem of stemming. The methods explored are Decision Tree (Schmid, 1994), HMM (Melucci and Orio, 2003), character n-gram (Mayfield and McNamee, 2003; Jordan et al., 2005), clustering (Garain and Datta, 2005; Majumder et al., 2007; Das and Bandyopadhyay, 2008) and many more novel techniques (Croft and Xu, 1995; Goldsmith et al., 2001; Bacchin et al., 2002; Gelbukh et al., 2004; Bacchin et al., 2005; Dasgupta and Ng, 2006; Hammarström, 2006; Bhamidipati and Pal, 2007; Pandey and Siddiqui, 2008; and Patel et al., 2010). However, we could not find any work where CRF has been used.

Conditional Random Field (Lafferty et al., 2001), a probabilistic model that helps in segmenting and labelling sequence data, is quite popular in various applications such as DNA sequencing (e.g. Culotta et al., 2005), image analysis (e.g. Wang et al., 2006) etc. In NLP field it is applied in word sequencing (e.g. Tseng et al., 2005) and POS tagging (e.g. Ekbal et al., 2007; Batuer and Sun 2009) extensively. However, we could not find any publication that applied CRF in stemming or lemmatisation task in Bengali or other language.

The problem of stemming can be seen as a labelling problem where a surface word needs to be labelled against its stem. Thus, our hypothesis was CRF can be a useful tool for stemming task. The objective of this work was to apply it in building stemmers and test its performance in Bengali natural language text.

2 Related Statistical Stemming Works

As part of his decision tree based POS tagger, Schmid (1994) discovered the co-relation between POS tags and inflections. With the aid of that he identified inflections while discovering POS from the surface words.

Croft and Xu (1995) proposed a statistical mechanism to improve the result of strong rule-based stemmers that suffered from overstemming problem. The proposed mechanism applied co-occurrence measure of two words. The idea was to build equivalence classes based on a standard strong stemmer and then to apply this measure among all the words in the class. The stemmer formed new classes by merging existing classes, if word pairs from those classes were found to have high co-occurrence.

Goldsmith et al. (2001) reported an automatic statistical stemmer that would analyse a corpus of any language and find out a set of suffix and stem possibilities. Such possibilities are assigned with empirical probabilities to determine the final stem. They developed a system, which was run on Italian, German and French corpora, and

The 4th Workshop on South and Southeast Asian NLP (WSSANLP), International Joint Conference on Natural Language Processing, pages 34–42,
Nagoya, Japan, 14-18 October 2013.

reported average precisions against interpolated recalls. However, in this publication, it was not clear how effective this stemmer was against no-stemming approach. Also they admitted that the system was at an early stage.

Next significant statistical stemmer was reported by Bacchin et al. (2002). The approach was to search the community of substrings, which were formed by interlinked prefixes and suffixes, for the best word splits. They compared the language independent statistical stemmer (SPLIT) with no-stemming approach and a Porter-like rule based stemmer (Porter, 1980). However, the reported results show that the SPLIT performed worse than the rule based stemmer and the performance improvement against no-stemming approach was 5% at best.

Mayfield and McNamee (2003) reported a language independent, character n-gram based approach to identify pseudo-stems. They compared the retrieval accuracy for pseudo-stems against unstemmed words and stems, which are obtained from Porter-like (Porter, 1980) stemmers, for Swedish, Dutch, Italian, French, Finnish, Spanish, English and German. For most of the cases, n-gram approach performed better than unstemmed words but underperformed for stemmed words.

Melucci and Orio (2003) reported another language independent statistical stemmer based on HMM. They ran a similar comparison exercise as reported in Mayfield and McNamee (2003) for German, English, Italian, French and Spanish. The relative performances were same as Mayfield and McNamee (2003).

Gelbukh et al. (2004) proposed a language-agnostic unsupervised statistical approach to discover simple stemming rules from a corpus. The approach was to identify the set of stems and inflections from a corpus, where every word of the corpus can be obtained by a concatenation of one member from stem and infection set respectively. Genetic algorithm was applied to keep the size of the stem and inflection sets minimal. Though Porter (1980) performed better than the proposed stemmer, it promised to be an acceptable approach for quick stemming tasks.

Bacchin et al. (2005) proposed a probabilistic mutual reinforcement enhancement on their previous work SPLIT (Bacchin et al., 2002). The hypothesis was that a valid stem-inflection pair would have more probability of occurrence than an invalid stem-inflection pair. They applied this model against an Italian corpus and compared that again with no-stemming approach and a Porter-like stemmer. The reported results showed that it performed better than no-stemming approach consistently. However, the comparison against Porter-like stemmer did not produce any consistent result.

Garain and Datta (2005) applied stemming in the context of Bengali image document retrieval system. The proposed approach was unsupervised clustering based on edit distance (Levenshtein, 1966) of two words. Once clusters are formed, stem of the cluster was identified as the longest substring common to all words in the cluster. The experiment was run on images of newspaper articles and achieved the stemming accuracy of 88.77%.

Jordan et al. (2005) proposed a character n-gram based unsupervised algorithm to find the morphemes from a corpus. The n-grams with highest probabilities of occurrence in a corpus became candidate morphemes. The test words were recursively split to compare with these candidate morphemes to identify the resultant morpheme composition of the word. It was run on English, Finnish and Turkish word lists. It was found that the IR retrieval performance improved in comparison to the no-stemming approach for English and Finnish, whereas, in case of Turkish, the performance was worse than no-stemming.

Dasgupta and Ng (2006) devised an unsupervised algorithm for any natural language text to induce the prefix, suffix and stems from an unannotated corpus without any prior morphotactics and morpho-phonological rules. The same algorithm was extended to detect composite suffixes. They reported an accuracy of 64.62% while the algorithm was run on Bengali text corpus.

Hammarström (2006) proposed an unsupervised algorithm for detecting suffixes and stems from any unannotated natural language corpus. The work suggested a ranking mechanism of potential suffixes using three measures – Frequency (how many times the suffix appeared), Curve Drop (whether the suffix is well segmented to the left), Random Adjustment (discriminates a random segment from a true suffix). It argued in favour of gold standard based accuracy measurement of stemmer rather than IR application based measurements. The algorithm was applied on Maori, English, Swedish and Kuku Yalanji and reported accuracy of more than 90% on a relatively small set of test data (200 words each). It compared the result with Porter stemmer (Porter, 1980) for English and the performance was found to be same.

Bhamidipati and Pal (2007) proposed a statistical approach to improve a given stemmer's (rule-based or statistical etc.) performance. The approach was to compute the distance between the multinomial distribution function of a word and that of a candidate stem. The words were sorted in descending order based on frequency. If the distance was small, then the word was put into the same class of the stem otherwise, the word was treated as a new stem class. When this approach was applied on top of Porter (1980) and Truncate(n) stemmers, it was observed that the stemming accuracies consistently improved.

Majumder et al. (2007) took a clustering approach to solve the stemming problem. The clusters were formed from the corpus based on the distance between two words. They argued that Levenshtein edit distance (Levenshtein, 1966) may not be appropriate for this purpose and thus proposed four different distance functions, which put weights on mismatches based on the character position in a decreasing manner. The nucleus of the cluster became the stem. For French this approach produced comparable results with respect to Porter-like (Porter, 1980) stemmer. For Bengali, in absence of a Porter-like stemmer, they showed that it significantly improved over no-stemming approach.

Pandey and Siddiqui (2008) proposed an unsupervised approach to identify the stem based on Goldsmith's model (Goldsmith et al., 2001). It calculated the probability of the split of the word based on all combinations of possible stem and inflection combination. It iterated the split probability based on a naïve Bayesian model. The split with maximum probability identified the stem. The accuracy for Hindi language was reported between 85% - 89%, which outperformed both Ramanathan and Rao (2003) (67% - 70%) and Larkey et al. (2003) (72%-78%).

Groenewald (2009) developed a stemmer for Setswana based on k-Nearest Neighbour algorithm using a relatively small training data set. The stemmer achieved 64.06% accuracy which was slightly better than that of Brits (2006), which was a rule based stemmer.

Like in Garain and Datta (2005), Das and Bandyopadhyay (2010) presented a clustering based stemming technique for Bengali. However, they applied this technique on text instead of image. The clusters were formed based on minimum edit distance (Levenshtein, 1966) based K-means clustering. The accuracy of the stemmer was reported to be 74.06%.

Patel et al. (2010) followed a hybrid approach to come up with a stemmer for Gujarati. The statistical unsupervised approach proposed by Goldsmith et al. (2001) was adopted, however, a hand-crafted suffix list was used to better understand the stem and inflection split probabilities. The accuracy of the stemmer was reported to be 67.86%, which received a 17% accuracy boost because of hand-crafted suffix.

3 Conditional Random Field

Among statistical models, Hidden Markov Model (HMM) is very popular for labelling and sequencing tasks. However, HMM is a generative model that defines a join probability distribution P(**X**, **Y**) where **X** and **Y** are random variables respectively ranging over the observation sequence **X** and state sequence **Y**. To define this joint probability, the generative model requires the enumeration of all possible observation sequences. Building such an extensive training set in a low privileged language like Bengali, is impractical.

Moreover, HMM assumes that the observation sequence is a set of isolated and independent observation units. In most applications such assumption is intractable as the observation units often are dependent based on several different features.

Maximum Entropy Markov Model (MEMM) addresses all the above problems. It defines the conditional probability P(**Y** | **X**) instead of joint probability. Moreover, for each source state it takes observation features as input and outputs a distribution over next possible states. However, it suffers from a problem called *label bias*. MEMM transitions leaving a particular state only compete against each other instead of competing globally across all the transitions. As a result, it creates a bias towards state with fewer outgoing transitions.

CRF, resolves both of the above problems. It works on the conditional probability P(**Y** | **X**). Therefore, it does not require any modelling effort on observation sequence. Moreover, unlike MEMM, which uses a per-state exponential model, CRF has a single exponential model for the joint probability of entire sequence of states for the given observation sequence. Hence it does not suffer from label bias problem.

CRF can be represented as an undirected graph that represent the conditional probability of the state sequence $\mathbf{Y} = \{y_1, y_2, \ldots, y_T\}$, for a

given observation sequence $\mathbf{X} = \{x_1, x_2, ..., x_T\}$, as depicted below:

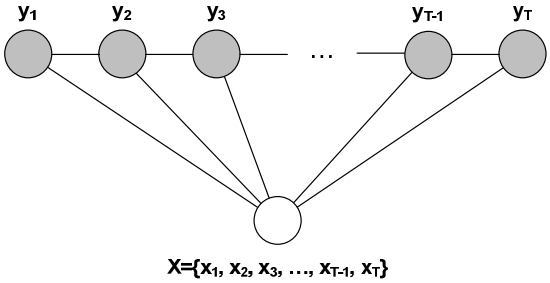

Figure 1: Graphical Structure of CRF

CRF makes a first-order Markov assumption on the state sequence – as the adjacent pair of state nodes (y_{t-1}, y_t) are linked by an undirected edge of the graph. However, it makes no assumption on the observation nodes, which is represented as a single node above.

3.1 Definition

We adopted the definition of CRF provided by Vail et al. (2007). The conditional probability P(**Y** | **X**) can be defined as the normalized product of strictly positive real-valued potential functions. The potential functions are computed over the cliques of CRF graph. As depicted in Figure 1, the cliques consist of the adjacent pairs of states and the entire observation sequence. Thus a potential function can be defined as $\psi(t, y_{t-1}, y_t, X)$, where t is an index in the state sequence. Since CRF is log-linear model (Wallach, 2004), the potential functions can be defined as follows:

$$\psi(t, y_{t-1}, y_t, X) = \exp(w^T, f(t, y_{t-1}, y_t, X))$$

Expression 1: Potential Function

where w presents a vector of weights and f is the vector of feature functions. The weight vector is estimated during training.

We define feature functions based on real-valued features. An example of a feature b from our area of application can be:

$$b(t, X) = \begin{cases} 1, \text{if } x_t = \text{a particular surface word} \\ 0, \text{otherwise} \end{cases}$$

Expression 2: Feature

Thus we can define a feature function as following:

$$f(t, y_{t-1}, y_t, X) = \begin{cases} b(t, X), \text{if } y_{t-1} = S_1 \text{ and } y_t = S_2 \\ 0, \text{otherwise} \end{cases}$$

Expression 3: Feature Function

where S_1 and S_2 are two example stems.

Going by the previous definition of the conditional probability,

$$P(Y \mid X) = \frac{1}{Z} \prod_{t=1}^{T} \exp(w^T, f(t, y_{t-1}, y_t, X))$$

Expression 4: CRF Conditional Probability

where Z is the normalization constant. The strictly positive real-valued potential functions are not guaranteed to satisfy the axioms of probability. Thus Z is used to ensure that the summation of all the probability is equal to 1. Z is defined as below:

$$Z = \sum_{Y} \prod_{t=1}^{T} \exp(w^T, f(t, y_{t-1}, y_t, X))$$

Expression 5: CRF Normalisation Factor

3.2 Training

CRF training is actually about estimation of the weight vector w, where the conditional likelihood of the training corpus, which is labelled with states. Maximizing the conditional likelihood can be approximately equated to maximization of the log-likelihood, which is more convenient to achieve. Thus we define the log-likelihood as:

$$L(Y \mid X ; w) = \sum_{t=1}^{T} w^T f(t, y_{t-1}, y_t, X) - \log(Z)$$

Expression 6: CRF Objective Function

The gradient of the above function is:

$$\frac{dL}{dw_i} = \sum_{t=1}^{T} f_i(t, y_{t-1}, y_t, X) - \sum_{Y} P(Y \mid X) f_i(t, y_{t-1}, y_t, X)$$

Expression 7: CRF Objective Function Gradient

Expression 6 is called the objective function. Optimization techniques (e.g. conjugate gradient, BFGS etc.) can be applied on the objective function to calculate the maximum log-likelihood.

3.3 Regularisation

Regularization norms can be applied on the above maximum likelihood calculation. Usually in CRF two different regularization norms are applied – L_1 and L_2.

In L_1 norm, instead of maximizing the log-likelihood alone a penalty term for each weight proportionate to $|w_i|$ are deducted from it. Thus

the penalized objective function can be defined as below:

$$\max_{w} L(Y|X;w) - \mu \sum_{i} |w_i|$$

Expression 8: L_1 Norm

where μ a parameter that controls the degree of smoothing.

In L_2 norm, the penalty term is proportionate to w_i^2. The penalized objective function can be defined as below:

$$\max_{w} L(Y|X;w) - \mu w^T w$$

Expression 9: L2 Norm

4 Experiment

4.1 Corpora

We used two corpora for the experiment:

1. **Classic Literature Corpus (CLC)**. This corpus comprised of first five short stories (ঘাটের কথা [*ghaaTer kathaa*], রাজপথের কথা [*raajapather kathaa*], মুকুট [*mukuT*], দেনাপাওনা [*denaapaaonaa*], and পোস্টমাস্টার [*posTamaasTaar*]) by Rabindranath Tagore [Tagore 1960]. It was written in traditional and colloquial dialects. It contained 15,347 tokens. We ourselves hand-tagged the corpus with POS.
2. **Contemporary Travelogue Corpus (CTC)**. This corpus comprised of four travelogues (আমাজনের গাছবাড়ি [*aamaajaner gaach-habaarhi*], বক্সা-জয়ন্তী [*baksaa-jayantee*], বনসুন্দর [*banasundar*] and বাগান [*baagaan*]) from contemporary travel magazines. It was written in colloquial dialects. It contained 11,561 tokens. The corpus is POS tagged by Indian Languages to Indian Languages Machine Translation System (IL-ILMT) developed by Jadavpur University as part of a DIT funded project. The corpus is also POS tagged by us again manually.

4.2 Strategy

We devised the following experiment strategy:

- The biggest benefit of statistical approach is language independence. Hence the CRF must not be training with any linguistic details, to avoid infusing language dependency.
- Train a CRF system so that it can discover the sequence of stems (**Y**) from the test corpus based on the following observation sequence (**X**)
    - UNI: Surface word only
    - POS: A combination of surface word and POS of the surface word
- Run the CRF system on both CLC and CTC corpora to observe the performance on different domains.
- Test the domain affinity, if any, of the CRF system. For that we decided to test CTC corpus using CRF trained using CLC and vice versa.

4.3 Experiment Setup

Since CRF is a supervised learning technique, we crafted two sets of corpora – for training (CLC_L and CTC_L) and test (CLC_T and CTC_T) purpose respectively. The details about these corpora are provided in the table below:

Table 1: Corpora Used in CRF Experiment

Detail	Value
# Surface Words in CLC_L	8953
# Surface Words in CTC_L	7493
# Inflections in CLC_L	192
# Inflections in CTC_L	131
# Stems in CLC_L	1650
# Stems in CTC_L	1856
# Surface Words in CLC_T	9607
# Surface Words in CTC_T	4410

Additionally, we also created combined corpora – $Combined_L$ and $Combined_T$ respectively by joining the two corpora from respective sets.

We chose CRF++ (Taku-ku, 2003), an open source implementation of CRF developed using C++ language. CRF++ takes few parameters, of which we used a couple, which are described below:

- Regularization Norm: We chose L_2.
- Regularization Parameter (μ): We chose 1.0 as value.

CRF++ requires an input file for both training and testing. The format of this file is like a table where each record is a set of fields, which carry individual semantics. It can have multiple input fields followed by a single output field. The usage of input fields depend on the features used in the model – potentially some input fields may remain unused. We defined two input fields: surface word (F_1), and POS of the surface word (F_2).

We had two options for the output column – stem or inflection. The natural output of a stemmer is stem. Thus initially we went for it. However, when we tried to train it on a 4 GB i5 quad-core Windows 7 64-bit laptop, the CRF++ tool crashed. We did an analysis to discover the reason behind it as explained below.

As explained earlier, CRF++ generates feature functions ($f(t, y_{t-1}, y_t, X)$) based on the training records and the features defined. The feature functions are a combination of features, records and output classes. Hence the number of feature functions generated by the tool can be estimated as N*M*K, where N = number of features, M = number of records in training file and K = number of output classes. We defined one feature (N) and used CLC_L as training corpus, which contained 8953 surface words (M) and 1650 stems (K). Hence it generated more than 14 million feature functions (1 * 8953 * 1650 = 14,772,450). Obviously, the configuration of the laptop used was not powerful enough to handle it. Soon, as the performance monitoring revealed, the training task consumed all 4 GB of available primary memory, and resulted in a crash of CRF++.

Next, the second output class option, which was inflection, was considered. In a stemmer, the inflection is not the final output. However, for evaluation purpose of the statistical approach, it would serve well.

The number of feature functions estimated for CLC_L was around 1.7 million (1 * 8953 * 193 = 1,718,976), which is less than previous estimate by at least one order of magnitude. CRF++ could manage to run it successfully with a decent memory usage, even though the CPU usage hit the 100% limit with regular valleys. A typical resource utilization pattern of the machine during training was shown in Figure 2:

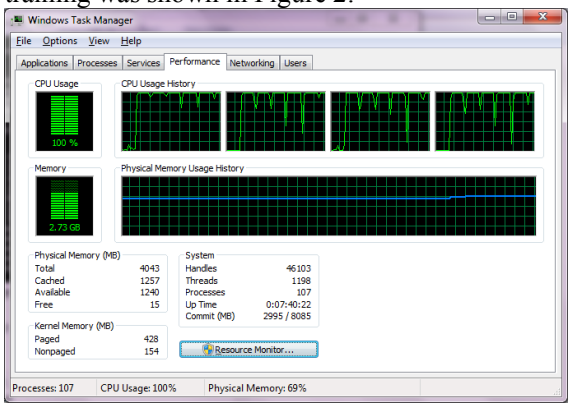

Figure 2: Machine Performance for the Training with Inflection as Output Class

In CRF++, the features (please refer to Expression 2) are defined in a template file. As we strategised, two different features, which do not have any linguistic details, were defined for two different experiment runs. They are defined below:

$$b_{UNI}(t, X) = \begin{cases} 1, \text{if } F_1 = \text{surface word of } x_t \\ 0, \text{otherwise} \end{cases}$$

Expression 10: Feature UNI

$$b_{POS}(t, X) = \begin{cases} 1, \text{if } F_1 = \text{surface word of } x_t \text{ and } F_2 = \text{POS of } x_t \\ 0, \text{otherwise} \end{cases}$$

Expression 11: Feature POS

5 Result

We trained the machine six times using different training corpora and features as summarized below:

Table 2: CRF Trained Models

Model	Training Corpus	Feature
TAGORE.UNI	CLC_L	UNI
TRAVEL.UNI	CTC_L	UNI
COMBINED.UNI	$Combined_L$	UNI
TAGORE.POS	CLC_L	POS
TRAVEL.POS	CTC_L	POS
COMBINED.POS	$Combined_L$	POS

We executed a set of test runs using different combinations of model and test corpus. Afterwards we compared the machine output with the manually determined gold standard and computed the accuracy. Following table presents the outcome:

Table 3: Accuracies Achieved in CRF Test Runs

Run	Model	Corpus	Result
CLC.TAGORE.UNI	TAGORE.UNI	CLC_T	84.7%
CTC.TAGORE.UNI	TAGORE.UNI	CTC_T	69.5%
CTC.TRAVEL.UNI	TRAVEL.UNI	CTC_T	79.8%
CLC.COMBINED.UNI	COMBINED.UNI	CLC_T	86.0%
CTC.COMBINED.UNI	COMBINED.UNI	CTC_T	81.6%
CLC.TAGORE.POS	TAGORE.POS	CLC_T	84.3%
CTC.TAGORE.POS	TAGORE.POS	CTC_T	68.2%
CTC.TRAVEL.POS	TRAVEL.POS	CTC_T	78.7%
CLC.COMBINED.POS	COMBINED.POS	CLC_T	85.6%
CTC.COMBINED.POS	COMBINED.POS	CTC_T	80.9%

Overall, the performance of the CRF stemmer is encouraging as it more or less consistently achieved an accuracy of more than 80%. It performed better than two other statistical stemmers

(Dasgupta and Ng, 2006; and Das and Bandyopadhyay, 2010), which reported 64.62% and 74.06% accuracy respectively for Bengali text.

We further analysed it on three different aspects as presented in the subsections below:

5.1 Effect of Features

We plotted different test runs to compare the effect of different features.

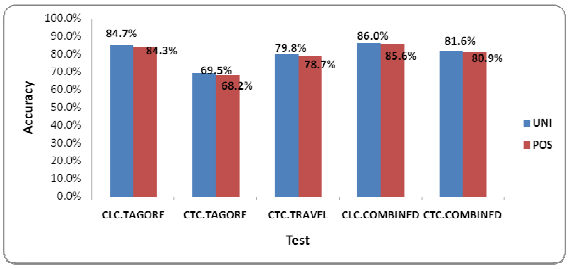

Figure 3: Effect of Features on Accuracy

As evident from above, having an additional feature in the form of POS, did not help the performance of the CRF stemmer. For all test runs, both of these features yielded almost similar performance.

5.2 Effect of Domains

We analyzed the effect of domains both on training and test corpus. We picked up the UNI feature based results as that was slightly better than POS based accuracies. The chart below depicts the result of this analysis:

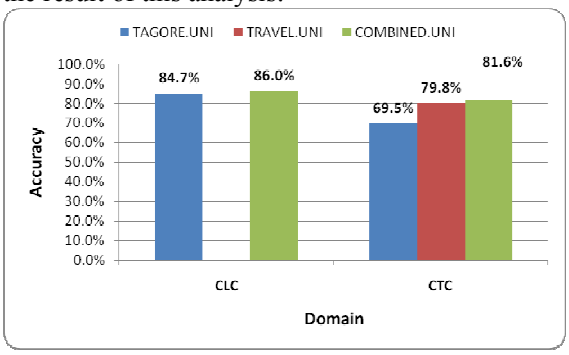

Figure 4: Effect of Domains on Accuracy

We made the following observation:

- CTC performed worse than CLC. We found that there were many spelling mistakes and malformed words present in this corpus. The CRF failed to find the right inflection pattern for such words.
- The performance of CTC against the model TAGORE.UNI produced worst result when compared against other runs of CTC. It shows that the statistical stemmer shows domain affinity. In this case the training and test corpus were from different domains – and that was the reason for bad accuracy.
- The combined model (COMBINED.UNI) yielded better performance than the respective trained models of domains. This observation matches the intuition that richness of the training data may improve the stemmer performance.

5.3 Rule-based vs. Statistical

In our survey, we could not find any work that compared the performances rule-based and statistical stemmers in the context of Bengali text. As third analysis step, we attempted the same.

We picked up the rule-based stemmer Mulaadhaar3 (M3) proposed by Sarkar and Bandyopadhyay (2012). M3 performances were reported on same set of test corpora, thus a fair comparison was possible. We compared the best results CRF achieved against the A_1 and A_2 accuracy measures of M3. The analysis result is depicted below:

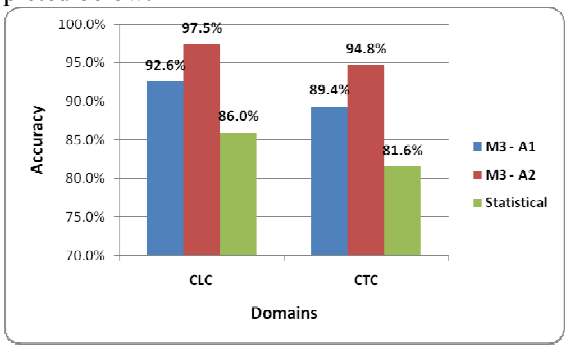

Figure 5: CRF vs. M3

As evident from above, M3 outperformed CRF on all the domains.

6 Conclusion

We tried a different statistical approach than the more popular options, in the form of a CRF based machine learning technique to produce a statistical stemmer for Bengali. The results found to be encouraging while comparing against other published works on Bengali statistical stemmers. However, we found that the rule-based stemmer M3 performed far better than the CRF stemmers.

However, the approach presented here is language independent. Thus it can be applied to languages where the deep linguistic rules are not yet formalised. It would be interesting to see its application in other Indo-Aryan languages like Oriya, Assamese etc. where linguistic rule-based stemmers are yet to arrive.

References

M. Bacchin, N. Ferro, and M. Melucci. 2002. *Experiments to evaluate a statistical stemming algorithm*. Proceedings of the Conference and Labs of the Evaluation Forum (CLEF).

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

B. Aisha, and M. Sun. 2009. *A Uyghur Morpheme Analysis Method based on Conditional Random Fields*. International Journal on Asian Language Processing, 19(2):69- 77.

N. L. Bhamidipati, and S. K. Pal. 2007. *Stemming via Distribution-Based Word Segregation for Classification and Retrieva*. IEEE Transactions on Systems, Man, and Cybernatics - Part B: Cybernatics, Vol. 37, No. 2.

J. H. Brits. 2006. *Outomatiese Setswana lemma-identifisering*. Master's Thesis. North-West University, Potchefstroom, South Africa.

W.B. Croft, and J. Xu. 1995. *Corpus-Specific Stemming using Word Form Co-occurrences*. In Fourth Annual Symposium on Document Analysis and Information Retrieval.

A. Culotta, D. Kulp and A. McCallum. 2005. *Gene Prediction with Conditional Random Fields*. Technical Report UM-CS-2005-028, University of Massachusetts, Amherst.

A. Das, and S. Bandyopadhyay. 2010. *Morphological Stemming Cluster Identification for Bangla*. Knowledge Sharing Event-1: Task 3: Morphological Analyzers and Generators.

S. Dasgupta, and V. Ng. 2006. *Unsupervised Morphological Parsing for Bengali*. Language Resources and Evaluation, Volume 40, Numbers 3-4, 311-330.

U. Garain, and A. K. Datta. 2005. *An approach for stemming in symbolically compressed Indian language imaged documents*. Proceedings of the Eighth International Conference on Document Analysis and Recognition.

A. Gelbukh, M. Alexandrov, and S. Y. Han. 2004. *Detecting Inflection Patterns in Natural Language by Minimization of Morphological Model*. Progress in Pattern Recognition, Image Analysis and Applications: Lecture Notes in Computer Science, Volume 3287/2004, pp. 110-14.

J. A. Goldsmith, D. Higgins, and S. Soglasnova. 2001. *Automatic Language-Specific Stemming in Information Retrieval*. Cross-Language Information Retrieval and Evaluation String Processing and Information Retrieval: Lecture Notes in Computer Science, Volume 2857/2003, pp. 238-251.

H. J. Groenewald. 2009. *Using Technology Transfer to Advance Automatic Lemmatisation for Setswana*. Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages – AfLaT 2009, pp. 32–37.

H. Hammarström. 2006. *Poor Man's Stemming: Unsupervised Recognition of Same-Stem Words*. Information Retrieval Technology: Lecture Notes in Computer Science, Volume 4182/2006, 323-337.

C. Jordan, J. Healy, and V. Keselj. 2005. *Swordfish: Using Ngrams in an Unsupervised Approach to Morphological Analysis*. Proceedings of Morpho Challenge.

L. S. Larkey, M. E. Connell, and N. Abduljaleel. 2003. *Hindi CLIR in Thirty Days*. ACM Transaction on Asian Language Information Processing, Vol-2, No. 2, pp. 130-142.

V. Levenshtein. 1966. *Binary codes capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady, 10: 707–10.

P. Majumder, M. Mitra, S. Parui, G. Kole, P. Mitra and K. Datta. 2007. *YASS: Yet another suffix stripper*. ACM Transactions on Information Systems (TOIS).

J. Mayfield, and P. McNamee. 2003. *Single N-gram Stemming*. Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03).

M. Melucci, and N. Orio. 2003. *A Novel Method for Stemmer Generation Based on Hidden Markov Models*. Proceedings of the twelfth international conference on Information and knowledge management (CIKM '03).

A. K. Pandey, and T. J. Siddiqui. 2008. *An Unsupervised Hindi stemmer with heuristic improvements*. Proceedings of the second workshop on Analytics for noisy unstructured text data (AND'08).

P. Patel, K. Popat, and P. Bhattacharyya. 2010. *Hybrid Stemmer for Gujarati*. Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), The 23rd International Conference on Computational Linguistics (COLING), pp. 51–55.

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

A. Ramanathan, and D. D. Rao. 2003. *A Lightweight Stemmer for Hindi*. Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics.

H. Schmid. 1994. *Probabilistic part-of-speech tagging using decision trees*. Proceedings of International Conference on New Methods in Language Processing.

H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning . 2005. *A Conditional Random Field Word Segmenter*. Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing.

D. L. Vail, J. D. Lafferty, and M. M. Veloso. 2007. *Feature selection in conditional random fields for activity recognition*. Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2007).

Y. Wang, K. F. Loe, and J. K. Wu. 2006. *A dynamic conditional random field model for foreground and shadow segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(2), pp. 279-189.

S. Sarkar, and S. Bandyopadhyay. 2012. *Mulaadhaar: Towards an Improved Stemmer and Its Effect on Machine Tagged Travelogue Corpus*. International Journal of Computational Linguistics and Natural Language Processing, Volume 1, Issue 5.

Taku-ku. 2003. *CRF++: Yet Another CRF toolkit*. [Online] Accessed on 11 Jun 2013 at http://crfpp.googlecode.com/svn/trunk/doc/index.html