Int J Data Sci Anal (2016) 1:123–136
DOI 10.1007/s41060-016-0010-5

REGULAR PAPER

A semi-supervised associative classification method for POS tagging

Pratibha Rani[1] · **Vikram Pudi**[1] · **Dipti Misra Sharma**[1]

Received: 30 March 2016 / Accepted: 12 May 2016 / Published online: 10 June 2016

Abstract We present here a data mining approach for part-of-speech (POS) tagging, an important natural language processing (NLP) task, which is a classification problem. We propose a semi-supervised associative classification method for POS tagging. Existing methods for building POS taggers require extensive domain and linguistic knowledge and resources. Our method uses a combination of a small POS tagged corpus and untagged text data as training data to build the classifier model using association rules. Our tagger works well with very little training data also. The use of semi-supervised learning provides the advantage of not requiring a large high-quality annotated corpus. These properties make it especially suitable for resource-poor languages. Our experiments on various resource-rich, resource-moderate and resource-poor languages show good performance without using any language-specific linguistic information. We note that inclusion of such features in our method may further improve the performance. Results also show that for smaller training data sizes our tagger performs better than state-of-the-art conditional random field (CRF) tagger using same features as our tagger.

This article is an invited extended version of paper [24] presented in IEEE International Conference on Data Science and Advanced Analytics 2014, Shanghai, China (DSAA'14).

✉ Pratibha Rani
pratibha_rani@research.iiit.ac.in

Vikram Pudi
vikram@iiit.ac.in

Dipti Misra Sharma
dipti@iiit.ac.in

1 International Institute of Information Technology, Hyderabad, India

Keywords Part-of-speech tagging · Association rule mining · Semi-supervised learning · Natural language processing

1 Introduction

A few languages like English and French have been extensively analyzed for NLP tasks such as POS tagging. Domain expertise is needed in order to study the properties of each language so as to build linguistic resources, select appropriate features, configure parameters and set exceptions in systems. To build systems to handle each of the remaining (6500+ or so) "resource-poor" languages [23] of the world would require the same intensive effort, expertise, expenses and time. In order to handle this challenge, we need to build domain-independent, language-independent and data-driven systems, which can work reasonably and effectively without much domain expertise.

POS tagging (henceforth referred to as just tagging) is an important NLP classification task that takes a word or a sentence as input, assigns a POS tag or other lexical class marker to a word or to each word in the sentence, and produces the tagged text as output. For this task, several rule-based [8], stochastic-supervised [7,33], and stochastic-unsupervised [3,17] methods are available for a number of languages. All of these (including the state-of-the-art taggers) require training data and linguistic resources like dictionaries in *large* quantities.

These taggers do not perform well for *resource-poor* languages, which do not have much resources and training data. So, there is a need to develop generic semi-supervised tagging methods which take advantage of untagged corpus and require less or no lexical resources. A few available techniques for this are discussed in Sect. 2. In order to perform well, these techniques require a *large* untagged corpus. Unfortunately, for many resource-poor languages, even obtaining this is hard.

This motivates us to explore data mining methods to build a generic POS tagger. Data mining, being composed of data-driven techniques, is a promising direction to explore or to develop language-/domain-independent tagging methods. Associative classification [31] is a well-known data mining-based classification approach which uses association rules [1] to build the classifier model. To the best of our knowledge, no semi-supervised association rule mining method exists. In this work, we apply semi-supervised associative classification approach to build a generic semi-supervised POS tagger.

However, direct application of data mining concepts for this task is not feasible and requires handling various challenges like (1) mapping POS tagging task to association rule mining problem, (2) developing semi-supervised methods to extract association rules from training set of *annotated* and *untagged* data combined and (3) handling challenges of POS tagging task (discussed in Sect. 4.2), like class imbalance, data sparsity and phrase boundary problems.

Our method uses a combination of a *small* annotated and untagged training data to build a classifier model using a new concept of *context-based association rule mining*. These association rules work as context-based tagging rules. Our experiments demonstrate that this new concept gives good performance even without using any linguistic resources—except for a small POS tagged corpus—for resource-rich English, resource-moderate Hindi and resource-poor Telugu, Tamil and Bengali languages.

Our method is generic in two aspects: (1) it does not use any language-specific linguistic information such as morphological features and there is ample scope to improve further by including such features and (2) it does not require a large, high-quality, annotated corpus and uses the POS tags of the annotated corpus only to calculate scores of "context-based lists" which are used to form association rules. This can be easily adapted for various languages. Also, as an additional benefit our tagger makes human-understandable model since it is based on association rules, which are simple to understand.

Our algorithm has the following advantages, especially suitable for resource-poor languages, arising due to the use of untagged data: (1) increased coverage of words and contexts in the classifier model increases tagging accuracy and tags unknown words without using smoothing techniques and (2) it creates additional linguistic resources from untagged data in the form of word clusters.

Remainder of this paper is as follows. Section 2 surveys related work. Section 3 formally presents the problem. Sections 4, 5 and 6 present details of our proposed approach. Section 7 gives details of the various datasets, experiments and discusses the performance. Section 8 presents case study of Hindi and Telugu languages and Sect. 9 finally concludes the paper.

2 Related work

Associative classifiers have been successfully applied for various classification tasks. For example, Zaïane et al. [36] present an associative classifier for mammography image classification and Soni and Vyas [29] use it for predictive analysis in health care data mining. Some of the associative classifiers worth mentioning are CBA [22], which integrates association rules and classification by finding class association rules, CMAR [21], uses concept of multiple class-association rules, CPAR [35], based on predictive association rules and ACME [32], exploits maximum entropy principle. A good review of various associative classifiers and the detailed analysis of this method can be found in [31]. For text classification, Kamruzzaman et al. [19] use association rules in a hybrid system of Naive Bayes and genetic classifier. Shaohong and Guidan [25] present a supervised *language-specific* hybrid algorithm of statistical method and association rule mining to increase the POS tagging accuracy of Chinese text. To the best of our knowledge, no semi-supervised method exists for mining association rules from a training set of annotated and untagged data combined.

One of the first semi-supervised POS tagging methods was proposed by Cutting et al. [10] which uses untagged corpus by incorporating features obtained from a small fraction of untagged data along with features obtained from a *large* annotated data. A good overview of the existing semi-supervised POS tagging methods and discussion on their limitations is provided by Subramanya et al. [30], which uses graph as a smoothness regularizer to train CRFs [20] in a semi-supervised manner from a *large* untagged data and a small annotated data. Authors of [27] present a tri-training [37]-based semi-supervised POS tagger that combines supervised SVMTool [16] and unsupervised Unsupos [6] taggers. Later in [28], they present a condensed nearest neighbor method for semi-supervised POS tagging and report 97.5 % accuracy on WSJ dataset of English. Most of the existing semi-supervised POS tagging methods use a combination of complex learning methods and existing supervised tagging methods to learn from large untagged data and moderate-sized annotated data. All these methods have been developed for resource-rich English and other European languages.

To the best of our knowledge, no semi-supervised tagger exists for resource-moderate Hindi and resource-poor Telugu and Tamil languages. Also to the best of our knowledge, no fully data mining-based generic POS tagger exists for any language. Baseline POS taggers for various languages are discussed below. We note that all the reported accuracy values were obtained for very small-sized test sets. All the mentioned POS taggers use linguistic (especially morphological) knowledge in some or the other form, while our approach uses only the POS tags of the annotated set in an indirect form and learns from the untagged data.

For Hindi language, Avinesh and Karthik [2] proposes a CRF model with transformation-based learning (TBL) with morphological features and reports 78.67 % accuracy on SPSAL corpus. Gadde and Yeleti [15] reports 92.36 % accuracy on ISPC corpus using special linguistic features in a hidden Markov model (HMM). Shrivastava and Bhattacharyya [26] propose an HMM with morphological features and report 93.05 % accuracy. For Telugu language, Avinesh and Karthik [2] apply transformation-based learning (TBL) on top of a CRF model and report 77.37 % accuracy on SPSAL corpus. Gadde and Yeleti [15] use various special linguistic features in an HMM and report 91.23 % accuracy on ISPC corpus.

For Bengali language, Dandapat et al. [11] present various supervised and semi-supervised maximum entropy models and HMMs using morphological features and report 87.9 % accuracy for semi-supervised HMM on CIIL corpus. Authors in [14] report 92.35 % accuracy using a voted approach among various models. For Tamil language, Dhanalakshmi et al. [12] present a linear programming-based support vector machine (SVM) model and reports 95.63 % accuracy.

3 Problem definition

Automated POS tagging is a classification task which takes a word or a sentence as input, assigns a POS tag or other lexical class marker to a word or to each word in the sentence, and produces the tagged text as output. In semi-supervised paradigm, the POS tagger is built from a corpus of untagged sentences and a set of annotated sentences. The POS tagging classification problem is formally defined as follows:

Given a set of tags $\Gamma = \{T_1, T_2, \ldots, T_n\}$, an *annotated* training set of tagged sentences $AS = \{St_1, St_2, \ldots St_N\}$, where $St_i = \langle W_1/T_i, W_2/T_j \ldots W_n/T_k\rangle$ (W_i is a word and T_i is a tag from Γ) and a *untagged* training corpus of sentences $D = \{S_1, S_2 \ldots S_M\}$, where $S_i = \langle W_1 W_2 \ldots W_m\rangle$, the goal is to build a classifier model Φ which outputs the best tag sequence $\langle T_1 T_2 \ldots T_l\rangle$ for an input sequence of words $\langle W_1 W_2 \ldots W_l\rangle$.

4 Our approach

4.1 Mapping POS tagging to association rule mining

According to the *one sense per collocation* [34] hypothesis, the sense of a word in a document is effectively determined by its *context*. The notion of context has been used in various methods of POS tagging [3,33]. A context can occur in multiple places in the text. We refer to this list of occurrences of a context as its *context-based list*. We use this idea for building our tagger. In our method, we mine context-based association rules from training data containing both annotated and untagged text. Our method works as follows:

- We collect all possible words occurring in the same context from the untagged data into a list called *context-based list* (formally defined later). In this way, we are able to find groups of words of *similar categories* from the untagged data.
- Using the annotated set and the tag finding algorithm of Algorithm 1, we find association rules of the form: $Context \Rightarrow Tag$ for the context-based lists. Each rule maps a context-based list to a suitable POS tag. These association rules work as the context-based classification rules.
- Lastly, we group these context-based association rules according to their POS tags to form clusters. This set of clusters is used as the classifier model to tag words using the method described in Sect. 6 (algorithm given in Algorithm 2).

By experimenting with two varieties of bi-gram (one with preceding word and the other with succeeding word) and trigram as possible contexts, we found that trigram works best for our method. For a word instance W_i, we fix its context as a trigram containing W_i in the middle and we use this context to find the *context-based list*. Any other notion of context can be used as long as it fits into the formalism given below.

Context-based list: If Ψ is a function mapping from a word instance W_i in the data to its context $\Psi(W_i)$, then inverse function $\Psi^{-1}(\Psi(W_i))$ is a list of words instances sharing the same context. We refer to this list as *context-based list* of $\Psi(W_i)$. It denotes words of similar category or type as W_i in a specific context and can store multiple instances of a word. For a given trigram $(W_{i-1}\ W_i\ W_{i+1})$ of words, $\Psi(W_i) = (W_{i-1}, W_{i+1})$. The preceding word W_{i-1} and succeeding word W_{i+1} are called *context words* and $\Psi(W_i)$ is called the *context word pair* of W_i.

Context-based association rule: For each *context-based list* L, our approach finds association rule of the form $L \Rightarrow T$. This rule maps the context-based list L to a POS tag T with *support* and *confidence* parameters defined below. Since each list L is obtained from a unique context word pair, so each association rule uniquely associates a context to a POS tag and works as the context-based tagging rule.

In the following definitions and formulas, we develop the intuition and the method to compute the interestingness measures of the significant association rules. The complexity in defining support is due to the presence of untagged training data required for semi-supervised learning. The support is the count of occurrences of the context in the dataset. Context-based lists are made from untagged data D, and we are interested in those words of these lists for which we know the tag in annotated set AS. Hence, we define support of a context as follows:

AllTagContextSupport: Number of unique words of a context-based list L whose tags are available (in annotated set AS) is denoted as $\mathit{AllTagContextSupport}(L)$. This measure gives the number of tagged words of L.

ContextSupport: For a list of words L in which duplicates may be present, $\mathit{ContextSupport}(L)$ is defined as the set of unique words present in L.

Coverage: For a *context-based list* L,

$$\mathit{Coverage}(L) = \frac{\mathit{AllTagContextSupport}(L)}{|\mathit{ContextSupport}(L)|} \tag{1}$$

This measure represents the confidence that enough number of tagged samples are present in L.

ContextTagSupport: Number of unique words of a *context-based list* L present in annotated set AS with a particular tag T is denoted as $\mathit{ContextTagSupport}(L, T)$.

Confidence: For a *context-based list* L and tag T,

$$\mathit{Confidence}(L, T) = \frac{\mathit{ContextTagSupport}(L, T)}{|\mathit{ContextSupport}(L)|} \tag{2}$$

This measure represents the confidence that considerable number of words in list L have a particular tag T and leads to rules of the form $\mathit{Context} \Rightarrow \mathit{Tag}$.

WordTagSupport: Frequency of tag T for a word W in the annotated set AS is denoted as $\mathit{WordTagSupport}(T, W)$.

WordTagScore: For a word W and tag T,

$$\mathit{WordTagScore}(W, T) = \frac{\mathit{WordTagSupport}(T, W)}{\max_{T_i \in \Gamma} \mathit{WordTagSupport}(T_i, W)} \tag{3}$$

This represents how good the tag fits the word on a scale of 0 to 1.

ListTagScore: For a tag T in *context-based list* L,

$$\mathit{ListTagScore}(L, T) = \frac{\sum_{W_i \in \mathit{ContextSupport}(L)} \mathit{WordTagScore}(W_i, T)}{|\{W_i \in \mathit{ContextSupport}(L) : W_i/T \in AS\}|} \tag{4}$$

where **AS** is the annotated set. This formula represents the average frequency of tag T in *context-based list* L. Intuitively, it represents how good the tag fits the list. Unfortunately, this is not always indicative of the correct tag for the list. For example, if a tag is overall very frequent, it can bias this score. Therefore, we compare this with the following score, inspired by the notion of *Conviction* [9].

BackgroundTagScore: For a tag T in annotated set AS,

$$\mathit{BackgroundTagScore}(T) = \frac{\sum_{W_i \in \mathit{ContextSupport}(AS)} \mathit{WordTagScore}(W_i, T)}{|\{W_i \in \mathit{ContextSupport}(AS) : W_i/T \in AS\}|} \tag{5}$$

This represents the average frequency of tag T in annotated set AS.

4.2 POS tagging challenges

POS tagging, especially for resource-poor languages, involves three major challenges listed below. In our approach, we handle each of them explicitly.

1. **Data sparsity problem**: Some POS tag classes are present in the annotated set with very few representations. This is not enough to derive statistical information about them. In our approach, the use of untagged data reduces this problem (shown in Sect. 7.4).
2. **Class imbalance problem**: POS tag classes are highly imbalanced in their occurrence frequency. While selecting a tag this may lead to biasing toward the most frequent tags. Existing solutions of class imbalance problem typically favor rare classes [13]. However, while tagging the *context-based lists*, we need to find POS tags for them in such a way that we neither favor frequent tags nor rare tags. We tackle this problem using a novel *Minmax* approach to find the best preferred POS tag instead of the most frequent one (described in Sect. 5.2).
3. **Phrase boundary problem**: Some lists are formed at phrase boundaries where the context comes from two different phrases. We need to filter out these *context-based lists* which do not contain words of similar categories. In this case, the context of a word instance need not represent strong context and so the context-based list may contain unrelated words. We use suitable parameters to handle this problem (explained in Sect. 5.3).

5 Building classifier model

5.1 Finding association rule for a context-based list

The first step in our classifier model building method is to compute *context-based lists* from an untagged train-

```
1.  for each tag $T_i \in \Gamma$ present in annotated set $AS$ do:
2.      Find $BackgroundTagScore(T_i)$          // Use Equation (5)
3.  for context based list $L$ do:
4.      Find $Coverage(L)$          // Use Equation (1)
5.      if $Coverage(L) \geq MinCoverage$:
6.          $ContextTagSupport(L, T_{max}) = \max_{T_i \in \Gamma} ContextTagSupport(L, T_i)$
7.          $Maxconf = Confidence(L, T_{max})$          // Use Equation (2)
8.          if $Maxconf > MinConfidence$:
9.              $MaxTset = \{T_i \mid ContextTagSupport(L, T_i) == ContextTagSupport(L, T_{max})\}$
10.             $BestPrefTag = FindBestPrefTag(L, MaxTset)$
11.             Return $BestPrefTag$
12.         else: Return NOTVALIST
13.     else: Return NOTVALIST

14. $FindBestPrefTag(L, MaxTset)$:
15.     Initialize $PrefTagset = \{\}$
16.     for each word $W$ of $ContextSupport(L)$ present in $AS$ do:
17.         $Tagset(W) = \{T_i \mid W$ has tag $T_i$ in $AS\}$
18.         $UnqTagset = Tagset(W) \cap MaxTset$
19.         Find $MaxWTag \mid WordTagSupport(MaxWTag, W) == \max_{T_j \in UnqTagset} WordTagSupport(T_j, W)$
20.         $PrefTagset = PrefTagset \cup MaxWTag$
21.     Find $MinTag \in PrefTagset \mid \exists\, W_{min} \in ContextSupport(L)$ with
            $\left(WordTagSupport(MinTag, W_{min}) == \min_{W_i \in ContextSupport(L)} WordTagSupport(MinTag, W_i)\right)$
22.     Find $ListTagScore(L, MinTag)$          // Use Equation (4)
23.     if $ListTagScore(L, MinTag) \geq BackgroundTagScore(MinTag)$: Return $MinTag$
24.     else: Return NOTVALIST
```

Algorithm 1: Algorithm to find POS tag for a *context-based list*

ing corpus D. It may be noted that a context-based list can store multiple instances of a word. We use a sliding window of size three to collect the context-based lists from D, in a single iteration, taking care of sentence boundaries.

In the next step, we use the algorithm shown in Algorithm 1 to find association rules for all the context-based lists. In this algorithm, *BackgroundTagScore* of all the POS tags present in the annotated set AS (lines 1–2) are computed first. Then, for a context-based list satisfying the threshold values of *Coverage* and *Confidence* (lines 3–9), function *FindBestPrefTag* (described in Sect. 5.2) finds the best preferred tag (lines 10–11, 14–24) from the set of tags with maximum *ContextTagSupport* (lines 7–9).

For a context-based list L present as antecedent in association rule $L \Rightarrow T$, tag T returned by this algorithm becomes the consequent. This algorithm outputs best preferred tags for all the context-based lists and hence finds association rules for all of them.

5.2 Handling class imbalance problem

We handle the *class imbalance problem* by using a novel *Minmax* approach in the function *FindBestPrefTag* (lines 14–24 in Algorithm 1) and parameters *BackgroundTagScore* and *ListTagScore*. In our *Minmax* approach, the preferred tag T_i for *context-based list* L is the one which has maximum *ContextTagSupport*(L, T_i), but it is also having minimum *WordTagSupport*(T_i, W) among those words of list L which have tag T_i as the best tag in AS. This takes care that the selected tag is supported by majority of the words in the list and is not biased by annotated set's most frequent tag.

To find the best preferred tag for list L in the function *FindBestPrefTag*, from the set of all the tags with maximum *ContextTagSupport* value (line 9), at first we find those tags which are best tags (having maximum *WordTagSupport* value) for the words of list L in AS (lines 15–20). Next, from this set of preferred tags we find the tag with minimum *WordTagSupport* value (line 21). Then, we check the tag scores using criteria specified in lines 22–23, which is,{ *ListTagScore*$(L, T_i) \geq$ *BackgroundTagScore*(T_i) }, to ensure that the selected tag has above-average support in the annotated set and the context-based list, both. If none of the tags satisfy this criteria, then we tag the list as "NOTVALIST" (line 24).

5.3 Handling phrase boundary problem

To filter out context-based lists with the *phrase boundary problem* (see Sect. 4.2), we use two suitable threshold values for parameters *Confidence* and *Coverage*. *Coverage* takes care of the fact that a context-based list has considerable number of words to map it to a tag and *Confidence* ensures that the tag found for the list is the one which is supported by majority of the words in the list.

If context-based list L has *Coverage* and *Confidence* values less than the corresponding threshold values *MinCoverage* and *MinConfidence*, we tag L as "NOTVALIST" (lines 3–8, 12, 13 in Algorithm 1). If L satisfies both of the threshold values, then only we find the set of all the tags which have maximum $\mathit{ContextTagSupport}(L, T_i)$ value and use this set (lines 9–10) to find the best preferred tag for the list (lines 14–24).

5.4 POS tag-wise grouping of association rules

In the last step, we group context-based lists according to their POS tags to get clusters of context-based lists as classifier model. We exclude context-based lists with tag "NOTVALIST" from the grouping process. Then, we process these clusters to store word frequencies, corresponding context word pairs and their frequencies in each cluster. We represent the set of clusters as *Clustset*.

Since we are highly confident about the tags of the words present in the annotated set AS so, to improve cluster quality we remove those words (present in AS) from each cluster which do not have a matching cluster tag in AS. Finally, we get a set of clusters in which each cluster has words, their counts and associated context word pairs with their counts. Each cluster has a unique POS tag. These clusters are overlapping in nature and words can belong to multiple clusters.

6 POS tagging of test data

Our tagging method is formalized in Algorithm 2. To tag the words of a test sentence, we make use of the test word's context word pair, preceding word, and the word frequency in a cluster to decide the tag of the word. When a test word is found in only one cluster then we output the cluster tag as the tag of the test word. But when a test word is found in many clusters, then to select the suitable clusters following priority order is followed:

1. **Criteria 1:** Highest priority is given to the presence of matching context word pair of the test word in the clusters.
2. **Criteria 2:** Second highest priority is given to the presence of matching preceding word of the test word as first word of the context word pairs in clusters.
3. **Criteria 3:** Last priority is given to the frequency of the test word in the clusters.

For test words not present in any cluster, we use criterion 1 and 2 to select appropriate clusters. Based on the priority order, only one of the criterion is used to select the suitable clusters. If we are not able to find any suitable cluster, then we return "NOTAG" as the tag of the test word.

Even when we find suitable clusters, to increase precision, our method finds POS tags only for those cases where it is confident. It avoids to wrongly classify non-confident cases and returns "NOTAG" for them. This is especially useful when the cost of misclassifying (false positive) is high. This also gives opportunity to integrate other language-/domain-specific POS taggers as they can be used for the non-confident cases.

After selecting the suitable clusters, we need to make sure that we have enough confidence in the highest probability tag obtained from the clusters. To ensure this, we use the parameter *TagProbDif*, which gives the fractional difference between the highest and the second highest cluster tag probabilities and is defined as follows:

$$\mathit{TagProbDif} = \frac{\mathit{TagProb}(C_{\max}) - \mathit{TagProb}(C_{\text{secmax}})}{\mathit{TagProb}(C_{\max})} \qquad (6)$$

where $C_{\max}$ is the cluster with highest $\mathit{TagProb}(C_i)$ value and C_{secmax} is the cluster with second highest $\mathit{TagProb}(C_i)$ value. $\mathit{TagProb}(C_i)$ of a cluster is defined as follows:

$$\mathit{TagProb}(C_i) = \frac{\text{Frequency of X in } C_i}{\sum\limits_{\forall C_j \in Clustset} \text{Frequency of X in } C_j} \qquad (7)$$

where X is set as follows: If the test word is present in cluster C_i then X = test word. For test word not present in any cluster, if the clusters are selected based on the presence of the context word pair of the test word then X = context word pair; if the clusters are selected based on the presence of the preceding word of the test word as first word of the context word pairs in clusters, then X = preceding word of the test word. In this way, we are able to tag some $\mathit{unseen/unknown}$ words also which are not present in the training data. This, in a way, acts as an alternative of smoothing technique for them.

After selecting the clusters (based on priority order), we compute their *TagProb* values using (7) and then compute *TagProbDif* using (6). For *TagProbDif* value above a suitable threshold value *MinprobDif*, we output the tag of cluster with highest *TagProb* value as the tag of the test word, otherwise we return "NOTAG'.

7 Experiments, results and observations

7.1 Dataset details

We have done our experiments on resource-rich English language (New York Times dataset of American National

Table 1 Statistics of all the five language datasets with *AverageAccuracy* values obtained by our tagger

	Hindi	Telugu	Tamil	Bengali	English
No. of words in untagged training set	393,303	104,281	169,705	85,796	1,293,388
No. of words in annotated training set	282,548	83,442	20,207	21,561	629,532
POS tag count in annotated training set	35	28	28	27	109
No. of words in test set	70,811	20,854	22,352	20,618	471,977
POS tag count in test set	32	24	27	29	105
Test words tagged as NOTAG by our tagger	1916	1634	2647	3448	9385
AverageAccuracy (%) (Eq. 8)	87.8	87.6	83.46	76.17	88.5
Resource type	Moderate	Poor	Poor	Poor	Rich

```
for each word $Wmid$ in sentence $S$ with context word pair $CW_p$ and $CW_s$ do:
    Initialize $PredClustset = \{\}$
    if $\exists$ cluster $C_i \in Clustset \mid Wmid \in C_i$:
        Find $PClustset = \{C_i \mid Wmid \in C_i\}$
        if $\exists$ cluster $C_j \in PClustset \mid CW_p$ and $CW_s$ pair is present as context word pair in cluster $C_j$:
            Find all such clusters from $PClustset$ and append to $PredClustset$    #Criteria 1
        else:
            if $\exists$ cluster $C_j \in PClustset \mid CW_p$ is present as preceding word in a context word pair in cluster $C_j$:
                Find all such clusters from $PClustset$ and append to $PredClustset$    #Criteria 2
            else: Append $PredClustset = PredClustset \cup PClustset$    #Criteria 3
    else:
        if $\exists$ cluster $C_i \in Clustset \mid CW_p$ and $CW_s$ pair is present as context word pair in cluster $C_i$:
            Find all such clusters from $Clustset$ and append to $PredClustset$    #Criteria 1
        else:
            if $\exists$ cluster $C_i \in Clustset \mid CW_p$ is present as preceding word in a context word pair in cluster $C_i$:
                Find all such clusters from $Clustset$ and append to $PredClustset$    #Criteria 2
            else: Return NOTAG
    $\forall C_i \in PredClustset$ Find $TagProb(C_i)$    // Use Equation (7)
    Find $C_{max}$ = cluster with highest $TagProb(C_i)$ value in $PredClustset$
    Find $C_{secmax}$ = cluster with second highest $TagProb(C_j)$ value in $PredClustset$
    Find $TagProbDif$    // Use Equation (6)
    if $TagProbDif \geq MinprobDif$: Return $PredTag$ = POS tag label of cluster $C_{max}$
    else: Return NOTAG
```

Algorithm 2: Tagging Algorithm – Classification method using set of clusters *Clustset* for tagging words of a sentence

Corpus[1] using Biber tag set [18]), resource-moderate Hindi language [4,5] and resource-poor Telugu[2] [4], Tamil[3] and Bengali[4] languages. Table 1 gives details of all the language datasets. All the five language datasets have flat tag sets present in annotated training and test sets without any hierarchy. The POS tag data distribution in the resource-moderate and resource-poor language datasets are highly imbalanced and sparse.

7.2 Performance analysis and observations

We observed that for the following set of threshold values *MinConfidence* = 60 %, *MinCoverage* = 60 % and *MinprobDif* = 30 %, the three parameters *Confidence*, *Coverage* and *TagProbDif* give best *AverageAccuracy* (defined below) values for all the five languages. Tables 1, 2, 3, 4, 5, 8 and 9 show the results for this set of threshold values for the respective parameters.

$$AverageAccuracy = \frac{\text{Number of correctly tagged test words}}{|\text{Test set}| - \text{No. of test words tagged as NOTAG}} \qquad (8)$$

where $|\text{Test set}|$ = No. of words in the test set.

[1] http://americannationalcorpus.org/FirstRelease/contents.html.

[2] Provided by IIIT Hyderabad, data is part of IL-ILMT project sponsored by MC&IT, Govt. of India Reference No: 11(10)/2006-HCC(TDIL).

[3] http://sanskrit.jnu.ac.in/ilci/index.jsp.

[4] http://sanskrit.jnu.ac.in/ilci/index.jsp.

Table 2 Tagging criteria selection and number of words correctly tagged using them by our tagging algorithm shown in Algorithm 2 for 70811 Hindi and 20854 Telugu test set words with classifier models built using training data specified in Table 1

Criteria	Test word in Clusters of classifier model				Test word not in any Cluster of classifier model (Unknown Words)			
	No. of Words Satisfying Criteria		No. of Words Correctly tagged		No. of Words Satisfying Criteria		No. of Words Correctly tagged	
	Hindi	Telugu	Hindi	Telugu	Hindi	Telugu	Hindi	Telugu
Criteria 1	15136	2604	14209	2581	1057	492	645	334
Criteria 2	27696	3500	25444	3355	3702	3716	821	1609
Criteria 3	22721	9522	19405	8959				

	Test Words tagged as NOTAG by our tagging algorithm	
	Hindi	Telugu
POS tag rejected due to *TagProbDif*	1417	614
No POS tag found	499	1020

Table 3 *AverageAccuracy* (%) values for all languages obtained by our tagger and CRF tagger for various annotated training set sizes ($\leq$ 25,000 words)

Lang.	Test set size	Annotated Training set size	CRF	Our Tagger		
			Average Accuracy (%)	Average Accuracy (%)	No. of NOTAG Words	Untagged Training set size
Hindi	20227	5730	74.6	**79.1**	3195	10025
		10030	78.4	**79.05**	2740	10025
		15771	81.3	**82.1**	2116	25020
		25591	84.7	**85.0**	1903	50019
Telugu	20854	4994	59.4	**81.4**	5617	9994
		9994	67.1	**82.6**	3897	23435
		14995	71.2	**84.4**	3240	43434
		23435	75.3	**84.3**	2419	104281
Tamil	22352	5006	48.9	**75.0**	6957	40988
		9941	59.9	**79.7**	4357	80004
		15007	65.9	**82.1**	3778	80004
		20207	69.4	**83.1**	3495	80004
Bengali	20618	5010	47.3	**73.5**	7081	49997
		10003	56.2	**74.6**	5332	85796
		15009	59.3	**75.2**	4269	85796
		21561	63.0	**77.8**	4170	85796
English	24952	10671	70.4	**79.7**	3774	50444
		15298	72.9	**82.3**	3424	50444
		24825	76.4	**82.5**	2574	93679

Table 2 shows that for both known and unknown test words, highest percentage of correct tagging is done by giving first priority to presence of context word pair in the cluster (**Criteria 1**). Here, *known* words means test set words which are present in untagged training set and *unknown* word means unseen test set words which are not present in the untagged training set. Note that words of annotated training set are not included in the classifier model, only their tags are used indirectly while building the model. This trend was observed for all the five languages. In the results shown in Table 1, around 46 % unknown English words, 60 % unknown Hindi words, 67 % unknown Telugu words, 52 % unknown Bengali words and 57 % unknown Tamil words were correctly tagged using their context word pair. This shows the strength of our tagger to tag unknown words without using any smoothing technique used by other POS taggers.

In Table 3, we compare our results with a supervised CRF[5] tagger [20]. This tagger uses words, their POS tag and context word pair information from annotated training data, while our tagger uses words and their context word pair information from untagged training data and POS tag information

[5] http://crfpp.googlecode.com/svn/trunk/doc/index.html, CRF model outputs tag for all test words. So, for CRF tagger AverageAccuracy = (No. of correctly tagged test words)/(No. of test words).

Table 4 Effect of annotated training data size on classifier model of Tamil language for 22,352 test set words built with 169,705 untagged words

No. of words in annotated set	No. of clusters (unique words) in model	No. of NOTAG test words	*Average accuracy* (%)
5006	22 (2021)	5317	72.2
9941	25 (3553)	3575	79.0
15,007	26 (4842)	2940	82.09
20,207	26 (5774)	2647	83.46

Table 5 Effect of untagged training data size on classifier model built using 282,548 Hindi and 14,995 Telugu annotated training set words for 70,811 Hindi and 20,854 Telugu test set words

Lang.	No. of words in untagged set	No. of clusters (unique words) in model	No. of NOTAG test words	*Average accuracy* (%)
Hindi	50019	25 (4366)	4714	87.3
	98,331	28 (6081)	3664	87.7
	128,329	28 (6865)	3220	87.9
	158,337	29 (7546)	2890	87.9
	188,326	29 (8112)	2793	88.0
	196,659	29 (8260)	2748	88.0
	282,554	30 (9517)	2484	88.0
	294,979	30 (9663)	2450	88.1
	393,303	30 (10817)	1916	87.8
Telugu	23435	24 (4619)	3318	84.4
	43,434	24 (4646)	3240	84.4
	63,436	24 (4629)	2962	83.7
	83,442	24 (4700)	2864	83.5
	104,281	23 (2799)	2032	82.3

from annotated training data. We observe that for annotated training data size $\leq$ 25000 words, our tagger gives better *AverageAccuracy* than CRF tagger. Our tagger also gives better POS tag precisions and better tagging accuracies than CRF tagger for unknown words and performance improves by increasing the untagged training data size up to a certain size. This shows that our tagger can be a better choice for the resource-poor languages. Also, as an additional benefit the model made by our tagger is more human understandable than that made by CRF tagger.

7.3 Effect of annotated (POS tagged) training data size

For 22,352 Tamil test set words we varied the size of annotated training set while keeping the untagged training set constant at 169,705 words (see Table 4). We observed that the coverage of words by the clusters (number of unique words in the cluster set) in the classifier model increases with the increase in the size of annotated training data. This happens because more *context-based lists* qualify for getting tagged which gets them included in cluster set and which in turn increases the number of unique words captured by the cluster set. This increases the tagging accuracy while the number of words missed by the model (tagged as "NOTAG") decreases. For all the five languages, we observed that increasing the annotated training data size improves cluster quality which increases the *AverageAccuracy* values but only up to a certain size. We also observed that there is only a slight decrease in *AverageAcuracy* value with decrease in annotated training set size, so performance does not decrease drastically when the annotated training set is made smaller. Our tagger gives above 70 % *AverageAccuracy* with annotated training data size as low as 5000 words and untagged training data size 10,000 words for all the languages. This justifies the use of small annotated training set to build a semi-supervised POS tagging model for the resource-poor languages.

7.4 Effect of untagged (raw) training data size

In Tables 1, 3 and 5 (for 70,811 Hindi and 20,854 Telugu test set words, using 282,548 Hindi and 14,995 Telugu annotated training set words), we observe that increasing the untagged training data size initially increases word coverage of clusters (number of unique words in the cluster set) which in turn increases the *AverageAccuracy* values but stabilizes after a certain size. After this if we increase the untagged data size then it decreases the performance. For all the five languages, we observed that while keeping the size of annotated training set constant if we increase the size of untagged training

Table 6 List of 35 POS tags present in Hindi annotated training set of 282,548 words and distribution of tagged words in it

JJ	**NEG**	**QCC**	**DEM**	**NULL**	**WQ**
15194	2022	625	3748	4	292
CCC	**RP**	**NN**	**NNC**	**VGF**	**VAUX**
7	4191	56050	5888	1	17716
PRP	**RB**	**NSTC**	**NST**	**QC**	**NNP**
11767	1260	21	4201	5017	22325
INTF	**RDP**	**CC**	**VM**	**UNKC**	**UNK**
274	207	11147	30011	1	141
RBC	**QFC**	**PRPC**	**VMC**	**SYM**	**SYMC**
21	16	45	40	19342	1
NNPC	**INJ**	**QF**	**PSP**	**JJC**	**QO**
12212	7	2160	55815	284	495

Table 7 List of 28 POS tags present in Telugu annotated training set of 83,442 words and distribution of tagged words in it

WQ	**DEM**	**JJ**	**RP**	**NN**	
1215	1004	1843	495	25047	
ECH	**CC**	**PRP**	**RB**	**NP**	
3	451	4241	1934	1	
NST	**NNP**	**INTF**	**RDP**	**CO**	**CL**
1906	2901	448	221	1	44
VAUX	**VM**	**INTJ**	**UNK**	**NNPP**	**QO**
532	17631	5	28	4	97
UT	**SYM**	**QC**	**INJ**	**QF**	**PSP**
781	18975	1571	110	658	1295

Table 8 Precision and Recall values (in %) of 32 POS tags of 70,811 Hindi test set words obtained by our classifier model built using 282,548 annotated and 393,303 untagged training set words

JJ	**NEG**	**QCC**	**DEM**
90.3, 85.9	**98.4, 98.8**	**46.1, 3.1**	**78.6, 94.5**
RP	**NN**	**NNC**	**VAUX**
97.3, 88.4	**82.0, 91.7**	**57.2, 23.8**	**76.0, 95.8**
PRP	**RB**	**NST**	**QC**
95.2, 87.3	**88.3, 76.8**	**97.8, 96.5**	**77.6, 85.4**
INTF	**CC**	**VM**	**UNK**
66.6, 80.0	**94.0, 95.1**	**95.3, 72.9**	**42.8, 11.5**
NNPC	**QF**	**PSP**	**JJC**
63.3, 53.2	**90.1, 91.1**	**96.2, 98.5**	**58.3, 25.9**
SYM	**NNP**	**QO**	**WQ**
99.2, 97.6	**82.3, 64.4**	**84.3, 82.2**	**96.0, 92.3**
RBC, QFC, PRPC, VMC, INJ, CCC, NSTC, RDP			
0, 0			

data then the coverage of words by the clusters in the classifier model increases up to a certain size. This accounts for the increase in tagging accuracy and decrease in the number of words missed by the model (tagged as “NOTAG”). Also, Table 2 shows that for Hindi with higher untagged training data size than Telugu, the number of test words for which there is no POS tag present in the cluster set is far less than Telugu. Other interesting observation is that *AverageAccuracy* does not vary much as the untagged training data size varies, so our algorithm is able to perform well even with a small-sized untagged training data.

Table 9 Precision and Recall values (in %) of 24 POS tags of 20,854 Telugu test set words obtained by our classifier model built using 83,442 annotated and 104,281 untagged training set words

DEM	**JJ**	**RP**	**NN**
96.19, 99.6	**81.38, 86.64**	**78.45, 70**	**82.07, 82.97**
ECH	**CC**	**PRP**	**RB**
0, 0	**83.43, 86.39**	**90.05, 90.78**	**78.24, 64.53**
NNP	**INTF**	**RDP**	**CL**
95.59, 17.22	**50.55, 93.88**	**40, 2.41**	**85.71, 94.74**
VM	**QO**	**PSP**	**UNK**
88.33, 69.96	**88.46, 74.19**	**91.14, 94.74**	**0, 0**
SYM	**QC**	**INJ**	**QF**
95.78, 100	**94.39, 79.57**	**91.66, 64.71**	**78.31, 71.82**
WQ	**NST**	**VAUX**	**UT**
92.47, 84.77	**84.6, 79.72**	**55, 5.82**	**95.67, 100**

7.5 Effect of various parameter values on classifier model

We made following observations about the effect of parameter values: (1) For parameter *Confidence*, increasing threshold value of *MinConfidence* increases the quality of clusters but at the same time it also increases the number of *context-based lists* tagged as “NOTVALIST” which decreases the word coverage of clusters. (2) Decreasing threshold value of *MinCoverage* for parameters *Coverage* although decreases the quality of clusters (by allowing inclusion of non-related words in a *context-based list*) but at the same time it increases the word coverage of clusters by decreasing the number of *context-based lists* tagged as “NOTVALIST”. (3) By varying the threshold value of *MinprobDif* from 5 to 30 % for parameter *TagProbDif*, we found that increasing the threshold value increases the Precision values of POS tags but slightly decreases their Recall because the number of words tagged as “NOTAG” increases. Practical advantage of this parameter is that it ensures that tagging of ambiguous and non-confident cases is avoided. (4) The number of POS tag clusters obtained in the classifier model is almost independent of the selected threshold values of the parameters. For the training data sizes given in Table 1 and for the range of threshold values *MinConfidence* = 60–90 % and *MinCoverage* = 0–75 %, number of POS tag clusters found for English was 100–101, for Hindi was 29–31, for Tamil was 22–26, for Bengali was 25 and for Telugu was 23. We noted that the POS tags missing from the set of clusters were the rare POS tags having very low frequencies.

7.6 Complexity analysis

Our tagger’s complexity is linear, $O(N)$, where N is the number of words in untagged training data. We note that while making the trained model, time required to build the

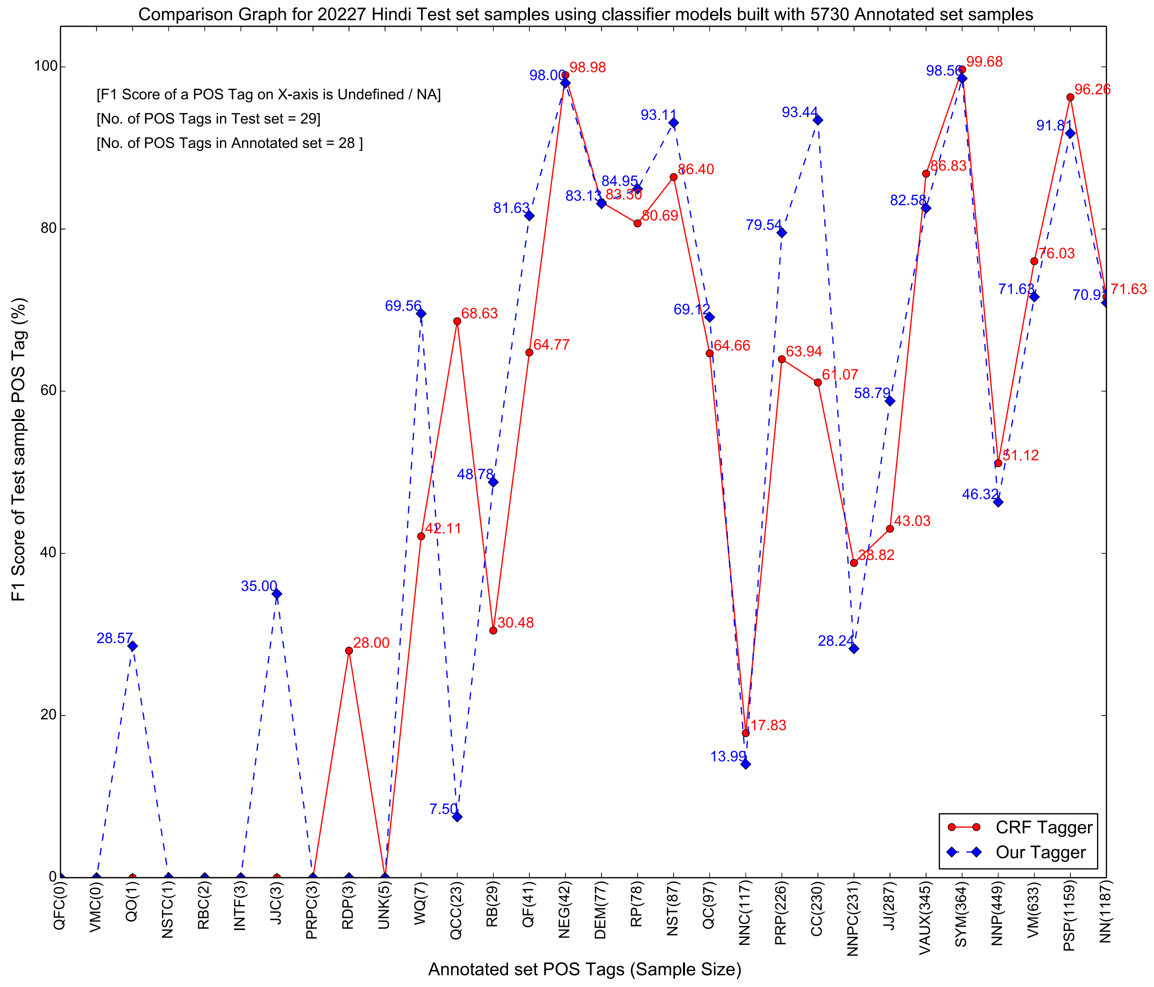

Fig. 1 F_1 *Score* values (in %) for 29 test set POS tags of Hindi (listed in Table 10) obtained by CRF tagger and our tagger using annotated training set of 5730 words with 393,303 untagged training set words

context-based lists constitutes the most time taking section of our algorithm. This section of the algorithm decides the overall complexity of our tagger stated above. Merging of the lists is also linear, $O(n)$, where, n is the number of obtained *context-based list*, but $n \ll N$. Tagging of the test words is an instant lookup in the trained model and hence constant, $O(1)$. So, bulk of the time is taken by model building section, in which *context-based list* building takes most of the time since it is linear in size of untagged data. Since, tagging time is almost instantaneous for a test word, our tagger is efficient.

8 Case study of resource-moderate Hindi and resource-poor Telugu languages

We present here, the detailed results obtained by applying our POS tagging method on *resource-moderate* Hindi and *resource-poor* Telugu languages and compare them with results obtained by applying supervised CRF[6] tagger [20].

List of POS tags and tag-wise distribution of words present in Hindi and Telugu annotated training data for the data sizes given in Table 1 are shown in Tables 6 and 7, respectively. Please note that "NULL" tag present in Hindi annotated and test sets is noise and hence it is not counted in the available tag set and is not included in the results. It is clear from the tables that there is a *class imbalance problem* in the data and we handle it using a specific method (discussed in Sect. 4.2) in our algorithm.

Precision and *Recall* values obtained for each POS tag of 70,811 Hindi and 20,854 Telugu test set words by our classifier models built using training data sizes specified in Table 1 (282,548 Hindi and 83,442 Telugu annotated training set words with 393,303 Hindi and 104,281 Telugu untagged

[6] http://crfpp.googlecode.com/svn/trunk/doc/index.html. For CRF tagger AverageAccuracy = (No. of correctly tagged test words)/(No. of test words).

Table 10 List of 29 POS tags present in Hindi test set of 20,227 words and distribution of tagged words in it

JJ	**NULL**	**QCC**	**DEM**	**WQ**
1172	5	64	244	15
RP	**NN**	**NNC**	**VAUX**	**NNP**
305	4053	443	1310	1557
PRP	**RB**	**VMC**	**NST**	**QC**
730	85	5	324	341
INTF	**RDP**	**CC**	**VM**	**UNK**
16	12	825	2056	2
RBC	**QFC**	**PRPC**	**NEG**	**SYM**
2	1	1	149	1408
NNPC	**QF**	**PSP**	**JJC**	**QO**
867	180	3994	31	30

training set words) are shown in Tables 8 and 9, respectively. Their *AverageAccuracy* values are already shown in Table 1. In each cell of these tables, the first value below the POS tag name gives Precision and second value gives Recall. It is well known that *Precision* is the number of correctly predicted positive results divided by the number of all positive results, and *Recall* is the number of correctly predicted positive results divided by the number of positive results that should have been predicted.

For Hindi, using training set data of Table 1, the obtained classifier model cluster set contained 30 POS tag clusters with 10,817 unique words (see Table 5). Analyzing the results in Table 8, we can say that, except for "RDP", only very rare POS tag classes, like "RBC", "QFC", "PRPC", "NSTC," etc., which have number of samples below 45 in the annotated training set (see Table 6) are missed by our tagger. It may be noted that the POS tag "RDP" is given to duplicated words of other POS tags. It is a difficult POS tag class because it has total overlapping with other tags. Apart from these, for almost all other tags our tagger performed well, although they also are not highly represented in the annotated set (see Table 6).

For Telugu, using training set data mentioned in Table 1, the obtained classifier model cluster set had 23 POS tag clusters with 7180 unique words. Results in Table 9 show that, excluding "ECH" and "UNK", which are exceptional POS

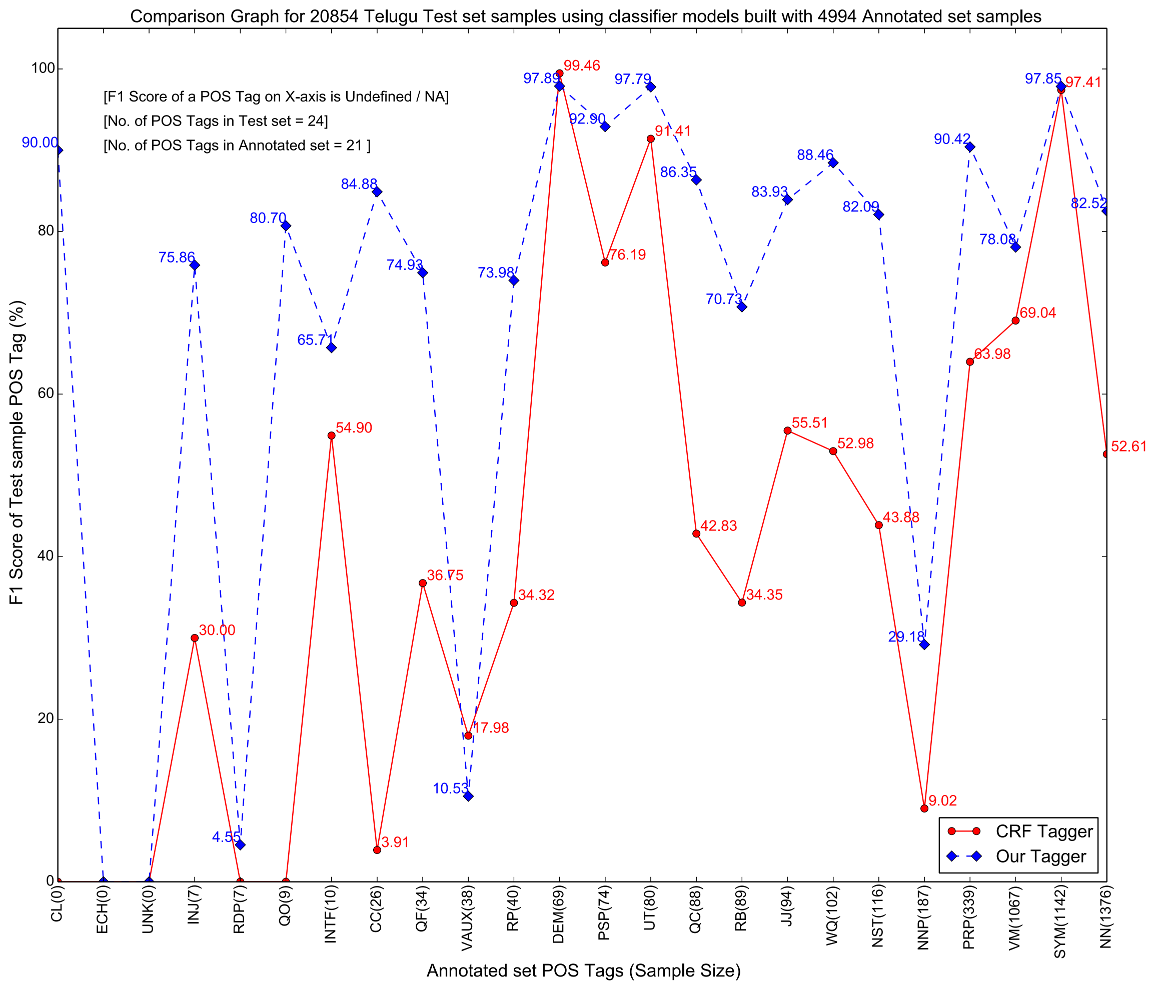

Fig. 2 F_1 *Score* values (in %) for 24 test set POS tags of Telugu (listed in Table 11) obtained by CRF tagger and our tagger using annotated training set of 4994 words with 104,281 untagged training set words

Table 11 List of 24 POS tags of 20,854 Telugu test set words and distribution of tagged words in it

WQ	**DEM**	**JJ**	**RP**	**NN**
348	279	464	130	6071
ECH	**PSP**	**CC**	**PRP**	**RB**
3	380	169	1247	468
CL	**NST**	**NNP**	**INTF**	**RDP**
19	503	755	98	83
UNK	**VAUX**	**VM**	**QO**	
1	189	4481	31	
UT	**SYM**	**QC**	**INJ**	**QF**
199	4273	465	17	181

tag classes, our tagger performed well although the POS tag classes had minimalistic representation (see Table 7) in this resource-poor language dataset. This proves the utility of our POS tagging method for resource-poor languages.

Figures 1 and 2 present comparison graphs of F_1 *Scores* (defined below) of our tagger and CRF tagger for Hindi and Telugu, respectively. F_1 *Score* of a POS tag T is the harmonic mean of its Precision and Recall values and is defined as follows:

$$F_1\ \textit{Score}\text{ of T} = \frac{2 \cdot (\text{Precision of T}) \cdot (\text{Recall of T})}{(\text{Precision of T}) + (\text{Recall of T})} \quad (9)$$

Figure 1 shows comparison graph for 20,227 Hindi test set words containing 29 POS tags (see Table 10). In this graph, the CRF tagger's values were obtained from a classifier model built with training set of 5730 annotated words, while our tagger used 393,303 untagged training set words along with this annotated set to build the classifier model. Our tagger obtained 81.1 % *AverageAccuracy* while CRF tagger obtained 74.67 %.

Figure 2 presents comparison graph of F_1 *Scores* of our tagger and CRF tagger for Telugu test set of 20,854 words with 24 POS tags (see Table 11). Our tagger's classifier model was built using 4994 annotated training set words along with 104,281 untagged training set words. CRF tagger used 4994 annotated training set words to build the classifier model. Our tagger obtained 77 % *AverageAccuracy* while CRF tagger obtained 59.4 %.

Since our motivation is to provide a practical good tagger for resource-poor languages, we present comparison results of small annotated training data sizes for both the languages. Figure 1 shows that, in case of Hindi language, except for "RDP" and "QCC", overall performance of our tagger is better than CRF tagger. Figure 2 clearly shows that for Telugu our tagger outperforms CRF tagger. In both the cases, our tagger performs well for maximum number of POS tags although the annotated training data sizes are considerably small.

9 Conclusions and future work

In this work, we developed a semi-supervised associative classification method for POS tagging. We used the concept of context-based list and context-based association rule mining. We also developed a method to find interestingness measures required to find the association rules in a semi-supervised manner from a combined training set of annotated and untagged data. We showed that our tagger gives good performance for resource-rich as well as resource-poor languages without using extensive linguistic knowledge. It works well even with less annotated and untagged training data. It can also tag unknown words. These advantages make it very suitable for resource-poor languages and can be used as an initial POS tagger while developing linguistic resources for them.

Future work includes (1) using other contexts instead of trigram, (2) finding methods to include linguistic features in the current approach, (3) mining tagging patterns from the clusters to find tag of a test word and (4) using this approach for other lexical item classification tasks.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD'93, pp. 207–216 (1993)
2. Avinesh, P.V.S., Karthik, G.: Part of speech tagging using conditional random fields and transformation based learning. In: Proceedings of IJCAI Workshop SPSAL, pp. 21–24 (2007)
3. Banko, M., Moore, R.C.: Part-of-speech tagging in context. In: COLING (2004)
4. Bharati, A., Sharma, D.M., Bai, L., Sangal, R.: AnnCorra: annotating corpora guidelines for POS and chunk annotation for Indian languages. Tech. Rep. TR-LTRC-31, Language Technologies Research Centre, IIIT, Hyderabad (2006)
5. Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D.M., Xia, F.: A multi-representational and multi-layered treebank for Hindi/Urdu. In: Proceedings of the Third Linguistic Annotation Workshop, pp. 186–189. ACL (2009)
6. Biemann, C.: Unsupervised part-of-speech tagging employing efficient graph clustering. In: ACL (2006)
7. Brants, T.: TnT: a statistical part-of-speech tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing, pp. 224–231 (2000)
8. Brill, E.: A simple rule-based part of speech tagger. In: ANLP, pp. 152–155 (1992)
9. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, SIGMOD '97, pp. 255–264 (1997)
10. Cutting, D., Kupiec, J., Pedersen, J., Sibun, P.: A practical part-of-speech tagger. In: Proceedings of the Third Conference on Applied Natural Language Processing, pp. 133–140 (1992)
11. Dandapat, S., Sarkar, S., Basu, A.: Automatic part-of-speech tagging for Bengali: an approach for morphologically rich languages in a poor resource scenario. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, pp. 221–224 (2007)

12. Dhanalakshmi, V., Kumar, A., Shivapratap, G., Soman Kp, R.S.: Tamil POS tagging using linear programming. Int. J. Recent Trends Eng. **1**(2), 166–169 (2009)
13. Dubey, H., Pudi, V.: Class Based Weighted K-Nearest Neighbor Over Imbalance Dataset. PAKDD **2**, 305–316 (2013)
14. Ekbal, A., Hasanuzzaman, M., Bandyopadhyay, S.: Voted Approach for Part of Speech Tagging in Bengali. In: PACLIC, pp. 120–129 (2009)
15. Gadde, P., Yeleti, M.V.: Improving statistical POS tagging using Linguistic feature for Hindi and Telugu. In: International Conference on Natural Language Processing (ICON) (2008)
16. Gimenez, J., Marquez, L.: SVMTool: a general POS tagger generator based on support vector machines. In: Proceedings of the 4th International Conference on Language Resources and Evaluation, pp. 43–46 (2004)
17. Goldwater, S., Griffiths, T.: A fully Bayesian approach to unsupervised part-of-speech tagging. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 744–751 (2007)
18. Ide, N., Suderman, K.: The American National Corpus First Release. In: LREC, pp. 1681–1684 (2004)
19. Kamruzzaman, S.M., Haider, F., Hasan, A.R.: Text Classification Using Association Rule with a Hybrid Concept of Naive Bayes Classifier and Genetic Algorithm. CoRR abs/1009.4976 (2010)
20. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, pp. 282–289 (2001)
21. Li, W., Han, J., Pei, J.: CMAR: accurate and efficient classification based on multiple class-association rules. In: Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM '01, pp. 369–376. IEEE Computer Society (2001)
22. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Knowledge Discovery and Data Mining, pp. 80–86 (1998)
23. Nakov, P., Ng, H.T.: Improved statistical machine translation for resource-poor languages using related resource-rich languages. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3, EMNLP '09, pp. 1358–1367. Association for Computational Linguistics (2009)
24. Rani, P., Pudi, V., Sharma, D.M.: A semisupervised associative classification method for POS tagging. In: IEEE International Conference on Data Science and Advanced Analytics, DSAA 2014, pp. 156–162 (2014)
25. Shaohong, Y., Guidan, F.: Research of POS tagging rules mining algorithm. Appl. Mech. Mater. **347**, 2836–2840 (2013)
26. Shrivastava, M., Bhattacharyya, P.: Hindi POS tagger using naive stemming: harnessing morphological information without extensive linguistic knowledge. In: International Conference on NLP (ICON) (2008)
27. Søgaard, A.: Simple semi-supervised training of part-of-speech taggers. In: ACL, pp. 205–208 (2010)
28. Søgaard, A.: Semisupervised condensed nearest neighbor for part-of-speech tagging. In: Proceedings of ACL HLT: Short Papers, Volume 2, pp. 48–52 (2011)
29. Soni, S., Vyas, O.: Using associative classifiers for predictive analysis in health care data mining. Int. J. Comput. Appl. **4**(5), 33–37 (2010)
30. Subramanya, A., Petrov, S., Pereira, F.: Efficient graph-based semi-supervised learning of structured tagging models. In: Proceedings of EMNLP'10, pp. 167–176 (2010)
31. Thabtah, F.: A review of associative classification mining. Knowl. Eng. Rev. **22**(1), 37–65 (2007)
32. Thonangi, R., Pudi, V.: ACME: an associative classifier based on maximum entropy principle. In: ALT, pp. 122–134 (2005)
33. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of NAACL HLT'03—Volume 1, pp. 173–180 (2003)
34. Yarowsky, D.: One sense per collocation. In: Proceedings of the Workshop on Human Language Technology, pp. 266–271 (1993)
35. Yin, X., Han, J.: CPAR: classification based on predictive association rules. In: SDM, pp. 331–335 (2003)
36. Zaïane, O.R., Antonie, M., Coman, A.: Mammography classification by an association rule-based classifier. In: MDM/KDD, pp. 62–69 (2002)
37. Zhou, Z.H., Li, M.: Tri-training: exploiting unlabeled data using three classifiers. IEEE Trans. Knowl. Data Eng. **17**(11), 1529–1541 (2005)