Chapter IV

Part of Speech Tagger for Kashmiri

4.0 Introduction

Corpus based natural language processing (NLP) tasks for popular languages like English, French etc. have been much worked on with success. On the contrary, very little or rather no work has been done on languages like Kashmiri which are at the primary level in the NLP realm. One of the main reasons is the absence of annotated corpus for such languages. Corpus annotation is the practice of adding interpretative especially linguistic information to a text corpus by coding, added to the electronic representation of the text itself.

A typical case of corpus annotation is that of morphosyntactic annotation (also called grammatical tagging), whereby a label or tag is associated with each word token in the text to indicate its grammatical classification. Annotated corpora serves as an important tool for investigators of natural language processing, speech recognition and other related areas. It is a basic building block for constructing statistical models for automatic processing of natural languages. Keeping in view the importance of NLP tasks, and in order to overcome the shortage of the annotated corpus for Kashmiri an attempt is made to build an annotated corpus for Kashmiri so that the ultimate goal of developing an automatic tagger is fulfilled.

A large number of current language processing systems use a part-of-speech tagger for pre-processing. The tagger usually annotates the given word or a token. In other words, a POS tagger assigns a (unique or ambiguous) part-of-speech tag to each token in the input and then passes it to the next processing

level (chunking, parsing etc.). Part-of-speech tagging is also important for corpus annotation projects, with the help of which valuable linguistic resources are created by a combination of automatic processing and human correction.

For both these applications, a tagger with the highest possible accuracy is required. The debate over the issue of which tagger solves the parts of speech problem in the best way is not over. Several approaches have been used to construct automatic taggers. Most of the work done is based on statistical methods using n-gram models for Hidden Markov Model–based tagger (Church 1988; De rose 1988; Cutting et al 1992; Merialdo 1994; Kupiec 1992; Brill 1992 and Voutilainen et al 1992). In these approaches a tag sequence is chosen for a sentence that maximizes the product of lexical and contextual probabilities as estimated from a tagged corpus.

4.1 Hidden Markov Model

A Hidden Markov Model (HMM) is a statistical construct that can be used to solve classification problems that have an inherent state sequence representation. The model includes an interconnected set of *states* which are connected by a set of *transition probabilities*. Transition probabilities indicate the probability of traveling between two given states. A process starts at a particular state and moves to a new state as governed by the transition probabilities in discrete time intervals. As the process enters into a state one of a set of *output symbol* (also known as *observation*) is emitted by the process. The symbol emitted, is dependent on the probability distribution of the particular state. The

output of the HMM is a sequence of output symbols. In an HMM, the exact state sequence corresponding to a particular observation sequence is unknown (i.e. *hidden*).

A POS tagger based on Hidden Markov Model (HMM) (Jurafsky and Martin, 2000) assigns the best sequence of tags to an entire sentence. Generally, the most probable tag sequence is assigned to each sentence following the Viterbi algorithm (Viterbi, 1967). The task of Part of Speech (POS) tagging is to find the sequence of POS tags $T = \{t_1, t_2, t_3, \ldots t_n\}$ that is optimal for a word sequence $W = \{w_1, w_2, w_3 \ldots w_n\}$. The tagging problem becomes equivalent to searching for $\text{argmax}_T P(T) * P(W \mid T)$, by the application of Bayes' law.

The probability of the tag i.e., P(T) can be calculated by Markov assumption which states that the probability of a tag is dependent only on a small, fixed number of previous tags. We have used tri-gram model, i.e., the probability of a tag depends on two previous tags, and then we have,

$$P(T) = P(t_1) * P(t_2 \mid t_1) * P(t_3 \mid t_1, t_2) * P(t_4 \mid t_2, t_3) * \ldots * P(t_n \mid t_n - 2 , t_n - 1).$$

An additional tag '$' (dummy tag) has been introduced in this work to represent the beginning of a sentence. So, the previous probability equation can be slightly modified as:

$$P(T) = P(t_1 \mid \$) * P(t_2 \mid \$, t_1) * P(t_3 \mid t_1, t_2) * P(t_4 \mid t_2, t_3) * \ldots * P(t_n \mid t_n - 2 , t_n - 1)$$

Due to sparse data problem the linear interpolation method has been used to smooth the trigram probabilities as follows:

$$P'(t_n \mid t_n - 2 , t_n - 1) = \lambda_1 P(t_n) + \lambda_2 P(t_n \mid t_n\text{-}1) + \lambda_3 P(t_n \mid t_n - 2 , t_n - 1)$$

such that the λ_s sum to 1.

The values of λ_s have been calculated by the following method (Brants, 2000):

set $\lambda_1 = \lambda_2 = \lambda_3 = 0$

for each tri-gram t_1, t_2, t_3 with freq(t_1, t_2, t_3)>0

depending on the maximum of the following three values:

case : $\frac{freq(t_1,t_2,t_3)-1}{freq(t_1,t_2)-1}$: $increment\ \lambda_3\ by\ freq(t_1, t_2, t_3)$

case : $\frac{freq(t_2,t_3)-1}{freq(t_2)-1}$: $increment\ \lambda_2\ by\ freq(t_1, t_2, t_3)$

case : $\frac{freq(t_3)-1}{N-1}$: $increment\ \lambda_1\ by\ freq(t_1, t_2, t_3)$

end

normalize $\lambda_1, \lambda_2, \lambda_3$

Here, N is the corpus size, i.e., the number of tokens present in the training corpus. If the denominator in one of the expression is 0, we define the result of that expression to be 0. The -1 in both the numerator and denominator has been considered for taking unseen data into account. By making the simplifying assumption that the relation between a word and its tag is independent of context, we can simplify P(W | T) as the following equation:

$$P(W \mid T) \approx P(w_1 \mid t_1) * P(w_2 \mid t_2) * \ldots * P(w_n \mid t_n)$$

The emission probabilities in the above equation can be calculated from the training set as:

$$\text{Emission Probability: } P(wi \mid ti) = \frac{freq(w_i, t_i)}{freq(t_i)}$$

4.1.1 Context Dependency

To make the Markov model more powerful, additional context dependent feature has been introduced to the emission probability in this work that specifies the probability of the current word depends on the tag of the previous word and the tag to be assigned to the current word. Now, we calculate P(W | T) by the following equation:

$$P(W \mid T) \approx P(w_1 \mid \$, t_1) * P(w_2 \mid t_1, t_2) * \ldots * P(w_n \mid t_{n-1}, t_n)$$

So, the emission probability can be calculated as

$$P(w_i \mid t_{i-1}, t_i) = \frac{freq(t_{i-1}, t_i, w_i)}{freq(t_{i-1}, t_1)}$$

Here also the smoothing technique is applied rather than using the emission probability directly. The emission probability is calculated as:

$$P'(w_i \mid t_{i-1}, t_i) = \theta_1 P(w_i \mid t_i) + \theta_2 P(w_i \mid t_{i-1}, t_i)$$

where θ_1, θ_2 are two constants such that all θs sum to 1.

The values of θs should be different for different words. But the calculation of θs for every word takes a considerable time and hence θs are calculated for the entire training corpus. In general, we can calculate the values of θs by the following method like λs:

set $\theta_1 = \theta_2 = 0$

for each bi-gram t_1, t_2 with $freq(t_1, t_2) > 0$

depending on the maximum of the following two values:

case : $\frac{freq(t_2,t_3)-1}{freq(t_2)-1}$: $increment\ \theta_2\ by\ freq(t_1,t_2,t_3)$

case : $\frac{freq(t_3)-1}{N-1}$: $increment\ \theta_1\ by\ freq(t_1,t_2,t_3)$

end

normalise θ_1, θ_2

Now, the emission probability and transition probability have been joined together to set up the modified Hidden Markov model as shown in below:

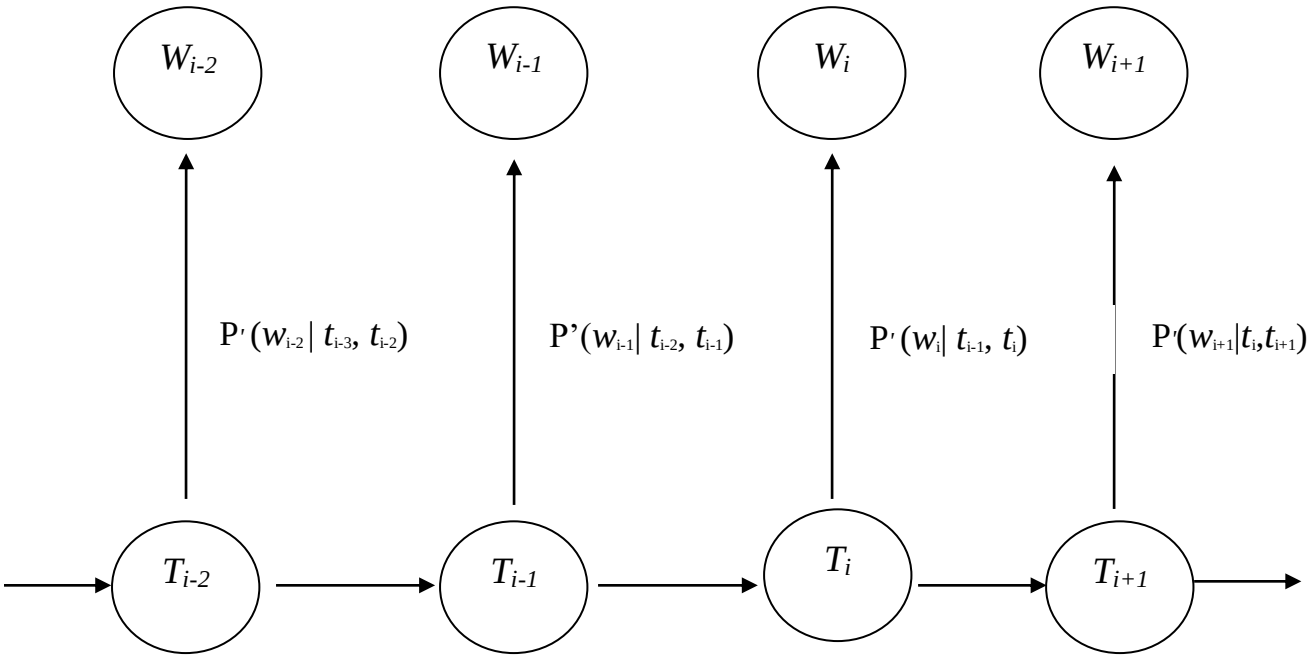

Figure 4.1: Modified Hidden Markov Model

4.1.2 Viterbi Algorithm

We now know how to derive the probabilities needed for the Markov model, and how to calculate P(T | W) for any particular (T, W) pair. But what we really need is to be able to find the most likely T for a particular W. The Viterbi algorithm (Viterbi, 1967) allows us to find the best T in the linear time. The idea behind the algorithm is that of all the state sequences, only the most probable of these sequences need to be considered. The trigram model has been used in the present work. The pseudo code of the algorithm is shown below.

For*i* = 1 to Number_of_Words_in_Sentence

for each state *c* ∈ Tag_Set

for each state *b* ∈ Tag_Set

for each state *a* ∈ Tag_Set

For the best state sequence ending in state *a* at time (*i* -2), *b* at time (i-1), compute the probability of that state sequence going to state *c* at time *i*.

end

end

end

Determine the most-probable state sequence ending in state *c* at time *i*.

end

So if every word can have *S* possible tags, then the Viterbi algorithm runs in **O**(*S*3*|*W*|) time, or linear time with respect to the length of the sentence.

4.1.3 Handling the Unknown Words

Handling of unknown words is an important issue in POS tagging. For words which have not been seen in the training set, $P(w_i \mid t_i)$ is estimated based on features of the unknown words, such as whether the word contains a particular suffix. The list of suffixes will also be created. The linguistic rules will also be used to assign the most possible tag to the unknown words.

4.2 Our Approach

We have used an HMM for automatic POS tagging of natural language text. The HMM uses three components and are depicted in Figure 4.2. First, the system requires some knowledge about the task of POS disambiguation. The knowledge may come from several resources and can be encoded in various representations. We call this representation as language model. In particular, to HMM, the language model is represented by the model parameters $\mu = (\pi, A, B)$. We aim to estimate the model parameters $\mu = (\pi, A, B)$ of the HMM using corpora. The model parameters of the HMM are estimated based on the labeled data during supervised learning. Unlabeled data are used to re-estimate the model parameters during semi-supervised learning. The taggers will be implemented based on both bigram and trigram HMM models.

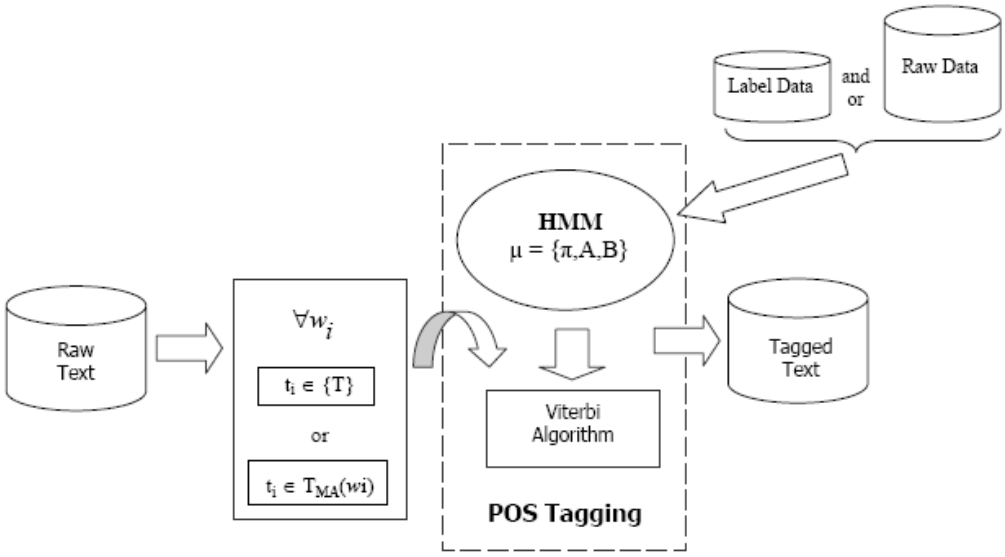

Figure 4.2: The HMM based POS tagging architecture

Secondly, there is a disambiguation algorithm, which decides the best possible tag assignment for every word in a sentence according to the language model. We use Viterbi algorithm for disambiguation. The third component estimates the set of possible tags {T}, for every word in a sentence. We shall call this as possible class restriction module. This module consists of a list of lexical units associated with the list of possible tags. In our approach, we first assume that every word can be associated with all the tags in the tagset. Further, we assume the POS tag of a word 'w' can take the values from the set T(w). These three components are related and we combine them into a single tagger description. The input to the disambiguation algorithm takes the list of lexical units with the associated list of possible tags. The disambiguation module provides the output tag for each lexical unit using the encoded information from the language model. The following subsections give a detailed design of the above three components in our work.

4.2.1 Models

There are several ways of representing the HMM based model for automatic POS tagging according to the way we acquire knowledge. The HMM models use the following three sources of information.

a. *Symbol emission probabilities*, i.e. the probability of a particular tag t_i, given a particular word w_i, $P(w_i \mid t_i)$.
b. *State transition probabilities*, i.e. the probability of a particular tag depending on the previous tags, $P(t_i \mid t_{i-1}t_{i-2}.....t_{i-k})$.
c. *Probability for the initial sate*, i.e. the probability of a particular tag as an initial state of a Markov model

The above parameters can be estimated using only labeled data during supervised learning. We shall call this model **HMM-S**. Further, semi-supervised learning can be performed by augmenting the labeled data with additional unlabeled data. We shall call this model **HMM-SS.**

The state transition probabilities are often estimated based on previous one (first-order or bigram) or two (second-order or trigram) tags. Depending on the order of the symbol transition probability we shall call the Markov process as first-order (**HMM1**) and second-order (**HMM2**) Markov process respectively. We adopt four different Markov models for representing the language model: (1) *Supervised first-order HMM* (**HMM-S1**) (2) *Semi-supervised first-order HMM* (**HMM-SS1**) (3) *Supervised second-order HMM* (**HMM-S2**) (4) *Semi-supervised second-order HMM* (**HMM-SS2**).

4.2.1.1 Supervised HMM (HMM-S)

In this model, the model parameters are estimated using only labeled training data. In a *k*-th order Markov model, the state transition probability of a particular tag t_i depends on the previous *k-1* tags in the sequence, $P(t_i \mid t_{i-1}t_{i-2}...t_{i-k-1})$. In the *supervised first-order HMM* (HMM-S1), the state transition probability of a particular tag t_i depends only on the previous tag t_{i-1}(i.e. $P(t_i \mid t_{i-1})$). The symbol emission and state transition probabilities are estimated directly from the labeled training data as follows.

$P(t_i \mid t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$ and $P(w_i \mid t_i) = \frac{C(w_i, t_i)}{C(t_i)}$, where C( ) denotes the number of occurrence in the labeled training data. As we are dealing with a small labeled corpora it is often possible that $C(t_{i-1}, t_i)$ and $C(w_i, t_i)$ will become zero. To cope with the above situation, state transition probabilities are smoothed and symbol emission probabilities are estimated for handling unknown words that are not in the labeled corpora.

Like supervised first-order HMM (HMM-S1), the model parameters of the supervised second-order HMM (HMM-S2) are also estimated simply by counting from the labeled training data. Here the state transition probabilities of a particular tag t_i depends on the previous two tags t_{i-1} and t_{i-2}, $P(t_i \mid t_{i-2}, t_{i-1})$. Experiments have been carried out with TnT tagger (Brants, 2000); a supervised trigram HMM tagger along with suffix tree information for unknown words. When a particular instance of a trigram state transition probability does not occur in the training data

the state transition probabilities are smoothed and the symbol emission probabilities for unknown words are computed using the probability distribution for a particular suffix generated from all words in the labeled corpora (Brants, 2000).

4.2.1.2 Semi-supervised HMM (HMM-SS)

In semi-supervised first-order HMM, we first make use of the labeled training data to train the initial model. Further we make use of semi-supervised learning by augmenting the labeled data with a large amount of unlabeled data. The semi-supervised learning uses Baum-Welch re-estimation (or equivalently the *expectation maximization* (EM)) algorithm by recursively defining two sets of probabilities, the forward probabilities and the backward probabilities. First, we determine the initial choice for model parameters *A*, *B* and π from the labeled data. After choosing the above starting values, we iteratively use Baum-Welch algorithm to compute the new values of model parameters until convergence.

Baum Welch, or forward backward algorithm, recursively define two sets of probabilities. The forward probabilities,

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(w_{t+1}) \qquad 1 \le t \le T, \text{ (where } \alpha_1(i) = \pi_i b_i(w_1) \text{ for all } i),$$

and the backward probabilities,

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(w_{t+1}) \beta_{t+1}(j) \qquad T-1 \le t \le 1, \text{ (where } \beta_T(j) = 1 \text{ for all j).}$$

The forward probability $\alpha_t(i)$ is the joint probability of the sequence up to time t, $\{w_1, w_2, \ldots, w_t\}$ and the Markov process is in state *i* at time *t*. Similarly, the backward probability $\beta_t(j)$ is the probability of seeing sequence $\{w_{t+1}, w_{t+2},$

$...,w_T\}$ and the Markov process is in state i at time t. It follows the probability of the entire sequence is

$$P = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(w_{t+1})\beta_{t+1}(j) \qquad \text{for any } t \text{ in the range } 1 \le t \le T\text{-}1.$$

After the initial choice of the model parameters $\mu = (\pi, A, B)$ from the training data, the expected number of transition γ_{ij} from state i to j conditions on the observation sequence W is computed as follows:

$\gamma_{ij} = \frac{1}{P}\sum_{t=1}^{T-1}\alpha_t(i)a_{ij}b_j(w_{t+1})\beta_{t+1}(j)$, which is expected number of transition from state i to j.

Hence, the expected transition probability from a particular state i to a particular state j (i.e. $\hat{a}_{ij}$) are estimated by:

$$\hat{a}_{ij} = \frac{\gamma_{ij}}{\sum_{j=1}^{N}\gamma_{ij}} = \frac{\sum_{t=1}^{T-1}\alpha_t(i)a_{ij}b_j(w_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T-1}\alpha_t(i)\beta_t(i)} \qquad \textbf{Eq. 1}$$

In particular, to POS tagging, the above probability is the ratio of the expected number of transitions from a particular tag t_i to another particular tag t_j and the total expected number of transition from tag t_i to t_j.

Similarly, the emission probability (i.e. $\hat{b}_j(k)$) and initial probability (i.e. $\hat{\pi}_i$) can be estimated as follows:

$$\hat{b}_j(k) = \frac{\sum_{W_t = w_k}\alpha_t(j)\beta_t(j)}{\sum_{t=1}^{T}\alpha_t(j)\beta_t(j)} \qquad \textbf{Eq. 2}$$

and

$$\hat{\pi}_i = \frac{1}{p}\alpha_1(i)\beta_1(i) \qquad \textbf{Eq. 3}$$

The Baum Welch algorithm uses EM algorithm. Starting from at the initial model $\mu = (\pi, A, B)$ obtained by the supervised learning using the small annotated data, we repeatedly compute the new values { $\hat{\mu} = (\hat{\pi}, \hat{A}, \hat{B})$ } applying the equation 2-4 until convergence. It has been shown that the algorithm will converge, possibly to a non global local maximum.

4.2.2 Disambiguation

The aim of the disambiguation algorithm is to assign the most probable tag sequence $t_1 \ldots t_n$, to a observed sequence of words $w_1 \ldots w_n$, that is

$$S = \arg\max_{t_1 \ldots tn} P(t_1 \ldots t_n \mid w_1 \ldots w_n)$$

The stochastic optimal sequence of tags $t_1 \ldots t_n$, are assigned to the word sequence $w_1 \ldots w_n$, can be expressed as a function of both lexical $P(w_i \mid t_i)$ and language model $P(t_i \mid t_{i-1})$ probabilities using Bayes' Theorem:

$$P(t_1 \ldots t_n \mid w_1 \ldots w_n) = \frac{P(w_1 \ldots w_n \mid t_1 \ldots t_n) P(t_1 \ldots t_n)}{P(w_1 \ldots w_n)} = \frac{\prod_{i=1,n} P(w_i \mid t_i) P(t_i \mid t_{i-1})}{P(w_1 \ldots w_n)}$$

Since the probability of the word sequence $P(w_1 \ldots w_n)$ is the same for all candidate tag sequences, the most probable tag sequence (S) satisfies:

$$S = \arg\max_{t_1 \ldots tn} \prod_{i=1,n} P(w_i \mid t_i) P(t_i \mid t_{i-1}) \qquad \textbf{Eq. 4}$$

We use the Viterbi algorithm to find out the most probable tag sequence for a given word sequence. It is a very effective dynamic programming algorithm which takes $O(TN^2)$ time. The algorithm works as follows:

Let $S = \{s(t)\}$ $1 \le t \le T$ is a state sequence (i.e. the tag sequence) that generates $W = \{w(t)\}$ (the word sequence or observation of the HMM). Then the probability that S generates W is,

$$P(S) = \pi_{s(1)} b_{s(1)}(w_1) \prod_{t=2}^{T} a_{s(t-1)s(t)} b_{s(t)}(w_t)$$

To find the most probable sequence, the process starts with $\phi_1(i) = \pi_i b_i(w_1)$ where $1 \le i \le N$, and then performs the following steps:

$$\left.\begin{array}{l} \phi_t(j) = \max_{1 \le i \le N} \left[\phi_{t-1}(i) a_{ij} b_j(w_t) \right] \\ \text{and} \\ \psi_t(j) = \arg\max_{1 \le i \le N} \left[\phi_{t-1}(i) a_{ij} b_j(w_t) \right] \end{array}\right\} \begin{array}{l} 2 \le t \le T \\ 1 \le j \le N \end{array}$$

The most probable sequence at state i in time t is the only consideration for each time t and state i. The probability of the most probable sequence is $\max_{1 \le i \le N} \left[\phi_T(i) \right]$. The most probable sequence is reconstructed by

$$s(T) = \arg\max_{1 \le i \le N} \phi_T(i) \text{ and } s(t-1) = \psi_t(s_t) \text{ for } T \ge t \ge 2.$$

4.2.3 Smoothing

It may be the case that all events are not encountered in the limited training corpus that we have. The probabilities corresponding to these events would be set to zero. However the event may occur during testing. The problem

can be solved using different smoothing algorithms. Initially simple add-one smoothing was used to estimate the state transition probabilities that are not in the training corpora. Further, linear interpolation of unigram and bigram has been implemented for smoothing the state transition probabilities. We smooth the n-gram state transition probability for various *n* as follows:

$$P_{li}(t_i \mid t_{i-1}, t_{i-2}, ..., t_{i-(n-1)}) = \lambda_1 P(t_i) + \lambda_2 P(t_i \mid t_{i-1}) + ... + \lambda_n P(t_i \mid t_{i-1}, t_{i-2}, ..., t_{i-(n-1)})$$

The values of λ_1, λ_2 , …, λ_n are estimated by deleted interpolation (Brants, 2000) and $\sum_{i=1}^{n} \lambda_i = 1$.

When some new text is processed, few words might be unknown to the tagger. In our model, words are unknown when they are not included in the training text. Initially, we estimated the symbol emission probability by simple add-one smoothing. Further, we use suffix information for handling unknown words which has been found to work well for highly inflected languages (Samuelsson, 1993). The term suffix is a sequence of last few characters of a word, which does not necessarily mean a linguistically meaningful suffix. First we calculate the probability of a particular tag t_i, given the last *m* letters (l_i) of an *n* letter word: $P(t_i \mid l_{n-m+1}, ..., l_n)$ .Based on the above hypothesis we calculate the symbol emission probabilities using Bayes' rule:

$$P(Unknown\_word \mid t_i) = \frac{P(t_i \mid Unknown\_word)P(Unknown\_word)}{P(t_i)}$$

$$= \frac{P(t_i \mid l_{n-m+1}, ..., l_n)P(Unknown\_word)}{P(t_i)}$$

The probability $P(Unknown\_word)$ is approximated in open testing text by measuring the unknown word frequency. Therefore the model parameters are adopted each time an open testing text is being tagged. The probability $P(t_i \mid l_{n-m+1},...,l_n)$ and the probability $P(t_i)$ are measured in the training text. We conducted different experiments varying the suffix length from 1 to 6 characters. It has been observed empirically that the suffix length of 4 gives better results for all the HMM based models. Based on our observations, the inclusion of suffix essentially captures helps to understand the morphological inflection of the surface form word and in Kashmiri most morphological inflections lies in the last 4 characters of the words. Finally, each symbol emission probability of unknown word has been normalized:

$$\sum_{1}^{N} P(wi \mid ti) + P(Unknown\_word \mid t_i) = 1$$

Where, N is the number of known words and $t_i \in \{T\}$.

4.3 Experiments

We have implemented baseline model to understand the complexity of the POS tagging task. In this model the tag probabilities depend only on the current word:

$$P(t_1...t_n \mid w_1...w_n) = \prod_{i=1,n} P(t_i \mid w_i)$$

The effect of this is that the each word in the test data will be assigned the tag which occurred most frequently for that word in the training data.

The experiments were conducted with five different sizes (100K, 120K, 140K, 160K and 180K words) of the training data to understand the relative performance of the models as we keep on increasing the size of the annotated data.

We have used Trigrams'n'Tags (TnT) tagger, which is based on a supervised trigram HMM with suffix tree information for unknown words. These experiments give us some insight about the performance of the tagging task in comparison with the order of the Markov model in a poor-resource scenario.

4.3.1 Training Data

The training data consists of 2,00,000 manually annotated words. The training data has been annotated using a Kashmiri POS tagset as discussed in section 3.5 (Table 3.37).The annotated corpus was then used by the tagger for creating the language model i.e. lexical and n-gram files were created. It has been observed that the corpus ambiguity (mean number of possible tags for each word) in the training text is 1.77 which is much larger compared to the figure reported for European languages (Dermatas et al.,1995).

Once the language model was developed, the untagged files were then given to the tagger for automatically annotating the corpus. The tagged files produced by the tagger are then manually checked for errors. The files after being manually corrected are then added to the training corpus to retrain the tagger in order to increase the accuracy of the tagger.

4.3.2 Test Data

The language models have been tested on a set of randomly drawn 50,000 words distinct from the training corpus. It has been noted that 14% words in the open testing text are unknown with respect to the training set, which is also a little higher compared to the European languages (Dermatas et al., 1995).

4.3.3 System Performance

We define the tagging accuracy as the ratio of the correctly tagged words to the total number of words.

$$Accuracy(\%) = \frac{Correctly\ tagged\ words\ by\ the\ system}{Total\ no.\ of\ \ words\ in\ the\ evaluation\ set} \times 100$$

The accuracy of the TnT tagger is checked by comparing two files, one is the manually tagged file and another is the file tagged by the tagger. The process of retraining the tagger continued till the highest level of accuracy is achieved. The schematic diagram given below shows the optimization process of the TnT tagger using Kashmiri corpus.

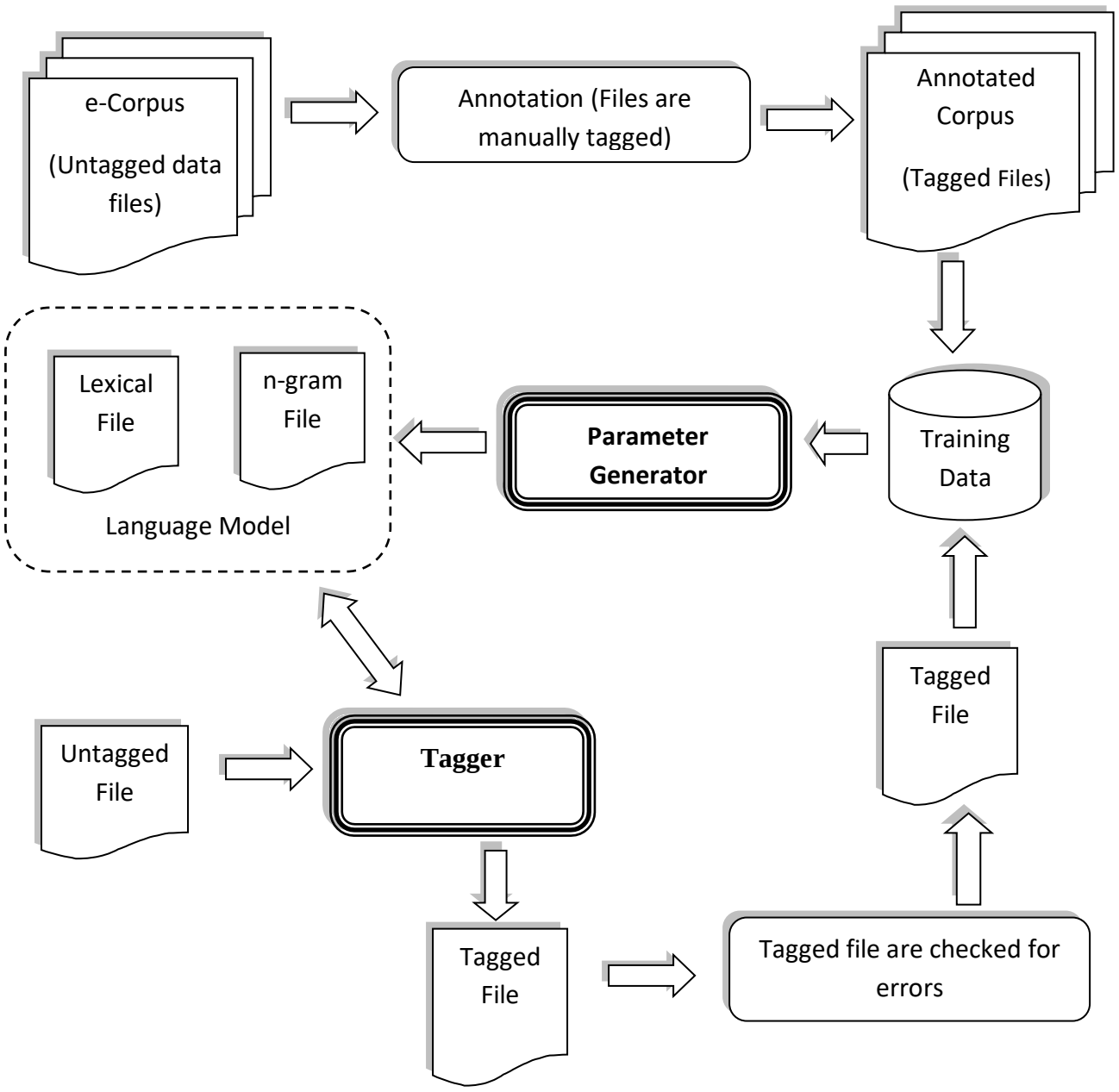

Fig. 4.3 Optimization process of TnT using Kashmiri Corpus.

4.3.4 Evaluation

The overall accuracy of the tagger depends upon the size of the corpus and the accuracy of the tags assigned. Therefore, training corpus should be large and the accuracy of tags should be as high as possible. Generally, the bigger the

corpus the higher the accuracy of the training corpus, with the result the overall performance of the tagger is better. Tagger's performance is evaluated at several steps. First of all tagging accuracy is determined over ten iterations. All tests are performed in partitions i.e. the overall corpus is divided into parts.

Initially corpus of around 2,00,000 words was taken. Then this corpus was divided into two parts out of which 50% was used for training the tagger and the remaining 50% of the corpus was used as test data for checking the accuracy of the tagger.The test data is unseen during training. By using these proportions, the overall accuracy of 62.55% was found. Then separate accuracies of known and unknown words were also calculated. In the test data 92.80% of words were known and 7.80% of the words were unknown to the tagger. And the accuracy of known and unknown words was 87.02% and 38.19% respectively. Each result was obtained by repeating the experiment 5 times with different partitions that is, the first partition was taken as 50%- 50% then next partition was 60%- 40% and so on and so forth to check the accuracy of the tagger. The results obtained are shown in the table given below.

Corpus			**Overall**		**Known Words**		**Unknown Words**	
Total Corpus	Training Data	Test Data	Acc. (%)	Errors (%)	Acc. (%)	Errors (%)	Acc. (%)	Errors (%)
2,00,000	**1,00,000**	**1,00,000**	**62.55**	**37.45**	**87.02**	**12.98**	**38.19**	**61.81**
	1,20,0000	**80,000**	**68.10**	**31.90**	**85.22**	**14.78**	**40.15**	**59.85**
	1,40,000	**60,000**	**76.39**	**23.61**	**84.78**	**15.22**	**42.11**	**57.89**
	1,60,000	**40,000**	**85.64**	**14.36**	**85.62**	**14.38**	**51.68**	**48.32**
	1,80,000	**20,000**	**96.28**	**03.72**	**87.62**	**12.38**	**52.35**	**47.65**

Table 4.1: Part-of-speech accuracy of Kashmiri POS tagger.

The learning curve for the above results is represented below:

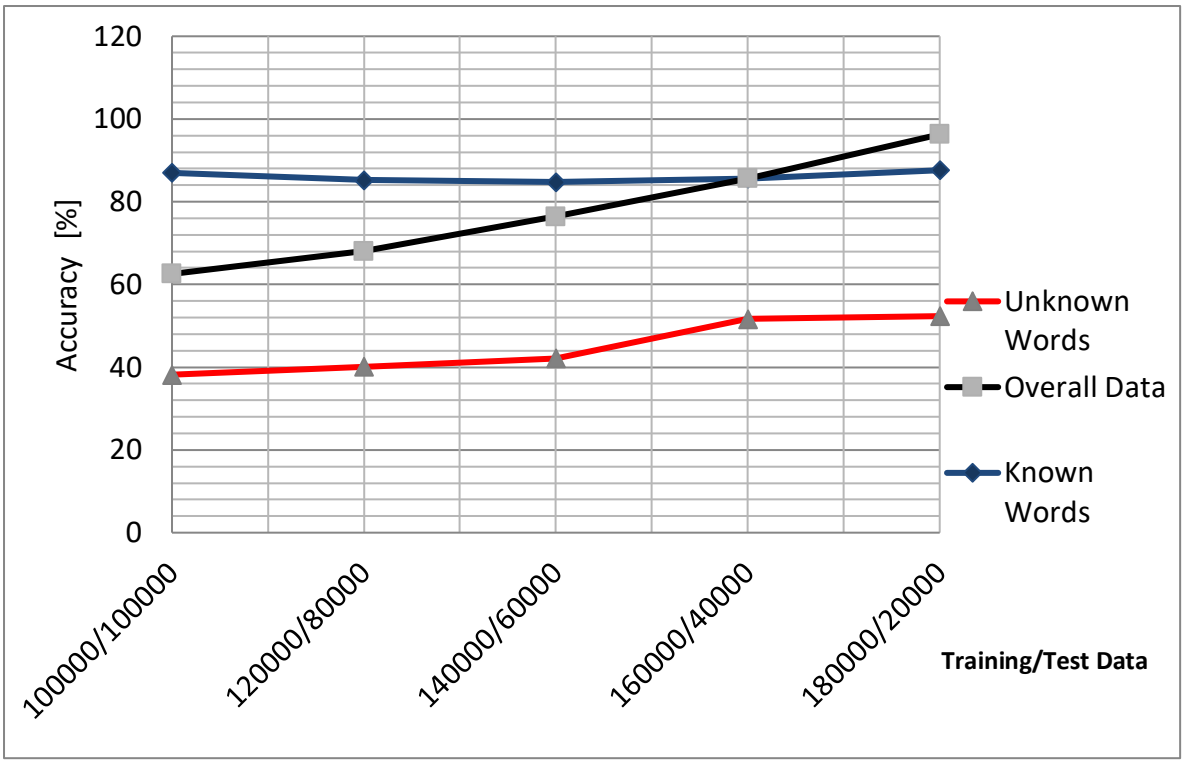

Fig. 4.4 Kashmiri Corpus: POS learning curve.

The curve given above shows the increase in accuracy as the training data is increased.Accuracy for known words is significantly higher than the unknown words.

4.4 Conclusion

The tagger developed for Kashmiri is in a preliminary stage as no NLP related work (corpus development, tagset development and other related works) has been done before in Kashmiri.

In this chapter, we have described an approach for automatic stochastic tagging of natural language text. The models described here are very simple and efficient for automatic tagging even when the amount of available labeled text is small.