Indian Journal of Science and Technology, *Vol 8(35), DOI: 10.17485/ijst/2015/v8i35/85367, December 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

HMM Chunker for Punjabi

Chandan Mittal*, Vishal Goyal and Umrinderpal Singh

Department of computer Science, Punjab University, Patiala – 147002, India;
chandanmittal29@gmail.com, vishal.pup@gmail.com, umrinderpal@gmail.com

Abstract

This paper presents a Hidden Markov Model (HMM) based Chunker for Punjabi. Chunking is the process of segmenting the text into syntactically correlated word groups known as chunks and then identifying the labels of the defined chunks. A robust Chunker is an important component for various applications requiring Natural Language Processing (NLP). In this research work, my goal is to develop an HMM based Chunker for Punjabi language. HMM Chunker is based on statistical probabilities. I have followed Hidden Markov Model in achieving my goal in which Viterbi Algorithm is used for calculating the highest probability of chunks and to train the system, Baum-Welch algorithm is followed and 25,000 lines of chunked Punjabi text are used. An annotated text file having 1,000 lines is used for testing the system. The accuracy of the system to find the chunk boundaries of the system is about 80% approx and the labelling is applied with an accuracy of about 98% and the labelling is applied with an accuracy of about 82%.

Keywords: Baum-Welch Algorithm, Chunking, Hidden Markov Model, Viterbi Algorithm

1. Introduction

1.1 Motivation and Problem Statement

Chunking process is one of the well studied problems in the field of Natural Language Processing (NLP). Different approaches have already been tried to automate the task of chunking for English and other languages. A robust Chunker has emerged as an important component in a variety of NLP applications. It is employed in information extraction, named entity identification, search, and in machine translation. Chunkers can be built using handcrafted linguistic rules; these need a relatively long time to develop because of many special cases. The task of chunking is ideally suited for machine learning because of robustness and relatively easy training.

Chunker identifies Noun Phrases (NP), Verb Groups Group Chunks (VGF, VGNF, VGINF, and VGNN), Adjectival Phrases (JJP) and Adverbial Phrases (RBP) in the annotated text corpus. In this work, the chunking process has been broken up into two sub processes: First, identifying the chunk boundaries and second, labeling the chunks with their syntactic categories.

The first sub-problem is to build a Chunker that takes a text in which words are tagged with Part of Speech (POS) tags as its input, and marks the chunk boundaries in its output. Moreover, the chunker is to be built by using machine learning techniques requiring only modest amount of training data. The second sub-problem is to label the chunks with their syntactic categories.

The presented work aims at building a Chunker for Punjabi. At present, no Chunkers are available for Punjabi.

1.2 Survey of Related Work

In this research work, to develop a chunker for Punjabi, I have studied Chunking for Hindi and English. Various approaches have been studied for developing a chunker such as rule based approach, statistical approach and hybrid approach. Various methods for boundary identification and labeling of the chunks have been studied.

**Author for correspondence*

The earliest work on Chunking for English using statistical approach has been done by Church K, 1988. This approach has been modified by Skut and Brants, 1998. They have used standard HMM based tagging methods in modeling the chunking process. Zhou, et al, 2000 continued the chunking process using the same methods and achieved an accuracy of 92% using contextual lexicon. Also, Chunking for Hindi has been done by Akshay Singh et al, 2005[1]. They have followed Hidden Markov Model to develop the chunker. Baum–Welch algorithm has been used to train the system. An annotated data set containing 2,00,000 tokens used for the training process and the system demonstrated about 92% accuracy.

In this work, I have used HMM based Chunking. Tokens as combination of Punjabi words and POS tags are given as an input to the system. The combination of words and POS tags gives the best result. Also, different methods are used and compared in order to get best results for labeling the chunks. This experience can also be used to build Chunkers for other languages.

The rest of the paper is structures as follows. Section 2, describes the Hidden Markov Model and Baum–Welch algorithm for training the system. Section 3, discusses problem formulation and reports the results of some initial experiments. Section 4 concludes experiments on identification of boundaries and labeling of chunks using statistical methods.

2. Hidden Markov Model

2.1 Introduction

Hidden Markov Models (HMMs) are a statistical tool used for modeling time series data. In HMM, the system being modeled is assumed to be a Markov process with unobserved (hidden) states. In Markov models, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters.

In Hidden Markov Model, the state is not directly visible, but output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states. Here, the adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; the model is still referred to as a 'hidden' Markov model even if these parameters are known exactly. The formal definition of a HMM is as follows:

$$\lambda = (A, B, \pi)$$

Where A is a set of state transition probabilities, B is a set of Emission probabilities and π is a set of Initial State probabilities.

There are three basic problems for HMMs.

2.2 Evaluation

The Evaluation problem of HMM determines the probability that a particular sequence of symbols was generated by that model.

Given a HMM, and a sequence of observations, we'd like to be able to compute the probability of the observation sequence given a model. This problem could be viewed as one of evaluating how well a model predicts a given observation sequence, and thus allow us to choose the most appropriate model from a set.

The probability that a model produces a sequence V^T of visible states:

$$\mathrm{P}(V^T) = \sum\nolimits_{\downarrow}(r=1)^{\uparrow}(r_{\downarrow}\ \max) \text{▒} [\![P(V^{\uparrow}T\text{-}|\,|w_{\downarrow}r^{\uparrow}T)]\!] \quad P(w_{\downarrow}r^{\uparrow}T)$$

Probability of sequence V^T is equal to the sum over all r_{max} possible sequences of hidden states of the conditional probability that the system has made a particular transition multiplied by the probability that it then emitted.

$$P(V^{\uparrow}T\text{-}|\,|w_{\downarrow}r^{\uparrow}T) = \prod\nolimits_{\downarrow}(t=1)^{\uparrow}(t=1) \text{▒} [\![p(v(t)\,|\,w(t))]\!] \tag{1}$$

$$P(w_r^T) = \prod_{t=1}^{t=T} P(w(t)\,|\,w(t-1)) \tag{2}$$

Where (1) output probabilities depend only upon hidden states and (2) first order hidden Markov Process.

$$P(V^T) = \sum_{r=1}^{r_{max}} \prod_{t=1}^{t=T} p(v(t)\,|\,w(t))P(w(t)\,|\,w(t\text{ - }1))$$

We can calculate the above probability as:

$$\alpha_j(t)\begin{cases} 0 & t=0 \text{ and } j \neq \text{initial state} \\ 1 & t=0 \text{ and } j = \text{initial state} \\ \left[\sum_i \alpha_i(t-1)\alpha_{ij}\right] b_{jk}v(t) & \text{otherwise} \end{cases}$$

$$a_{ij} = P(w_j(t+1)\,|\,w_i(t))$$

$$b_{jk} = P(v_k(t) \mid w_j(t)$$

Where b_{jk} $v(t)$ means the symbol probability b_{jk} corresponding to $v(t)$.

2.3 Decoding

The Decoding problem is to find the most likely sequence of hidden states w^T, given a sequence of visible states V^T.

The decoding problem can be resolved by using Viterbi Algorithm. The decoding algorithm finds at each time step t the state that has the highest probability of having come from the previous step and generated the observed visible state v_k. The full path is the sequence of such states.

It can be expressed as:

$(w(1), w(2), \ldots., w(T)^{argmax} P[w(1), w(2), \ldots., w(T), v(1), v(2), \ldots., v(T)|\theta]$

2.4 Learning

We are given with the structure of the model i.e. number of states and number of symbols but we do not know the probabilities a_{ij} and b_{jk}. The learning problem determines these parameters from an ensemble of training samples. The learning problem can be resolved by using Baum-Welch Algorithm or Forward-Backward Algorithm.

We have calculated $\alpha_i(t)$ as the probability that the model is in state $w_i(t)$ and has generated the target sequence up to step t. Similarly, $\beta_i(t)$ is the probability that the model is in state $w_i(t)$ and will generate the remainder of the given target sequence from t+1 to T.

$$\beta_i(t) = \begin{cases} 0 & w_i(t) \neq w_0 \text{ and } t = T \\ 1 & w_i(t) = w_0 \text{ and } t = T \\ \left[\sum_j \beta_j(t+1) a_{ij}\right] b_{jk} v(t+1) & otherwise \end{cases}$$

$\alpha_i(t)$ and $\beta_i(t)$ are the estimates of their true values since we do not know the actual value of a_{ij} and b_{jk}.

The probability of transition between $w_i(t-1)$ and $w_i(t)$, given the model generated the entire training sequence V^T by any path is:

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1) a_{ij} b_{jk} \beta j(t)}{P(V^T \mid \theta)}$$

Improved estimate of a_{ij}:

$$Imp\, a_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_k \gamma_{ik}(t)}$$

Improved estimate of b_{jk}:

$$Imp\, b_{jk} = \frac{\sum_{\substack{t=1 \\ v(t)=v_k}}^{T} \sum_l \gamma_{jl}(t)}{\sum_{t=1}^{T} \sum_l \gamma_{jl}(t)}$$

The learning algorithm consists of three steps. First is to start with the rough estimates of a_{ij} and b_{jk}. Second is to calculate the improved estimate using the above formulas. Third is to repeat the above step until there is sufficiently small change in the estimated values of the parameters.

3. Experiments

We have used stochastic approach to develop a language chunker. The chunking algorithm is divided into two phases, namely: Chunk boundary identification and Chunk labeling.

3.1 Chunk Boundary Identification

Suppose we have a sentence:

ਇਹਨਾਂ ਨਦੀਆਂ ਤੋ ਸ਼ਹਰਿ ਦੀਆਂ ਸਾਰੀਆਂ ਜ਼ਰੂਰਤਾਂ ਦੀ ਪੂਰਤੀ ਵੀ ਹੁੰਦੀ ਹੈ | ehna ndiyan ton shehr diyan sarian jruratan di purti vi hundi hai |

It is a sequence of words $W_n = (w_1, w_2, \ldots\ldots, w_n)$, $w_i \in W$, where W is the word set. Each word has its part of speech (POS) Tag:

ਇਹਨਾਂ\DM_DMD ਨਦੀਆਂ\N_NN ਤੋ\PSP ਸ਼ਹਰਿ\N_NN ਦੀਆਂ\PSP ਸਾਰੀਆਂ\QT_QTF ਜ਼ਰੂਰਤਾਂ\N_NN ਦੀ\PSP ਪੂਰਤੀ\V_VM_VF ਵੀ\RP_RPD ਹੁੰਦੀ\V_VM_VF ਹੈ\V_VAUX |\RD_PUNC

Ehna\DM_DMD ndiyan\N_NN ton\PSP shehr\N_NN diyan\PSP sarian\QT_QTF jrurtan\N_NN di\PSP purti\V_VM_VF vi\RP_RPD hundi\V_VM_VF hai\V_VAUX |\RD_PUNC

The sequence of corresponding part of speech (POS) tags $T_n = (t_1, t_2, \ldots.., t_n)$, $t_i \in T$ where T is the POS tag set.

Now, our aim is to create most probable chunks of the sequence W_n. The chunks are marked with chunk tag sequence $C_n = (c_1, c_2, \ldots\ldots, c_n)$ where c_i stands for the chunk tag corresponding to each word w_i, $c_i \in C$.

C here is the chunk tag set which depends upon various tagging schemes:

Two – Tag Scheme: It consists of symbols such as STRT and CNT.

Three– Tag Scheme: It consists of symbols such as STRT, CNT and STP.

Four– Tag Scheme: It consists of symbols such as STRT, CNT, STP and STRT_STP.

Where Tag stands for:

STRT – Chunk starts at this token.

CNT – This token lies in middle of the chunk.

STP – This token lies at the end of the chunk.

STRT_STP – This token lies in a chunk of its own.

Here, we combine the corresponding words and POS tags to get a sequence of new tokens $V_n = (v_1, v_2,\ldots\ldots, v_n)$ where $v_i = (w_i, t_i) \in V$.

Thus the problem is to find the sequence C_n given the sequence of tokens V_n which maximizes the probability

$P(C_n \mid V_n) = P(c_1, c_2,\ldots\ldots\ldots, c_n \mid v_1, v_2,\ldots\ldots\ldots, v_n)$,

which is equivalent to maximizing

$$P(V_n \mid C_n)\, P(C_n)$$

We assume that given the chunk tags, the tokens are statistically independent of each other and that each chunk tag is probabilistically dependent on the previous k chunk tags ((k + 1)-gram model). Using chain-rule, the problem reduces to that of Hidden Markov model (HMM) given by $\max_{ci\in C} \prod\downarrow(i = 1)^{\uparrow} n \,[\![P([\![\;\; v_i \mid c_i)\, P(c_{i+k} \mid c_i,\ldots\ldots, c_{i+k-1})$

where the probabilities in the first term are emission probabilities and in the second term are transition probabilities.

Under 1st order HMM, prediction of chunk tag at i^{th} token is conditional only on the previous chunk tag.

$$a_{ij} = P(c_j(t+1) \mid c_i(t))$$

$$b_{jk} = P(v_k(t) \mid c_j(t))$$

The optimal sequence of chunk tags can be found using the Viterbi algorithm. For the training of HMM we have used the Baum – Welch algorithm.

3.2 Chunk Labeling

Once the chunk boundaries are marked, the next task is to classify the chunk. The components within a chunk help to assign the label on the chunk. In our scheme there are 5 types of chunks – NP (Noun Phrase), VG (Verb Group), JJP (Adjectival Phrase) RBP (Adverbial Phrase) and BLK (others). We tried a method based on machine learning for deciding chunk labels.

3.3 HMM based Chunk Labeling

In this method, the chunk boundary tags are augmented with the chunk labels while learning. For example, the tags for the last token in a chunk could have additional information in the form of the chunk label. [[ਇਹਨਾਂ\DM_DMD ਨਦੀਆਂ\N_NN ਤੋ\PSP]]_NP [[ਸ਼ਹਰਿ\N_NN ਦੀਆਂ\PSP ਸਾਰੀਆਂ\QT_QTF ਜ਼ਰੂਰਤਾਂ\N_NN]]_NP [[ਦੀ\PSP]]_CCP [[ਪੂਰਤੀ\V_VM_VF ਵੀ\RP_RPD]]_NP [[ਹੁੰਦੀ\V_VM_VF ਹੈ\V_VAUX]]_VGF [[|\RD_PUNC]]_BLK [[Ehna\DM_DMD ndiyan\N_NN ton\PSP]]_NP [[shehr\N_NN diyan\PSP sarian\QT_QTF jrurtan\N_NN]]_NP [[di\PSP]]_CCP [[purti\V_VM_VF vi\RP_RPD]]_NP [[hundi\V_VM_VF hai\V_VAUX]]_VGF [[|\RD_PUNC]]_BLK

4. Conclusion

Chunker divides a sentence into its major-non-overlapping phrases and attaches a label to each chunk. The capability for a computer to automatically POS tag and chunk a sentence is very essential for further analysis in many approaches to the field of NLP. Chunking is used in many areas such as linguistic acquisition, Psychology, Sequence Learning, and Memory Architecture, etc. Also, in NLP, many applications are based on chunking such as Information Extraction, Question Answering, providing features for Machine Learning e.g., for building Named Entity recognizers, etc.

In this research work, I have developed HMM based chunker. The reasons behind choosing HMM for developing the chunker instead of other models such as CRF or Maximum Entropy Model are HMM works efficiently for many other languages. Also, we are not having sufficient training data as large training data is required to train the systems based on other models.

5. Acknowledgements

The annotated corpus for Punjabi has been provided by TDIL, DeitY, Ministry of Communication and Information Technology, New Delhi has been provided.

6. References

1 Akshay S, Sushma B, Rajeev S. HMM based Chunker for Hindi. Proceedings of 2nd International Joint Conference on Natural Language Processing; 2005. p. 126–31.

2. Ashish T, Arnab S, Sudeshna S. A New Approach for HMM Based Chunking for Hindi; 2005.
3. Dhanalakshmi V, Padmavathy P, Anand KM, Soman KP. Chunker for Tamil. International Conference on advances in recent technologies in Communication and Computing; 2009. p. 436–8.
4. Avinesh PVS, Karthik FG. Part-of-Speech Tagging and Chunking using Conditional Random Fields and Transformation based Learning. Proceedings of International Joint Conference on Artificial Intelligence – 07. Workshop on Shallow Parsing in South Asian Languages; 2007. p. 21–4.
5. Pranjal A, Delip R, Balaraman R. Parts Of Speech Tagging and Chunking with HMM and CRF. Proceedings of CoNLL – 2000 and LLL – 2000.