Chapter 5

5 CHUNKING USING HIDDEN MARKOV MODE(uni-gram Technique)

Chunking is a natural language processing (NLP) task that focuses on dividing a text document into syntactically correlated non-exhaustive and non-overlapping groups of words, i.e., a word can only be a member of one chunk and not all words are in chunks (Tjong et al, 2000). Chunking is widely used as an intermediate step to parsing with the purpose of improving the performance of the parser. It also helps to identify non-overlapping phrases from a stream of data, which are further used for the development of different NLP applications such as information retrieval, information extraction, named entity recognition, question answering, text mining, text summarization, etc. These NLP tasks consist of recognizing some type of structure which represents linguistic elements of the analysis and their relations. In text chunking the main problem is to divide text into syntactically related non-overlapping groups of words (chunks).

The main goal of chunking is to divide a text into segments which correspond to certain syntactic units such as noun phrases, verb phrases, prepositional phrases, etc. Abney (1991) introduced the concept of chunk as an intermediate step providing input to further full parsing stages. Thus, chunking can be seen as the basic task in full parsing. Although the detailed information from a full parse is lost, chunking is a valuable process in its own right when the entire grammatical structure produced by a full parse is not required. For example, various studies indicate that the information obtained by chunking or partial parsing is sufficient for information retrieval systems rather than full parsing (Yangarber and Grishman,1998). Partial syntactical information can also help to solve many NLP tasks, such as text summarization, machine translation and spoken language understanding (Molina and Pla, 2002). For example, Kutlu (2010) stated that finding noun phrases and verb phrases is enough for information retrieval systems. Phrases that give us information about agents, times, places, objects, etc. are more significant than the complete configuration-al syntactic analyses of a sentence for question-answering, information extraction, text mining and automatic summarization.

Chunkers do not necessarily assign every word in the sentence like full

parses to a higher-level constituent. They identify simple phrases but do not require that the sentence be represented by a single structure. By contrast full parsers attempt to discover a single structure which incorporates every word in the sentence. Abney (1995) proposed to divide sentences into labelled, non overlapping sequences of words based on superficial analysis and local information. In general, many of NLP applications often require syntactic analysis at various NLP levels including full parsing and chunking. The chunking level identifies all possible phrases and the full parsing analyses the phrase structure of a sentence. The choice of which syntactic analysis level should be used depends on the specific speed or accuracy of an application. The chunking level is efficient and fast in terms of processing than full parsing (Thao et al., 2009). Chunkers can identify syntactic chunks at different levels of the parser, so a group of chunkers can build a complete parser (Abney, 1995). Most of the parsers developed for languages like English and German use chunkers as components. Brants (1999) used a cascade of Markov model chunkers for obtaining parsing results for the German NEGRA corpus. Today, there are a lot of chunking systems developed for various languages such as Turkish (Kutlu, 2010), Vietnamese (Thao et al., 2009), Chinese (Xu et al, 2006), Urdu (Ali and Hussain, 2010), etc.

5.1 Chunk Representation

The tag of chunks can be noun phrases, verb phrases, adjectival phrases, etc. in line with the language construction rules. There are many decisions to be made about where the boundaries of a group should lie and, as a consequence, there are many different style of chunking. There are also different types of chunk tags and chunk boundary identifications. Nevertheless, in order to identify the boundaries of each chunk in sentences, the following boundary types are used (Ramshaw and Marcus, 1995): IOB1, IOB2, IOE1, IOE2, IO, “[”, and “]”. The first four formats are complete chunk representations which can identify the beginning and ending of phrases while the last three are partial chunk representations. All boundary types use “I” tag for words that are inside a phrase and an “O” tag for words that are outside a phrase. They differ in their treatment of chunk-initial and chunk-final words.

IOB1: the first word inside a phrase immediately following another phrase receives a B tag.

IOB2: all phrase- initial words receive a B tag.

IOE1: the final word inside a phrase immediately preceding another same phrase receives an E tag.

IOE2: all phrase- final words receive an E tag.

IO: words inside a phrase receive an I tag, others receive an O tag.

"[": all phrase-initial words receive "[" tag, other words receive "." tag.

"]": all phrase-final words receive "]" tag and other words receive "." tag.

5.2 Architecture of the CHUNKER

To implement the chunker component, we used hidden Markov model (HHM) enhanced by a set of rules to prune errors. The HMM part has two phases: the training phase and the testing phase. In the training phase, the system first accepts words with POS tags and chunk tags. Then, the HMM is trained with this training set. Likewise in the test phase, the system accepts words with POS tags and outputs appropriate chunk tag sequences against each POS tag using HMM model. Fig 3 illustrates the work flow of the chunking process. In this work, chunking is treated as a tagging problem. We use POS tagged sentence as input from which we observe sequences of POS tags represented as T. However, we also hypothesize that the corresponding sequences of chunk tags form hidden Markovian properties. Thus, we used a hidden Markov model (HMM) with POS tags serving as states. The HMM model is trained with sequences of POS tags and chunk tags extracted from the training corpus. The HMM model is then used to predict the sequence of chunk tags C for a given sequence of POS tag T. This problem corresponds to finding C that maximizes the probability $P(C|T)$, which is formulated as:

$$C^{'} = arg_c max P(C|T) - - - - - - - (1)$$

where C' is the optimal chunk sequence. By applying Baye's rule can derive, Equation (1) yields:

$$C^{'} = arg_c max P(C|T) * P(C) - - - -(2)$$

which is in fact a decoding problem that is solved by making use of the Viterbi algorithm. The output of the decoder is the sequence of chunk tags which groups words based on syntactical correlations. The output chunk

sequence is then analysed to improve the result by applying linguistic rules derived from the grammar of Amharic. For a given Amharic word w, linguistic rules (from which sample rules are shown in Algorithm 1) were used to correct wrongly chunked words ("w-1" and "w+1") are used to mean the previous and next word, respectively).

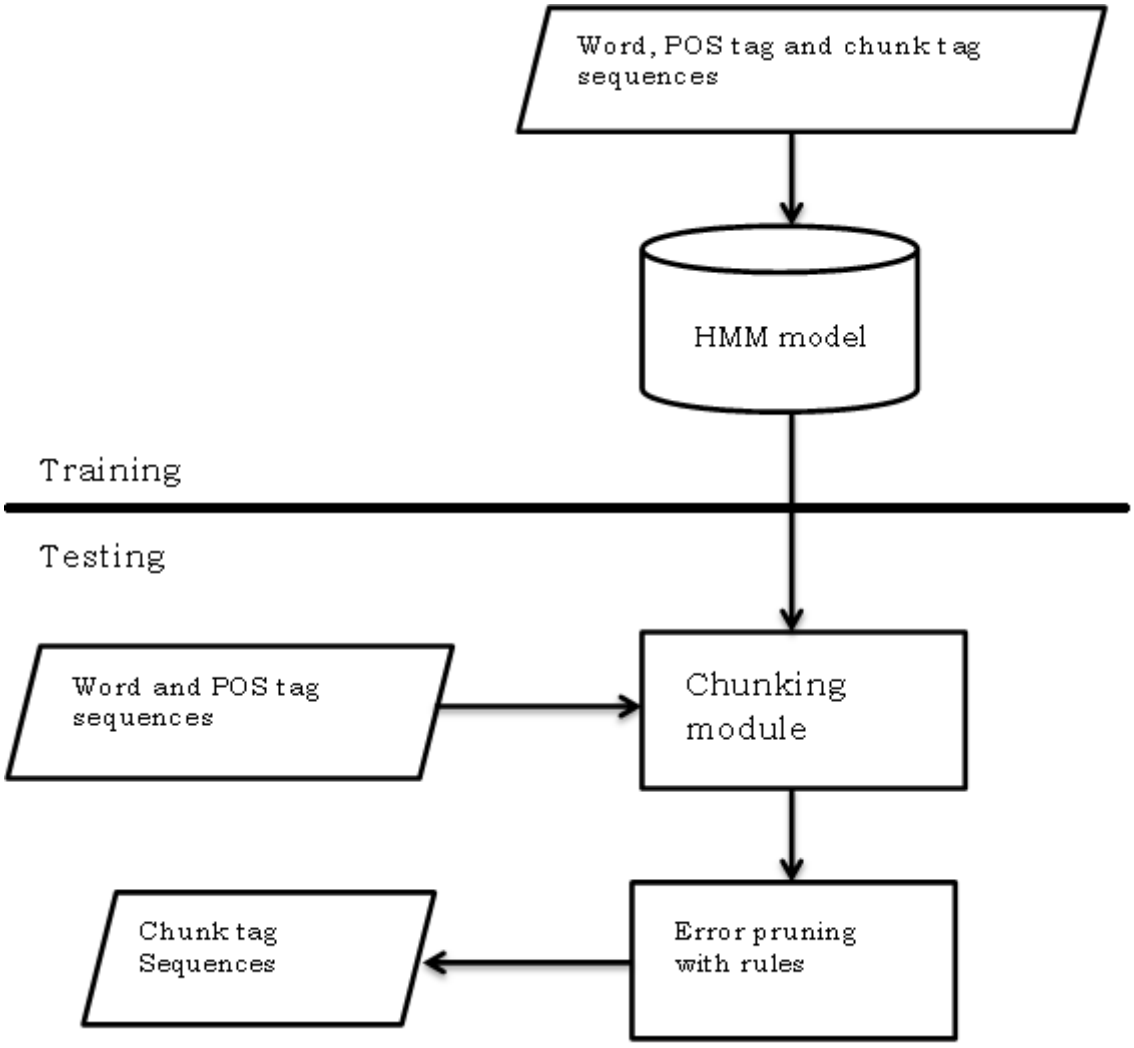

Fig 3: Work flow of the chunking process.

5.3 The Corpus

The major source of the data-set we used for training and testing the system was Nepali news corpus which is at present widely used for research on Nepali natural language processing. The corpus contains 8067 sentences where words are annotated with POS tags. Furthermore, we also collected additional text from a Nepali grammar book authored by Banu Oja and Shambhu Oja (2004). The sentences in the corpus are classified as training data set and testing data set using 10 fold cross validation technique.

5.4 Test Results

In 10-fold cross-validation, the original sample is randomly partitioned into 10 equal size sub-samples. Of the 10 sub-samples, a single sub-sample is used as the validation data for testing the model, and the remaining 9 sub-samples are used as training data. The cross-validation process is then repeated 10 times, with each of the 10 sub-samples used exactly once as the validation data. Accordingly, we obtain 10 results from the folds which can be averaged to produce a single estimation of the model's predictive potential. By taking the average of all the ten results the overall chunking accuracy of the system is presented in Table 2.

Table 8: Test Result for Nepali Based Phrase Chunker.

Chunking model	Accuracy
HMM	85.31%