Improved part of speech tagging model of hindi language

Jagjit Singh

A thesis submitted in partial fulfillment for the
degree of Masters of Technology

in the

North West Institute of Engineering and Technology, Moga

Supervisor: Lakhvir Singh Garcha

June 2016

CANDIDATE'S DECLARATION

I hereby certify that I am student of M.Tech (regular) of Computer Science and engineering Department and declare that I own the full responsibility for the information, results etc. provided in this thesis titled **"Improved Part of speech tagging model for hindi language"** Submitted to **North west institute of engineering and technology** for the award of **Master of Technology in Computer Science and Engineering** degree. I have taken care in all respect to honor the intellectual property right and have acknowledged the contributions of others for using them in this academic purpose. I further declare that in case of any violation of intellectual property right or copyright, I as a candidate will be fully responsible for the same. My supervisors and institute should not be held for full or partial violation of copy right if found at any stage of my degree.

Place: Moga | Jagjit Singh
Date: | Reg.No.

CERTIFICATE

This is to certify that the thesis work entitled **"Improved Part of speech tagging model for hindi language"**, submitted by Jagjit Singh, Reg. no _ _ _ _ _ _ to the North west institute of engineering and technology, Moga, for the partial fulfillment of the requirement of the degree of Masters of Technology in Computer Science and Engineering is a record of student's own study under my supervision and guidance.

This thesis has not been submitted to any other university or institution for the award of any other degree.

Supervisor
Pr. lakhvir Singh Garcha
Assistant Professor
Department of Computer Science and Engg.
NWIET, Moga

The M.Tech Viva-Voice examination of **Jagjit Singh** has been held on ______.

Sign. of Supervisor **Sign. of External Examiner**

Sign. of H.O.D

ACKNOWLEDGEMENT

First of all I would like to thank Almighty God. It's only because of the blessings of the God that I have been able to complete my thesis work successfully. I would like to thank **Pr. Lakhvir Singh Garcha** for being my advisor, and for her guidance and support throughout my research work.Also i like to thank **Dr. Mohita Garg** Head of Department for her valuable suggestions about the research work. I am deeply grateful for all the help I have received during the course of this thesis at North west institute of engineering and technology, Moga. I am also thankful to all the staff members of the CSE department who have helped me in many ways directly or indirectly for this research work.

Finally I would like to thank my dear parents whose moral support and care always encouraged me to proceed.

Jagjit Singh

ABSTRACT

Part-of- Speech tagging is the way to tag every word in a text as a particular part of speech, e.g. proper verb, adverb etc. POS tagging is the first important step in the processing of NLP applications. This paper reports the improved model POS tagging for Hindi Languages. Due to complex structural effect, the number of problems occurs when tagging the sentences written in various languages. So we, in this research try to make a model, which could improve the word sentence tagging to a gereater extent as compare to the previous models. We define 25 types of tags in the research, and details of these tag are briefly discussed in this book.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

This chapter provides the introduction of NLP (natural language processing) and architecture of POS tagging process. It presents features and applications of POS taggers. It also describes different methods of tagging.

Context

With the advancement of technology, the demand of Natural Language Processing (NLP) is also increasing and it becomes very important to find out correct information from collection of huge data only on the basis of queries and keywords. Sometimes user tries to search data with help of query and get unimportant or irrelevant data instead of correct data. Due to complex structural effect, this problem occurs mostly with Indian languages as compared to others. To avoid this problem, POS tagging is the best application of NLP that assigns exact part of speech to each word of a text (Mohnot, K, 2014). It is the process of marking up a word in a corpus as corresponding to a particular part of speech use its definition, as well as its relation. POS tags are also known as word classes, morphological classes, or lexical tags to choose correct grammatical tag for word on the basis of linguistic feature.

There are a number of approaches to implement part of speech tagger, i.e. Rule Based approach, Statistical approach and Hybrid approach. Rule-based tagger use linguistic rules to assign the correct tags to the words in the sentence or file. Statistical Part of Speech tagger is based on the probabilities of occurrences of words for a particular tag. Hybrid based Part of Speech tagger is combination of Rule based approach and Statistical approach. Part of Speech tagging is an important application of natural language processing. It is used in several Natural Languages processing based software implementation. Accuracy of all NLP tasks like grammar checker, phrase chunker, machine translation etc. depends upon the accuracy of the Part of Speech tagger. Tagger plays an important role in speech recognition, natural language parsing and information retrieval (Mehta, D. N, 2015).

NLP (natural language processing) is the process that provides the facility of interaction between human and machine. It is a component of computer science, linguistics and artificial intelligence. It is difficult task to build NLP application because human speech is not always specific. The main objective of NLP is to develop such a system that can understand text and translate between human language and another. The work in area of Part-of-Speech (POS) tagging has begun in the early 1960s. Part of Speech tagging is an important tool for NLP. It is one of the simplest as well as statistical models for many NLP applications. POS Tagging is an initial step of information extraction, summarization, retrieval, machine translation, speech conversion.

POS tagging is the process of assigning the best grammar tag to each word of text like verb, noun, pronoun , adjective , adverb, conjunction , preposition etc. some unknown words exist in every language so it is very difficult task to assign the appropriate POS tag to each word in a sentence [3]. The mostly work that has been done for Indian languages was one of the rule based approaches and other empirical based POS tagging Approach. But the fact was that rule-based approach requires proper language knowledge and hand written rule. Due to morphological effect of Indian languages, researchers faced a great problem to write proper linguistic rules and many cases it was noticed that results were not good. Most of natural language processing work has been done for Hindi, Tamil, Malayalam and Marathi and several part-of-speech taggers have been applied for these languages. After this, researchers moved to stochastic based approach. However the stochastic methods requires large corpora to be effective, but still many successful POS were developed and used in various natural language processing tasks for Indian language.

The main issue after morphological richness of Indian Languages is Ambiguity. It is very time consuming process to assign a correct POS tag to different context words. Due to this reason, POS Tagging is becoming a challenging problem for study in the field of NLP.

The main objective of Natural Language Processing is to facilitate the interaction between human and machine. POS tagging is the process of attaching the best grammar tag like to each word of a sentence of some language. A word in a sentence can act as a verb, noun, pronoun, adjective, adverb, conjunction, preposition etc so POS is defined as the grammatical information of each word of a sentence. While assigning a POS tag it is necessary to determine the context of the word i.e. whether it is acting like a noun, adjective, verb etc. Sometime a word can act as a noun in one sentence and in another sentence it can give the sense of verb. So before selecting a POS tag for a word the exact context of the word must be clear. For Indian languages it is a difficult task to assign the correct POS tag to each word in a sentence because of some unknown words in Indian languages. The earlier work that has been done for Indian languages was based rule based approaches. But the rule-based approach needs proper language knowledge and hand written rule. Most of natural language processing work has been done for Hindi, Tamil, Malayalam and Marathi and several part-of-speech taggers have been applied for these languages. The set of tags assigned by a part of speech tagger may contain just a dozen tags so such a big tagset can arise the difficulty in the tagging process. POS tagging is helpful in various NLP tasks like Information Retrieval, Machine Translation, Information Extraction, Speech Recognition etc. For Indian languages researchers find difficulty in writing linguistic rules for rule based approaches because of morphological richness. The other main issue after morphological richness of Indian Languages is Ambiguity. It is very time consuming process to assign a POS tag to each word according to its context in sentence by hand and that is why POS Tagging is becoming a challenging problems for study in the field of NLP

Tagging

Automatic assignment of descriptors to the given tokens is called tagging. The descriptor is called tag. The tag may indicate one of the part of speech, semantic information and soon. So tagging is kind of classification.

POS Tagger

The broad utilization of internet for making search of information is difficult due to the search systems consist container of words which causes problem in retrieval due to synonyms. There is need to accept the word boundary between what kinds of query information are submitted by humans and what kinds further result get (Tapaswi, N., & Jain, S., 2012). So for text indexing and retrieval uses POS information. POS tagging is used as an early stage of text analysis in many applications such as subcategory acquisition, text to speech synthesis and alignment of parallel corpora. POS tagging is a necessary pre-module and building block for various NLP tasks like Machine translation, Natural language text processing and summarization, User interfaces, Multilingual and cross language information retrieval, Speech recognition, Artificial intelligence, Parsing , Expert system and so on. Parts of speech (POS) tagging are one of the most well studied problems in the field of Natural Language Processing (NLP). Different approaches have already been tried to automate the task for English and other western languages there are large numbers of POS tagger available for English language which has got satisfactory performance but cannot be applied to Marathi language. Part-of-speech tagging in Marathi language is a very complex task as Marathi is highly inflectional in nature and free word order language (Mehta, D. N., & Desai, N. P. 2015). The process of assigning description to the given word is called Tagging. The descriptor is called tag. The tag may indicate one of the parts-of-speech like noun, pronoun, verb, adjective, adverb, preposition, conjunction, and interjection. The input (Raw Text) is tokenized and a corpus is used for detecting the corresponding part of speech of each token in the sentence. For correct POS tagging, training the tagger, corpus and a proper tagset is also important Disambiguation is the most difficult problem in tagging. The ambiguity which is identified in the tagging module is resolved using the grammar rules.

Architecture of POS tagger

- **Tokenization:** Tokenization is the process of separating tokens from raw text. Words are separated by white spaces or punctuation marks. The sentence is segmented by using white space because the occurrence of white space indicates the existence of a word boundary. There are various morphological problems where this approach fails. So by using this we can easily find out the tokens from the sentence. The given text is divided into tokens so that they can be used for further analysis. The tokens may be words, punctuation marks, and utterance boundaries (Bagul P. et al., 2014).

- **Ambiguity look-up:** This is to use lexicon and a guesser for unknown words. While lexicon provides list of word forms and their likely parts of speech, guessers analyze unknown tokens. Compiler or interpreter, lexicon and guesser make what is known as lexical analyser.

- **Ambiguity Resolution:** This is also called disambiguation. Disambiguation is based on information about word such as the probability of the word. Disambiguation is also based on related information or word/tag sequences. For example, the model might prefer noun analyses over verb analyses if the preceding word is a preposition or article. Disambiguation is the most difficult problem in tagging. The ambiguity which is identified in the tagging module is resolved using the Marathi grammar rules.

- **WordNet:** The main relation among words in WordNet is synonymy. WordNet is an electronic database which contains parts of speech of all the words which are stored in it. It is trained from the corpus for higher performance and efficiency (Bagul, P. et al., 2014). WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The majority of the WordNet's relations connect words from the same part of speech (POS). Thus, WordNet really consists of four sub-nets, one each for nouns, verbs, adjectives and adverbs, with few cross-POS pointers. Cross-POS relations include the "morph semantic" links that hold among semantically similar words sharing a stem with the same meaning.

- **Corpus:** For correct POS tagging, training the tagger well is very important, which requires the use of well annotated corpora. Annotation of corpora can be done at various levels which include POS, phrase or clause level, dependency level etc. Corpus linguistics is the study of language as expressed in samples (corpora) of "real world" text. Corpus is a large collection of texts. It is a body of written or spoken material upon which a linguistic analysis is based. The plural form of corpus is corpora. Some popular corpora are British National Corpus (BNC), COBUILD/Birmingham Corpus, and IBM/Lancaster Spoken English Corpus (Bagul, P. et al., 2014).

- **Tagset:** Apart from corpora, a well-chosen tagset is also important. The language tagset represents parts of speech and consist on syntactic classes. According to contextual and morphological structure, natural languages are different from each other.In the top level the following categories are identified as universal categories for all ILs and hence these are obligatory for any tagset. Some common tags: [N] Nouns [V] Verbs, [PR] Pronouns, [JJ] Adjectives, [RB] Adverbs, [PP] Postpositions, [PL] Participles, [QT] Quantifiers, [RP] Particles, [PU] Punctuations (Mahar, J. A., & Memon, G. Q., 2010).

POS Tagging Techniques

The POS tagger can be implemented by using either a supervised technique or an unsupervised technique. Supervised POS taggers are based on pre-tagged corpora, which are used for training to learn information about the word-tag frequencies, rule and tag set, sets etc. The performance of the models generally increases with the increase in size of these corpora.

Unsupervised POS tagging models do not require pretagged corpora. Instead, they use those methods through which automatically tags are assigned to words. Advanced computational methods like the Baum-Welch algorithm to automatically include tag sets, transformation rules etc. Under these two categories different approaches have been used for the implementation of POS taggers such as:

1. **Rule Based Approach / Transformation Based:** The rule based POS tagging approach that uses a set of hand written rules. Rule base taggers depend on word list or lexicon or dictionary to assign appropriate tag to each word. The tagger divided into two stages. First, it search words in dictionary and second, it assigns a tag by removing disambiguity of words using linguistic features of word.

   On the basis of level rule divided as lexical rules act in a word level, each sentence splits into small words called lexeme or token And, the context sensitive rules act in a sentence level, to check the grammar for the sentence. The transformation based approach is similar to the rule based approach in the sense that it depends on a set of rules for tagging. The transformation based approaches use a pre-defined set of handcrafted rules as well as automatically induced rules that are generated during training (Bagul P. et al., 2014).

   The main drawback of rule based system is that it fails when the text is not present in lexicon. Therefore the rule based system cannot predict the appropriate tags.

2. **Statistical Approach / Stochastic Tagger:** A stochastic approach assign a tag to word using i frequency, probability or statistics. From the annotated training data it “selects the most likely tag for the word” and uses same information to tag that word in the unannotated text (Bagul P. et al., 2014). Stochastic tagger as a simple generalization of the stochastic taggers generally resolves the ambiguity by computing the probability of a given word (or the tag). The drawbacks of this approach are that it can come up with sequences of tags for sentences that are not acceptable according to the grammar rules. So, it determines the best tag for a word by calculating the probability of previous tags on n value, where the value of n is set to 1, 2 or 3 are known as the Unigram, Bigram and Trigram models.

   - **Hidden Markov Model:** HMM stands for Hidden Markov Model. HMM is a generative model. The model assigns the joint probability to paired observation and label sequence. Then the parameters are trained to maximize the joint likelihood of training sets .It is advantageous as its basic theory is elegant and easy to understand.

Hence it is easier to implement and analyze. It uses only positive data, so they can be easily scaled. It has few disadvantages. In order to define joint probability over observation and label sequence HMM needs to enumerate all possible observation sequence. Hence it makes various assumptions about data like Markovian assumption i.e. current label depends only on the previous label. Also it is not practical to represent multiple overlapping features and long term dependencies. Number of parameter to be evaluated is huge. So it needs a large data set for training (Bagul P. et al., 2014).

- **Maximum Entropy Markov Model:** MaxEnt stands for Maximum Entropy Markov Model (MEMM). It is a conditional probabilistic sequence model. It can represent multiple features of a word and can also handle long term dependency. It is based on the principle of maximum entropy which states that the least biased model which considers all known facts is the one which maximizes entropy. Each source state has an exponential model that takes the observation feature as input and output a distribution over possible next state. Output labels are associated with states.

  The large dependency problem of HMM is resolved by this model. Also, it has higher recall and precision as compared to HMM. The disadvantage of this approach is the label bias problem. The probabilities of transition from a particular state must sum to one. MEMM favors those states through which less number of transitions occurs (Mohnot K & Singh S P, 2014).

- **Conditional Random Field Model:** CRF stands for Conditional Random Field. It is a type of discriminative probabilistic model. It has all the advantages of MEMMs without the label bias problem. CRFs are undirected graphical models (also know as random field) which is used to calculate the conditional probability of values on assigned output nodes given the values assigned to other assigned input nodes.

3. **Hybrid Approach:** This approach combines the advantages of both of the above approaches namely rule based approach and stochastic approach. Words in this technique are first tagged probabilistically and then as post processing, linguistic rules are applied to tag tokens. Accuracy of taggers based on this approach generally gives good results than other techniques (Mehta, D. N, 2015).

4. **Neural Tagger:** Neural taggers are based on neural networks which learn the parameters of POS tagger from a representative training data set. The MLP-tagger is trained with error back-propagation learning algorithm. The performance has shown better than stochastic method (Raju, S. B., 2002).

Applications of POS tagger

The POS tagger can be used as a preprocessor. Text indexing and retrieval uses POS information. Speech processing uses POS tags to decide the pronunciation. POS tagger is used for making tagged corpora.

Features for POS Tagging

The Following features have been found to be very useful in POS tagging:

Suffixes The next word of Current token is used as feature.

Prefixes The previous word of Current token is used as feature.

Context Pattern based Features Context patterns are helpful for POS tagging. Eg. Word prefixes and suffix context patterns.

Word length Length of particular word is useful feature.

Static Word Feature The previous and next words of a particular word are used as features.

Presence of Special characters Presence Special characters surrounding the current word are used as features.

Organization of Thesis

The structure of the rest of the Thesis is as follows:

Chapter 2, presents the background of various POS tagging approaches for various languages and it covers the detail about. It also includes literature review of study.

Chapter 3, Tells about the present work, methodology in detail. It explains the algorithm and flowchart of present study.

Chapter 4, presents the results of study and compares this with existing techniques on the basis of different output parameters.

Chapter 5,contains the conclusion and future work. In the end references are marked.

Chapter 2

CONCEPTS OF SPEECH

Introduction

Speech is a complex phenomenon. People rarely understand how is it produced and perceived. The naive perception is often that speech is built with words, and each word consists of phones. The reality is unfortunately very different. Speech is a dynamic process without clearly distinguished parts. It's always useful to get a sound editor and look into the recording of the speech and listen to it. Here is for example the speech recording in an audio editor.

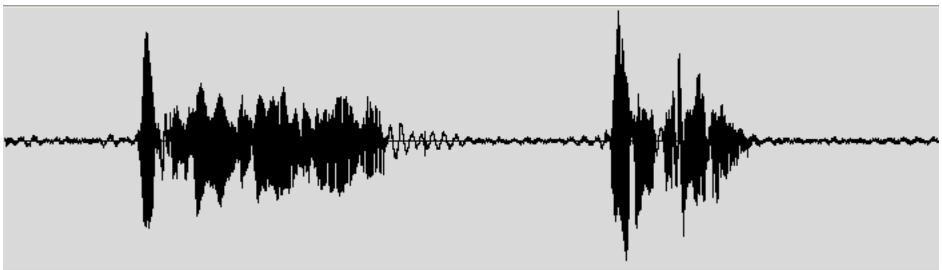

Figure 2.1: Speech Waveform

All modern descriptions of speech are to some degree probabilistic. That means that there are no certain boundaries between units, or between words. Speech to text translation and other applications of speech are never 100 percent correct. That idea is rather unusual for software developers, who usually work with deterministic systems. And it creates a lot of issues specific only to speech technology [4].

Concept of Speech

Speech is a continuous audio stream where rather stable states mix with dynamically changed states. In this sequence of states, one can define more or less similar classes of sounds, or phones. Words are understood to be built of phones, but this is certainly not true. The acoustic properties of a waveform corresponding to a phone can vary greatly depending on many factors - phone context, speaker, style of speech and so on. The so called coarticulation makes phones sound very different from their "canonical" representation. Next, since transitions between words are more informative than stable regions, developers often talk about diphones- parts of phones between two consecutive phones. Sometimes developers talk about subphonetic units - different substates of a phone. Often three or more regions of a different nature can easily be found [5].

The first part of the phone depends on its preceding phone, the middle part is stable, and the next part depends on the subsequent phone. That's why there are often three states in a phone selected for speech recognition.Sometimes phones are considered in context. Such phones in context are called triphones or even quinphones. For computational purpose it is helpful to detect parts of triphones instead of triphones as a whole, for example, to create a detector for a beginning of triphone and share it across many triphones. The whole variety of sound detectors can be represented by a small amount of distinct short sound detectors. Usually we use 4000 distinct short sound detectors to compose detectors for triphones. We call those detectors senones.

Recognition Precess

The common way to recognize speech is the following: we take waveform, split it on utterances by silences then try to recognize what's being said in each utterance. To do that we want to take all possible combinations of words and try to match them with the audio. We choose the best matching combination. There are few important things in this match [6].

First of all it's a concept of features. Since number of parameters is large, we are trying to optimize it. Numbers that are calculated from speech usually by dividing speech on frames. Then for each frame of length typically 10 milliseconds we extract 39 numbers that represent the speech. That's called eature vector. They way to generates numbers is a subject of active investigation, but in simple case it's a derivative from spectrum.

Second it's a concept of the model. Model describes some mathematical object that gathers common attributes of the spoken word. In practice, for audio model of senone is gaussian mixture of it's three states - to put it simple, it's a most probable feature vector. From concept of the model the following issues raised - how good does model fits practice, can model be made better of it's internal model problems, how adaptive model is to the changed conditions.

The model of speech is called Hidden Markov Model or HMM, it's a generic model that describes black-box communication channel. In this model process is described as a sequence of states which change each other with certain probability. This model is intended to describe any sequential process like speech. It has been proven to be really practical for speech decoding.

Third, it's a matching process itself. Since it would take a huge time more than universe existed to compare all feature vectors with all models, the search is often optimized by many tricks. At any points we maintain best matching variants and extend them as time goes producing best matching variants for the next frame.

Models

According to the speech structure, three models are used in speech recognition to do the match:

- **Acoustic Model** contains acoustic properties for each senone. There are context-independent models that contain properties (most probable feature vectors for each phone) and context-dependent ones (built from senones with context).

- **Phonetic Dictionary** contains a mapping from words to phones. This mapping is not very effective. For example, only two to three pronunciation variants are noted in it, but it's practical enough most of the time. The dictionary is not the only variant of mapper from words to phones. It could be done with some complex function learned with a machine learning algorithm.

- **Language Model** is used to restrict word search. It defines which word could follow previously recognized words (remember that matching is a sequential process) and helps to significantly restrict the matching process by stripping words that are not probable. Most common language models used are n-gram language models-these contain statistics of word sequences-and finite state language models-these define speech sequences by finite state automation, sometimes with weights. To reach a good accuracy rate, your language model must be very successful in search space restriction. This means it should be very good at predicting the next word. A language model usually restricts the vocabulary considered to the words it contains. That's an issue for name recognition. To deal with this, a language model can contain smaller chunks like subwords or even phones. Please note that search space restriction in this case is usually worse and corresponding recognition accuracies are lower than with a word-based language model [7].

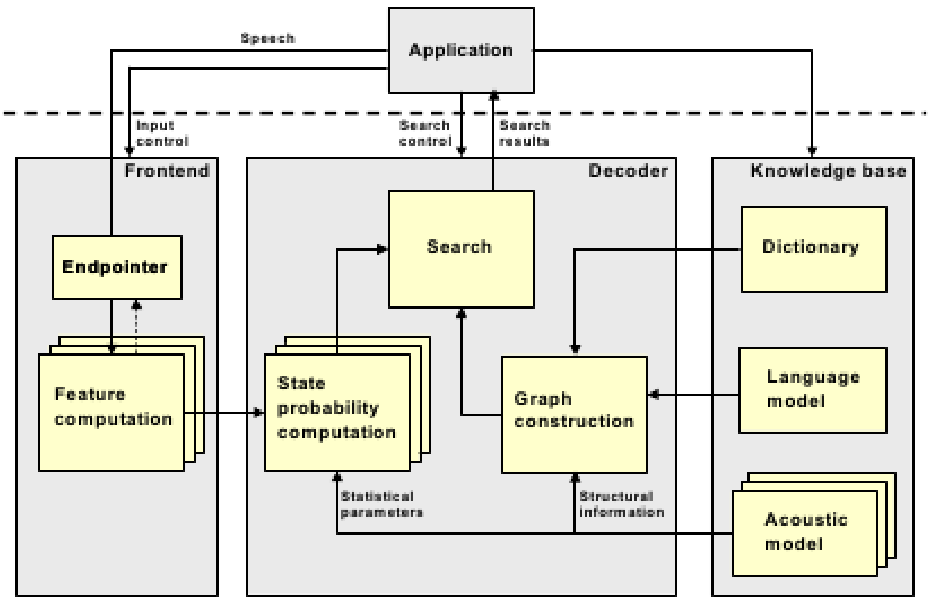

Figure 2.2: Sphinx Architecture

Most current speech recognition systems use hidden Markov models(HMMs) to deal with the temporal variability of speech and Gaussian mixture models(GMMs) to deter- mine how well each state of each HMM fits a frame or a short window of frames of coefficients that repre- sents the acoustic input. An alternative way to evaluate the fit is to use a feed-forward neural network that takes several frames of coefficients as input and produces posterior probabilities over HMM states as output.

PocketSphinx is a flexible, modular and pluggable framework to help foster new innovations in the core research of hidden Markov model(HMM) speech recognition systems. The design of PocketSphinx is based on patterns that have emerged from the design of past systems as well as new requirements based on areas that researchers currently want to explore. To exercise this framework, and to provide researchers with a "research- ready" system, Sphinx-4 also includes several implementations of both simple and state-of-the-art techniques. The framework and the implementations are all freely available via open source.

PocketSphinx is a flexible, modular and pluggable framework to help foster new innovations in the core research of hidden Markov model (HMM) speech recognition systems. The design of PocketSphinx is based on patterns that have emerged from the design of past systems as well as new requirements based on areas that researchers currently want to explore. To exercise this framework, and to provide researchers with a "research- ready" system, Sphinx-4 also includes several implementations of both simple and state-of-the-art techniques. The framework and the implementations are all freely available via open source.

When researchers approach the problem of core speech recognition research, they are often faced with the problem of needing to develop an entire system from scratch, even if they only want to explore one facet of the field. Open source speech recognition systems are available, such as HTK , ISIP , AVCSR and earlier versions of the Sphinx systems. The available systems are typically optimized for a single approach to speech system design. As a result, these systems intrinsically create barriers to future research that departs from the original purpose of the system. First and foremost, PocketSphinx is a modular and pluggable framework that incorporates design patterns from existing systems, with sufficient flexibility to support emerging areas of research interest. The framework is modular in that it comprises separable components dedicated to specific tasks, and it is pluggable in that modules can be easily replaced at run time. To exercise the framework, and to provide researchers with a working system, PocketSphinx also includes a variety of modules that implement state-of-the-art speech recognition techniques.

Advantages of Speech Recognition and Dictation systems

- Speech is a very natural way to interact, and it is not necessary to sit at a keyboard or work with a remote control.
- No training required for users.
- With speech recognition software, become a lot more productive – you'll be able to get your work done much more quickly.

- Most of us can dictate at least 3 times faster than we can type.
- It's like having your own Personal Assistant available 24/7.

Principles of HMM

Speech recognition systems generally assume that the speech signal is a realisation of some mes-sage encoded as a sequence of one or more symbols. To effect the reverse operation of recognising the underlying symbol sequence given a spoken utterance, the continuous speech waveform is first converted to a sequence of equally spaced discrete parameter vectors. This sequence of parameter vectors is assumed to form an exact representation of the speech waveform on the basis that for the duration covered by a single vector (typically 10ms or so), the speech waveform can be regarded as being stationary. Although this is not strictly true, it is a reasonable approximation. Typical parametric representations in common use are smoothed spectra or linear prediction coefficients plus various other representations derived from these.

The role of the recogniser is to effect a mapping between sequences of speech vectors and the wanted underlying symbol sequences. Two problems make this very difficult. Firstly, the mapping from symbols to speech is not one-to-one since different underlying symbols can give rise to similar speech sounds. Furthermore, there are large variations in the realised speech waveform due to speaker variability, mood, environment, etc. Secondly, the boundaries between symbols cannot be identified explicitly from the speech waveform. Hence, it is not possible to treat the speech waveform as a sequence of concatenated static patterns.

The second problem of not knowing the word boundary locations can be avoided by restricting the task to isolated word recognition. This implies that the speech waveform corresponds to a single underlying symbol (e.g. word) chosen from a fixed vocabulary. Despite the fact that this simpler problem is somewhat artificial, it nevertheless has a wide range of practical applications. Furthermore, it serves as a good basis for introducing the basic ideas of HMM-based recognition before dealing with the more complex continuous speech case. Hence, isolated word recognition using HMMs will be dealt with first.

Application of Speech Recognition

- **Automation of Operator Services** Systems like the Voice Recognition Call Processing (VRCP) system introduced by AT and T or the Automated Alternate Billing System (AABS) introduced by Nortel enabled operator functions to be handled by speech recognition systems. The VRCP system handled so-called 'operator assisted' calls such as Collect, Third Party Billing, Person-to- Person, Operator Assisted Calling, and Calling Card calls. The AABS system automated the acceptance (or rejection) of billing charges for reverse calls by recognizing simple variants of the two word vocabulary Yes and No.

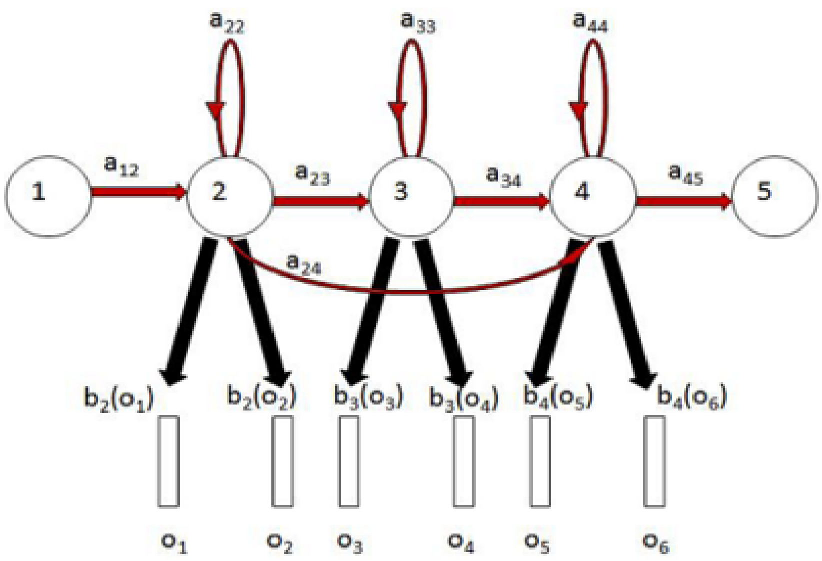

Figure 2.3: The Markov Generation Model

- **Automation of Directory Assistance** Systems were created for assisting operators with the task of determining telephone numbers in response to customer queries by voice. Both NYNEX and Nortel introduced a system that did front end city name recognition so as to reduce the operator search space for the desired listing, and several experimental systems were created to complete the directory assistance task by attempting to recognize individual names in a directory of as many as 1 million names. Such systems are not yet practical (because of the confusability among names) but for small directories, such systems have been widely used (e.g. in corporate environments).

- **Voice Dialing** Systems have been created for voice dialing by name (so- called alias dialing such as Call Home, Call Office) from AT and T, NYNEX, and Bell Atlantic, and by number (ATandT SDN/NRA) to enable customers to complete calls without having to push buttons associated with the telephone number being called.

- **Voice Banking Services** A system for providing access to customer accounts, account balances, customer transactions, etc. was first created in Japan by NTT (the ANSER System) more than 10 years ago in order to provide a service that was previously unavailable. Equivalent services have been introduced in banks worldwide over the last several years.

- **Voice Prompter** A system for providing voice replacement of touch-tone input for so-called Interactive Voice Response (IVR) systems was introduced by AT and T in the early 1990's (initially in Spain because of the lack of touch- tone phones in that country). This system initially enabled the customer to speak the touch-tone position (i.e. speak or press the digit one); over time systems have evolved so that customers can speak the service associated with the touch-tone position (e.g. say reservations or push the 1-key, say schedule or push the 2-key, etc.

- **Directory Assistance Call Completion** This system was introduced by both AT and T and NYNEX to handle completion of calls made via requests for Directory Assistance. Since Directory Assistance numbers are provided by an independent system, using Text-to-Speech synthesis to speak out the listing, speech recognition can be used to reliably recognize the listing and dial the associated number. This highly unusual use of a speech recognizer to interface with a speech synthesizer is one of the unusual outgrowths of the fractionation of the telephone network into local and long distance carriers in the United State.

- **Reverse Directory Assistance** This system was created by NYNEX:, Bellcore, and Ameritech to provide name and address information associated with a spoken telephone number.Information Services. These type of systems enable customers to access information lines to retrieve information about scores of sportilng events, traffic reports, weather reports, theatre bookings, restaurant reservations, etc.

- **Agent Technology** Systems like Wildfire and Maxwell (AT and T) enable customers to interact with intelligent agents via voice dialogues in order to manage calls (both in-coming and out-going calls), manage mesaages (both voice and email), get information from the Web (eg, movie reviews, calling directories), customize services (e.g., first thing each morning the agent provides the traffic and weather reports), personalize services (via the agent personality, speed, helpfulness), and adapt to user preferences (e.g., learn how the user likes to do things and react appropriate.

- **Customer Care** The goal of customer care systems is to replace Interactive Voice Response systems with a dialogue type of interaction to make it easier for the user to get the desired help without having to navigate complicated menus or understand the terminology of the place being called for ihelp. The How May I Help You (HMIHY) customer care system of AT and T is an excellent example of this type of system.

- **Computer Telephony Integration** Since the telecommunication network of the future will integrate the telephony (POTS) and computer (Packet) networks, a range of new applications will arise which exploit this integration more fully. One prime example is registry services where the network locates the user and determines the most appropriate way to communicate with them. Another example is providing a user cache of the most frequently accessed people in order to provide a rapid access mechanism for these frequently called numbers.

- **Voice Dictation** Although the desktop already supports voice dictation of documents, a prime telecommunications application of speech recognition would be for generating voice responses to email queries so that the resulting message becomes an email message back to the sender (rather than a voice mail response to an email message).

Summary

In this chapter, we describes about the Introduction of Speech Recognition concept. Then further we discussed about Hidden mrkov's Model. In chapter we discussed about advantages of Speech Recognition systems, application of Speeh Recognition System.

Chapter 3

LITERATURE REVIEW

Literature Review

The main objective of the present study was: "To develop a software for speech to text conversion". The researcher had studied a lot of related literature to identify the different works done in the related area. The study had given a more clear vision in the task. In this present chapter, the brief information about the related research works for the present study is given.

A glance over Related Literature

Initial computers were design just to do simple calculations; scientists have been trying to endow the computer with more and more intelligence. That is to make the digital machine, do things, which require human-like intelligence. Thus the term Artificial Intelligence (AI) was coined in early 1950s. Gradually scientists came to know the immense power of the device and its limitations. More and more systems have been developed as a consequence of incremental advances in computer science. Slowly these electronics devices have been spreading their roots in the soils of this society. Now computer systems are available for applications of civil amenities to medical diagnosis; home entertainment to space programs; simple calculations to complex mathematical modeling etc [7].

In today's fast placed world where every human being has to combat a race against time – a direct interface between the spoken words and computer user will gleefully accept the same words spontaneously typed on the computer screen with minimum lapse of time. The phenomenon is known as speech (voice) recognition.

Speech recognition is an emerging technology where a speaker speaks to computer with the microphone and computer understands the given words. Speech recognition is a great supplement to traditional mouse and keyboard input; it will boost productivity and provide a new option for people who have difficulty using a keyboard.

Speech is the most common mode of communication among human beings. The objective of speech recognition is to recognize the message being spoken. Differentorganization and researchers define the term "speech recognition", and it can be summarized as follow:

- Speech recognition extracts the message information in a speech signal to control the actions of a machine in response to spoken commands.

- Speaker recognition identifies or verifies a speaker for restricting access to information, network or physical premises [7].

- Speech recognition is the process of finding a linguistic interpretation of a spoken utterance, typically, this means finding the sequence of characters, which forms the word.

Some Empirical Studies

A life without speech (voice) is a picture without color. Speech is viewed in two different ways; one is generation of the voice by human beings and other is to understand the speech by the opponent. The God has gifted both the things to all the creature of the world and the same is more and efficiently utilized by human beings. In today environment speech is the best way for communication. But to reach to this level, speech had to passed from lots many revaluations. The computer was introduced and with the applications of AI human had tried to talk to computers. Lots many applications related to speech is developed. The development process can be summarized as follow:

An Unsurpassed Heritage in Speech

Bell telephone researchers began with Alexander Graham Bell and his assistant Watson held their historic telephone conversion in 1876. In the area of speech technology, the company has achieved numerous breakthroughs over the years. The groundbreaking research in the early years was to improve the quality and clarity of telephone conversations, understanding speech in telephone handsets. As time passes AT and T labs. Led the way with a number of other major innovations related speech, hearing and acoustics, taking machine (1939), speech coding and transmissionmachine, stereo recording and playback, speech synthesizer, speech recognition, taskoriented speech translation system, natural language voice enabled customer care system, called How May I Help You (HMIHY) (Figures of speech, white paper AT and T labs)

AT and T, Bell's lab, developed electronics speech synthesis in 1936. However, the first commercial voice to 1978 when Texas instruments introduced the first speech synthesizer in the form of children's toy. The toy could spell the works back to the end user. The first voice recognition software for computers, was PLAINTALK by Apple Computers for Macintosh. In July of 2001, AT and T introduced the first in what will be a family of pioneering text to speech (TTS) products.

IBM's voice technology research originated back in the late 1950. In 1992, IBM's AIX based speech server series supported the dictation of reports and letters. Next, the IBM continuous speech series provided continuous recognition of spoken commands and phrase. In 1993, a personal voice product IBM's Personal Dictation System, was released for OS/2, In 1996 IBM voice Type Simply Speaking introduced high accuracy spoken dictation technology. In 1997, VivaVoice, a dictation product using continuous voice technology was introduced. User no longer had to pause between words, and could speak at a natural pace. IBM offers several voice software one of them is IBM WebSphere software which consists of IBM WebSphere Voice Server, IBM WebSphere Voice Response and IBM Message Center [10].

Language Technology in India

As the computers are being slowly woven into the fabrics of the society, more and more applications are being explored. Till recently, usage of computers was limited to English speaking community. Having found this limitation, as a proactive effort,many programmers were designed in India, with the initiatives of the Government and academia. Some R and D projects were also funded by the Department of Electronic (DoE), Government of India in this area.

The Early Efforts (During 1980-90)

DoE organized a symposium on “Linguistic Implications of Computer Based Information Processing” in 1979. Probably this was the first symposium in this area, supported by DoE. As the outcome about ten projects were initiated.

GIST development: In the early 1980’s the development of GIST (Graphics and Intelligence – Based Script Technology) was started as a DoE sponsored project at the IIT, Kanpur (IITK) and later the technology was further matured at the Center for Development of Advanced Computing (CDAC) Pune. GIST allows display of various Indian Scripts on the computer monitor screen based on the information entered through a keyboard having an overlay of the concerned Indian scripts. Based on these developments, codes for various keys used in Indian Scripts and their layout has been standardized by the Bureau of Indian Standard. This development has led to a number of software products that are currently available in the market to enable diverse users to carry out word processing, DTP, Spread sheet, Spell checkers etc [23].

In addition to GIST, CDAC has also brought out a software called “LEAP” for Indian Language word processing on Windows. Variety of fonts for Indian Language have also been developed by CDAC known as ISFOC fonts.

Corpora development: Machine readable corpora of texts in Indian language (i.e. Tamil, Telugu, Kannada, Malayalam, Assamese, Bengali, Oriya, English, Hindi, Punjabi, Sanskrit, Kashmiri and Urdu have been developed. The software for grammatical tagging of corpora, word count and frequency count and spell checkers are also developed.

W. Byrne: "Towards language independent acoustic modeling" this paper describes procedures and experimental results using speech from diverse source languages to build an ASR system for a single tar-get language. This work is intended to improve ASR in languages for which large amounts of training data are not available. they have developed both knowledge-based and automatic methods to map phonetic units from the source languages to the target language. We employed HMM adaptation techniques and Discrimi- native Model Combination to combine acoustic models from the individual source languages for recognition of speech in the target language[9].Byrne, William, et al. "Towards language independent acoustic modeling." Acoustics, Speech, and Signal Processing, 2000. ICASSP’00. Proceedings. 2000 IEEE International Conference on. Vol. 2. IEEE, 2000 [7].

Richard P. Lippmann: "Speech recognition by machines and humans" Comparises six modern speech corpora with vocabularies ranging from 10 to more than 65,000 words and content ranging from read isolated words to spontaneous conversations. Error rates of machines are often more than an order of magnitude greater than those of humans for quiet, wideband, read speech. Machine performance degrades further below that of humans in noise, with channel variability, and for spontaneous speech. Humans can also recognize quiet, clearly spoken nonsense syllables and nonsense sentences with little high-level grammatical information. These comparisons suggest that the human–machine performance gap can be reduced by basic research on improving low-level acoustic-phonetic modeling, on improving robustness with noise and channel variability, and on more accurately modeling spontaneous speech. Lippmann, Richard P. "Speech recognition by machines and humans [8].

Tanja Schultz: "Language-independent and language-adaptive acoustic modeling for speech recognition" this paper decribes the distribution of speech technology products all over the world, the portability to new target languages becomes a practical concern. As a consequence our research focuses on the question of how to port large vocabulary continuous speech recognition (LVCSR) systems in a fast and efficient way. More specifically we want to estimate acoustic models for a new target language using speech data from varied source languages, but only limited data from the target language. For this purpose, they introduce different methods for multilingual acoustic model combination and a polyphone decision tree specialization procedure. Recognition results using language- dependent, independent and language-adaptive acoustic models are presented and discussed in the framework of our GlobalPhone project which investigates LVCSR systems in 15 languages. Schultz, Tanja, and Alex Waibel. "Language-independent and language-adaptive acoustic modeling for speech recognition" [9].

Willie Walker, Paul Lamere:"Sphinx-4: A Flexible Open Source Framework for Speech Recognition" this paper describes about sphinux that the phinx-4 is a flexible, modular and pluggable framework to help foster new innovations in the core research of hidden Markov model (HMM) speech recognition systems. The design of Sphinx-4 is based on patterns that have emerged from the design of past systems as well as new requirements based on areas that researchers currently want to explore. To exercise this framework, and to provide researchers with a “research- ready” system, Sphinx-4 also includes several implementations of both simple and state-of-the-art techniques. The framework and the implementations are all freely available via open source [10].

Fuliang Weng: in his paper "Efficient Lattice Representation and Generation" describe two new techniques for reducing word lattice sizes without eliminating hypotheses. The first technique is an algorithm to reduce the size of non-deterministic bigram word lattices. The algorithm iteratively combines lattice nodes and transitions if local properties show that this does not change the set of allowed hypotheses. On bigram word lattices generated from Hub4 Broadcast News speech, it reduces lat- tice sizes by half on average. It was also found to produce smaller lattices than the standard finite state automaton determinization and minimization algorithms. The second technique is an improved al- gorithm for expanding lattices with trigram language models. In- stead of giving all nodes a unique trigram context, this algorithm only creates unique contexts for trigrams that are explicitly repre- sented in the model. Backed-off trigram probabilities are encoded without node duplication by factoring the probabilities into bigram probabilities and backoff weights [11].

Mohit Dua1, R.K.Aggarwal: " Punjabi Automatic Speech Recognition Using HTK" aims to discuss the implementation of an isolated word Automatic Speech Recognition system (ASR) for an Indian regional language Punjabi. The HTK toolkit based on Hidden Markov Model (HMM), a statistical approach, is used to develop the system. Initially the system is trained for 115 distinct Punjabi words by collecting data from eight speakers and then is tested by using samples from six speakers in real time environments. To make the system more interactive and fast a GUI has been developed using JAVA platform for implementing the testing module. The paper also describes the role of each HTK tool, used in various phases of system development, by presenting a detailed architecture of an ASR system developed using HTK library modules and tools. The experimental results show that the overall system performance is 95.63percent and 94.08percent [12].

Kumar Ravinder: "Comparison of HMM and DTW for Isolated Word Recognition System of Punjabi Language" describes about te issue Issue of speech interface to computer. Issue of speech interface to computer has been capturing the global attention because of convenience put forth by it. Although speech recognition is not a new phenomenon in existing developments of user-machine interface studies but the highlighted facts only provide promising solutions for widely accepted language English. This paper presents development of an experimen- tal, speaker-dependent, real-time, isolated word recognizer for Indian regional language Punjabi. Research is further extended to comparison of speech recog- nition system for small vocabulary of speaker dependent isolated spoken words in Indian regional language (Punjabi) using the Hidden Markov Model (HMM) and Dynamic Time Warp (DTW) technique. Punjabi language gives immense changes between consecutive phonemes. Thus, end point detection becomes highly difficult. The presented work emphasizes on template-based recognizer approach using linear predictive coding with dynamic programming computa- tion and vector quantization with Hidden Markov Model based recognizers in isolated word recognition tasks, which also significantly reduces the computa- tional costs. ref/cite Ravinder, Kumar. "Comparison of hmm and dtw for isolated word recognition system of punjabi language." Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer Berlin Heidelberg, 2010. 244-252 [13].

Peter Grasch: this paper we present recomment, our approach to natural language based unit critiquing. it discuss the developed prototype and present the corresponding user interface. In order to show the applicability of our concepts, we present the results of a user study. This study shows that speech interfaces have the potential to improve the perceived ease of use as well as the overall quality of recommendations [14].

R Kuman and R K Sharma: In this paper "An efficient post processing algorithm for online handwriting gurmukhi character recognition using set theory" a post processor for accuracy of character recognition of real-time online Gurmukhi script has been developed. analysis is based on dataset consisting of 184 samples of each 45 characters of Gurmukhi script collected from four different categories of writers. Based on this extensive study, they propose an efficient algorithm for online handwritten Gurmukhi character recognition that achieves promising recognition accuracy of 95.6 percent for single character stroke sequencing. Beside character recognition the contribution in this paper is summarized in two folds as (i) the proposed scheme resolves stroke sequencing, (ii) overwritten strokes are identied and resolved. Moreover, for every stroke, complexity of adding new stroke for Gurmukhi character formation has been computed [15].

Divya Bansal1, Ankita Goel: their papaer "Punjabi speech synthesis system using htk" describes an Hidden Markov Model-based Punjabi text-to-speech synthesis system (HTS), in which speech waveform is generated from Hidden Markov Models themselves, and applies it to Punjabi speech synthesis using the general speech synthesis architecture of HTK (HMM Tool Kit). This Hidden Markov Model based TTS can be used in mobile phones for stored phone directory or messages. Text messages and caller's identity in English language are mapped to tokens in Punjabi language which are further concatenated to form speech with certain rules and procedures [16].

Virender Kadyan: "Punjabi speech to text system for connected words" This paper discusses the implementation of a connected word Speech to Textsystem (STT) for the Punjabi language.Hidden Markov model toolkit (HTK)has been used to develop the system. A Java platform based Graphical User Interface (GUI) has been developed to make the system fast and user friendly.The implemented system performs well with Word Recognition Rate (WR) 95.8 percent and 95.4 percent, Word Accuracy Rate (WA) 94.1 percent and 91.6 percent and Word Error Rate (WER) 5.9 percent and 8.35 percent in class room and open environment respectively [17].

Wiqas Ghai: in their paper " Continuous Speech Recognition for Punjabi Language " work has been done in the field of isolated word and connected word speech recognition for Punjabi language. Acoustic template matching and Vector quantization have been the supporting techniques. Continuous speech recognition is one area where no work has been done so far for Punjabi language. In this paper, an effort has been made to build automatic speech recognizer to recognize continuous speech sentences by using Tri-Phone based acoustic modeling approach on HTK 3.4.1 speech engine. Overall recognition accuracy has been found to be 82.18 percent at sentence level and 94.32 percent at word level [18].

Chapter 4

PROPOSED SCHEME

Motivation

Speech is the most common mode of communication among human beings. Speech offers a new way of interfacing with a computer that lends itself very well to solving the problems of field-based computing. It's natural and intuitive.

Before we take a look at what this technology is all about and how it works, we should make a note of the fast pace at which interactive speech technologies have been coming into the market of late. Whenever a person has to use the PC for typing a document on any word processing software, he/she has to slog hard and spend his valuable time in the dull and boring job of typing the document. Time is wasted, firstly because the typing speed does not match the speed with which a person can think, speak or hear.

In today's fast paced world where every human being has to combat a race against time a direct interface between the spoken word and Pc users will gleefully accept the same words spontaneously typed on the PC screen with minimum lapse of time. These technologies are getting easier to work with too. In the future, speech will definitely be the standard input mode that most of us will replace the traditional keyboard and mouse as the standard input device [19].

The 1990s saw the first commercialization of spoken language understanding systems. Computers can now understand and react to humans speaking in a natural manner in ordinary languages within a limited domain. Basic and applied research in signal processing, computational linguistics and artificial intelligence have been combined to open up new possibilities in human-computer interfaces [21].

So all this benifits motivates me to produce a dictation system for our motheer tongue "Punjabi" language. People from different social and economic backgrounds speak with different different dialects.Computers can now understand and react to humans speaking in a natural manner in ordinary languages within a limited domain. Basic and applied research in signal processing, computational linguistics and artificial intelligence have been combined to open up new possibilities in human-computer interfaces.

System Framework

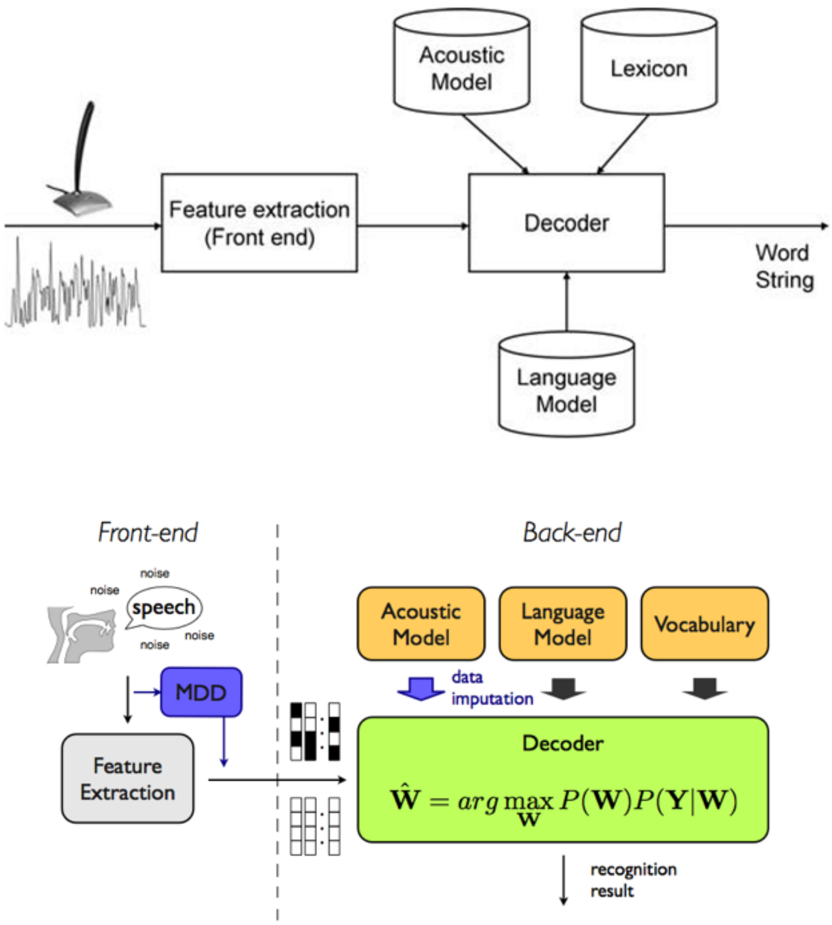

Figure 4.1: System Framework

Chapter 5

EXPERIMENTAL RESULTS AND ANALYSIS

Platform

Operating System is Ubuntu 14.04

Ubuntu is a Debian-based Linux operating system, with Unity as its default desktop environment. It is based on free software and named after the Southern African philosophy of ubuntu (literally, "human-ness"), which often is translated as "humanity towards others" or "the belief in a universal bond of sharing that connects all humanity".

Development of Ubuntu is led by UK-based Canonical Ltd. a company owned by South African entrepreneur Mark Shuttleworth. Canonical generates revenue through the sale of technical support and other services related to Ubuntu. The Ubuntu project is publicly committed to the principles of open-source software development; people are encouraged to use free software, study how it works, improve upon it, and distribute it [24].

Bison

Bison is a general-purpose parser generator that converts an annotated context-free grammar into a deterministic LR or generalized LR (GLR) parser employing LALR(1) parser tables. As an experimental feature, Bison can also generate IELR(1) or canonical LR(1) parser tables. Once you are proficient with Bison, you can use it to develop a wide range of language parsers, from those used in simple desk calculators to complex programming languages.

Bison is upward compatible with Yacc: all properly-written Yacc grammars ought to work with Bison with no change. Anyone familiar with Yacc should be able to use Bison with little trouble. You need to be fluent in C or C++ programming in order to use Bison. Java is also supported as an experimental feature [25].

Swig

To help build extension modules, SWIG is packaged with a library of support files that you can include in your own interfaces. These files often define new SWIG directives or provide utility functions that can be used to access parts of the standard C and C++ libraries. This chapter provides a reference to the current set of supported library files [26].

Gstreamer

GStreamer is a pipeline-based multimedia framework written in the C programming language with the type system based on GObject.

GStreamer allows a programmer to create a variety of media-handling components, including simple audio playback, audio and video playback, recording, streaming and editing. The pipeline design serves as a base to create many types of multimedia applications such as video editors, streaming media broadcasters and media players.

It is designed to work on a variety of operating systems, e.g. Linux kernel-based operating systems, the BSDs, OpenSolaris, Android, OS X, iOS, Windows, OS/400.

GStreamer is free and open-source software subject to the terms of the GNU Lesser General Public License (LGPL) and is being hosted at freedesktop.org [27].

GStreamer has been ported to a wide range of operating systems, processors and compilers. These include but are not limited to Linux on x86, PPC and ARM using GCC. Solaris on x86 and SPARC using both GCC and Forte, MacOSX, Microsoft Windows using MS Visual Developer, IBM OS/400 and Symbian OS.

GStreamer can bridge to other multimedia frameworks in order to reuse existing components (e.g. codecs) and use platform input/output mechanism [28].

Gcc

GCC was originally written as the compiler for the GNU operating system. The GNU system was developed to be 100 percent free software, free in the sense that it respects the user's freedom.

We strive to provide regular, high quality releases, which we want to work well on a variety of native and cross targets (including GNU/Linux), and encourage everyone to contribute changes or help testing GCC. Our sources are readily and freely available via SVN and weekly snapshots [27].

Automake

Automake is a tool for automatically generating Makefile.in files compliant with the GNU Coding Standards. Automake requires the use of Autoconf [31]. The Automake manual can be read on-line or downloaded in PDF format; also, more formats are offered for download or on-line reading. If you have installed Automake on your system, you may also find more information about it by looking at your local documentation; for example you might use info automake at the shell prompt [27].

Autoconf

Autoconf is an extensible package of M4 macros that produce shell scripts to automatically configure software source code packages. These scripts can adapt the packages to many kinds of

UNIX-like systems without manual user intervention. Autoconf creates a configuration script for a package from a template file that lists the operating system features that the package can use, in the form of M4 macro calls [28].

Libtool

GNU libtool is a generic library support script. Libtool hides the complexity of using shared libraries behind a consistent, portable interface.

To use libtool, add the new generic library building commands to your Makefile, Makefile.in, or Makefile.am [28].

Speech Recognition Engine

Library

Sphinxbase

This package contains the basic libraries shared by the CMU Sphinx trainer and all the Sphinx decoders (Sphinx-II, Sphinx-III, and PocketSphinx), as well as some common utilities for manipulating acoustic feature and audio files.

Sphinxbase uses the standard unix autogen system, and there's a script included, 'build for iphone.sh' that will setup configure to create binaries that are XCode friendly.

./autogen.sh

Python-dev

Header files, a static library and development tools for building Python modules, extending the Python interpreter or embedding Python in applications [20].

Decoder

PocketSphinx

PocketSphinx is CMU's fastest speech recognition system. It's a library written in pure C which is optimal for development of your C applications as well as for development of language bindings. At real time speed it's the most accurate engine, and therefore it is a good choice for live applications.

It's good for desktop applications, command and control and dictation where fast response and low resource consumption are the goals.

Also it includes support for embedded devices with fixed-point ariphmetics and is successfully used on IPhone, Nokia devices and on Windows Mobile. You can find further documentation about PocketSphinx in the release documentation, or at the online documentation [29].

Sphinx-4

Sphinx-4 is a state-of-the-art speech recognition system written entirely in the Java™ programming language. It's best for implementation of complex server or cloud-based system with deep interaction with NLP modules, web services and cloud computing [29].

Trainer

SphinxTrain

New openfst-based G2P trainer and decoder, supported by Sphinx4 too. It includes Parallel feature extraction. Package can be installed now just like any application. Single 'sphinxtrain' command to access all training process. Increased reuse of sphinxbase functions [30].

Experimental Setup

1. Installation of OS(operating system) i.e Ubuntu
2. Install Bison
3. Install Swig
4. Install Python-dev
5. Install Automake
6. Install Autoconf
7. Install Libtool
8. Download Sphinxbase-5prealpha and compile it and then install from the source
9. Download PocketSphinx-5prealpha and compile it and then install from the source
10. Install Gstreamer
11. Download SphinxTrain-5prealpha and compile it and then install from the source

Dictation System Setup

Dictation system setup includes the making of Acoustic model and Language model for the Punjabi language. Also the use of CMULTK and testing on the PocketSphinx decoder.

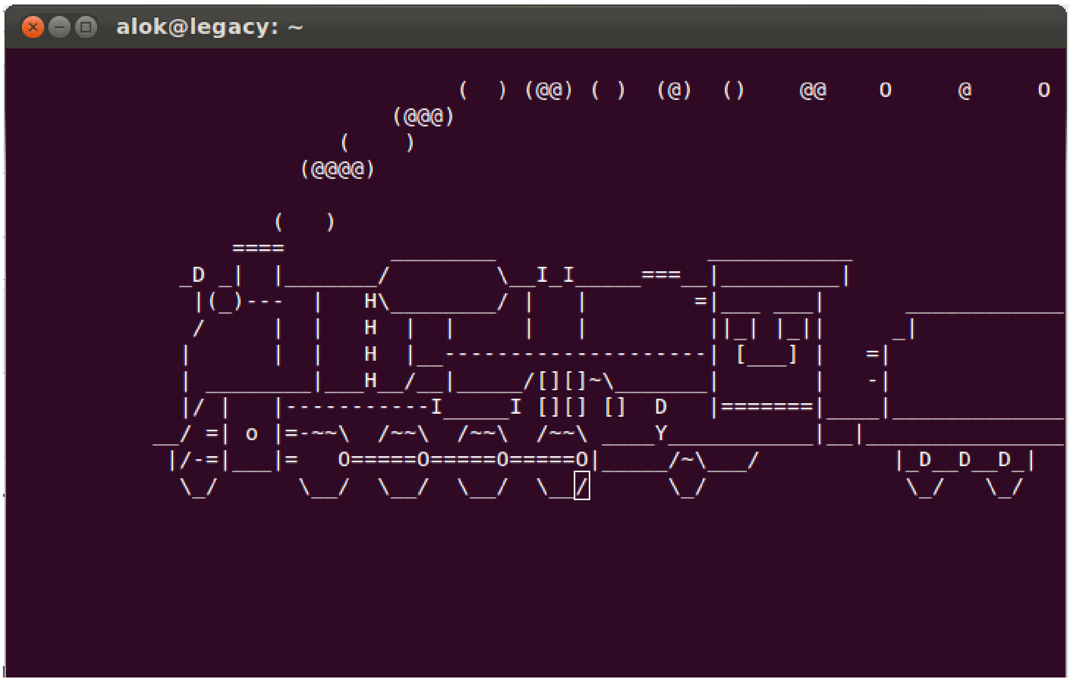

Figure 5.1: Ubuntu Terminal

Testing the Decoder

Before proceeding further it is important to check the existing decoder for the existing language model, so i test it with the exixting English model. So i used Ubuntu terminal for this purpose.

The sphinxbase will be installed in /usr/local/ folder by default. Not every system loads libraries from this folder automatically. To load them you need to configure the path to look for shared libaries. It can be done either in the file /etc/ld.so.conf or with exporting environment variables:
export LD_LIBRARY_PATH=/usr/local/lib
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig

To test installation, run 'pocketsphinx_continuous -inmic yes' and check that it recognizes words you are saying to the microphone.

Building Language Model

There are two types of models that describe language - grammars and statistical language models. Grammars describe very simple types of languages for command and control, and they are usually written by hand or generated automatically with scripting code. Grammars usually do not have probabilities for word sequences, but some elements might be weighed. Grammars could be created with JSGF format and usually have extension like .gram or .jsgf.

Statistical language models describe more complex language. They contain probabilities of the

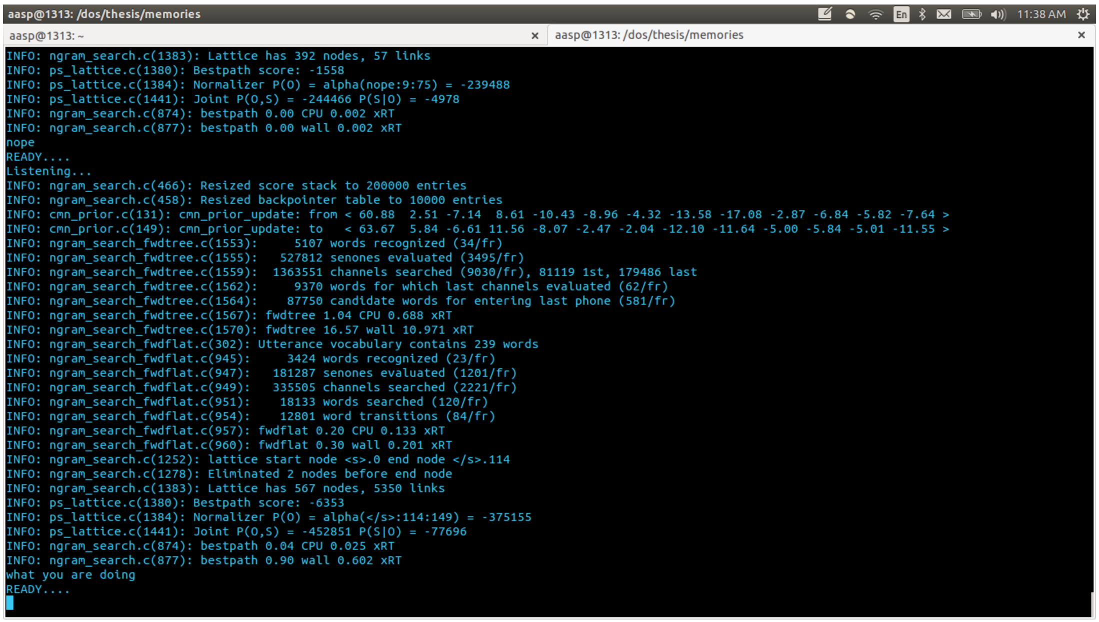

Figure 5.2: PocketSphinx Test

words and word combinations. There are many ways to build the statistical language models. When your data set is large, there is sense to use CMU language modeling toolkit. When a model is small, you can use an online quick web service. When you need specific options or you just want to use your favorite toolkit which builds ARPA models, you can use it.

Language model can be stored and loaded in two different format - text ARPA format and binary DMP format. ARPA format takes more space but it is possible to edit it. ARPA files have .lm extension. DMP format takes significantly less space and faster to load. DMP files have .lm.dmp extension. It is also possible to convert between formats.

Text Preparation

First of all you need to cleanup text. Expand abbreviations, convert numbers to words, clean non-word items. Language modeling for Mandarin is largely the same as in English, with one addditional consideration, which is that the input text must be word segmented. A segmentation tool and associated word list is provided to accomplish this.

ARPA Model Training

The process for creating a language model is as follows:

1. Prepare a reference text that will be used to generate the language model. The language model toolkit expects its input to be in the form of normalized text files, with utterances delimited by <s> and </s> tags. A number of input filters are available for specific corpora such as Switchboard, ISL and NIST meetings, and HUB5 transcripts. The result should be

the set of sentences that are bounded by the start and end sentence markers: <s> and </s>. Here's an example:

<s> ਮੰਨ ਲਈ ਜੋ ਕਰਦਾ ਰੱਬ ਪਾਕਿ ਐ </s>
<s> ਆਉਂਦੀ ਯਾਦ ਵਤਨ ਦੀ ਖ਼ਾਕ ਐ </s>
<s> ਟੁੱਟ ਫੁੱਟ ਟੁਕੜੇ ਬਣ ਗਏ ਦਿਲ ਦੇ </s>
<s> ਹਾਏ ਮੈਂ ਭੁੱਜ ਗਿਆ ਵਾਂਗੂੰ ਖਿੱਲ ਦੇ </s>
<s> ਭੜਥਾ ਬਣ ਗਈ ਦੇਹੀ ਐ </s>
<s> ਵਿਛੜੇ ਯਾਰ ਪਿਆਰੇ, ਬਣੀ ਮੁਸੀਬਤ ਕੇਹੀ ਐ </s>

Figure 5.3: Text Prepration

More data will generate better language models.

2. Generate the vocabulary file. This is a list of all the words in the file:

   text2wfreq < weather.txt | wfreq2vocab > weather.tmp.vocab

3. Edit the vocabulary file to remove words (numbers, misspellings, names).

4. Remove sentences from your input transcript that contain words that are not in your vocabulary file.

5. Generate the arpa format language model with the commands:

   text2idngram -vocab weather.vocab -idngram weather.idngram < weather.closed.txt
   idngram2lm -vocab_type 0 -idngram weather.idngram -vocab
   weather.vocab -arpa weather.lm

6. Generate the CMU binary form (DMP)

   sphinx_lm_convert -i weather.lm -o weather.lm.DMP

Using Language Model with PocketSphinx

pocketsphinx_continuous can be run from the command-line to recognize speech. Assuming it is installed under */usr/local*, and your language model and dictionary are called *.dic* and *.lm* and placed in the current folder, try running the following command:

pocketsphinx_continuous -inmic yes -lm weather.lm -dict weather.dic

Training Acoustic Model For Sphinx

CMUSphinx project comes with several high-quality acoustic models. Before starting with training we need to prepared the language model and need to train the model and have resources to do that.

Resources

- 1 hour of recording for command and control for single speaker
- 5 hour of recordings of 200 speakers for command and control for many speakers
- 10 hours of recordings for single speaker dictation
- 50 hours of recordings of 200 speakers for many speakers dictation
- Knowledge on phonetic structure of the language
- Time to train the model and optimize parameters (1 month)

Data Preparation

The trainer learns the parameters of the models of the sound units using a set of sample speech signals. This is called a training database. A choice of already trained databases will also be provided.

The database contains information required to extract statistics from the speech in form of the acoustic model.

The trainer needs to be told which sound units want it to learn the parameters of, and at least the sequence in which they occur in every speech signal in training database. This information is provided to the trainer through a file called the transcript file, in which the sequence of words and non-speech sounds are written exactly as they occurred in a speech signal, followed by a tag which can be used to associate this sequence with the corresponding speech signal.

The trainer then looks into a dictionary which maps every word to a sequence of sound units, to derive the sequence of sound units associated with each signal.

Thus, in addition to the speech signals, also be given a set of transcripts for the database (in a single file) and two dictionaries, one in which legitimate words in the language are mapped sequences of sound units (or sub-word units), and another in which non-speech sounds are mapped to corresponding non-speech or speech-like sound units. We will refer to the former as the language dictionary and the latter as the filler dictionary.

After training, it's mandatory to run the decoder to check training results. The Decoder takes a model, tests part of the database and reference transcriptions and estimates the quality (WER) of the model. During the testing stage we use the language model with the description of the order of words in the language.

First of all, design a database for training or download an existing one. For example, you can purchase a database from LDC. You'll have to convert it to a proper format.

The file structure for the database is:

▽ etc

- some_db.dic - *Phonetic dictionary*
- some_db.phone - *Phoneset file*
- some_db.lm.DMP - *Language model*
- some_db.filler - *List of fillers*
- some_db_train.fileids - *List of files for training*
- some_db_train.transcription - *Transcription for training*
- some_db_test.fileids - *List of files for testing*
- some_db_test.transcription - *Transcription for testing*
- some_
- some_

▽ wav

  ▽ speaker_1

  - file_1.wav - *Recording of speech utterance*

  ▽ speaker_2

  - file_2.wav

Fileids (some_db_train.fileids and some_db_test.fileids) file is a text file listing the names of the recordings (utterance ids) one by line.

Transcription file (your_db_train.transcription and your_db_test.transcription) is a text file listing the transcription for each audio file. t's important that each line starts with <s> and ends with </s> followed by id in parentheses. Also note that parenthesis contains only the file, without speaker_n directory. It's critical to have exact match between fileids file and the transcription file. The number of lines in both should be identical. Last part of the file id (speaker1/file_1) and the utterance id file_1 must be the same on each line.

Speech recordings (wav files) Recording files must be in MS WAV format with specific sample rate - 16 kHz, 16 bit, mono for desktop application, 8kHz, 16bit, mono for telephone applications. Double-check that, wrong audio file format is the most common source of training issues. Audio files shouldn't be very long and shouldn't be very short. Optimal length is not less than 5 seconds and not more than 30 seconds. Amount of silence in the beginning of the utterance and in the end of the utterance should not exceed 0.2 second.

Phoneset file (your_db.phone) should have one phone per line. The number of phones should match the phones used in the dictionary plus the special SIL phone for silence.

Language model file (your_db.lm.DMP) should be in ARPA format or in DMP format. *Filler dictionary (your_db.filler)* contains filler phones (not-covered by language model non-linguistic sounds like breath, hmm or laugh).

Setting up the training scripts

To start the training change to the database folder and run the following commands:

sphinxtrain -t <model_name> setup

This will copy all the required configuration files into etc subfolder of your database folder and prepare database for training, the structure after setup will be:

- etc
- wav

After training other data folders will be created, the database should look like this:

▷ etc

- ▷ feat
- ▷ logdir
- ▷ model_parameters
- ▷ model_architecture
- ▷ result
- ▷ wav

Setup the Format of Database Audio

After setup, we need to edit the configuration files in etc folder, there are many variables but to get started we need to change only a few. First of all find the file *etc/sphinx_train.cfg*
$CFG_WAVFILES_DIR = "$CFG_BASE_DIR/wav";
$CFG_WAVFILE_EXTENSION = 'sph';
$CFG_WAVFILE_TYPE = 'nist'; one of nist, mswav, raw

Configure Path to Files

See the following lines in your etc/sphinx_train.cfg file:
$CFG_DICTIONARY = "$CFG_LIST_DIR/$CFG_DB_NAME.dic";
$CFG_RAWPHONEFILE = "$CFG_LIST_DIR/$CFG_DB_NAME.phone";
$CFG_FILLERDICT = "$CFG_LIST_DIR/$CFG_DB_NAME.filler";
$CFG_LISTOFFILES = "$CFG_LIST_DIR/$CFG_DB_NAME_train.fileids";
$CFG_TRANSCRIPTFILE = "$CFG_LIST_DIR/$CFG_DB_NAME_train.transcription"

Configure Model Type and Model Parameters

$CFG_HMM_TYPE = '.cont.'; # Sphinx4, Pocketsphinx
#$CFG_HMM_TYPE = '.semi.'; # PocketSphinx only
#$CFG_HMM_TYPE = '.ptm.'; # Sphinx4, Pocketsphinx, faster model

If you are training continuous models for large vocabulary and have more than 100 hours of data, put 32 here. It can be any degree of 2: 4, 8, 16, 32, 64. *$CFG_FINAL_NUM_DENSITIES = 8;*

This value is the number of senones to train in a model. The more senones model has, the more precisely it discriminates the sounds. But on the other hand if you have too many senones, model will not be generic enough to recognize unseen speech. That means that the WER will be higher on unseen data. That's why it is important to not overtrain the models. In case there are

Table 5.1: Continuous Model Senones and Density

Number of Senones and Number of Densities				
Vocabulary	Hours in db	Senones	Densities	Example
20	5	200	8	Tidigits Digits Recognition
100	20	2000	8	RM1 Command and Control
5000	30	4000	16	WSJ1 5k Small Dictation
20000	80	4000	32	WSJ1 20k Big Dictation
60000	200	6000	16	HUB4 Broadcast News
60000	2000	12000	64	Fisher Rich Telephone Transcription

too many unseen senones, the warnings will be generated in the norm log on stage 50 below:
ERROR: "gauden.c", line 1700: Variance (mgau= 948, feat= 0, density=3, component=38) is less then 0. Most probably the number of senones is too high for such a small training database. Use smaller $CFG_N_TIED_STATES.

It might seem that diversity could improve the model but it's not the case. Diverse speech requires some artificial speech prompts and that decrease the naturalness of the speech. Artificial models don't help in real life decoding. In order to build a best database you need to try to reproduce real environment as much as possible. It's even better to collect more speech to try to optimize the database size.

It's important to remember, that optimal numbers depends on your database. To train model properly, you need to experiment with different values and try to select the ones which give best WER for a development set. You can experiment with number of senones, number of gaussian mixtures at least. Sometimes it's also worth to experiment with phoneset or number of estimation iterations.

Configure Sound Feature Parameters

The default for sound files used in Sphinx is a rate of 16 thousand samples per second (16KHz). If this is the case, the etc/feat.params file will be automatically generated with the recommended values.

 # Feature extraction parameters $CFG_WAVFILE_SRATE = 16000.0;
$CFG_NUM_FILT = 31; # For wideband speech it's 40, for telephone 8khz reasonable value is 31
$CFG_LO_FILT = 200; # For telephone 8kHz speech value is 200
$CFG_HI_FILT = 3500; # For telephone 8kHz speech value is 3500

Configure Parallel Jobs to Speedup Training

If you are on multicore machine or in PBS cluster you can run training in parallel, the following options should do the trick:
 # Queue::POSIX for multiple CPUs on a local machine # Queue::PBS to use a PBS/TORQUE queue $CFG_QUEUE_TYPE = "Queue";

Configure Decoding Parameters

Open *etc/sphinx_train.cfg*, make sure the following is properly configured:
$DEC_CFG_DICTIONARY = "$DEC_CFG_BASE_DIR/etc/$DEC_CFG_DB_NAME.dic";
$DEC_CFG_FILLERDICT = "$DEC_CFG_BASE_DIR/etc/$DEC_CFG_DB_NAME.filler";
$DEC_CFG_LISTOFFILES = "$DEC_CFG_BASE_DIR/etc/$DEC_CFG_DB_NAME_test.fileids";
$DEC_CFG_TRANSCRIPTFILE = "$DEC_CFG_BASE_DIR/etc/
$DEC_CFG_DB_NAME_test.transcription";
$DEC_CFG_RESULT_DIR = "$DEC_CFG_BASE_DIR/result";
These variables, used by the decoder, have to be user defined, and
may affect the decoder output
$DEC_CFG_LANGUAGEMODEL_DIR = "$DEC_CFG_BASE_DIR/etc";
$DEC_CFG_LANGUAGEMODEL = "$DEC_CFG_LANGUAGEMODEL_DIR
/model_name.lm.DMP";

Training

First of all, go to the database directory: *cd some_dir*

To train, just run the following commands:

sphinxtrain run

and it will go through all the required stages. It will take a few minutes to train. On large databases, training could take a month. During the stages, the most important stage is the first one which checks that everything is configured correctly and your input data is consistent. Do not ignore the errors reported on the first 00.verify_all step.

The typical output during decoding will look like:

Training Internals

This section describes what happens during the training. In the scripts directory (./scripts_pl), there are several directories numbered sequentially from 00 through 99. Each directory either has a directory named slave*.pl or it has a single file with extension .pl. The script sequentially goes through the directories and executes either the the slave*.pl or the single .pl file, as below.
perl scripts_pl/000.comp_feat/slave_feat.pl

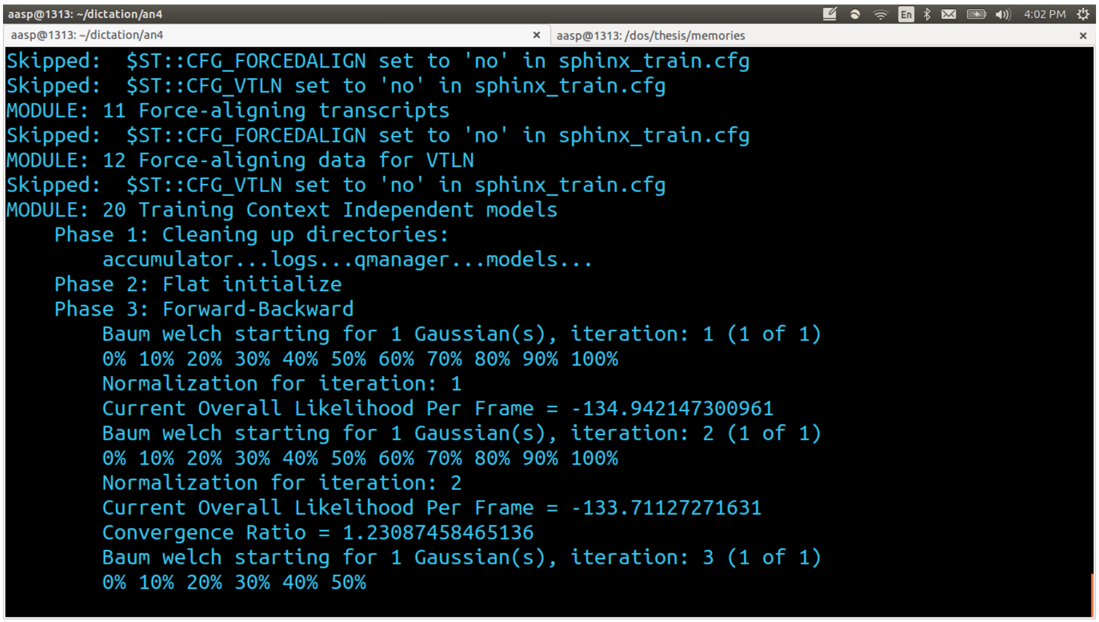

Figure 5.4: Training Model

perl scripts_pl/00.verify/verify_all.pl
perl scripts_pl/10.vector_quantize/slave.VQ.pl
perl scripts_pl/20.ci_hmm/slave_convg.pl
perl scripts_pl/30.cd_hmm_untied/slave_convg.pl
perl scripts_pl/40.buildtrees/slave.treebuilder.pl
perl scripts_pl/45.prunetree/slave-state-tying.pl
perl scripts_pl/50.cd_hmm_tied/slave_convg.pl
perl scripts_pl/90.deleted_interpolation/deleted_interpolation.pl

Scripts launch jobs on your machine, and the jobs will take a few minutes each to run through.

Before you run any script, note the directory contents of your current directory. After you run each slave*.pl note the contents again. Several new directories will have been created. These directories contain files which are being generated in the course of your training. At this point you need not know about the contents of these directories, though some of the directory names may be self explanatory and you may explore them if you are curious.

One of the files that appears in your current directory is an .html file, named model.html, depending on which database you are using. This file will contain a status report of jobs already executed. Verify that the job you launched completed successfully. Only then launch the next slave*.pl in the specified sequence. Repeat this process until you have run the slave*.pl in all directories.

Note that in the process of going through the scripts in 00* through 90*, you will have generated several sets of acoustic models, each of which could be used for recognition. Notice also that some of the steps are required only for the creation of semi-continuous models. If you execute these steps while creating continuous models, the scripts will benignly do nothing.

On the stage 000.comp_feat the feature feles are extracted. The system does not directly work with acoustic signals. The signals are first transformed into a sequence of feature vectors, which are used in place of the actual acoustic signals.

This script slave_feat.pl will compute, for each training utterance, a sequence of 13-dimensional vectors (feature vectors) consisting of the Mel-frequency cepstral coefficients (MFCCs). Note that the list of wave files contains a list with the full paths to the audio files. Since the data are all located in the same directory as you are working, the paths are relative, not absolute. You may have to change this, as well as the model_test.fileids file, if the location of data is different. The MFCCs will be placed automatically in a directory called 'feat'. Note that the type of features vectors you compute from the speech signals for training and recognition, outside of this tutorial, is not restricted to MFCCs. You could use any reasonable parameterization technique instead, and compute features other than MFCCs. CMUSphinx can use features of any type or dimensionality. The format of the features is described on the page MFC Format .

Testing

It's critical to test the quality of the trained database in order to select best parameters, understand how application performs and optimize performance. To do that, a test decoding step is needed. The decoding is now a last stage of the training process.

You can restart decoding with the following command: *sphinxtrain -s decode run* This command will start a decoding process using the acoustic model you trained and the language model you configured in the *etc/sphinx_train.cfg* file.

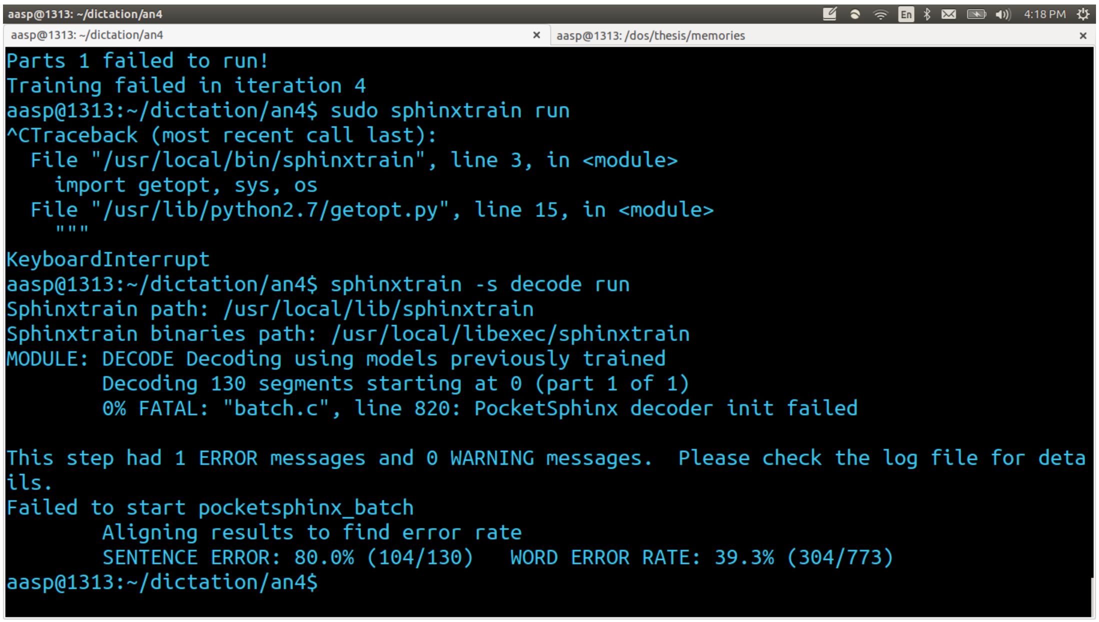

Figure 5.5: Testing Model

Using the Model

After training, the acoustic model is located in

model_parameters/<your_db_name>.cd_cont_<number_of senones>
or in
model_parameters/<your_db_name>.cd_semi_<number_of senones>
The model should have the following files:

- mdef
- feat.params
- mixture_weights
- means
- noisedict
- transition_matrices
- variances

Depending on the type of the model you trained. To use the model in pocketsphinx, simply point to it with the -hmm option:

pocketsphinx_continuous -hmm <model_folder> -lm <some_lm> -dict <some_dict>.

Output

The output the model trained for Punjabi language is given in the figure below:

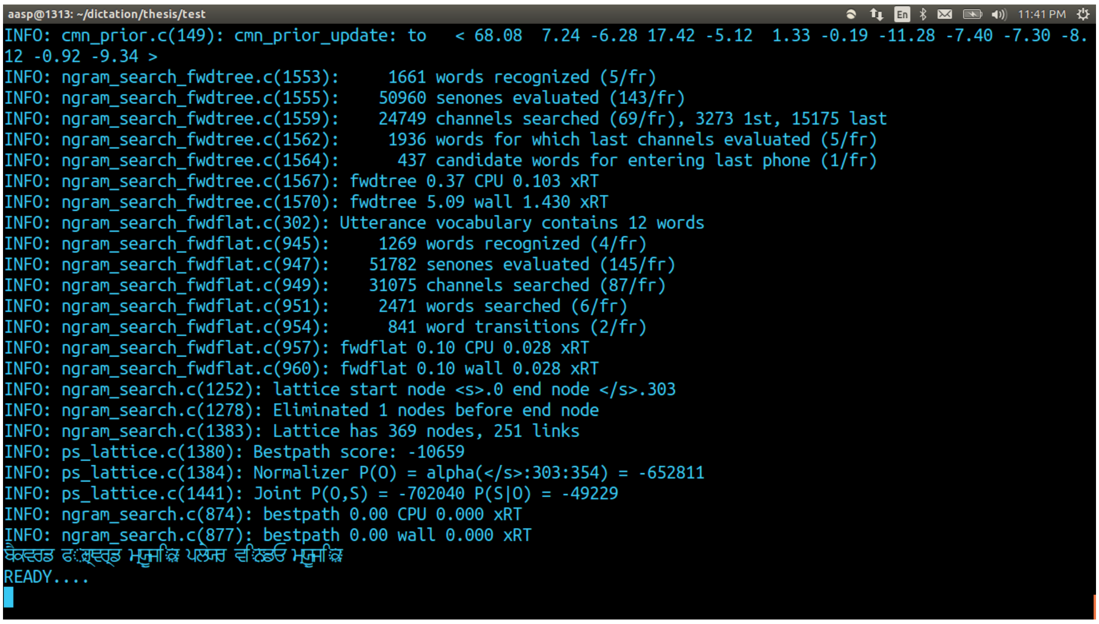

Figure 5.6: Punjabi Dictation

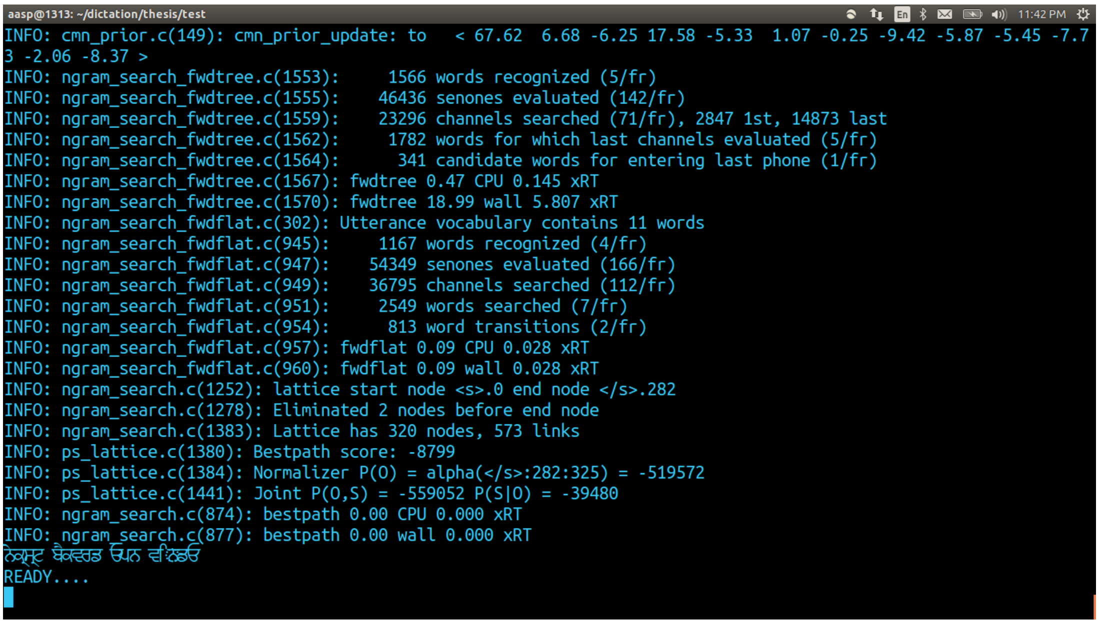

Figure 5.7: punjabi Dictation

Chapter 6

CONCLUSION

Conclusion

For the past many years, several mechanisms for the communication among human- machine have been explored. There is much evidence that human speech dictation in machine involves the integration of a great variety of knowledge sources, including knowledge of the world or context, knowledge of the speaker and/or topic and many more. Although there have been significant recent gains in speech dictation, current technology is far from human-like: only systems in limited domains can be envisioned in the near term, and the portability of existing techniques is still rather limited. Application areas that appear to be a good match to technology on the near horizon include those that are naturally limited.

The objective of the thesis was “speech pattern recognition for speech to text conversion”. The researcher desire was to pronounced the Punjabi character from any user, identify it and print it in the text editor. In other word, the recognition of the Punjabi character or one can say the dictation of the Punjabi character.

Future Work

The speech dictation is a very huge research work area and required a team of skilled researcher. Although, the researcher has tried his level best to detect the Punjabi character. When the idea of the speech dictation was born in the mind of the researcher, his dream was to dictate the Punjabi continuous speech. The work is of considerable domain and hence the beginning to end of the work has made with the detection of Punjabi characters and with its total analysis. So the present study has scope for the continuation of research for all the remaining Punjabi characters, which further can be extended in the Punjabi word dictation and ultimately the dictation of words can again be extended for the continuous speech dictation in Punjabi language.

Although many recent advances and successes in speaker recognition have been achieved, there are still many possible enhancement for which good solutions are to be found. Most of these problems arise from variability, including speaker-generated variability and variability in channel and recording conditions. It is very important to investigate feature parameters that are stable over time, insensitive to the variation of speaking manner, including the speaking rate and level, and robust against variations in voice quality due to causes such as voice disguise or colds and many more ...

Bibliography

[1] Chowdhury, Gobinda G. "Natural language processing." *Annual review of information science and technology 37.1,* 2003: 5189.

[2] Kaur, Navdeep,Vandana Pushe, and Rupinderdeep Kaur. Natural Language Processing Interface for Synonym." *from outside the contents of the document,* 2014.

[3] Kaur, Kamaldeep, and Vishal Gupta. "Name and Entity Recognition for Punjabi Language." *Machine translation 2.3,* 2012.

[4] Necula, George C."Translation validation for an optimizing compiler." *ACM sigplan notices. Vol. 35. No. 5. ACM,* 2000.

[5] Mac, David, Tom, and Alexendre. "GNU Automake." *User Manual, for Automake version 1,* 1995.

[6] Calcote, John. "Autotools: A Practitioner's" *Guide to GNU Autoconf, Automake, and Libtool. No Starch Press,* 2010.

[7] Byrne, William, et al. "Towards language independent acoustic modeling." *Acoustics, Speech, and Signal Processing,* 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on. Vol. 2. IEEE, 2000.

[8] Lippmann, Richard P. "Speech recognition by machines and humans." *Speech communication 22.1,* 1997.

[9] Schultz, TAnja, and Alex. "Language independent and languge adaptive acoustic model for speech recognition." *Speech Communication 35.1,* 2001.

[10] Walker, Willie, et al. "Sphinx4: A flexible open source framework for speech recognition,", 2004.

[11] Sankar, Fuliang Weng Andreas Stolcke Ananth. "Efficient Lattice Representation and Generation.", 2006.

[12] Dua, Mohit, et al. "Punjabi automatic speech recognition using HTK." *IJCSI International Journal of Computer Science Issues 9.4,* 2012.

[13] Ravinder, Kumar. "Comparison of hmm and dtw for isolated word recognition system of punjabi language." *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer Brelin,* 2010.

[14] Grasch, Peter, Alexander Felfernig, and Florian Reinfrank. "ReComment: Towards critiquing- based recommendation with speech interaction." *Proceedings of the 7th ACM Conference on Recommender Systems. ACM,* 2013.

[15] Ravinder, and Rajendra Kumar Sharma. "An efficient post processing algorithm for online handwriting Gurmukhi character recognition using set theory." *International Journal of Pattern Recognition and Artificial Intelligence 27.04,* 2013.

[16] Bansal, Divya, Ankita Goel, and Khushneet Jindal. "Punjabi Speech Synthesis System Using Htk." *International Journal of Information 2.4,* 2012.

[17] Dua, Mohit, et al. "Punjabi speech to text system for connected words.", 2012.

[18] Ghai, Wiqas, and Navdeep Singh. "Continuous Speech Recognition for Punjabi."2013.

[19] http://www.programmershare.com/

[20] http://cmusphinx.sourceforge.net/wiki/

[21] [6]Kumbharana, Chandresh K."Speech Pattern Recognition for Speech to Text Conversion." *Diss. Saurashtra University,* 2007.

[22] Electronics for U may 2000

[23] IBM, Developing voice applications.

[24] https://en.wikipedia.org/wiki/Ubuntu_(28operating_system)

[25] http://www.gnu.org/software/bison/

[26] http://www.swig.org/Doc1.3/Library.html

[27] https://en.wikipedia.org/wiki/GStreamer

[28] http://gstreamer.freedesktop.org/features/

[29] https://launchpad.net/ubuntu/precise/+package/pythondev

[30] http://sourceforge.net/projects/cmusphinx/files/sphinxtrain/5prealpha/

[31] https://www.gnu.org/software/automake/