A COMPARISON OF PUBLIC DOMAIN SOFTWARE TOOLS FOR SPEECH RECOGNITION

Samudra Vijayakante and Maria Barot*

School of Technology and Computer Science
Tata Institute of Fundamental Research
Homi Bhabha Road, Mumbai 400005
chief@tifr.res.in, mariabarot@yahoo.com

ABSTRACT

HTK and Sphinx are two freely downloadable software packages with the capability of implementing a large vocabulary, speaker independent, continuous speech recognition system in any language. While HTK has been in use by various groups for about a decade, and has gone through the refinement cycles necessary for a commercial software, Sphinx was released about an year ago and is still undergoing development in a university environment. However, due to certain advanced features and the license for unrestricted use, Sphinx appears to be more attractive. These two software packages have been compared by implementing a Hindi speech recognition system. Although recognition accuracies of the two systems are comparable, we observe that the acoustic modeling of Sphinx is superior.

1. INTRODUCTION

Software tools have made it easy to implement systems and to test new ideas quickly. Availability of public domain software for automatic speech recognition (ASR) has increased in the recent past. A notable trend is the distribution of not only the executable version, but also the source code of the toolkit for experimentation and personal use. The latter enables a researcher to modify the source code to implement a new feature vector or model architecture or pattern matching strategy.

Of course, the software architecture of the toolkit does put in constraints to the variety of changes that can be made and the ease with which these changes can be made. Nevertheless, public domain software toolkits are useful as they reduce the programming load of the user and permits him to concentrate more on the experiment on hand. In addition, discussion groups associated with such software facilitates exchange of ideas for improvement, solutions to difficulties faced and contribution of revised or new modules for incorporation into future versions of the software.

The most successful model for large vocabulary, speaker independent, continuous speech recognition system is hidden Markov model (HMM) [6]. The first user-friendly, versatile HMM toolkit for speech recognition made available to speech community is HTK from Entropic Research Laboratory Inc. Following the takeover of Entropic by Microsoft, HTK became freely available in source form for personal use from Cambridge University Engineering Department web site [1].

Another ASR software available in the public domain is Sphinx from Carnegie Mellon University (CMU) [2]. For the past several decades, speech recognition systems from CMU were in the forefront in performance evaluations [7] by Defense Advanced Research Projects Agency, USA. CMU Sphinx is a large vocabulary, speaker independent speech recognition code base and suite of tools. The code can be downloaded freely; there is no restriction against commercial use or redistribution.

Yet another ASR software package is from Institute of Information and Signal Processing (ISIP),
Mississippi State University [3]. The ultimate goal of the project as stated at the web site is to develop public domain software capable of implementing Large Vocabulary Conversational Speech Recognition system. The ongoing efforts are building an efficient decoder capable of handling cross-word triphones, and generation of word-graphs starting from N-Gram language models.

While Sphinx and HTK are written in C language, the ISIP system is built on top of a vast hierarchy of general purpose C++ classes. HTK has been used by the speech community for more than a decade. The recognition engine of Sphinx was released in early 2000, and an acoustic training module was released in June 2001. Thus Sphinx has been available to the public for experimentation for more than an year. The ISIP system is under development and a “production” version system was released in April 2002. This version came with a tutorial for recognition of isolated digits. The TIFR speech group has been using HTK for several years. Since the public domain Sphinx package became available an year ago, we wanted to compare and contrast the various facets of these two systems in the context of their use for recognition of spoken Hindi. The preliminary observations of such an exercise is reported here.

2. EXPERIMENTAL DETAILS

A comparative experiment was conducted using HTK and Sphinx software packages. The aim was to compare the performances as well as the ease of implementation of the two systems in a given task domain. The details of the task domain, speech database, the signal processing and the configuration of models are given in this section.

2.1. Speech Database

The experiments were carried out in the context of a railway reservation enquiry task. The database consists of Hindi sentences spoken by 48 volunteers - 32 male and 16 female. The text corpus consists of 320 sentences and the size of the vocabulary was 161 words. The sentences in the corpus were chosen such that the corpus contains at least

3 repetitions of each word in the lexicon. The following are two typical sentences in the text corpus.

kyaa rajdhani express me teis ac
two tier sleeper kaa ticket parson
ke liye mil_saktaa hei
bambai se chennai ke liye ac first
class kaa kirayaa kitnaa hogaa

Each speaker read a minimum of 10 sentences. A few volunteers participated in more than one recording session. In addition, the database contains one or two sentences spoken by visitors to the lab. There are a total of 999 Hindi utterances in the database.

The speech data was recorded using a high quality, directional microphone (Shure SM48 low impedance), sampled at 16kHz, and quantised with 16 bits using sound blaster Live card. Standard signal processing methods were employed in the experiment. Speech was blocked into overlapping frames of 25msec size with 10msec shift between successive frames. Energy and Mel frequency cepstral coefficients (MFCC) [4] were extracted from each frame. The time derivatives of static features were also used. Cepstral mean normalisation was used to minimise the speaker dependent effects.

2.2. Acoustic model

Each word was represented as a sequence of phone-like units. A pronunciation dictionary contained transcription of each word in the vocabulary in terms of 50 phone-like units. Each unit was modeled by a left-to-right HMM with 3 emitting states. Context dependent (‘triphone’) models were used. The transition matrices of all triphones of a given phone are tied. In addition, emission distributions of similar acoustic states are also tied to ensure that all state distributions can be robustly estimated with the limited training data.

In HTK based system, continuous density output distributions were used. Each state was represented by a single multivariate Gaussian density with diagonal covariance matrix. The training data could be represented by a total of 928 context dependent models. HTK provides two mechanisms for state tying. The data driven approach
uses a similarity measure between states. The decision tree based approach is based on asking questions about the left and right contexts of each triphone. The latter approach was used since Sphinx also uses the decision tree mechanism for state tying. The number of context dependent states reduced from 928 to 400 after tying states using a decision tree.

While implementing an ASR system using Sphinx package, we faced a few impediments. So, we had to devise means of going around these barriers. In order to appreciate these steps, a few words about the two versions of Sphinx package is in order. The Sphinx2 is a real-time system. Sphinx3 is a later, slower but more accurate recognizer. Sphinx2 employs semi-continuous models (uses tied mixtures), and Sphinx3 can use fully continuous observation densities. Sphinx3 is still under development, but the code base is cleaner than that of Sphinx2. Also, the phone sets of two versions are slightly different.

SphinxTrain is an environment for building acoustic models for the CMU Sphinx2 and Sphinx3 engines. This trainer produces Sphinx3 compatible models but includes tools for converting semi-continuous models to Sphinx2 format. Since Sphinx3 decoder is yet to be released, one has to train models using SphinxTrain to generate Sphinx3 compatible models and then convert them to Sphinx2 format so that the Sphinx2 decoder can be used for recognition. Figure 1 shows the steps involved in training a speech recognition system using Sphinx.

While the generation of regression tree based on linguistic questions was straightforward in HTK, the current version of Sphinx is capable of generating regression tree only for continuous density HMMs. However, the converter tool can convert only semi-continuous models to Sphinx2 format. So, We trained 5-state context independent (CI) models with continuous density initially. Linguistic questions were generated and a regression tree for state tying was derived using these CI models. Then, 5-state CI models with semi-continuous densities were trained and the corresponding context dependent models (CD) were trained using Sphinx3. The states of these CD models with semi-continuous densities were tied using the regression tree derived using CI models with continuous densities. These tied models in Sphinx3 format were then converted to Sphinx2 format for recognising test data.

Sphinx employs deleted interpolation for robust estimation of parameters. Deleted interpolation is an iterative process to interpolate between CD and CI mixture-weights to reduce the effects of over fitting. The data are divided into two sets, and the data from one set are used to estimate the optimal interpolation factor between CI and CD models trained from the other set. Then the two data sets are switched; this procedure is repeated using the last estimated interpolation factor as an initialization for the current step. The switching is continued until the interpolation factor converges.

Figure 1: A block diagram of generating trained models using Sphinx software package. Here, CI and CD refer to Context Independent and Context Dependent models respectively. Since the current training module generates models in the Sphinx3 format, and the Sphinx3 decoder is yet to be released, the trained models need to be converted to the Sphinx2 format. The converter tool can convert only semi-continuous models. In the current version of Sphinx, decision tree for state tying could be generated using only continuous density (and not semi-continuous density) models. Therefore, CI continuous models need to be trained for the generation of regression tree.
2.3. Language model

In the system implemented using HTK, the language model is a word network specified using extended BNF. For example, the second example sentence mentioned in section 2.1 is represented by the following grammar fragment.

\(<\text{cityname}>\) se \(<\text{cityname}>\) ke liye
\(<\text{class}>\) ka kirayaa kitnaa hogaa

Here, words inside angular brackets are class exemplars. Sphinx uses CMU-Cambridge Statistical Language Modeling Toolkit to generate back-off trigram language model from a file containing valid sentences of the task domain. The entire database was used to estimate the trigram grammar. Usage of both train and test sentences for training the language model is compatible with the word network grammar of HTK which represents the test data as well as training data.

3. EVALUATION AND RESULTS

For every sentence fed to the Sphinx decoder, a best sentence hypothesis was always generated. This was not so with HTK. On several occasions the HTK decoder could not find a valid sequence of word models matching the test feature sequence. In order to make a fair comparison, the test sentences fed to Sphinx decoder was pruned. Only those test sentences which have a valid transcription from the HTK system were tested by the sphinx system. Then, for each system, the recognition accuracies in terms of words and sentences were computed using HResults toolkit of HTK. We define

\[
\text{%word} = \frac{H}{N} \times 100\%
\]

where \(H\) and \(N\) are the number of words correctly recognised and the total number of words in the test subset respectively. Similarly, we define \(\text{%sent}\) as 100 times the ratio of the number of complete sentences recognised correctly to the total number of sentences in the test suite.

Experiments were carried out in a round robin fashion. The set of 999 spoken sentences was divided into 3 subsets of equal size. Two subsets

<table>
<thead>
<tr>
<th>Round Robin experiment 1</th>
<th>Feature set</th>
<th>%sent</th>
<th>%word</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
<td>Train</td>
</tr>
<tr>
<td>MFCC_DZ</td>
<td>81.7</td>
<td>68.4</td>
<td>97.5</td>
</tr>
<tr>
<td>MFCC_E_DZ</td>
<td>82.7</td>
<td>65.4</td>
<td>97.4</td>
</tr>
<tr>
<td>MFCC_D_A</td>
<td>82.9</td>
<td>64.1</td>
<td>97.6</td>
</tr>
<tr>
<td>MFCC_E_D_A</td>
<td>85.4</td>
<td>64.7</td>
<td>97.9</td>
</tr>
<tr>
<td>MFCC_E_D_A_Z</td>
<td>85.0</td>
<td>67.5</td>
<td>97.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Round Robin experiment 2</th>
<th>Feature set</th>
<th>%sent</th>
<th>%word</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
<td>Train</td>
</tr>
<tr>
<td>MFCC_DZ</td>
<td>81.1</td>
<td>73.6</td>
<td>96.9</td>
</tr>
<tr>
<td>MFCC_E_DZ</td>
<td>81.1</td>
<td>72.9</td>
<td>96.9</td>
</tr>
<tr>
<td>MFCC_D_A</td>
<td>79.6</td>
<td>70.3</td>
<td>96.9</td>
</tr>
<tr>
<td>MFCC_E_D_A</td>
<td>82.2</td>
<td>72.6</td>
<td>97.5</td>
</tr>
<tr>
<td>MFCC_E_D_A_Z</td>
<td>82.2</td>
<td>75.4</td>
<td>97.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Round Robin experiment 3</th>
<th>Feature set</th>
<th>%sent</th>
<th>%word</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
<td>Train</td>
</tr>
<tr>
<td>MFCC_DZ</td>
<td>81.7</td>
<td>74.2</td>
<td>97.5</td>
</tr>
<tr>
<td>MFCC_E_DZ</td>
<td>82.0</td>
<td>74.5</td>
<td>97.5</td>
</tr>
<tr>
<td>MFCC_D_A</td>
<td>82.3</td>
<td>73.0</td>
<td>97.6</td>
</tr>
<tr>
<td>MFCC_E_D_A</td>
<td>84.1</td>
<td>76.3</td>
<td>98.0</td>
</tr>
<tr>
<td>MFCC_E_D_A_Z</td>
<td>83.9</td>
<td>75.5</td>
<td>97.5</td>
</tr>
</tbody>
</table>

Table 1: The sentence and digit recognition accuracies of the HTK system for various feature sets. The performance figures corresponding to 3 round robin experiments with different test data are shown separately.

were used for training the system and the remaining subset was used for testing. In the first round, the first two subsets were used for training and the third subset was designated as the test set. In the second and third rounds, the second and the first subsets were designated as the test data respectively.

The performances of the systems based on HTK and Sphinx were measured for various feature sets. The sentence and digit recognition accuracies for various feature sets for 3 round robin experiments are shown in Table 1. Here, MFCC denotes a feature vector comprising of 12 mel frequency cepstral
coefficients. The suffix “_D” denotes addition of 12 delta-MFCCs which measure the slope of trajectories of individual MFCCs. The suffixes “_E” and “_A” denote the addition of log frame energy and 12 acceleration coefficients respectively. The suffix “_Z” indicates that the static features in an utterance are normalised so that the mean of each feature across the utterance is zero.

An examination of Table 1 shows that the sentence accuracy of test data in the first round robin experiment is poorer than that in other two rounds. Thus, it is necessary to conduct round robin experiments so that the performance figures are not biased by the unbalanced partition of small database into test set and train set.

Figure 2 shows the sentence accuracies, averaged over 3 round robin experiments for various feature sets in the form of a bar graph. The bars occur in pairs. The thin and thick bars show the accuracies of the HTK system for train and test data respectively. For the test data, the 42 dimensional feature vector MFCC_E_D_A_Z comprising of frame energy, 12 cepstral coefficients and their delta and acceleration coefficients appear to give the best sentence accuracy.

Table 2 shows the sentence recognition accuracies of the HTK and Sphinx systems for train and test data in 3 round robin experiments. Each experiment used distinct 1/3 of the database as test data. Here, the 42 dimensional feature vector MFCC_E_D_A_Z was used. The sentence accuracy of test data in the first round robin experiment is poorer than that in other two rounds in case of both systems. The last row of the table shows the sentence accuracies averaged over 3 round robin experiments. The average sentence recognition accuracy of the Sphinx system for the train data is much better than that of HTK. However, the accuracies for the test data are not very different considering the variation in performance figures over the 3 sets of test data.

Although accuracies of the the two systems are comparable, it should be noted that HTK uses network grammar while Sphinx uses trigram grammar. Hence, the perplexity of the language model in the HTK system is lower than that of Sphinx. Yet, the sentence accuracies of the two systems are comparable. In continuous speech recognition systems, major performance improvements come from constraints imposed by linguistic knowledge. A measure of linguistic constraints is the perplexity of the language model used. Despite the fact that the perplexity of the language model in the HTK system is smaller than that of the Sphinx system, the sentence accuracies are comparable. This indicates that the quality of acoustic models generated by Sphinx is much better than that of HTK. While it is possible to construct a network grammar for the current task, it would be very difficult to do so for other tasks in general. A N-gram type statistical grammar appears to be the most successful language model for large vocabulary systems.

<table>
<thead>
<tr>
<th>Round robin</th>
<th>train data</th>
<th>test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>expt Id</td>
<td>HTK</td>
<td>Sphinx</td>
</tr>
<tr>
<td>1</td>
<td>85.2</td>
<td>67.5</td>
</tr>
<tr>
<td>2</td>
<td>82.1</td>
<td>75.2</td>
</tr>
<tr>
<td>3</td>
<td>83.8</td>
<td>75.5</td>
</tr>
<tr>
<td>average</td>
<td>83.7</td>
<td>72.7</td>
</tr>
</tbody>
</table>

Table 2: The sentence recognition accuracies of the HTK and Sphinx systems for train and test data in 3 round robin experiments. The performance figures averaged over 3 round robin experiments are shown in the last row.
Our observation that the acoustic models of Sphinx are superior to that of HTK is supported by another experiment. We conducted a number recognition experiment in which 7 digit long spoken numbers are recognised by systems implemented using HTK as well as Sphinx. In a number recognition system, any digit can occur at any time. Thus, there is no linguistic constraint such as syntax. So, one has to use an ergodic language model. The digit and number recognition accuracies of HTK system on unseen test data was 86% and 42% respectively [5]. The respective figures for the Sphinx system were 97% and 75% respectively\(^1\). While the difference in recognition accuracy can partly be attributed to better trained acoustic models of the Sphinx system, the performance figures of the two systems are not strictly comparable due to some differences in experimental conditions.

There were some differences in the database and configuration of the two digit recognition systems based on HTK and Sphinx. The speech database for training and testing the system, the perplexity of the language model and the HMM states. The database used by the HTK based number recognition system, implemented last year, contained numerals such as “fifteen” and “hundred” in addition to the 10 digits. Thus the perplexity of the language model of the HTK number recognition system was 11.3 [5]. Some speakers participated in more than one recording session. The spoken numbers in the first recording session was every speaker formed the train data set. The speech data of additional recording sessions from the test data set. The HTK system used 804 sentences from 22 speakers for training and 190 sentences for testing. Due to continued data collection effort, the database size increased over time. For training and testing the Sphinx system, we included only those utterances which contained only the 10 digits-zero to nine. Thus the perplexity of the language model of Sphinx is 10. The sizes of train and test data set used in Sphinx experiments are 641 and 402 respectively. In addition, the number of emitting states were different in two systems. The number of states is fixed as 5 in the current version of Sphinx. On the other hand, the performance of the HTK system was found optimal with 7 emitting states. In spite of these differences, the increase in sentence accuracy from 42% to 75% is remarkable. Most of this difference has come about due to difference in acoustic models of HTK and Sphinx systems.

4. DISCUSSION

The quality of acoustic models generated by Sphinx is better than that of HTK. Major differences in the implementation of two systems are the use of deleted interpolation technique for robust estimation of parameters of context dependent models and the use of semi-continuous models in Sphinx. HTK permits training of semi-continuous models. If we use semi-continuous models, the performance gap may reduce. A few other differences between HTK and Sphinx software tools are worth noting.

Features: Let us suppose that there are 12 static features, say 12 MFCCs. If a user wants to augment the feature set by dynamic features, an user of HTK is forced to append all the 12 delta-MFCCs. In contrast, Sphinx permits an user to append only a subset of dynamic features, say the first 5. However, in HTK, one can compute and use a variety of features such as linear prediction coefficients and cepstral coefficients derived from them.

Model: A user of HTK has the flexibility of specifying the number of states for each unit. For number recognition system, 7-state model for each digit performed the best in number recognition experiments [5]. In the current version of Sphinx, one has to use 5-state models. HTK can train and test models with discrete and continuous densities, in addition to semi-continuous models.

Training: Sphinx employs deleted interpolation for robust estimation of parameters. This assumes significance when train data size is small.

Grammar: Language model in the form of bi-

\(^1\)A digit recognition error of 3% means that about 1 out of 30 digits will be misrecognised. This will translate to misrecognition of about 1 digit in 4 numbers since a number comprises of 7 digits. Thus, a digit recognition error of 3% is equivalent to number recognition error of 25%.
gram grammar is supported by both toolkits. Sphinx supports trigram grammar as well. On the other hand, HTK supports network grammar.

While anyone can download HTK and use it for any purpose within his organisation, redistribution of HTK in either source or binary format is not permitted. Although models trained by HTK can be freely distributed, the recipient of such trained models has to download the HTK decoder before using it even for personal use. This puts a constraint on the use of HTK as an ASR tool for mass usage. On the other hand, Sphinx does not have any such restriction. Also, the CMU speech group has consistently performed well in ASR evaluations conducted by DARPA. The stated goal of Sphinx project is to make such advanced software freely available for public use. Thus, one can expect future versions of Sphinx to be even more powerful. Even HTK is under development by the Cambridge University Engineering Department (the original authors of the software) as well as by several volunteer contributors. Version 3.1 of HTK, for example, has facilities for vocal tract length normalisation and perceptual linear prediction analysis.

HTK is much better than Sphinx as far as user-friendliness is concerned. Thanks to its commercial avatar before the recent release in the public domain, the documentation, the flexibility of setting/changing configuration parameters, the ease of invocation of tools and the quality of source code is very good. Therefore, we believe that HTK will be preferred, for sometime at least, by researchers who want to quickly explore a new idea using a toolkit. Of course, any such experiment has to be performed within the constraints imposed by the architecture of the software toolkit. In this respect, the software package under development at ISIP is worth watching. The documentation and the quality of source code is good. A functional package incorporating training as well as decoding tools was released recently. We hope that this system would be as clean and user-friendly as HTK, and as powerful as Sphinx.

5. CONCLUSIONS

The availability of public domain automatic speech recognition software with source code has enabled researchers to quickly test new ideas, and also to modify the source code to suit the needs of the experiment. The license for unrestricted use and redistribution of software facilitates the development of powerful speech recognition systems for wider distribution. While the quality of acoustic models trained by Sphinx was found to be better than that by HTK, the user-friendliness of Sphinx needs definite improvement. One can hope that automatic speech recognition software such as Sphinx and the ISIP system would meet such expectations in the near future.

REFERENCES