Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 6, June 2014, pg.964 – 970

RESEARCH ARTICLE

Language Identification and Locale Based Web Browser: New Approach

Ms. Shobhna Singh[1], Mr. Mahesh Singh[2], Er. Pritam Kumar[3]

[1]M.Tech Research Scholar, Department of Computer Science Engineering, ACTM, PALWAL, HARYANA, INDIA
[2]Assistant Professor, Department of Computer Science Engineering, ACTM, PALWAL, HARYANA, INDIA
[3]Lecturer, Department of Computer Science, IMT, FARIDABAD, INDIA
[1] shobhnatewatia39@yahoo.com, [2] mahesh100nucs@gmail.com, [3] parashar.pritam@gmail.com

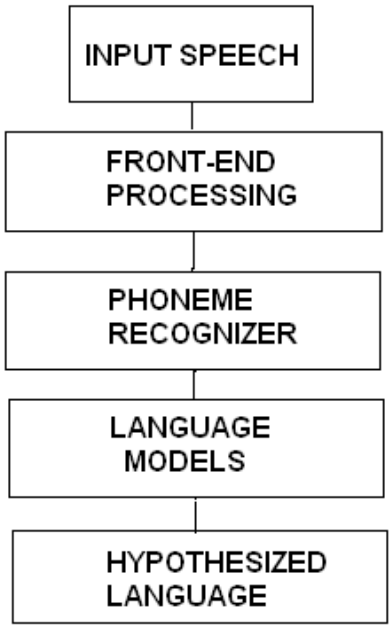
ABSTRACT: "Voice recognition has come a long way in the last few years," as said plain and simple by Judge Thomas C. Smith, author of Dictating to your Computer (for judges and lawyers) (1). What was once a sceptical and highly inaccurate, almost useless device, voice recognition is now an incredibly helpful and advanced tool.
Voice recognition is defined by www.techtarget.com as simply, "the ability of a machine or program to receive and interpret dictation, or to understand and carry out spoken commands." In voice recognition, analog audio is converted into digital signals. A vocabulary of words and/or syllables and a quick way to compare the vocabulary with signals is necessary for a computer to change the human voice into a digital computer signal. The speech patterns of the user that has "trained" the voice recognition software are saved to the hard drive and loaded into memory when the program is running. Language Identification is process of identifying the language being spoken from a sample of speech by an unknown speaker. Most of the previous work in this field is based on the fact that phoneme sequences have different occurrence probabilities in different languages, and all the systems designed till now have tried to exploit this fact.

Keywords— Hidden Markov Model, Language Model, Phoneme Recognizer, Vector Quantization

I. INTRODUCTION

The problem of Language Identification (language ID) is defined as recognizing the language being spoken from a sample of speech by an unknown speaker [3]. The human is by far the best language ID system in operation today, with accuracy as high as hundred percent in case if they know the language and can make a pretty reasonable guess about them in case if they don't. This project has tried to develop this ability in machines. Speech is basically consists of small units of sound called as phonemes. For example if you speak word BAT, then \b, \@, \t are three phonemes which together forms this sound. Now for language identification using phonetic characteristics, first a speech is converted into phoneme sequences, which can be done using various methods. In this project the phoneme recognizer used, is based on Hidden Markov Models (HMM). Once speech is converted into phoneme sequences then a probabilistic analysis is done which in turn is divided on three phases; training, tuning & testing, and for that the corpus is also divided accordingly. In second phase of project I have actually attempted to supersede the conventional HMM way of converting speech into phoneme sequences by using more abstract classes derived using statistical tools like Gaussian Mixture Method (GMM) and HMM and then passing it through the same language models, which were designed in first phase.

II. RELATED WORK



Language IDs works as a single entity in many applications, but it is, in itself a set of three black boxes; *front-end processing system, phoneme recognizer, and language models*. Speech Data is given as an input to these set of boxes and then it flows into the system as shown in the figure. Implementation of every system is hidden from others; only interfaces are standardized as we do in case of OSI Layers of networking. By standardization, we mean the format of data, which will be passed from one system to another, is fixed.

2.1 Front-End Processing is used for the purpose of vector extraction. Many different algorithms exist for speech recognition and language identification. A common need between them is some form of parameterized representation (feature vectors) of the speech input. These feature vector streams may then be used to train or interrogate the language models which will follow the feature extraction module in a typical language identification system [6]. It is obvious that there exist an infinite number of ways to encode the speech, depending upon which particular numerical measures are deemed useful. Over the many years of speech recognition research, there has been a convergence towards a few (spectrally based) features that perform well. Of these, Linear Prediction (LP) and Cepstral measures are most widely used [5].

2.2 Phoneme Recognizer

The basic aim behind this system is to generate the phoneme sequences from the vector sequences. There are 56 phonemes, and their different combinations can represent all possible speeches in various languages. We used Hidden Markov Models (HMMs) for this purpose. HMM models are primarily probabilistic state machines, in which each state represents a phoneme. Now there are two kinds of probabilities attached with each state. First, is the probability with which we can say which will be the next state (B_{xx}, as shown in fig.) and second, is the probability with which we can say what will be the output sound when this state is reached, which are represented by $A_x(O_y)$ in the figure. Basically HMMs are used for three problems, out of which one which we will be using it for is to find out the most probable state sequence given a sequence of output sound [1]. So basically what it does is that when processed speech vectors are passed through this system it gives sequence of phonemes.

2.2 Language Models

These are the most important aspect of a language ID. There basic aim is to predict the language given a phoneme sequence, and for this purpose some kind of probabilistic analysis is done which is implementation dependent. This is precisely my area of work. There are various ways of implementing this. One of the properties that most methods have in common is that they are made up of two phases: training and recognition. The latter may only be performed after the former, which involves presenting the system with speech from target languages (i.e. those that we are trying to recognize). During recognition, these features are compared to those of utterances being tested, in order to decide which language is the correct one. The simplest form of training uses only a sampled speech wave, and the true identity of the language being spoken. More complex approaches use phonetic transcriptions (a sequence of symbols representing the sounds in each utterance), or orthographic transcriptions (the text of the words spoken), along with a pronunciation dictionary, which would map each word to its

representation. Such methods are obviously more costly and time-consuming, not least because fluent speakers for each target language are required.

III. PROPOSED METHOD

These instructions give you guidelines for Till now most of research done in field of language ID was focused on the first two stages of it like that of M Zissman [2] or L Schwardt [4]. Little work is done in the field of language models which makes the last phase. I have first studied all the existing methods for this purpose and then designed some new algorithms for the same purpose. Input given to these models is the phoneme sequences obtained from the recognizer and the output expected from them is language in which the input speech is. Usually language models are designed in two phases: Training and recognition phase. In the design which I have proposed I have introduced one more phase of Tuning, in which system parameters are optimized. So now phases are:

- ***Training Phase***
- ***Tuning Phase***
- ***Testing Phase***

In the first phase of language models large amount of training data is passed through the model along with the language information and no recognition takes place in this phase. Now based on training data various probabilities are calculated like probability of occurrence of given phoneme in given language. Now once all these probabilities are calculated then next phase of recognition starts. In this phase also speech data is passed through models but now language information is not given to the system, in fact it's the system which predicts the language of the speech data by using the probabilities calculated in training phase. For example suppose speech data for English language was passed then in training phase what model will calculate is the probability like P(ph/E) which what is the probability of phoneme 'ph' occurring in English speech data. Now during recognition phase suppose some phoneme sequence ph1 ph2 ph3 occurred in the test data then the probability that this sequence is in English will be

P (ph1 ph2 ph3 | Eng) = P (ph1 | Eng) * P (ph2 | Eng) * P (ph3 | Eng)

3.1 Training Phase

Training phase is the most important phase among all the three, and decides how well or bad the whole system is going to perform. Main aim of this phase is to extract maximum possible information about a language from its training data. There are number of ways to do it. One of them is by finding the probability with which a given phoneme from a set occurs in that language. There are two issues to deal with in this probabilistic approach which are as follows:

i. **Method of finding the probabilities**: There are number of ways in which probabilities related to a phoneme can be found. One of them is *P (ph|X)*, which represent the probability of phoneme ph occurring in language X. This value is found by counting the number of times a given phoneme occurs in training data and then dividing it by the total number phonemes in the whole data and is calculated for every phoneme.

ii. **Type of probabilities**: Now there are various ways of capturing the language specific information from the training data. While selecting the appropriate method there are some parameters you should consider. First of them is the kind of application, language models are designed for. And second is the computational resource available. Like in our case where we are trying to design a language ID, we know that it's the order in which phonemes occur, makes one language different from other and then there are some phonemes which are specific to some languages and never occur in others.

So considering these factors a training model was designed in which when data is passed three types of probabilities are calculated.

1. **Unigram**: These are the probability of occurrence of single phoneme in language. These basically try to capture the distinct phonemes which are special to particular language like phonemes ending with \h\ are more probable in Hindi than in English.

Uni_Prob (ph|X) =

No. of time phoneme 'ph' occur in the training data

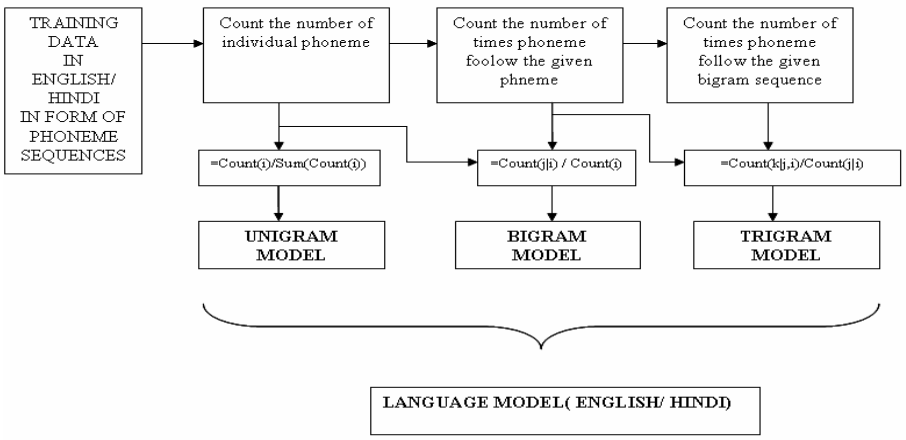
Count of total no of phonemes in the training data for 'X'

2. **Bigram & Trigram**: This is the probability of a phoneme being followed by a given phoneme or pair of phonemes in a given language. This basically tries to capture the sequential information related to phonemes which is also specific to a language. This method is called as n-Gram approach, and we can go till any value of n but as you increase value of n complexity increases exponentially. Therefore we have calculated till n=3.

$$Bi\_Prob\ (ph2 \mid X, ph1) = \frac{\text{No. of time phoneme ph1 is followed by phoneme ph2 in the data}}{\text{No of times phonemes ph1 occurs in the training data of language X}}$$

$$Tri\_Prob\ (ph3 \mid X, ph1, ph2) = \frac{\text{No. of time phoneme ph1 is followed by ph2 and then ph3 in the data}}{\text{No. of time phoneme ph1 is followed by ph2 in the training data of language X}}$$

So in a simple language in case of trigrams what you do is that instead of considering phoneme as single unit you consider sequence of three phonemes as a single unit to calculate the probability. Following diagram gives a pictorial view of the whole process:



So this was all about training phase of language models. The outputs of this phase are three probability files for each language as described above. Now before discussing about other phases lets discuss briefly how these are going to use these probability files. Let's suppose "a b c d e f "is a phoneme sequence, where a, b, c, d, e and f are different phonemes, came for recognition during testing phase, then different probabilities related to it will be:

$$Unigram\_Prob = P(a|X)*P(b|X)*P(c|X)*P(d|X)*P(e|X)*P(f|X)\ldots\ldots\ldots\ldots\ldots (1)$$
$$Bigram\_Prob = P(b|X, a)*P(c|X, b)*P(d|X, c)*P(e|X, d)*P(f|X, e)\ldots\ldots\ldots\ldots (2)$$
$$Trigram\_Prob = P(c|X, a, b)*P(d|X, b, c)*P(e|X, c, d)*P(f|X, e, f)\ldots\ldots\ldots\ldots (3)$$

So this way we calculate three probabilities related to each phoneme sequence. So the first problem which we should deal is the problem of zero-probability penalty, which deals with cases when a given phoneme actually never occurred in training data of a language. Then second problem which we should deal now is how to interpret these probabilities and combine them into single value. For this we need to find weights for different probabilities. Let's call alpha, beta and gamma as the required weights for unigram, bigram and trigram probability, then the final probability of some phoneme sequence *P* belonging to a specific language *X* will be:

$$Prob\ (P|X) =$$

$$alpha*Unigram\_Prob\ (P|X) + beta*Bigram\_Prob\ (P|X) + gamma*Trigram\_Prob(P|X)$$

So now our first task is to find out values of alpha, beta and gamma. Then the third problem which we should deal is that to find optimal value of duration for which phoneme sequence should be recorded. See in this problem the ideal solution will be to take phoneme sequence of the whole available data. But we should consider two graphs, first is the graph between the complexity and duration which is an ever increasing graph. Second is the graph between accuracy with which language is predicted for a given phoneme sequence with its duration. This graph is expected to be asymptotic in nature. All these issues are dealt in following section.

3.2 Tuning Phase

In training phase all system parameters were calculated. But in a system there are some hyper-parameters for which, neither is there any empirical way of calculating their values nor can they be calculated on the same data set on which system has been trained. For this reason an intermediate step of tuning the system has been proposed by us. In this step, optimal values of all the system's hyper-parameters are found out, and all the experiments which are done in this phase are performed on

tuning data which is totally disjoint from the one which is used for training and testing. Let's look at three problems discussed in last section one by one:

Penalty for zero probability phonemes

Consider a case when a phoneme sequence P (a b c d) occurs in the testing data. Then its unigram probability according to (1) will be:

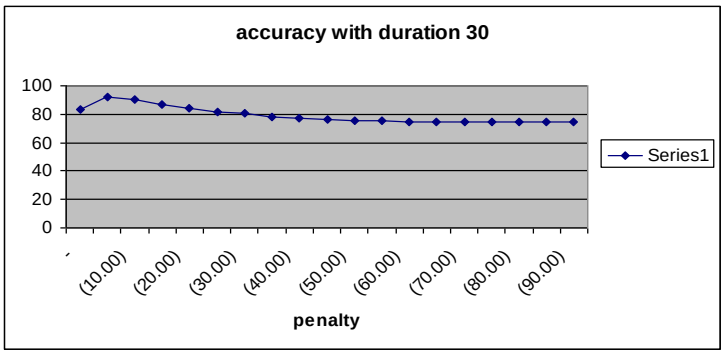
Unigram_Prob(P|X) = $P(a|X)* P(b|X) *P(c|X)*P(d|X)$

Now it might be possible that one of the phoneme in the test sequence actually never appeared in the whole training data of a given language *X*, and thus the probability of that phoneme is zero for *X*. Then what will happen? Now ideally we should not do anything in this regard and whenever there is such phoneme in a sequence, we should make the probability of whole sequence as zero, but the catch is that our phoneme recognizer is also not perfectly accurate so we should take a case a wrong conversion of speech to phoneme sequence. So what we should actually do is to assign a very small but non-zero value to all phonemes having zero probability. This approach has two advantages. First that now despite having a zero-probability phoneme, a given phoneme sequence is still eligible for recognition, and second because of a small value of probability it will try to bias the results in negative side, which ideally should have been the case.

Given below is graph for different values of penalty on x-axis and accuracy with which system works in y-axis. Penalty values are represented in terms of

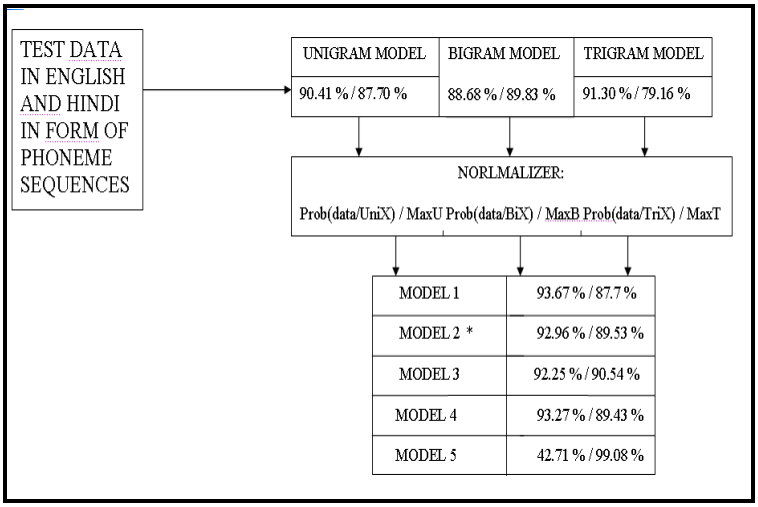
x = exponent (p)

Where *p* is actual penalty in terms of probability and values in x axis are its logarithmic counterpart. Now you can clearly see a peak in the graph at x= -10.



3.3 Testing Phase

This phase is the last phase of this process and in this all the models are evaluated against a fresh data set. Following figure shows the comparative study of different models which were designed in the tuning phase along with their performances.



The test data is first passed from three models and three probabilities are calculated separately for each n-Gram and phoneme sequence. Then these values are normalized by dividing them by the maximum value of probability which was observed during

tuning phase. Then these values are combined using five different models. Now from the above figure we can make out its *Model 3* which is performing best, but because of the overhead included in it makes *Model 2*'s performance as best. Therefore we conclude the model 2's way of combining different n-Gram probabilities as optimal

IV. PROPOSED ARCHITECTURE

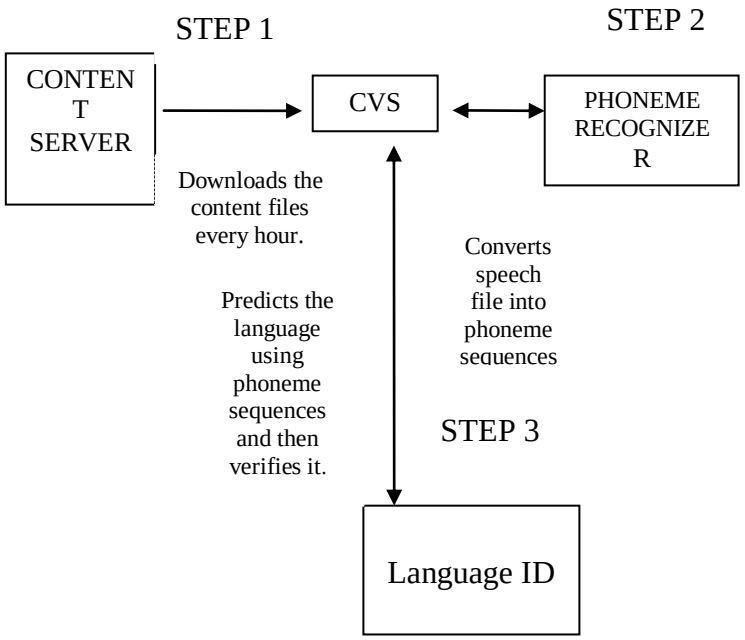


Fig. Flow Diagram of CVS

Speech applications are getting more and more popular with each passing day. All these applications involve lots of speech data stored on some server and used whenever a customer requests for it. For example suppose there is an application that plays news and a customer request for it in Hindi, so this application will play the file stored in server corresponding to that and everything will go smooth. But the problem arise is that how to make sure that the file which is stored on server is actually in Hindi only and not in any other language, because at the end it will be some human who will be uploading these files and human have this tendency to do something called as mistake. So we need some mechanized way of testing the content stored at server. With this aim CVS (Content Verification System) came into existence. Though this CVS has many aspects and things to verify, but language being the most important of them. So we have designed a tool for this purpose. Figure given below gives a graphical view of system. Currently it is in its testing phase.

V. CONCLUSION

The purpose of developing language ID is to make the process of language recognition mechanized and hence enable many speech based applications to use it as a black box. And the absence of absolutely correct way of recognizing languages by machine makes this field of language ID, a challenging research area.

Although the recent progress in language ID has been promising, currently existing systems are still not very reliable in distinguishing between a set of 10 or 11 languages. All reported systems still perform much better when exposed to about 50s of speech, compared to 10s. In this report a self designed novel method for language models was also discussed which improved the recognition accuracy by significant percentage. On comparing the systems that uses 'broad phonetic category' rather than actual phonetic as base units for modeling, it has been found that the latter perform significantly better.

Looking at the current growth of mobile usage, researchers will have to come up with a reliable solution for language Id very soon. This project was a step towards this direction. Though it did not solve the complete problem, but was able to come out with an improved algorithm for one of its important aspect along with one tool, 'Content Verification System' for speech data verification. Reliance Infocomm will be using this tool for verification data stored for speech applications.

.

REFERENCES

[1] Steve Young, "The HTK Book", Cambridge University Technical Services Ltd, December 1995.

[2] M.A. Zissman, "Language Identification using Phoneme Recognition and Phonotactic Language Modeling", in Proceedings ICASSP '95, 1995.

[3] Y.K. Muthusamy, E. Barnard, and R.A. Cole, "Reviewing Automatic Language Identification", in IEEE Signal Processing Magazine, October 1994.

[4] L Schwardt, "Automatic language identification using mixed order HMMs and Untranscribed corpora", ICSLP 2000.

[5] J.R. Deller, J.G. Proakis, J.H.L Hansen, "Discrete-Time Processing of Speech Signals", MacMillan, New York, 1993.

[6] W.A. Ainsworth, "Speech Recognition by Machine", Peter Peregrinus, London, 1988.

[7] S.B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", in IEEE Transactions, Vol. ASSP-28, No. 4, August 1980.

[8] M.A. Zissman, "Automatic Language Identification using Gaussian Mixture and Hidden Markov Models, in Proceedings ICASSP '93, vol 2, pp.399-402, April 1993.

[9] M.A. Zissman, "Comparison of Four Approaches to Automatic Language Identification of Telephone Speech", in IEEE Transactions on Speech and Audio Processing, Vol. 4, No. 1, January 1996.

[10] Y.K. Muthusamy, "Segmental Approach to Automatic Language Identification", Ph.D. thesis, Oregon Graduate Institute of Science & Technology, 1993.

[11] M A Kohler, "Approaches to Language Identification using Gaussian Mixture Models and Shifted Delta Cepstral Features", ICSLP 2002.

[12] George Constantinides and Uras Rapajic, "Language Identification in multilingual systems", Surprise_96, vol 4, gac1.Link: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/gac1/report.html

[13] H. Hermansky, N.Morgan, A. Bayya, and P. Kohn, "RASTA-PLP Speech Analysis Technique", in Proc. ICASSP '92, Vol. 1, March 1992, pp. 121-124.

[14] J Ajmera and C Wooters, "A Robust Speaker clustering algorithm", Tech. Rep. 38, IDIAP, 2003. Link: www.icsi.berkeley.edu/ftp/global/ pub/speech/papers/asru03-ajmewoot.pdf

[15] T.J. Hazen and V.W. Zue, "Automatic Language Identification using a Segment-Based Approach", in Proceedings 3rd European Conference on Acoustics, Speech, and Signal Processing '89, Glasgow, Scotland, May '89.

