ON STRING MINING SPEECH RECOGNITION

Levon Aslanyan, Minoosh Heidari

Abstract: *Automatic Speech Recognition (ASR) is a well developed technical area, conjoining technologies such as DSP, Machine Learning, HS Codesign, and Linguistics. For scientific research ASR is an invaluable source of ideas and solutions integrated in one important product. Unfortunately, being a business interest, advanced ASR technologies are hidden by companies. The open literature is focused round the Hidden Markov Models (HMM) which was developed earlier in 1960's and then incorporated into the HMM as an attractive technique. Open ASR systems with HMM report surprisingly low level recognition – 75%. After the beginning of HMM era many things changed in HS area – now very large memories are available, even the portable devices use multiprocessors with high computational power. It is to analyze what is achieved and what reminds to be understood with ASR in new situation and probably this helps in increasing the accuracy of recognition. Our target after this analysis will be the setting up of advanced ASR system (for Armenian language). In fact the system will be learnable, when large linguistic corpora are available, so that it can be multilingual. The feature ASR technology that is visible today is related to the Sequence Data Mining (SDM) technique. Being part of the structural data mining this technique is already studied and developed. Studies involve the topics of LCS, gene alignment and similar, but speech signal analysis provide a specific situation that is to be studied deeply, with consequent optimization and implementation. All the related thoughts are openly presented in the text below. Future work will enlarge the given ongoing prototype with means of advanced SDM and with interfaces.*

Keywords: *hidden Markov model, string mining, and speech recognition.*

ACM Classification Keywords: *I.2.7 Natural Language Processing.*

Introduction

Two technologies are our objective below – the Hidden Markov Models (HMM) and the String Data Mining (SDM). The former is a well known tool for many applications, including speech recognition that is our application target. SDM is specific case of Data Mining mostly used in search systems. Our discussion wants to understand the real advanced technical resource of speech recognition today. The experimental platform is presented by open tool set at [Becchetti & Ricotti, 1999].

Paper is composed in three parts. At first the HMM is considered and analyzed in order to determine the means of the best correlatedness between the state and observation sequences. Analysis shows that although probabilities are given to all pairs of states and observations, a unique observation sequence generates one dominating sequence of states moreover this sequence of states is built by constructing of individual – isolated states. We suppose that this might be the main bottleneck of the known low recognition rate in ASR with HMM. The third part of the paper considers an alternative to HMM technique for use in ASR. This is the technique, well known as the string data mining, and well developed for search purposes. Here an ASR string of observations is considered as intervals in whole. The learning set helps ASR to correspond to such intervals their classes by the ordinary means of the supervised pattern recognition. And our supposition is that by the given reason this technique fits better to the ASR needs. Finally, the section two of the work describes the open, accessible software environment of ASR, where we find all the input information tackled by HMM and SDM algorithms. [Becchetti & Ricotti, 1999] provides an open code software ASR together with the detail description of all the stages of its work. The valuable parts of the work are learning databases TIMIT (Texas Instruments and MIT) developed by a NIST project, with support of DARPA-ISTO, and ATIS (Air Travel Information System) developed by ARPA-SLS project. TIMIT provides phoneme annotated acoustic recordings to train the ASR, and ATIS is commonly used for the evaluation of word error performances. We refer to the chapter 3 “Speech signal analysis” of [Becchetti & Ricotti, 1999] that presents the DSP part of the work with signal windowing and algorithmic analysis, that outputs the multidimensional numerical vector sequences/strings. These strings correspond to the observation sequences considered by HMM and SDM.

Analysis of Hidden Markov Models in ASR

Start with defining the basic elements of the HMM.

Hidden Markov model (HMM) is a statistical model, extended form of Markov Chain model in which the system being modeled is assumed to be a Markov process, probably with unknown parameters, and the challenge is to determine the hidden states from the observable symbols.

In regular Markov model, the stats are directly visible to the observer, and therefore the state transition probabilities are the only parameters of the model. In a HMM, unlike a regular Markov model, the states are not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the set of possible output symbols. Therefore the sequence of symbols generated by an HMM gives some information, in an intermediary way, about the sequence of the states.

Formal Definition of Hidden Markov Models. The HMM is characterized by the following elements:

1) N, the number of states of the model. Although the states are hidden, for many practical applications there is often some physical significance attached to the states or to sets of states of the model. Generally the states are interconnected in such a way that any state can be reached from any other state (e.g., an ergodic model); however, we will see later that other possible interconnections of states are often of interest. We denote the individual states set as $S = \{S_1, S_2, \dots, S_N\}$, and the state at time t as q_t.

2) M, the number of distinct observation symbols per state, i.e., the size of a finite alphabet. The observation symbols correspond to the physical output of the system being modeled. We denote the individual symbols set as $V = \{v_1, v_2, \dots, v_M\}$.

3) The state transition probability distribution

$$A = (a_{ij}) \tag{1}$$

where

$$a_{ij} = P\left[q_{t+1} = S_j \middle| q_t = S_i\right], 1 \le i, j \le N.$$

Here we see that the probabilities do not vary by the time. For the special case where any state can reach any other state in a single step, we have $a_{ij} > 0$ for all i, j. For other types of HMMs, we would have $a_{ij} = 0$ for one or more (i, j) pairs.

4) The observation symbol probability distribution in state $i, B = \{b_i(k)\}$, where

$$b_i(k) = P[v_k \; at \; t | q_t = S_i], 1 \le i \le N, 1 \le k \le M. \tag{2}$$

5) The initial state distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P[q_1 = S_i], 1 \le i \le N. \tag{3}$$

Concerning (1) to (3) some relations exist among the model parameters: $\sum_{j=1}^{N} a_{ij} = 1$, $\sum_{k=1}^{M} b_j(k) = 1$, and $\sum_{i=1}^{N} \pi_i = 1$. The points 3 and 4 can be demonstrated as follows (Figure 1):

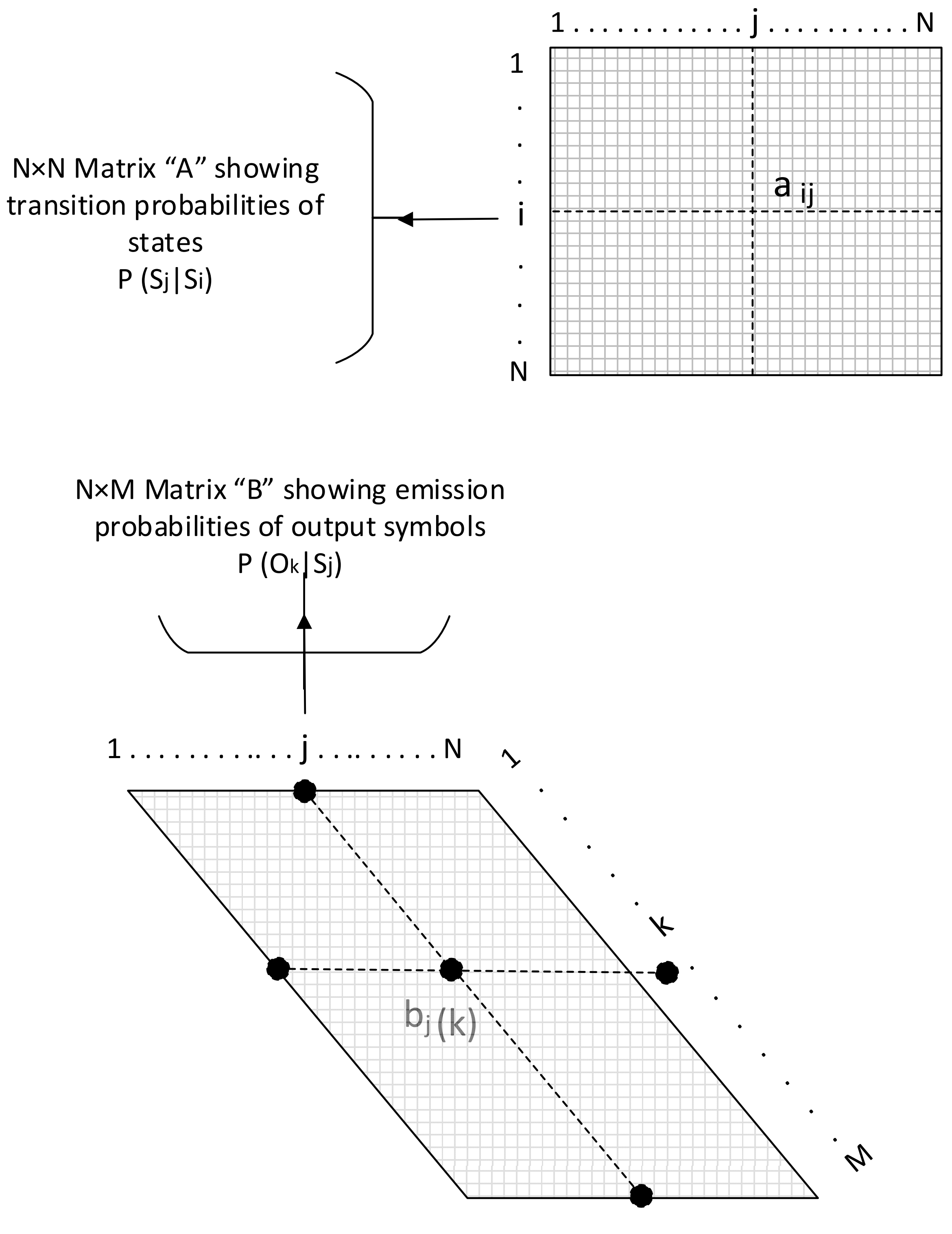

1 . . . . j . . . . N
1 . . . i . . . N
a ij
N×N Matrix "A" showing transition probabilities of states
P (Sj|Si)
N×M Matrix "B" showing emission probabilities of output symbols
P (Ok|Sj)
1 . . . . j . .. . N
1 . . . . . k . . . . M
bj (k)

Figure 1. State transition probability matrix and Symbol observation probability matrix

So a complete specification of an HMM requires specification of two model parameters (N and M), specification of the state and observation symbols, and the specification of the three probability measures A, B, and π. For convenience, we use the compact notation

$$\lambda = (A, B, \pi) \tag{4}$$

to indicate the complete parameter set of the model.

An introductory example. Figure 2 shows a Hidden Markov model that represents urn-and-ball system frequently used to illustrate HMM. We use, after a correction, the example of point 4.3.3 of [Becchetti & Ricotti, 1999]. We use the simplified and the complete versions.

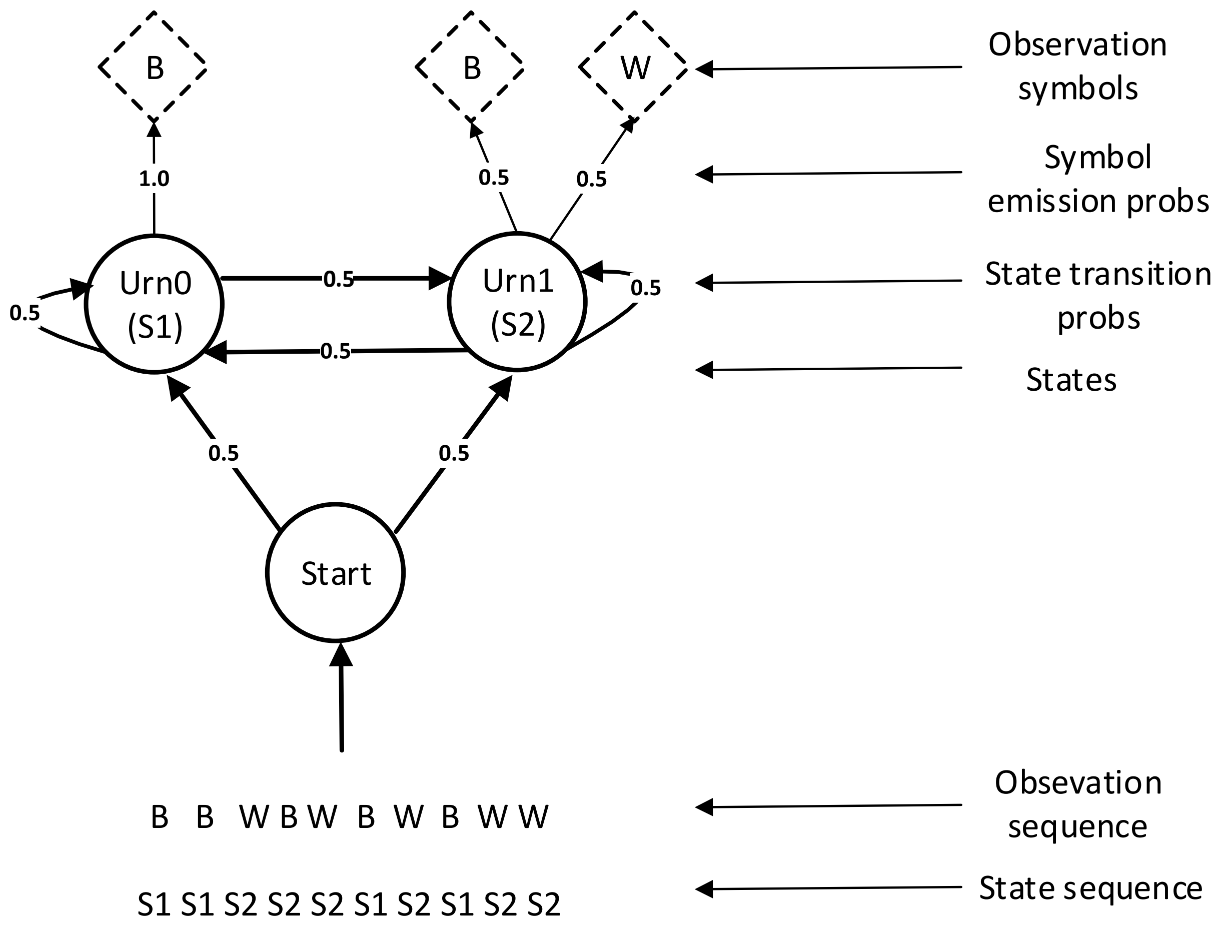


Figure 2. An HMM that models two urns containing Black and White balls.

Let us assume that there are 2 ($N = 2$ states) urns with black and white balls inside, i.e. 2 distinct colors ($M = 2$ observations as output symbols). Within each urn there is a large quantity of different balls. From urn, a ball is chosen at random, and its color is recorded as an observation. The ball is then replaced in the urn form which it was selected. A new urn is then selected according to the random selection procedure associated with the current urn. Ball selection process is repeated. This entire process generates a finite observation sequence of colors, which we would like to model as the observable output of an HMM.

Consider in particular the example when the first urn 0 is filled only with black balls (symbol 1), and the second urn 1 is filled with an equal number of the two color balls. Each extraction from urn 0 is followed by an extraction from urn 0 or 1 with equal probabilities. And the extraction from urn 1 determines a successive extraction from urn 0 and 1, with the same equal probability ½. The initial urn probabilities are supposed to be equal.

A physical process for obtaining observations is as shown on Figure 2.

Revisiting Hidden Markov Model for Figure 2, the formal definition is (S, V, N, M, A, B, π), where

1. $S = \{Urn0(s1), Urn1(s2)\}$
2. $V = \{white/(0), black/(1)\}$

1. $A = (a_{ij})$ is given as

	S1	S2
Start	0.5	0.5
S1	0.5	0.5
S2	0.5	0.5

2. $B = \{b_j(k)\}$ is given as

	B	W
S1	1.0	0.0
S2	0.5	0.5

3. $\pi_i = 0.5$

What is the probability of occurring of the sequence B W B?

To compute this probability, all the possible paths of the states to produce the output sequence (B W B) should be taken into account. The four possible paths are:

$s1 \rightarrow s2 \rightarrow s1$		$s1 \rightarrow s2 \rightarrow s2$		$s2 \rightarrow s2 \rightarrow s1$		$s2 \rightarrow s2 \rightarrow s2$	
$\downarrow$		$\downarrow$		$\downarrow$		$\downarrow$	
.5×.5×.5×.5	+	.5×.5×.5×.5×.5	+	.5×.5×.5×.5×.5	+	.5×.5×.5×.5×.5×.5	= .140925

With this simple example, we tried to understand the capability of HMM modeling which is larger in reality. The theory differs the output symbol generation – being it related to the states or to the state transitions correspondingly. The former (that we considered above) is called state-output HMM while the output generation in state transitions is known as the edge-output HMM. In this form, the output symbol is produced by the edges. So, it is called the edge-output HMM which is defined as a quadruple $\Theta = (S, Y, \{T^k\}, \pi)$, where:

S is the set of N states, Y is the set of M output symbols.

$\{\mathrm{T}^{\mathrm{k}}\} = \{\mathrm{T}^{\mathrm{k}} | \mathrm{k} = 1, \ldots, \mathrm{M}\}$ is a set of N×N (and implicitly N×M) matrices, with the elements $\mathrm{t_{ij}(k)}$ of the joint probability distribution of states and output symbols. $\{\mathrm{T}^{\mathrm{k}}\}$ must satisfy:

- For all $\mathrm{i, j}$ such that $1 \leq \mathrm{i, j} \leq \mathrm{N}$, $a_{ij} = \sum_{\mathrm{k}} \mathrm{t_{ij}(k)}$;
- For all i such that $1 \leq \mathrm{j} \leq \mathrm{N}$, $b_i(k) = \sum_{\mathrm{j}} \mathrm{t_{ij}(k)}$;
- For all i such that $1 \leq \mathrm{i} \leq \mathrm{N}$, $\sum_{\mathrm{j,k}} \mathrm{t_{ij}(k)} = 1$;
- For all $\mathrm{i, j}$ such that $1 \leq i, j \leq N$ and $1 \leq k \leq M$, $t_{ij}(k) \geq 0$.

And π is the initial state probability distribution.

The joint matrices of this form of HMM can be demonstrated as shown on(Figure 3):

Consider again the example above. Replace the part of description “And the extraction from urn 1 determine a successive extraction from urn 0 and 1, with the same equal probability ½.” by the new one “The extraction of a white ball (symbol 0) or a black ball (symbol 1) from urn 1 determine a successive extraction from urn 0 and 1 correspondingly, with the equal probability ½”. Model is the same. Now also change the equal probabilities. It is easy to see that the state output model is unable to model this case which is to consider as an edge-output model.

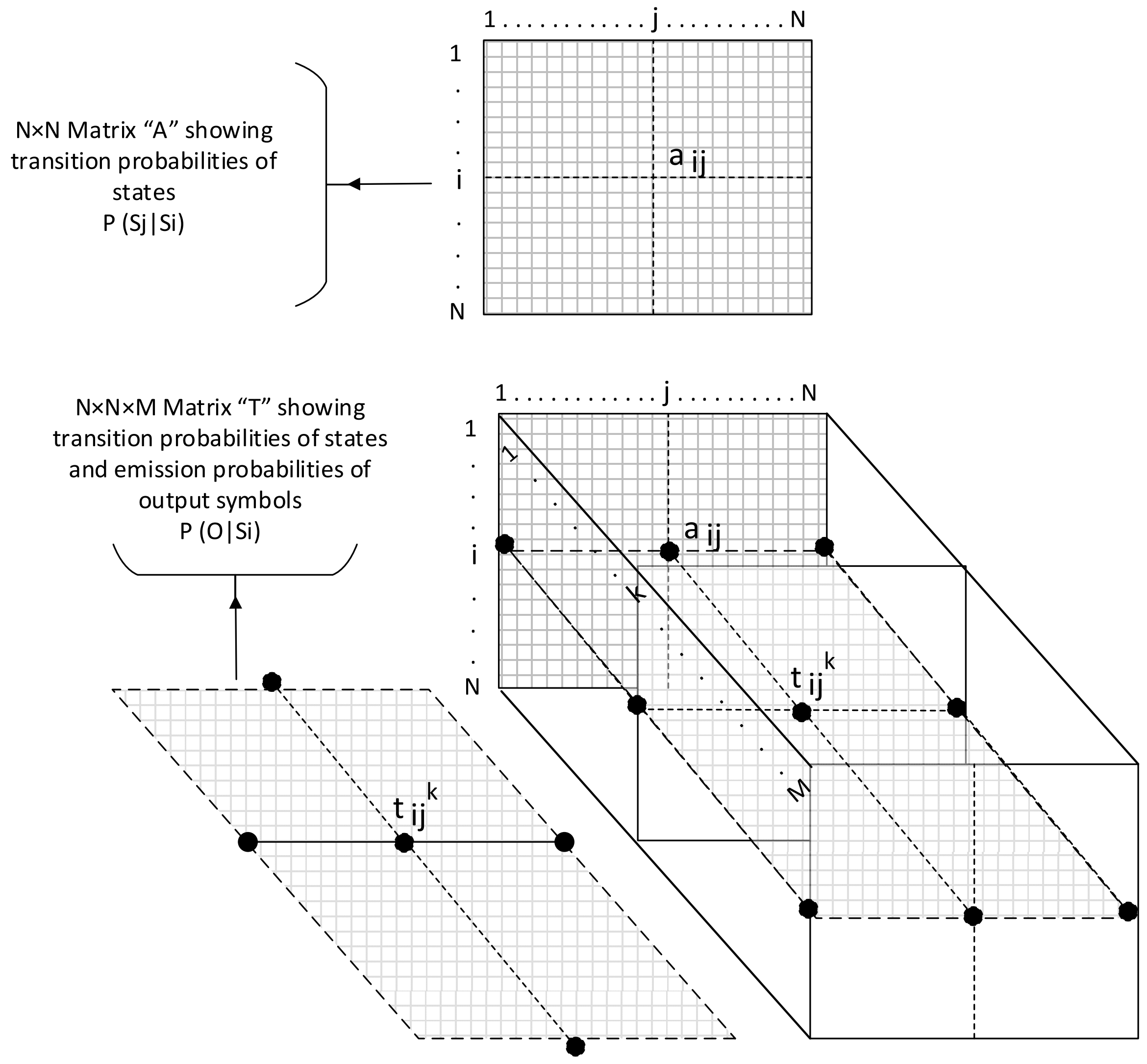

N×N Matrix "A" showing transition probabilities of states P (Sj|Si)
1 . . . . j . . . . N
1
i
N
a ij
N×N×M Matrix "T" showing transition probabilities of states and emission probabilities of output symbols P (O|Si)
1 . . . . j . . . . N
1
i
N
a ij
1
k
M
t ij k
t ij k

Figure 3. Joint Matrices Demonstration

The Three Basic Problems that Serve the HMM Applications. Given the HMM formalism of the previous section, mention the three basic problems of interest that must be solved for the model to be useful in real-world applications. These problems are the following:

P1. Observation Sequence Probability Computation

Given the observation sequence $O = O_1, O_2, \ldots, O_T$ and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

It is well known that the direct calculation of $P(O|\lambda)$ by its definition requires in its order $2T \cdot N^T$ operations. But fortunately, by the developed Forward-Backward Procedure this computation requires only $T \cdot N^2$ operations.

We will use in our analysis some internal objects and definitions of this Procedure, so we have to bring them again.

That is the so called forward variable

$$\alpha_t(i) = P(O_1, O_2, \dots, O_t, q_t = S_i|\lambda), \tag{5}$$

And the similar backward variable, that is

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots, O_T, q_t = S_i|\lambda). \tag{6}$$

P2. Optimal State Sequence Associated to the Acquired Observation Sequence

Given the observation sequence $O = O_1, O_2, \dots, O_T$, and the model $\lambda = (A, B, \pi)$, how do we choose a corresponding state sequence $Q = q_1, q_2, \dots, q_T$ which is optimal in some meaningful sense (i.e., best “explains” the observations)?

P3. Model Parameter Estimation that Maximize the Acquired Observation Sequence Probability

How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

Among the basic HMM problems P2. is the most meaningful. Problems P1 and P3 are computational, algorithmic, optimization tasks. Indeed these tasks are very much important to set up the final effective HMM application environment but the P2 is the place where the states and observations meet each other.

Analysis of the P2.

Here, as mentions [Becchetti & Ricotti, 1999], the difficulty lies with the definition of the optimality of a state sequence, i.e., there are several possible optimality criteria. For example, one possible optimality criteria is to choose the states, which are individually and independently most likely.

To implement this solution define the variable

$$\gamma_t(i) = P(q_t = S_i|O, \lambda) \tag{7}$$

that is the probability of being in state S_i at time step t, given the observation sequence O, and the model λ. Equation (7) can be expressed simply in terms of the forward-backward variables, i.e.

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{8}$$

since $\alpha_t(i)$ counts for the partial observation sequence $O_1, O_2, \dots, O_t$ and state S_i, at time step t, while $\beta_t(i)$ accounts for the remainder of the observation sequence $O_{t+1}, O_{t+2}, \dots, O_T$, given state S_i at t. The normalization factor $\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$ makes $\gamma_t(i)$ a probability measure so that

$$\sum_{i=1}^{N} \gamma_t(i) = 1 \tag{9}$$

Using $\gamma_t(i)$, we can solve for the individually most likely state q_t, at time t, as

$$q_t = arg \max_{1\le i\le N} [\gamma_t(i)],\ 1 \le t \le T. \tag{10}$$

Analysis:

a) Consider the Figure 4. explaining $\gamma_t(i)$ around the time step t.

The formula counterpart to $\alpha_t(i)\beta_t(i)$ in this picture is

$$([\textstyle\sum_{k=1}^{N} \alpha_{t-1}(k) \cdot a_{ki}] b_i(O_t)) \cdot \beta_t(i). \tag{11}$$

Having S_i fixed at time step t, and the sequence $O_{t+1}, O_{t+2}, \dots, O_T$ defined, maximization of $\beta_t(i)$ becomes an independent task from the left part of the formula (11). $b_i(O_t)$ is also fixed value, by the conditions given at time step t. So the part of formula to be maximized is the part included in square brackets. Rewrite $\alpha_{t-1}(k)$ in the form $\alpha'_{t-1}(k) \cdot b_k(O_{t-1})$ (which is valid by the definition of $\alpha_{t-1}(k)$) and then consider the transformation of the base formula into the:

$$([\textstyle\sum_{k=1}^{N} \alpha'_{t-1}(k) \cdot b_k(O_{t-1})\ a_{ki}] b_i(O_t)) \cdot \beta_t(i). \tag{12}$$

Formulas of $\alpha'_{t-1}(k)$ in (12) depend on parameters $O_1, O_2, \ldots, O_{t-1}$ and the requirement that $q_{t-1} = S_k$. Our idea is to understand, and outline, the tendency here in recursive maximization of terms of type $b_k(O_{t-1})a_{ki}$. These is exactly the product of two probabilities – state probability and observation probability out of some current state S_k. The real formula requires finding a set of similar optimized pairs. In more detail, in one fragment, by O_t it is to take such state q_k for step $t-1$ that provides greater probability $b_k(O_{t-1})$ for O_{t-1} and then it is to choose the next to the q_k state with the highest probability a_{ki}. The formal description requires that the product of these two probabilities is maximal. If states and observations are correlated this can have a meaning. While so, then this is a hypothesis, functional dependency, and it is mandatory to formulate this at the beginning. Without it - states and observations - having no connection to each other - can enter into the same game.

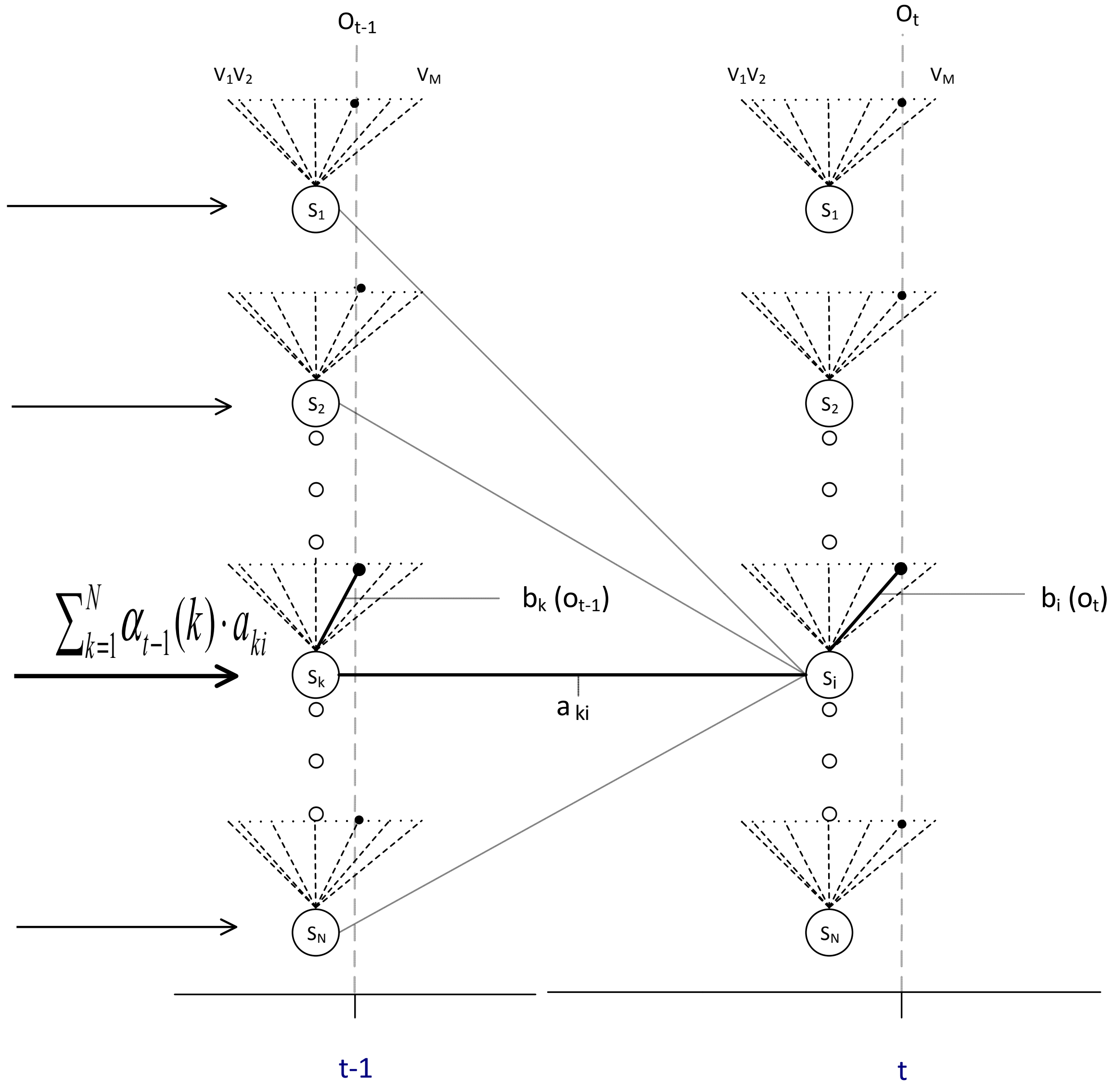


Figure 4. An Illustration for Forward and Backward Algorithm

b) Another caution to the considered algorithm is raised in [Rabiner, 1989]. Independent from one to the other construction of the optimal states at time steps $t, 1 \leq t \leq T$ can draw to a situation that the resulting state sequence can be not even valid. The simplest explanation of this is in use of the zero state transition probabilities.

In this section we analyzed the issues of applicability of HMM in ASR, in part of determination of best correlatedness between the state and observation sequences. Questions rise as it is mentioned in a), Figure 4 and in b). The formal technique for finding the single best state sequence is based on dynamic programming methods, and is called the Viterbi algorithm. The Viterbi algorithm combats the notion b) but not the a). The basic construction is again the pair $(b_k(O_{t-1}), a_{ki})$ and it is a question if consideration of 2 nearby states can be effectively manages the long state sequences. As an alternative technique, in last section of the work we will describe the use of String Data Mining technique in ASR.

Automatic Speech Recognition

This section brings the ASR internal information necessary to understand the use of HMM and SDM. Speech recognition is a type of the supervised recognition scenario. The final target is to create the alphanumeric information counterpart to the uttered speech but the way to this consists of different local steps such as setting the learning data and training, time and spectral domain analysis of signals, model composition that links the signal parameters to the linguistic elements like phonemes and words, and then can be some orthography and grammatical checks and empowerments. ASR uses both deterministic and statistical technique in model compositions. In statistical speech recognition it is wide spread to consider the Hidden Markov Models. Markov Models are really statistical but whole ASR framework in this case is quiet deterministic. And it is correct to mention another approach, statistical, when large speech corpuses are created, analyzed and characterized, and when the input signal and its parts are compared statistically with the fragments of these learning data. Anyway, our interest is to study a real example of ASR system with the use of this traditional technology, to analyze it, trying to understand the alternatives and empowerments. The base of our study in this part is the well made book [Becchetti & Ricotti, 1999]. We use it as a prototype environment to understand the whole pros and cons in area, determining novel research targets and developing technologies and solutions to them. Probably one of the starting points is that the maximal correct recognition percentage mentioned here is 75%. Usually it is 60-65%. Why the result of such hard work is so low? Is this because of incorrect or misuse of the selected models and is there a room for empowerment?

What we learn from [Rabiner, 1989]?

First, in “RES” project (Recognition Experimental System), that implemented in the book [Becchetti & Ricotti, 1999], there is information about the hardly made training data available for future experimentation such as TIMIT, and ATIS.

They are the most important speech databases used to build acoustic models of American English. TIMIT is an acronym composed by TI (Texas Instrument) and MIT, the two research centers involved in the project at NIST, National Institute of Standards and Technology, sponsored by DARPA-ISTO.

TIMIT contains a total of 6300 sentences that 630 American speakers have spoken 10 sentences. The database has been made up of the recorded waveform of the sentence, with a sampling frequency of 16 kHz, together with a time-aligned phonetic transcription of the sentence. The speakers in TIMIT have been subdivided into training and test sets using some criteria such as: Almost 20 to 30% of the database is considered for the test set, and the remaining 70 to 80% for training.

Every phrase is associated with:

.txt	the orthographic transcription of the phrase (spelling)
.wav	the wave file of the sound
.phn	the correspondence between the phonemes and the samples (the number of all Phonemes (included many times in different contexts in the mentioned speech corpora) is around 50 to 60).
.wrd	the correspondence between the words and the samples

Furthermore, there is the pair of letters (SX, SI and SA) before the name of files:

SX	are phonetically compact phrases in order to obtain a good coverage of every pair of phones
SI	phonetically varied phrases, for different allophonic contexts
SA	for dialectal pronunciation

The other mentioned database, ATIS, stands for Air Travel Information System, distributed from 1989 by ARPA-SLS project. It contains 10 722 utterances by 36 speakers.

Every phrase is associated with the following file types:

.cat	category of the phrase
.nli	phrase text with point describing what the speaker had in mind
.ptx	text in prompting form (question, exclamation,…)
.snr	SNOR (Standard Normal Orthographic Representation) transcription of the phrase (abbreviations and numbers explicitly expanded)
.sql	additional information
.sro	detailed description of the major acoustic events
.lsn	SNOR lexical transcription derived from the .sro
.log	scenario of the session
.wav	the waveform of the phrase in NIST_1A format (sampling rate, LSB or MSB byte order, min max amplitude, type of microphone, etc…)
.win	references for the interpretation

Furthermore, there are several labels (S, C, X and R):

.cat	category of the phrase
's'	close-speaking (Sennheiser mic)
'c'	table microphone (Crown-mic)
'x'	lack of direct microphone, 's' spontaneous speech
'r'	read phrases

In general, the project (RES) has the following general specifications:

It works on recorded speech files and it basically includes:

- The batch modules for acoustic model initialization and training;
- Grammar models training;
- Phoneme/word recognition;
- Performance evaluation.

It performs in speaker independent phonetic recognition:

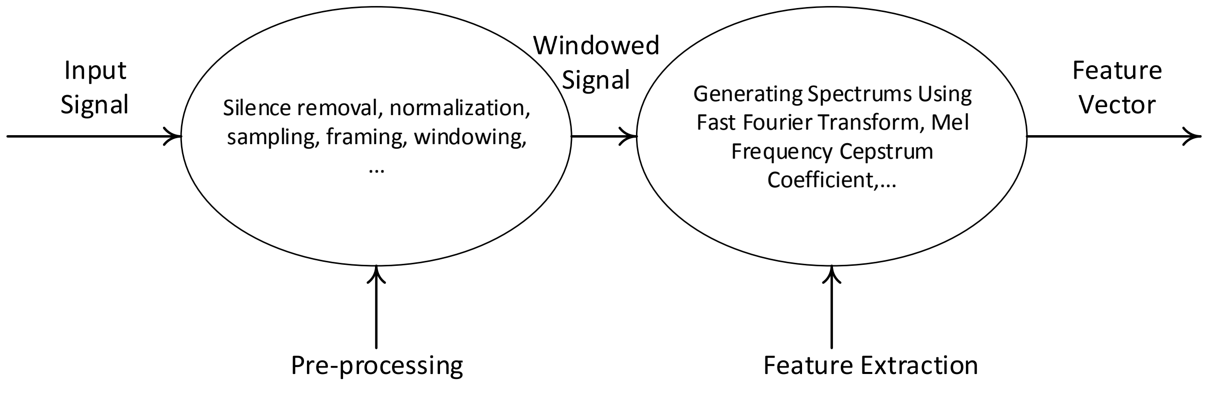
- With 69.2% of percent correct using all TIMIT test data using context independent phonetic models.

It yields 87.83% of percent correct in speaker independent word recognition on ATIS using context independent phonetic models not optimally tuned on this database.

Second are the input signal and its analysis. The base where speech signals and their fragments are compared is selected 16kHz. Then the recommended window length and the overlap is 512 and 384. The number of time spectral characteristics produced equals 39 including 13 base characteristics with their first and second order differences.

At this point it is seen that the base element for recognition is a fixed 39 length numerical vector. The input signal is coded/presented as the “overlapping” sequence of the numerical vectors. And surely the similar interpretation is possible to apply on training set signals, those already provided with the notations and transcriptions.

All the above information is acquired during the standard signal processing and feature extraction procedures:


Input Signal
Silence removal, normalization, sampling, framing, windowing, ...
Windowed Signal
Generating Spectrums Using Fast Fourier Transform, Mel Frequency Cepstrum Coefficient,...
Feature Vector
Pre-processing
Feature Extraction

Speech is analyzed over short analysis window.

For each short analysis window, a spectrum is obtained using FFT (Fast Fourier Transform).

Spectrum is passed through Mel-Filters to obtain Mel-Spectrum.

Cepstral analysis is performed on Mel-Spectrum to obtain Mel-Frequency Cepstral Coefficients (MFCC).

Thus speech is represented as a sequence of Cepstral vectors.

It is these Cepstral vectors which are given to pattern classifiers for speech recognition purpose.

The following block diagram shows algorithmic internals of the time spectral domain analysis of input signals by [Becchetti & Ricotti, 1999]:

Speech Recognbition Software System Initialisation Project Block-Diagram

```
DbaseVoc dbase;
dbase.Configure(config_file, TRUE); gets dbase configuration options \to soundlab.cpp\
\includes reading phoneme positions \in phn\ at labelcl.cpp
NTimit39LabelClass::Open_Sym(const String & file_name);\

    conf.Open_File(config_file); \to resconf.cpp Where it opens\
                  file_ini.open(file_name,ios::in|ios::_Nocreate); \"res.ini" & "res.opt"\
    conf.Get_String_Opt("DBaseOptions", "ListOfSoundFNames", db_file); \resconf.cpp\
    Initialize(config_file,"DBaseOptions",read_transcription); \soundlab.cpp\ size 2048
    Inizialize_File_List(); \to soundlab.cpp "file_list" then vets_cardinality\
                           soundfil.cpp reading 'headers" of sound files, 44

ini_options.Set_Options(config_file, dbase.Get_Num_Of_Symbols()); \iniopt.cpp symbol models\

symbol_models_initialization.Configure(ini_options, config_file, dbase.Get_Num_Of_Symbols());
                \to initiali.cpp\ … feature.Configure(config_fname); \to feature.cpp\ …

symbol_models_initialization.Symb_Model_Calculation(ini_options.InitializedModelsFName,
ini_options.models_file_input, dbase, ini_options.full_covariance, ini_options.load_one_mixture,
ini_options.unif_sect, ini_options.model_type); \to initiali.cpp\ … where:

                 Write_Header_Of_File_Model (out_fname, dbase.Snd_Type(), dbase.Label_Type(),
   dbase.Db_File_List_Name(), dbase.Window_Lenght(), dbase.Window_Overlap(), stat_dim, full_cov);
                          tspecbas.cpp opens phonemes_1.spt, outputes the header and closes the file
     states_info.Initialize(num_sections_per_symbol[i], num_mix_per_symbol[i],stat_dim, full_cov);
                                            \initiali.cpp initialize (*this)[0-2][0] with zeros\
     Calculate_One_Mixture_Codebook(act_phon, num_frames, dbase, unif_sect); \at initiali.cpp\ in
                                                                                           what:

  not_end_of_dbase=dbase.Get_Filtered_Sequential_VetSmp_And_Its_Predecessors(vetsmp,act_phon,
  is_new_phone, pred_list); \at initiali.cpp Ln163 which goes to soundlab.cpp where:\

    while(not_end_of_dbase AND NOT
    SoundLabelledFile::Get_Filtered_Sequential_VetSmp_And_Its_Predecessors(vet,sym,
    is_new_fone, pred_list)) \to in the same soundlab.cpp, where:\
          act_smp=snd_file->Get_Actual_Position(); \to soundfil.cpp\ position=44 then
                            SetSymbol(sym, act_smp, new_smp); to labelcl.cpp, new_smp_pos
                                                        Set_Absolute_Position(new_smp);
                                             Backshift=numpred*(len_win_sample-overlap);
             Set_bsolute_Positin(new_smp OR new_smp-backshift); \by soundfil.cpp and
                                                                 operation seekg();\
          Get_Sequential_Vet(vet); \to soundlab.cpp where \Read at soundfil.cpp\ and
             \Set_Relative_Position(-overlap) at soundfil.cpp by use of seekg()\;

  feature.Get_Previous_Info(pred_list, dbase.Smp_Rate()); \perform sequential
  transformations over prev_vetsmp_list to the memory of all the modules\
  feature.Extract(vet_features, vetsmp, dbase.Smp_Rate()); \retrieve configuration from
  configuration file apply all the required transformations\
  states_info[k][cluster].Do_Averages(); \utilize mean & cov accumulators to explicitly
  calculate.. for covariance, square of population, external product mean vector,
  main diagonal\
         States_info.Compute_Whole_Codebook_Clusters_Distortions(); \eigenvectors, eigenvalues\
                                                       states_info.Compute_Cluster_Weights();
  states_info.Store_Codebook(out_fname, act_phon, load_one_mixture, model_type); \model computation
                                                                                      and output\
```

This section described in short the internals of [Becchetti & Ricotti, 1999]. The basic initial information is the speech databases TIMIT and ATIS. All speech records are digitized and analyzed with help of DSP algorithms over the proper sliding windows. Multidimensional numerical vectors and their sequences are the observations, given as input to the HMM tools.

Modeling and Learning with Substring Data Mining

This section aims at introducing the elements of the Data Mining technique [Han & Kamber, 2006] in the form applicable to the ASR design. We mention the ordinary [Li et al, 2006], structural [Srivatsan & Sastry, 2006], sequential [Nizar & Ezeife, 2010] and finally the String [Ji & Bailey, 2007] Data Mining. We consider String Data Mining as an alternative to the HMM for ASR related applications.

Association Rule Mining. Consider a finite set $A = \{a_1, \dots, a_m\}, m \geq 1$. The elements of A we will also call items, due to example applications used in this area (supermarket transactions and market basket analysis). Further, let $\mathcal{D}: 2^A \rightarrow \mathbb{N}_0$ defines a multi-set over the power-set of A in a way that for each $X \in 2^A$ the number $\mathcal{D}(X)$ indicates how many times the subset (item-set) X occurs in multiset $\mathcal{D}$. We will refer to $\mathcal{D}$ as *the database* and to its elements as *transactions*. It is realistic to suppose the $\mathcal{D}$ to be finite in each time frame, although as a database it is a dynamically evolving relational table. We will use also the n-cube notation, where $\mathcal{D}$ induces natural item-set coding and their weights/labels to the n-cube vertices. In some cases we will consider and analyze only none zero occurrences/weights.

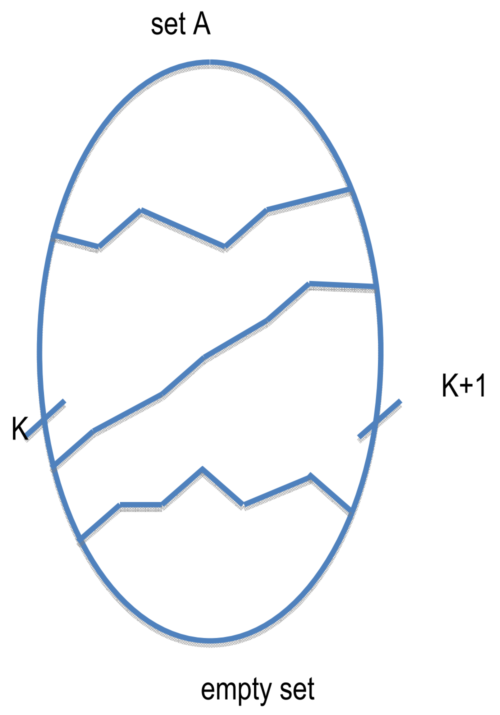


Figure 5. Hasse type diagram of power set of set A with isoareas by values of support function $s(X)$

For $X \subseteq A$ we define $s(X) \coloneqq \sum_{Y \supseteq X} \mathcal{D}(Y)$ to be the *support* of X in $\mathcal{D}$. We say that a subset $X \subseteq A$ is t-*frequent* in $\mathcal{D}$, $t \geq 0$, if $s(X) \geq t$, that is when there are at least t transactions in $\mathcal{D}$ containing X.

Monotonity is the important property of function $s(X)$,

$$(X \subseteq Y) \Rightarrow s(X) \geq s(Y).$$

Izo-areas of $s(X)$ can be presented by the sequences of embedded monotone Boolean functions. But such sequences can't behave arbitrarily. For example, if to consider a simple 2 row database then these rows correspond to 2 vertices of n-cube: v_1 and v_2. Consider subcubes $I(v_1, 1)$ and $I(v_2, 1)$ formed by these vectors and by the all 1 vector (set A). All points of these subcubes 1-support one of the rows, but the points in intersection support both rows. So it is not possible to design a database that gives homogeneous support value over the area covered by $I(v_1, 1)$ and $I(v_2, 1)$. So the mentioned "sequences of embedded monotone Boolean functions" are a specific inclusion-exclusion type object to be described and studied in deep.

Frequent subsets are the basis of the large research area of data mining. More precisely the association rule mining uses frequent subsets generating rules of some given confidence. The basic approach of association rule mining was formed inside the approach known as the APRIORI algorithm [Agrawal & Srikant, 1994], [Agrawal & Srikant, 1999]. An alternative approach is formulated in [Aslanyan & Sahakyan, 2009], [Aslanyan, 1976] using the well known technique of chain split [Hasel, 1966] and chain computations [Tonoyan, 1976], [Aslanyan & Khachatryan, 2008] of n-dimensional unite cube.

Today the real application problems that use the data mining technique deal with more complex information structures that the simple set of elements we considered so far. One typical problem is the trajectory analysis. Having the vertex set representing some real geographical points and considering movements of objects among these vertices we see that the valid trajectories are to be connected entities. Considering this in a graph we face the problem of mining connected sub-graphs [Rosen, 2010]. Another problem can consider only Sperner type subsets of a set trying to mine the frequent sets of this type. Interesting problems appear with sequences and its sub-sequences. The dominating part of research here supposes consideration og the so called embedded subsequences in manner of the known combinatorial LIS and LCS problems. Besides these mentioned structural diversity an attractive task is the algorithmic complexity issue. Bring an example. Consider the so called closed accessible set systems of the transactions database.

A subset $X \subseteq A$ is called *closed* (maximal subset) if for each $Y \subseteq A$, $Y \supset X$, it holds $s(Y) < s(X)$. For $X \subseteq A$ we define $\rho(X) \coloneqq \bigcap\{Y \subseteq A : Y \supseteq X \; and \; \mathcal{D}(Y) > 0\}$ to be the *closure* of X in $\mathcal{D}$. It can be checked that $X \subseteq \rho(X)$, $X \subseteq Y \Rightarrow \rho(X) \subseteq \rho(Y)$ and $\rho\big(\rho(X)\big) = \rho(X)$. It also can be checked that the closed subsets are the closures.

Obviously $\mathcal{D}: 2^A \rightarrow \mathbb{N}_0$ defines a monotonic decreasing integer function when X increases by set-inclusion, over the Boolean cube $(2^A, \subseteq)$, and it can be checked that the closed subsets are the upper k-points of $\mathcal{D}$, for $k \geq 0$. Also observe that the frequent subsets are all the subsets of the closed frequent subsets, and thereby the closed frequent subsets determine the frequent subsets. This technique is minimizing the mining constructions and computations.

Among the diverse problems of structural, sequential, time domain data mining there is a specific domain, known as string data mining (SDM) which is the one directly related to the speech recognition domain. There are plenty of algorithms dealing with sequential data mining. Each data structure with specific particularities of the applied problem inserts additional specifics in constructing the efficient mining algorithms. In speech recognition there appear vector sequences. Vectors are of fixed length and they have numerical coordinates (time/spectral domain coefficients). Particular coordinates and the vectors in whole are comparable – having a similarity/dissimilarity measure. One can imagine to cluster the vectors but the target is not the individual vector but their sequences, the sequence of utterances, moreover the important subsequences are of different length. We may imagine the Hasse diagram that interprets this situation. It is very simple.

The upper vertex (level 0) represents the entire sequence of length m, itself. In level 1, below the level 0, there are only 2 subsequences of length $m-1$. In k-th level there are k subsequences of length $m-k$. Graphically this can be represented as a triangular halve of a rectangular grid construction. Having the voce signal one have to create the vector subsequence counterpart, and scan all subsequences to accumulate their appearances on the grid vertices. Given a voice signal database we determine thresholds for each vector component (quantization, where rounding of coordinates by threshold values is applied) and accumulate subsequence repetitions into the grid points.

Our next postulation is about the validity of monotonity property on frequent subsequence area. This is easy to check and is very useful as an algorithmic construction of SDM. In global terms such algorithms identify all subsequences in groups by their initial vectors, then in each group the maximal frequent subsequence is sought. In real speech domain these frequent subsequences must obey additional requirements (properties). They will not involve one the other and they will not intersect in time axis. These are not absolute propositions but they are recommended and are almost mandatory.

In conclusion we bring the descriptions related to the technique we designed concerning the enhancements of the experimental speech recognizers.

String Mining. Let Σ denote a finite alphabet. A sequence S is an ordered list composed by letters from Σ. Denote the i-th item of a sequence S as $S[i]$. We will consider two types of structures:

subsequences and substrings. A sequence $S_1 = a_1 a_2 \ldots a_m$ is called a subsequence of sequence $S_2 = a_1 a_2 \ldots a_n$ if $m \le n$ and there exist integers $1 \le j_1 < j_2 < \ldots < j_m \le n$ such that $a_i = b_{j_i}$ for $1 \le i \le m$. We denote this relation as $S_1 \subseteq S_2$. In a similar way, if there exist an index $j, 1 \le j \le n - m + 1$ so that $a_i = b_{j+i-1}$ for $1 \le i \le m$, then sequence S_1 is called a substring of S_2 denoting this relation as $S_1 \sqsubseteq S_2$. We also express this relation as $S_1 = S_2[j, j + m - 1]$. In both cases there can be large number of subsequence and/or substring insertions within one general string. Subsequence and substring data mining algorithms in an initial stage discover frequent subsequence and substring insertions correspondingly. Sequence mining is known with its application in market basket type data analysis. String mining, as it is easy to understand, can be an important tool of speech signal analysis.

In this application the alphabet Σ, as it was described in previous section, consists of signal window characteristic-observation vector quantization. Let us insert, in addition, a concept of similarities of letters of Σ. We do not justify it but denote it as $\mathrm{d}(\mathrm{a}_{\mathrm{i},}\mathrm{a}_{\mathrm{j}})$, $\mathrm{a}_{\mathrm{i},}\mathrm{a}_{\mathrm{j}} \in \Sigma$. Given a threshold ε we may identify $\mathrm{a}_{\mathrm{i},}\mathrm{a}_{\mathrm{j}}$ when $\mathrm{d}(\mathrm{a}_{\mathrm{i},}\mathrm{a}_{\mathrm{j}}) \le \varepsilon$. In this way we come to the approximate subsequence/substring mining concept.

Given a sequence database SDB and a minimum support threshold α, a string Q is frequent substring pattern of SDB if it holds

$$\frac{|\mathrm{S} \in \mathrm{SDB} | \mathrm{Q} \sqsubseteq \mathrm{S}|}{|\mathrm{SDB}|} \ge \alpha.$$

In this case it is used that SDB lists strings of our interest – words, phoneme code-words, etc. The same formula is applied in a case when SDB presents a long speech signals recording or a set of them. The variety is expressed by the use of the term “episode”. Frequent episodes are like a clustering model while the substring mining tends to the supervised learning. Technically substring mining is based on suffix tree models, well developed, and importantly linear in time and space complexities. And indeed still there are particular problems to understand and manage with this type of algorithms. Consider a couple of scenarios.

Let we are given a speech corpus scanned and computed the window based observations. Frequent substring mining in this case that gives a limited number of episodes is an equivalent to a cluster analysis procedure. Cluster analysis can not be applied directly due to different length of target subsequences. And the episodes derived can be used as the initial basis in manual notification of phonemes.

As the second scenario consider the case when a phoneme database is given and when it is to be enlarged by the use of speech corpus analysis. Then how the frequent episodes and the existing base can be combined? This seems like a supervised substring mining procedure. Here one possible approach may apply the well known parametric optimization of voting estimation algorithms by Yu. Zhuravlev.

And of course the main case is the phoneme search and partitions by phonemes that is based on suffix tree constructions and their extensions. Just mention two extensions of these types [Fischer et al, 2005] inserts minfreq and maxfreq in constraint-based frequent string mining computing all strings that are frequent in one database and infrequent in another. The technique used is the suffix tree and the longest common prefix (LCP) array constructions. [Tsuboi, 2003] proposed a divide-and-conquer type algorithm that decomposes the mining task into a set of smaller tasks by using a ternary partitioning technique. It brings to memory minimization and to the computation reduces.

Next group of technique we bring is about classification, recognition. An evident strategy is the use of search by phonemes (observation vector sequences) that can be effectively implemented by the suffix tree based algorithms. The necessary extension may address the use of distances and similarities, the idea of scaling over the time domain, and the none-intersection (separation) requirement of the phonemes. In addition, [Chan et al, 2003] introduces the emerging substring concept that aims at mining data classes, substrings, which occurs more frequently in one particular class rather than in other classes. In this model, above the support threshold, a growth rate threshold is determined. And again the technique is the prefix suffix trees and their transformations.

String Recognition. We use the special case of general data mining, in our case the association rule mining technique that tries to generate if-then type rules with a property of having satisfactory support and confidence. The rule has left hand attributes with their constraints and a similar set of right hand attributes with constraints. Previously, the same construction appeared in relational databases in part of functional dependency generation. Even that period, in several applications, appeared an interest in generating rules and dependencies with only one right hand attribute [Armstrong et al, 1998]. Now, [Liu et al, 1998] use this scheme in associated rule mining with one right side attribute rules. This attribute corresponds to the class label. Parallel frequency determination for the part of observation attributes and then for the complete attribute set – the class label included, forms a proper base for class associated rule mining. This can correspond to the phoneme classification rules by the use of window based observation vectors and their analytics.

Conclusion

Having that the recognition rate is quite low in HMM based ASR systems it is to try to find out the real bottleneck of the problem. HMM analysis shows that even being well defined and an attractive technique for applications it raises question with their implementation in ASR system design. Even the Artificial Neural Network (ANN) can be a good alternative. Due to ANN is not well interpretable the SDM technique might be a better choice. For experimentation on this it is necessary to set up an open source research HS environment with necessary learning databases, with DSP and interfaces. An open source prototype is selected. The additional models and components to be inserted into these environment were sought and determined as SDM (and ANN potentially).

Bibliography

[Agrawal & Srikant, 1994] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 487–499.

[Agrawal & Srikant, 1999] R. Agrawal, R. Srikant, Mining sequential patterns. In Proc. 11th Int. Conf. on Data Engineering, 1995, Washington, DC, IEEE Comput. Soc.

[Armstrong et al, 1998] T. Armstrong, K. Marriott, P. Schachte, H. Sondergaard, Two classes of Boolean functions for dependency analysis, Selected Papers of the First International Static Analysis Symposium, Science of Computer Programming, vol. 32, issue 1, 1998, pp. 3 - 45.

[Aslanyan, 1976] L. Aslanyan. Isoperimetry problem and related extremal problems of discrete spaces, Problemy Kibernetiki, 36, 1976, pp. 85-126.

[Aslanyan & Khachatryan, 2008] L. Aslanyan and R. Khachatryan. Association rule mining inforced by the chain decomposition of an n-cube, Mathematical Problems of Computer Science, XXX, 2008, ISSN 0131-4645.

[Aslanyan & Sahakyan, 2009] L. Aslanyan, H. Sahakyan, Chain Split and Computations in Practical Rule Mining, International Book Series, Information Science and Computing, Book 8, Classification, forecasting, Data Mining, Sofia, Bulgaria, 2009, pp. 132 - 135.

[Becchetti & Ricotti, 1999] C. Becchetti, K. P. Ricotti, Speech Recognition: Theory and C++ Implementation, Wiley, the University of Michigan, 1999, 428p.

[Chan et al, 2003] Sarah Chan, Ben Kao, Chi Lap Yip, Michael Tang, Mining Emerging Substrings, Eighth International Conference on Database Systems for Advanced Applications, 26-28 March 2003, Proceedings, pp. 119-126.

[Fischer et al, 2005] J. Fischer, V. Heun, S. Kramer, Fast Frequent String Mining Using Suffix Arrays, Proceedings of the 5th IEEE International Conference on Data Mining, IEEE Computer Society, 2005.

[Han & Kamber, 2006] J. Han and M. Kamber. Data Mining: Concepts and Techniques, 2nd ed. The Morgan Kaufmann Series in Data Management Systems, ISBN 1-55860-901-6, 2006, 743p.

[Hasel, 1966] G. Hansel. Sur le nombre des functions booleennes monotones de n variables, C.R. Acad. Sci. Paris, 262, serie A (1966), 1088.

[Ji & Bailey, 2007] X. Ji, J. Bailey, An efficient technique for mining approximately frequent substring patterns, Seventh IEEE International Conference on Data Mining – Workshops, 2007, pp. 325-330.

[Li et al, 2006] H. F. Li, S. Y. Lee, and M. K. Shan, “DSM-PLW: single-pass mining of path traversal patterns over streaming web click-sequences”, Proc. of Computer Networks on Web Dynamics, pp. 1474–1487, 2006.

[Liu et al, 1998] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In KDD’98, New York, NY, Aug. 1998, pp. 80 - 86.

[Nizar & Ezeife, 2010] M. Nizar R., and C. I. Ezeife, A taxonomy of sequential pattern mining algorithms, ACM Computing Surveys (CSUR) 43.1, 2010: 3.

[Rabiner, 1989] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE, 1989, pp. 257-286

[Rosen, 2010] Kenneth H. Rosen editor, Handbook of Discrete and Combinatorial Mathematics, 2010, 1248 p.

[Srivatsan & Sastry, 2006] L. Srivatsan, and P. Shanti Sastry, A survey of temporal data mining, Sadhana 31.2, 2006, pp. 173-198.

[Tonoyan, 1976] G. P. Tonoyan. Chain decomposition of n dimensional unit cube and reconstruction of monotone Boolean functions, JVM&F, v. 19, No. 6, 1976, pp. 1532-1542.

[Tsuboi, 2003] Yuta Tsuboi, Mining Frequent Substring Patterns with Ternary Partitioning, IBM Research Report, Tokyo Research Laboratory, Sept. 2003, 8p.

Authors' Information

Levon Aslanyan *– Principal Researcher, Discrete Mathematics Department, Institute for Informatics and Automation Problems, NAS RA; e-mail: lasl@sci.am*

Major Fields of Scientific Research: Discrete optimization, Pattern recognition.

Minoosh Heidari *– PhD student, Discrete Mathematics Department, Institute for Informatics and Automation Problems, NAS RA; e-mail: meinoosh@gmail.com*

Major Fields of Scientific Research: HMM, NLP, Software engineering.