Project no. 034362

ACORNS

Acquisition of COmmunication and RecogNition Skills

Instrument:	STREP
Thematic Priority:	IST/FET

Deliverable D2.2
Methods for Enhanced Pattern Discovery in Speech Processing

Due date of deliverable: 30 November 2008
Actual submission date: 12 December 2008

Start date of project: 1 December 2006 Duration: 36 months

Project coordinator name:	Prof. Lou Boves
Project coordinator organisation name:	Radboud University,
Revision [version 1.0]	

Final *version: 12.12.2008*

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Del 2.2: Methods for Enhanced Pattern Discovery in Speech Processing

Unto K. Laine, *Okko Räsänen*, *Toomas Altosaar*, TKK
Gustav Henter, KTH
Guillaume Aimetti, University of Sheffield
Joris Driesen, Katholieke Universiteit Leuven

The main authors of the sections:

Unto K. Laine	0., 1., 1.1-1.5, 2.6.2, parts of 3.
Okko Räsänen	1.6, 2.4, 2.5, 2.6.1, 3.
Joris Driesen	2.1,
Guillaume Aimetti	2.2
Gustav Henter	2.3
Toomas Altosaar	Review and comments

Table of contents:

0. Introduction

This report consists of three main sections. We start in section 1 with general, introductory notes on pattern theory and pattern discovery including some special topics and questions of the field arisen during the ACORNS project. Different aspects of the pattern discovery are discussed first on a general, principal level, in order to help the reader to orientate to the methods applied and the common problems found during the journey.

Section 2 deals with more practical and technical issues and describes the present status of the different pattern discovery activities in ACORNS, especially those that are not reported elsewhere (e.g., the NMF-method).

Section 3 gives a summary regarding the main results obtained by the different approaches. It collects both the positive and critical issues related to the different methods, and makes some general conclusions.

1. General notes on pattern theory, pattern discovery, and other related issues arisen during the project

This section consists of some general, introductory, notes and discussion that covers some of the most fundamental aspects of pattern discovery. Some aspects are connected to general pattern and information theory, some to the theory of knowledge, and some may reach up to the problems of epistemology.

This general discussion attempts to illuminate the common background for all the different efforts made in the ACORNS project that concern studying, understanding, and solving real-world pattern discovery problems. A general discussion with different viewpoints is necessary to understand the nature of problems arising out of practical work and to better comprehend the possible differences and implications found between the various approaches.

As discussed below, pattern discovery currently does not rely on a solid methodological basis and theory. This causes the field to be very diverse with all those different methods, tools and approaches. At the current phase of the development this may be the only way to find better solutions, to create deeper understanding, and to take steps toward a universal pattern theory.

Also in the ACORNS project we have worked in parallel in the field of pattern discovery, we have applied different tools and methods *to the same data* in order to understand the nature of the problem, perhaps even the nature of the speech, and to evaluate critically the positive and negative properties found in the different approaches.

In the following subsections we start with the general questions of how to define a pattern and is there any proper pattern theory available. Then we continue with other issues like: how to cope with noisy sequential patterns, what kind of representations and models are needed, and how to cope with the problem of multiple keywords.

1.1 How to define pattern?

Many well-known textbooks related to pattern discovery, classification, and recognition, do not discuss *pattern theory* at all. Even the concept of *pattern* is not always defined (see e.g., [31]). The fundamental problem in the field is not only how to formulate a good definition for a *pattern*, but how to create a unified view or even a *theory* that is able to provide a robust framework for all pattern based information processing.

If we are not able to define the central concepts so crucial to the field and fail to derive a new, comprehensive and solid theory on which we are able to illuminate the concepts, their relations and constructions, we risk going on forever performing different kinds of more or less *ad hoc* experimentations without understanding the deeper meanings and aspects of the problem. Even more frustrating, we may end up fumbling in the dark without guidance of a solid methodology supported by a solid theory.

This is a fundamental question and maybe therefore *most of the time it is not touched upon at all*. In the following subsections we try to find answers to this and other related question with the help of the recent literature.

1.1.1 Attempts to define Pattern and general Pattern Theory

An example of a team working towards a *pattern theory* is the Brown University pattern theory group which was founded by *Ulf Grenander* (http://www.dam.brown.edu/ptg/) in 1972. They define the field as well as their goals and methodology in the following way:

> **The Brown University pattern theory group** is working with the belief that *the world is complex*, and to understand it, or a part of it, requires realistic representations of knowledge about it. We create such representations using a mathematical formalism, **pattern theory**, that is compositional in that the representations are built from *simple primitives*, combined into (often) *complicated structures* according to **rules** that can be *deterministic or random*. This is similar to the formation of molecules from atoms connected by various forms of bonds.
>
> **Pattern theory is transformational** in that groups or semigroups of transformations operate on the primitives. These *transformations express the invariances* of the worlds we are looking at.
>
> Pattern theory is *variational* in that it describes the variability of the phenomena observed in different applications in terms of *probability measures* that are used with a *Bayesian interpretation*. This leads to inferences that will be realized by computer algorithms. Our aim is to realize them through codes that can be executed on currently available hardware.
> (Text in **bold**, *italic*, and underlined was added by ukl)

The methods and goals of this team clearly reflect the ideas presented in the book: *General Pattern Theory* by Ulf Grenander [32]. In March 1995 Professor *Josef Kittler* from the University of Surrey, who is a specialist in machine intelligence, evaluated the book, e.g., with these words:

> *Grenander's General Pattern Theory is the result of some 25 years of research that puts forward a unified representational model for sensory patterns. The representation is an algebraic structure with topological, probabilistic and statistical aspects. Its basic building blocks are generators equipped with bonds allowing their spatial interconnection. The representational capacity of the structure lies in the combinatorial nature of the configuration spaces of these generators. Invariance to transformations and deformations is handled by equivalence classes.*

Kittler concluded:

> *Grenander's book is neither light reading, nor necessarily the final answer to the pattern representation issue.*

In any case, up to now this work is still one of the few attempts to create a universal theory for patterns and mathematical methods for processing them. It tries to give answers to the hard problems like: How to deal with noisy patterns and how to deal with transformed (distorted) patterns. Grenander's answer is to study the topological properties of the algebraic structures selected to form representations for the patterns.

It could be added that the statistical methods adopted by Grenander are mainly limited to Markovian approaches. However, in ACORNS we have selected more practical tools and we have built on one of the most general and classical definition of the pattern:

A pattern is an organized whole made of elements (parts) or pattern primitives.

Here we consider a pattern as something that can always be *decomposed* into a set of primitives. This also includes primitives that may be reused in order to construct new patterns. An abundance of examples exist in nature: six different types of quarks and leptons composed electrons, protons and neutrons in the early universe. Electrons, protons and neutrons form atoms, atoms combine into molecules, etc.

The process where the primitives self-organize to form a new system, a new *whole*, exhibiting new properties is called *emergent*. At the same time *new qualities* appear. Qualities, that are *not explicitly present* in the parts but can only appear (emerge) through combinations and connections between proper elements (primitives).

For this type of integration to occur the parts must have a property “to be reactive”, or “to be able to interact”. They have to be *sensitive* to detect the other potential parts in order to form intensive interactions needed in construction of relatively stable, new structures. They must “sense” and “recognize” the other elements in order to form bindings or connections with them. Passive and neutral elements without the ability to interact strongly and permanently cannot form any stable construction, *a new whole*. An ideal gas at room temperature can never form permanent patterns from its elements.

A whole with its novel qualities and properties can only be defined by its *parts* and their interactions (the way they are organized), and *a part* (element, primitive) can often be defined by the whole (construction) only.

One interesting direction of thinking related to these questions is called *hologenesis* which attempts to describe the relationships between the primitives and their constructions. These principles and views have evolved in the studies of human perception. *Van Leeuwen* who first coined this concept around 1997 summarizes his views in the following way [33]:

> In the process of setting up *a system approach to perception*, brain processes cannot be neglected. The problem is *to find a broad, general characterization of these processes*. In accordance with the system approach, the dynamics of perceptual organization in the brain could be approached from the perspective of self-organization. I proposed *hologenesis* as a uniform principle of self-organization in the perception of object structure and suggested that this principle is embodied in the chaotic activity of the brain (Italics added by ukl).

Naturally, van Leeuwen is not the only one searching for *a broad, general characterization of these processes* for perception of patterns by humans and agents. *Jeff Hawkins* has communicated similar ideas, views and hopes. Many different branches of science have generated knowledge, models and hypotheses around this *hot point*. We are still lacking an integrating theory able to illuminate at least to some extent processes like:

- Efficient modeling, storing, memorizing, recognizing and reconstructing of patterns based on their common, *essential* elements (primitives) and the *hierarchic interactions* within
- Noise robust, incremental, latent, multimodal learning
- Emergent processes created by these latent learning and self-organizing principles

On the practical level we may conclude that at the present most researchers are looking for *statistical* or *rule-based methods* to describe and analyze important connections between the parts that make up a whole, i.e., regularities in chaos.

More generally, from philosophy, nuclear physics, and branches of mathematics to pattern processing engineering and cognitive sciences, researchers are presently looking for new views, models and theories on how to deal with patterns, *the whole vs. part problem*, how to extract the *relevant information* hidden in *noise*; clean *wholes* from their *noisy components*. At the moment no one knows if there is a universal theory waiting to be discovered just behind the corner, or, are we trying to capture too general things not present in this complex hierarchic world with so many different qualities and layers (see, e.g., [34]).

1.2 Pattern as a regular part of noisy sequence

Language, in written as well as in spoken form, can be represented by sequences of discrete elements. The *discretization* of speech in such a way that it helps the recognition is naturally an essential and also a demanding problem. Our auditory, cognitive processes are able to find discrete units from the continuous dynamic flow of speech. More generally, this reflects a universal cognitive principle: learning and constructing of internal *invariant representations*. Without this skill, e.g., the evolution of written forms of languages with different orthographic coding systems is difficult to explain. However, how the cognition has solved the discretization and the learning of invariant representations is still an open issue, and how to model and mimic these skills in language acquisition systems are the largest challenges of the ACORNS project.

In many practical situations patterns can be treated as *regularities* in noisy sequences. In ACORNS sequences with local regularities have been studied by the following methods:

- Multigrams
- N-grams
- DP N-grams
- Discrete transition matrices of VQ-indices (also delta & delta^2) combined to NMF
- Computational Mechanics (CMM/CSSR)
- Discrete transition matrices (concept matrix) of segmental indices
- Discrete transition matrices (concept matrix) of VQ and SLVQ indices

Also ACORNS with these different approaches demonstrates the importance of processing noisy sequences created by a limited alphabet efficiently. The activities related to the different approaches, the benefits and current problems of the method, and future plans are closer described in more detail in Section 2.

1.3 Information hierarchy and pattern discovery

Objects in our environment as well as processes these objects are creating and participating to, do often have a spatio-temporal hierarchy, which helps us to discover not only the whole event but also the elements and their way to form an organized whole through their hierarchy. In technical pattern discovery we should not only focus on the discovery or recognition of the elements, but also try to discover and model the way they are organized to form a larger totality.

The cognitive literature gives us examples where the totality is first recognized and after that its details and elements (see, e.g., [35]). However, is this effect just the conscious part of the recognition process and could there still be a "hidden" (unconscious) process before the conscious one, which uses a strong bottom-up strategy and starting from the elements proceeds to the totality and then "displays" the outcome to the conscious level before the details? If this is accepted then the bottom-up method could still be plausible even if the top-down approach seems to fit the subjective experience better.

From the practical point of view it is difficult even to imagine a pattern discovery process able to **start** from *the total pattern* (the whole) without *first* looking at and analyzing any of its components.

On the other hand cognitive science has plenty of examples where based on the holistic view a person can fill the caps in an audio, or a visual stream. This is a relatively strong argument for a kind of top-down processing.

When the recognizing system is able to construct a hierarchic model for the object or event, the hierarchy contains information of the internal structure of the object (or event), too, thus providing a mechanism to "guess" the missing spatial or temporal data by providing predictions from the model in the memory or by creating associations between the information provided and the total pattern (its representation) in the memory.

A tree structure is an illustrative example of a hierarchy able to do this kind of task. Up to now in the ACORNS project these aspects are touched, e.g., in connection to different memory architectures and not directly in connection to the pattern discovery methods discussed in this report. However, during the last project year it may be possible to integrate the mentioned memory prediction/association mechanisms at least to some of the methods discussed in this report.

1.4 Statistical approach vs. deterministic model elements

One of the most difficult questions in the pattern discovery is how to cope with the noise present in the input data. For example, in the standard N-gram and computational mechanistic approach we try to model all the details in the data stream equally closely as if they were equally important and true. This leads to an ever-increasing complexity in the modeling, which has to be limited, e.g., by a proper pruning method. However, pruning needs good rules and theory in order to be efficient and to do *the right thing*.

Randomness in the input data leads to complex models if we try to model everything given with the same accuracy, and what is even worse, when the complexity of the model in the number of model elements increases, we need larger and larger amounts of data in order to build up *a sufficient statistics* for all these elements, their appearance and their combinations. An ever-increasing amount of data is needed and the data will bring in more and more noise, which causes that new model elements have to be learned and stored. In this way a circle of increasing problems is created.

One possible way out of this circle is to apply a proper statistical approach where the model elements are no more “exact”, rigid representations of the segments (N-grams) of the input stream, but a somewhat more “fuzzy” picture of it. This is one argument for the usage of a different kind of *statistical model*. Instead of looking after equal N-grams or causal states (in CMM approach) appearing in the input stream, we may construct a discrete statistical model able to estimate the probability of that N-gram or causal state not based on the whole pattern but based on its elements and their organization.

However, we have to remember that the limitations described above may not be *fundamental*. Both the N-gram approach and the CMM approach may be equipped with a new mechanism able to help to overcome the problem of the increasing complexity. A developmental work is presently running on both fields.

In principle, the HMM-method provides one possible solution to this question, too. However, it seems to be difficult to apply the method to model human kind of incremental language learning and acquisition. Furthermore, as discussed in the Section 3, speech does not realize the Markov property. Wider temporal dependencies must be taken into account in order to reach higher performance and robustness. For example, in Finnish the first word of a sentence may predict last phone of another word in the same sentence – sometimes even the last phones of the whole sentence!

These questions motivated the TKK team to search for a new statistical method, which could learn more efficiently and could handle the statistics in a more rich and flexible way without any limiting pre-assumptions about the nature of speech (see section 2.5).

1.5 On the problem of multiple keywords

The second year (Y2) ACORNS speech material is richer than Y1. Also sentences with multiple keywords are present. This evokes the question how to select the methods so that we can cope with this more complicated speech material (see also WP5 document: D5.2).

The first look to the problem of multiple keywords was done by studying the statistics of different N-grams found in the VQ label sequences of the UK-Y1 material (see section 2.6.2). After these preliminary studies we found a better method, which seems to solve the problem of multiple keywords and provides promising recognition results. It is a novel method to apply the state transition matrices (concept matrices) as statistical models for the keywords (see preliminary results in section 2.5).

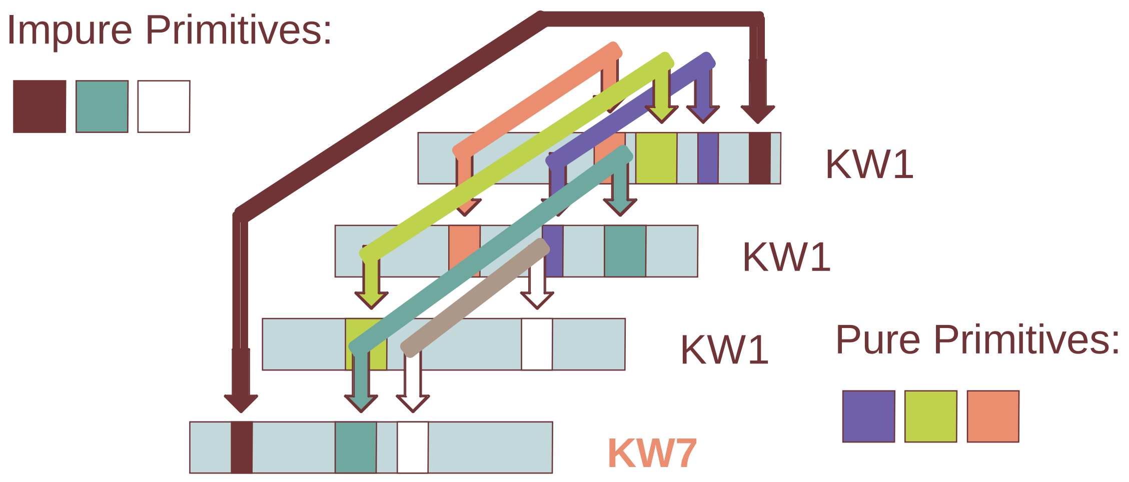

Figure 1.1: Sentences of different keywords are analyzed and some common N-grams discovered. Some of them are *pure primitives* appearing only in the sentences of the same keyword. Some other primitives may appear in sentences of different keywords (*impure primitives*).

Figure 1.1 illuminates the usage of N-gram primitives. When the same primitive is found in several sentences it helps to build statistical connections (associations) between the sentences and possibly between the keywords, too. Most of the primitives can be found in sentences of any keyword (impure primitives) whereas some other may always be found only in sentences of the same keyword (pure primitive). Typically a pure primitive can stay “pure” only in a limited speech material. Most primitives are located somewhere on the scale: pure-impure, outside the absolute pure point. High number of occurrences of the same primitive in sentences with the same keyword naturally helps to construct a useful statistical model for the keyword.

One possible method studied preliminary could approximately go in the following way:

1° Create a library of common primitives.
2° Create statistics of the occurrences of the primitives in sentences with different keywords.
3° Search after maximally pure primitives which occur often.
4° Construct statistical word models based on primitives found in 3°, where it is possible to take into account the temporal ordering of the primitives, too.

From a statistical point of view, the carrier sentences should provide a “noisy background” for all keywords. Ideally, the distribution of this background material over sentences of different keywords is flat. Thus selecting those primitives having the highest selectivity between keywords we could construct selective word models. To some extent the case of multiple keywords in the sentence does not make any difference so far the other keywords being present simultaneously with the actual one form a closely uniform distribution in the statistics collected separately for each keyword. In this case the other words, as well as the other keywords of the sentences, can be modeled just as an additional background noise. In other words, the combinations of the keywords and the words of the carrying sentences should be rich in order to help their modeling and recognition.

Under these conditions we may say that the issue of multiple keywords is quite irrelevant from the keyword discovery, spotting, and recognition point of view (see also multiple keyword learning experiments in ACORNS deliverable D5.2).

1.6 Signal representations for pattern discovery in speech

1.6.1 Representations and their relation to language

The nature of representations derived from the stimuli is a very important aspect in pattern discovery. Almost always some sort of conversion from the original raw data to a more compact form is required in order to do meaningful statistical or rule based analysis of the input. In case of speech signals, the first step in computational processing is to convert continuous acoustic waveforms into a discrete-time finite-resolution representation that tries to capture all the necessary information required for pattern discovery from the speech stream. This leads to a re-description of the original signal by a series of units defined at specific points in the signal timeline.

For the task of pattern recognition, the important question is then: what is the type of information that required in the discovery process and how it should be represented? Especially, how to describe the original signal so that the information required for performing the task is maximized and unnecessary (noisy) aspects are discarded in the created representations? The demand above is also true for hierarchical systems, where the aim is to incrementally refine the representations at all levels of processing by extracting detectable structure and possible converting representational forms based on information embedded in these structures.

To begin with possible representational options for pattern discovery, three coarse classes for signal representations can be described: 1) *Continuous representations*, e.g., MFCC-vectors, where distances between units (vectors and vector elements) can be computed with some defined metric in a continuous space. 2) *Discrete element representations*, e.g., sequences of VQ-labels, where each unit represents an outcome of some sort of categorical decision and lower level information used in the classification process is discarded. 3) *Hybrid representations*, where discrete elements are combined with additional information, e.g., spectral distances between the elementary units are defined and can be utilized on demand.

In case of speech, continuous representations always precede discrete element representations, since classification to some discrete world requires comparison of the properties of the signal at different times, which again requires a well-defined space in which the comparison can be performed. On the other hand, it is a predominant supposition that at least one symbolic level at the top of the processing hierarchy is required for meaningful modeling, since our cognitive level thinking is often considered as a symbolic process (see, e.g., [8,9]). The nature of these human-like internal symbolic representations has a conceptual link to definitions of discrete categories (e.g., categorical perception) and states (classical probabilities, automata theory) that are often utilized in cognitive modeling. As Dietrich & Markman [12], in their paper discussing cognitive representations, put it: "*A system has discrete representations if and only if it can discriminate its inputs.*". This discrimination process is something that human-like cognitive beings are surely able to do in order to make decisions. Also when we are dealing with our special cognitive skill, language, we are often dealing with discrete elements (phonemes, syllables, words, morphemes etc.) and especially symbols (meaning conveyed by the message). This assumption about symbolic nature of language is well established from several perspectives, including cognitive psychology [8] and general linguistics [10].

If we consider human thought as a symbolic process, it has important consequences for the representations of speech we are seeking for. Since "low level" acoustic signals are statistical in nature, and if at the other end the "thought" that formulates or receives that signal is a symbolic collection of meanings and memories, it is evident that a conversion has to occur at some point in the process from probabilistic descriptions to discrete categories. In the end the input is described as a collection of high-level symbolic representations that are then used to adjust the behavior of the receiving agent. This type of discrete form, or invariance, of outputs from perceptual processing can be also considered as extremely efficient for interaction between different cognitive and perceptual functions [11].

In linguistics, the invariance (conversion from variable input to a fixed set of categories) seems to appear first at the phonemic level. It seems that actual context dependent speech sounds are too variable to be used as building blocks for language, since all natural languages share a rather limited set of basic elements that differentiate meaning and that can be explicitly formulated as a phonemic alphabet. Varying sounds are mapped into these invariant categories, phonemes, which behave systematically and contain all the necessary information in order to form larger stable representations. The phoneme level is often seen as a prerequisite for word learning [2], and it is indeed difficult to think of modeling of, e.g., exemplar based learning without the systems ability to generalize from a single pronunciation dependent acoustic example to all possible appearances of such word. Learning a link between invariant level units (phonemes) and all their possible realization variants (allophones) would naturally solve this problem with ease.

The problem of invariance is also something that is supported by findings in ACORNS. Pattern discovery experiments with, e.g., multigrams (section 2.1), N-grams (section 2.6.2), DP-N-grams (section 2.2), and Computational mechanics (section 2.3), all trying to build discrete models using quantized speech signals, lead all to a common problem: the discrete element sequences related to specific keywords *are too variable* for finding strong common recurring structure between keywords spoken at different contexts and by different speakers. This makes generalization and comparison of patterns difficult, and most importantly, makes building of internal word models inefficient due to number of possible combinations of discrete units for each word. Combinatorial explosion resulting from variation in VQ and SLVQ label-sequences implies that it may be impossible to build good generalizing models using a strict sequential approach where long fixed stretches of adjacent discrete elements based on spectral properties are considered as meaningful models or prototypes for words.

Instead of going for fixed sequential models, approaches using fuzzier probabilistic-like internal representations, e.g., NMF or concept matrices (section 2.5), due to their nature, are robust to variance at phonetic level and therefore lead to good recognition results with limited vocabulary whenever there is sufficient training data available to cover the different variations of the keyword realizations. However, probabilistic representations of words in concept matrices or NMF base vectors cannot either escape the fact that they cannot generalize from exemplars to all different possible realizations, but they have to be trained with the possible realizations of each keyword in advance. This is something that contradicts strongly with human language learning capabilities, as *humans can learn several novel words per day using single exemplar tokens* [17,18].

Overall, in the light of current knowledge about human processing of patterns occurring in the external input , this all suggests that invariant levels can be and should be striven for. If humans can learn discrete invariances from probabilistic input, there is no reason why a pattern-learning algorithm could not do the same (although it might require modeling of the entire embodied human-like agent). It will be interesting to see, whether pattern discovery methods in ACORNS can be harnessed to find such strong structures through probabilistic analysis.

1.6.2 Temporal aspects of representations

In addition to the properties of single elements in the sequences of frames produced and possibly refined from the original signal, the temporal domain is also extremely important in speech. Especially important question is how can we extract meaningful information from a product of a continuous process like articulation without losing essential aspects of language en route?

In ACORNS WP2, the issue of temporal representation has been centered on two approaches: fixed frame descriptions and segmental descriptions. Both of these approaches embed temporal aspect into relations of sequential discrete units. In the former a signal is windowed using a pre-defined step size (e.g., 10 ms) and signal features are extracted from each window. In the latter approach a signal is first segmented using a blind segmentation algorithm [3] into phone-like units, and then a number of representations are created for each segment.

The advantage of the segmental representation is that it can be approximated as synchronous to phonetic aspects of the speech signal since each segment can be approximated as a phone [3]. Moreover, the segmentation algorithm creates these segmental boundaries at points where there are significant changes in the spectrum. These boundaries stand as landmarks for locations where a static description of the spectrum is not informative, whereas locations between large spectral changes are internally more coherent in speech and more provide reliable spectral information. On the other hand, information about maximal spectral change at phone boundaries can be used to provide efficient spectral change features. Segmental representation also enables compression of the data compared to fixed frame representations, if each segment is described with a handful of representations. This aspect becomes emphasized in signals where there is plenty of silence or stationary noise instead of continuous speech, as the stationary background will not create continuous data input to higher levels of processing unless there are changes in the signal content. The drawback of the current segmental approach is that it relies on the segmentation algorithm that again, relies blindly on sudden spectral changes without the ability to use real learned linguistic knowledge in its decisions.

Fixed frame representations, on the other hand, can be utilized to provide continuous small time-scale resolution flow of features. An advantage of this approach is its systematical behavior and linearity in time, as the temporal relationships between frames are always explicitly defined in the sequences of frames. A fixed frame approach provides reliable detection of fine-grained details in the signal (limited only by size and step of the window), as the extraction of representations is not dependent on any other process that tries to detect what is meaningful and what is not. A small window step enables high data-rate for higher levels and is especially suitable for data-hungry algorithms for pattern discovery. As a drawback, the fixed frame approach does not guarantee anything about the content of the frames, producing large amounts of “garbage”-frames, which can either represent meaningless events, or that are extracted from locations in the signal that cannot be efficiently described with the chosen set of features.

The word learning experiments performed with the so-called concept matrices (section 2.5) and NMF indicate that the segmental representations are too coarse-grained for high-accuracy performance. While approximately 98% of correct keyword recognitions can be obtained using segmental representations with the Y1 corpus, it still falls behind 10 ms step fixed-frames in accuracy. Insertions and deletions in the segmentation process may lead to missing of some important details and loss of synchrony in tracking of co-occurences of speech sounds stored in statistics of word models, as each word consists only of three to ten segments on average.

On the other hand, experiments of Driesen & Räsänen in an NMF-word learning task have shown that the segmental information can become useful if noise is introduced to the signal before and/or after the quantization of speech signals. Temporal integration of information over larger units helps to avoid local distortions in the vector quantized representations, and blind segmentation of speech can be utilized to perform this integration inside phone-like temporal blocks. Combining of both fine-grained fixed frame representations and temporally integrated segmental level representations in NMF word-recognition has significant effect on recognition accuracy especially at high levels of noise (paper in progress).

In current line of pattern discovery development in WP2, the segmental representations are used for quantization of the speech signals. The cluster space is first created using only segmental frames, but afterwards new signals are quantized with 10 ms fixed frames in order to obtain high temporal resolution label sequences, where each label is a typical spectral representation of a phone-like unit. This type of combined fixed-frame and segmental processing is discussed in more detail in section 2.4.

Next we will move to discuss different approaches to pattern discovery studied in the ACORNS project. As we will see, the properties of different methods impose different types of constraints to the type of representations they are efficiently able to deal with.

2. ACORNS activities in pattern discovery

This chapter introduces a group of pattern discovery methods that have been intensively studied in the ACORNS project. First we will discuss the use of *multigrams* for learning of word models (section 2.1). Then we will move to *DP-ngrams* (section 2.2) and *Computational Mechanics* (section 2.3). Section 2.4 will introduce a novel, cognitively inspired, incremental speech quantization method called *self-learning vector quantization* (SLVQ) that is used in section 2.5 in combination with yet another pattern discovery method called *concept matrices*. Finally, section 2.6 reviews briefly two other pattern discovery methods that were investigated and that gave important insight into the signal representations regarding bottom-up pattern discovery from speech (representations were discussed in section 1.6). These methods are *segmental dynamic programming* (SDP) and more traditional *ngrams*.

2.1 Multigrams

In the multigram algorithm [16], a sequence of input symbols that represents a speech signal is explained by a finite set of underlying basic units, so-called multigrams that stochastically emit symbol sequences. The stochastic behaviour of each of these multigrams is modelled by a Hidden Markov Model. The goal is then to find a set of models (multigrams) that

- Provide a good fit on the data: the symbolic input sequence needs to be explained with a sufficiently high likelihood by the converged set of models
- Are likely to be an adequate set for explaining language

The latter condition is necessary as a counterweight for the tendency to make the set as big and detailed as possible, which stems from the first condition. Unfortunately, whereas the fitness of a multigram set in terms of the first condition can readily be calculated (facilitated by our choice to model the multigrams as HMM's), this is not the case for the second condition. There, we apply a heuristic of parsimony, stating that more elements in the converged set is not always better.

2.1.1 The algorithm

When applying this method to language acquisition, the first step is to convert speech to symbols. In our experiments we have initially chosen to use phones (modelled by an HMM-based acoustic model). This is a simplification, since a real infant has no such knowledge to start with. The input then consisted of a somewhat inaccurate phone recognition of the speech signal (error rate ca. 25 %). The phonetic alphabet we used contained 43 phones, including noise (indicated by the symbol #).

The initial set we start from is then derived from an exhaustive listing of all possible subsequences of the original input sequence. Each of these subsequences can easily be converted to an HMM that is most likely to emit that particular subsequence. In our experiments, this was done as follows: each phone X was converted to a corresponding state S_X with emission probabilities defined as $p(X|S_X) = 0.5$, $p(\#|S_X) = 0$ and $p(\text{other symbol}|S_X) = 0.5/41$. For each transition between states, a dummy state D was defined that can be visited before going to the next state. Because of this, it is possible to model single insertions. The emission probabilities for this dummy state are: $p(\#|D) = 0$ and $p(X|D) = 1/42$. Aside from this dummy state to model a single insertion, we also add a transition for each state that skips the next state, modeling a single deletion. The transition probabilities for each state to go to its following dummy state, its next normal state or the state after that, are respectively initialized at 0.1, 0.8, and 0.1.

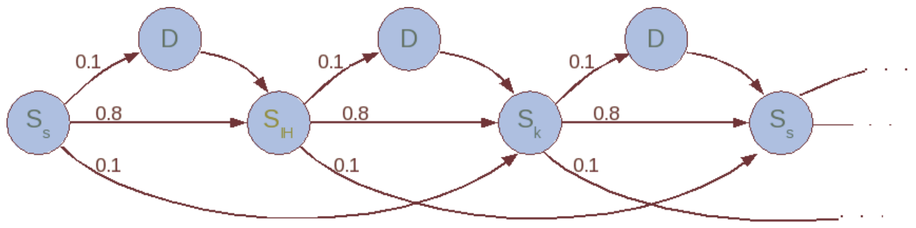

Figure 2.1: The HMM that was derived from the symbolic sequence "S IH K S".

The set of models we have at this moment is very large, and therefore not likely to be a good solution. It will be necessary to remove a number of elements. A sensible way to do this is to remove the patterns that occur the least. This can be straightforwardly done by determining for each element in the set the probability that it occurs in the speech signal and thresholding this with a certain value. The probability that a pattern occurs can be calculated as the relative number of occurrences of that pattern in the symbolic sequence derived from the signal, on the condition that we make a correction for the length of the pattern. Such a correction can be obtained with the following reasoning: if P is the probability of a certain subsequence being seen in the symbolic input without errors and Q is the probability of a certain pattern appearing in the speech signal, then we can say that approximately $P=QA^L$, with L being the number of symbols in the pattern and A the approximate probability of a symbol being right.

After that, the remaining models are again fitted to the input data. A new probability is calculated for each pattern (this time the correction is not needed) and the multigram set is once again pruned with the same threshold as before. This process is repeated until the set remains the same.

2.1.2 Results

The results on TIDIGITS looked very promising. If we determine the most likely symbol sequence of each of the remaining patterns, we get: W AH N, T UW, TH R IY, F AO R, F AY V, S IH K S, S EH V AH N, EY T, N AY N, OW, Z IH R OW, L and ER IY.

It is very clear to see that these phonetic patterns correspond to the actual words in the input. The presence of "ER I" can be explained by the tendency to recognize "TH R IY" as "T UW ER IY". The presence of "L" can be explained by the fact that it emerges as a natural garbage model, since "L" doesn't appear in any of the digits.

Mind that this is the result obtained when performing multigrams on a single sequence of symbols. When applying multigrams on phone lattices, obtained from an automatic phone recognizer, a lot more uncertainty is retained. As a consequence, the converged set becomes substantially bigger, although the correct patterns are all there and have a higher probability than the spurious ones.

2.1.3 Applicability of the approach for unsupervised learning

As will also be shown in section 2.6.2, the amount of variability in a stream of VQ-labels (or presumably any other speech representation with a high data rate and a large codebook) is enormous. The number of initial multigram candidates is too large. This severely limits the applicability of the multigram algorithm in the context of language acquisition without prior knowledge.

Our only hope to overcome the difficulties would be to make the learning process hierarchical, in which shorter (e.g. phone-sized) units with consequently less variation are discovered first. Word-level patterns would then be learned from the output of this layer.

Because of its serious problems with self-discovered inputs, its critical dependency on preset parameters (e.g. the probability threshold) and its lack of flexibility (e.g. removed patterns cannot be relearned), research into the multigram algorithm has been put on hold, in favor of other, more promising methods.

2.2 DP N-grams

The acoustic DP-ngram algorithm reported in this section is a modification of the preceding DP-ngram algorithm [5,6]. The original DP-ngram model was developed by Sankoff and Kruskal ([6]) to find two similar portions of gene sequences. Nowell and Moore ([5]) then modified this model to find repeated patterns within a single phone transcription sequence through self-similarity. Expanding on these methods, the author has developed a variant that is able to segment speech, directly from the acoustic signal; automatically segmenting important lexical fragments by discovering 'similar' repeating patterns. Speech is never the same twice and therefore it is impossible to find exact repetitions of importance (e.g. phones, words or sentences).

The use of DP allows this algorithm to accommodate temporal distortion through dynamic time warping (DTW). The algorithm finds partial matches, portions that are similar but not necessarily identical, taking into account noise, speed and different pronunciations of the speech.

Traditional template based speech recognition algorithms using DP would compare two sequences, the input speech vectors and a word template, penalising insertions, deletions and substitutions with negative scores. Instead, this algorithm makes use of quality scores, positive and negative, to reward matches and prevent anything else; resulting in longer, more meaningful sub-sequences. Initial test results show that there is significant potential with this approach, as it segments in an unsupervised manner, therefore not relying on a predefined lexicon or acoustic phone models.

What the acoustic DP-ngram model does:

- Finds similar repeating patterns directly from the acoustic signal
    - Works on the assumption that common words and phrases are acoustically similar, finding low distortion alignments between spectral representation of different regions of time in words and phrases
- Uses Dynamic Programming techniques to find partial matches
    - Partial matches allows the algorithm to discover alignments that are longer and more meaningful
- Clusters similar alignments
    - Alignments with the same underlying meaning will be grouped together. The centroid of each group is the ideal representation, which will constantly evolve and become more accurate as more data is processed
- Output can be used to create a symbolic representation of audio stream
    - Compresses the audio stream which can be used to feed back into the algorithm to find hierarchical repeating structure over a longer time range

2.2.1 Acoustic DP-ngram Method

Figure 2.2 shows the simplified architecture of the acoustic DP-ngram method. There are four main stages to the basic process:

1. Two utterances are fed to the DP-ngram algorithm as two sets of feature vectors.
2. A frame-to-frame distance matrix is calculated.
3. Accumulative quality scores are calculated for successive frame steps.
4. Local alignments are then discovered within the quality matrix.

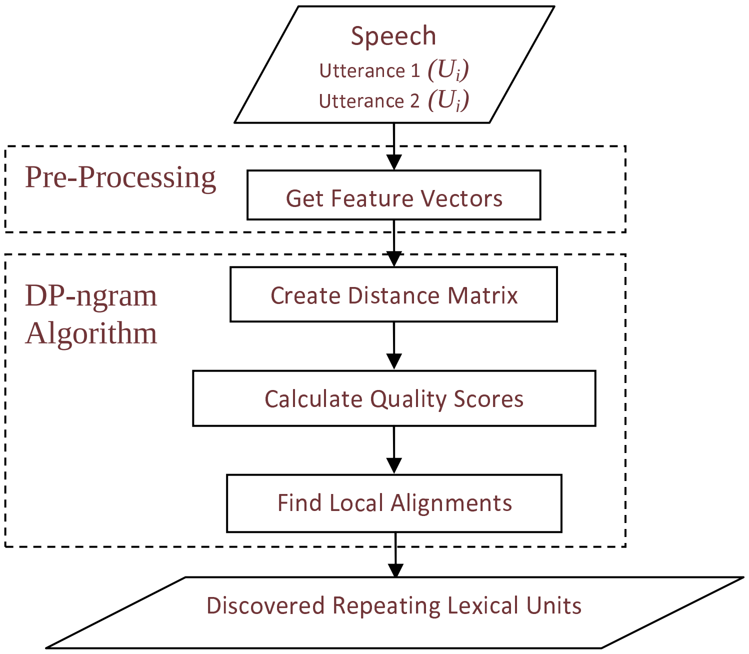

Figure 2.2: Acoustic DP-ngram architecture.

2.2.1.1 Feature Vectors

The test data being used is utterances of speech that have been specifically recorded by the ACORNS project during the first year. The ACORNS MFCC front-end has been used to parameterise the raw speech signal. The default settings have been used to output a series of 37-element feature vectors. The front-end is based on Mel-Frequency Coefficients (MFCC), which reflects the frequency sensitivity of the auditory system, to give 12 MFCC coefficients. A measure of the raw energy is added along with 12 differential (Δ) and 12 2nd differential (ΔΔ) coefficients. The front-end also allows the option for cepstral mean normalisation (CMN) and cepstral mean and variance normalisation.

2.2.1.2 Distance Matrix

A local-match distance matrix is then calculated by measuring the cosine distance between each pair of frames (v_1, v_2) from the two sequences, which is defined by:

$$d(v_1, v_2) = (v_1^T \times v_2)/(||v_1||^T \times ||v_2||) \tag{2.1}$$

Where,

$T =$ Transpose

Figure 2.3 is a plot of the frame-frame similarity for two utterances. The distance measure is on a scale of 0-1, where 1 is an exact match.

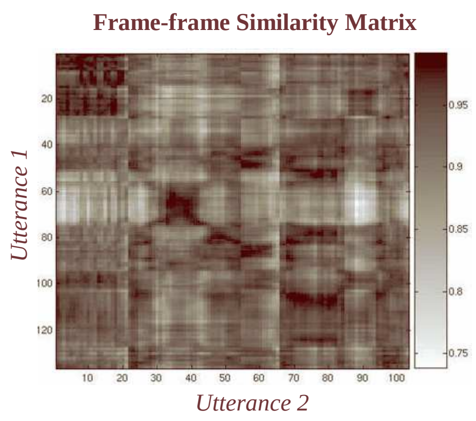

Figure 2.3: Frame-frame similarity matrix between two utterances.

2.2.1.3 Quality Scores

Traditional DP template based recognition systems use negative scores to penalise insertions, deletions and mis-matches to find the shortest distance. This method uses positive scores to reward matches and negative scores to discourage anything else, allowing us to find longer and more meaningful alignments.
The following recurrence is used to find all quality values $q_{(i,j)}$:

$$q_{i,j} = max \begin{cases} q_{i-1,j} + \left(s(a_i,\emptyset) \times \left(\left|d_{i-1,j-1} - 1\right|\right) \times q_{i-1,j}\right), \\ q_{i,j-1} + \left(s(\emptyset,b_j) \times \left(\left|d_{i-1,j-1} - 1\right|\right) \times q_{i,j-1}\right), \\ q_{i-1,j-1} + \left(s(a_i,b_j) \times d_{i-1,j-1} \times q_{i-1,j-1}\right), \\ 0, \end{cases} \tag{2.2}$$

Where,

$s(a_i,\emptyset) = -1.1$	Score for alignment ending with an insertion
$s(\emptyset,b_j) = -1.1$	Score for alignment ending with a deletion
$s(a_i,b_j) = +1.1$	Score for alignment ending with a substitution
$d_{i,j}$	Cosine distance between frames (i,j)

In order to maximize on quality, substitution scores must be positive and insertions/deletions must be negative. The recurrence also stops past dissimilarities causing global effects by setting all negative scores to zero, therefore starting a fresh new homologous relationship between local alignments. Figure 2.4 shows the plot of the quality scores after carrying out the recurrence (eq. 2.2).

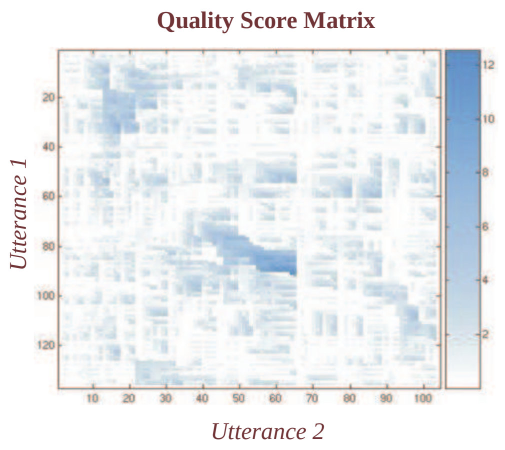

Figure 2.4: Quality score matrix for two utterances.

Applying a substitution score of 1 causes the quality scores to grow as a linear function. The current settings use a substitution score greater than 1 (1.1), thus allowing the quality scores to grow exponentially, giving longer alignments more importance.

Altering the negative insertion/deletion scores greater or less than -1 allows the model to increase or decrease the spread of quality scores, therefore allowing control over the tolerance for distortion. By setting insertion/deletion scores to values less than -1, the model will find closer matching repetitions, whereas a value greater than -1 allows the model to find repeated patterns that are longer and less accurate.

2.2.1.4 Finding Local Alignments

Backtracking pointers (bt) are maintained at each step of the recursion:

$$bt_{(i,j)} = \begin{cases} (i-1, j), & (deletion) \\ (i, j-1), & (insertion) \\ (i-1, j-1), & (substitution) \\ (0,0) & (initial\ pointer) \end{cases} \quad (2.3)$$

When the quality scores have been calculated with the recursion defined by equation 2.2, it is possible to backtrack from the highest score to obtain the local alignments in order of importance. A threshold is set so that only alignments of a desired quality are to be retrieved.

Figure 2.5 displays the steps taken to find the local alignments in order of importance:

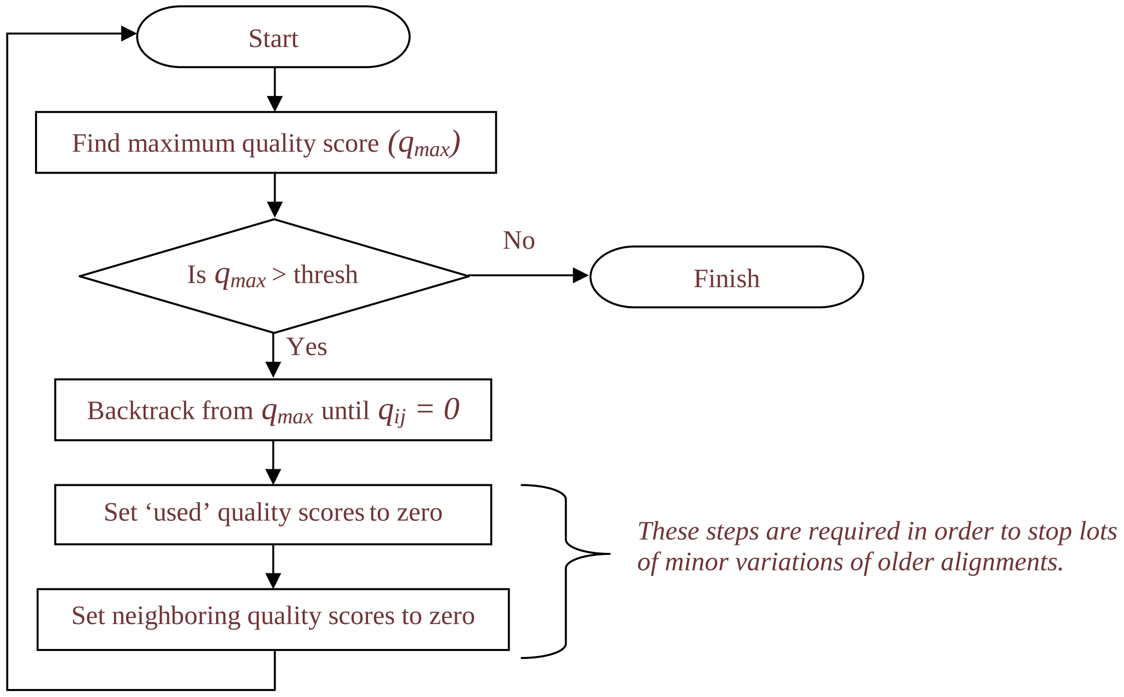
Start
Find maximum quality score (q_{max})
Is q_{max} > thresh
No
Finish
Yes
Backtrack from q_{max} until $q_{ij} = 0$
Set 'used' quality scores to zero
Set neighboring quality scores to zero
These steps are required in order to stop lots of minor variations of older alignments.

Figure 2.5: Flowchart of backtracking process to find local alignments.

Figure 2.6 presents the optimal local alignment that was discovered by the DP-ngram model for two utterances:

utt_1 "Finally Ewan was there"

utt_2 "But Ewan does"

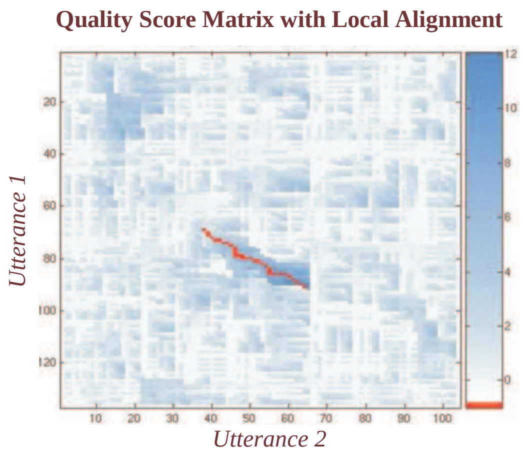

Figure 2.6: Quality score matrix with optimal local alignment plot.

The discovered repeated pattern is [y uw ah n]. Start and stop times are collected which allows the model to retrieve the alignment from the original audio signal in full fidelity when required.

2.2.2 Keyword Discovery

The milestone for the end of the first year of the ACORNS project is that LA should have learned 10 keywords. A keyword discovery (KWD) method has been added to the acoustic DP-ngram algorithm that continues the theme of a general statistical learning mechanism. The acoustic DP-ngram algorithm exploits the co-occurrence of similar acoustic patterns within different utterances; in its turn, the KWD method exploits the co-occurrence of semantic features to build internal representations of keywords. Both of these processes combine to achieve a system that is able to discover important word-like units from a cross-modal environment.

KWD is a simple approach that creates a class for each keyword, in which all discovered exemplar units representing each keyword are stored. With this list of episodic segments we can perform a clustering process to derive an ideal representation of each keyword.

For a single iteration of the DP-ngram algorithm, the current utterance (Utt_{cur}) will be compared with another utterance in memory (Utt_n). KWD hypothesises whether the segments found within the two utterances are potential keywords, by simply comparing the associated semantic tags. There are three possible paths for a single iteration:

1. If the tag of Utt_{cur} has never been seen before - create a new keyword class and store the whole utterance as an exemplar of it. Do not carry out the DP-ngram process and proceed to the next utterance in memory (Utt_{n+1}).
2. If both utterances share the same tag - proceed with the DP-ngram process and append discovered local alignments to that keyword class. Proceed to the next utterance in memory (Utt_{n+1}).
3. If both utterances contain different tags - do not carry out DP-ngram process and proceed to the next utterance in memory (Utt_{n+1}).

By creating an exemplar list for each keyword class we are able to carry out a clustering process that will allow us to create a model of the ideal representation. Currently, the clustering process implemented simply calculates the ‘centroid’ exemplar, finding the local alignment with the shortest distance from all the other local alignments within the same class. The ‘centroid’ is updated every time a new local alignment is added, therefore the system is creating internal representations that are continuously evolving and becoming more accurate with experience.

For recognition tasks the system can be set to either use the ‘centroid’ exemplar or all the stored local alignments for each keyword class. Using all the stored alignments gives more accurate results, but the processing required for this method increases exponentially with experience until it is not a viable method anymore. Using the ‘centroid’ is less accurate as it does not model the variance of the input. Future work will be carried to try and build a single ideal representation of each keyword class that models the variance.

Another important point to mention is that for the year 1 database there is only one tag associated with each utterance which makes this a binary decision for the KWD process. Utterances from the year 2 database are more complex and contain multiple keywords, but KWD will still run in the similar fashion with the addition of an ‘uncertain’ class; this is where segments are temporarily stored before a decision has been made on their correct keyword class.

2.2.2.1 DP-ngram - Batch Process

For a set of utterances the acoustic DP-ngram method compares every incoming utterance (U_i) with all past utterances $(U_{j=1\rightarrow(i-1)})$ to find local alignments. The search space for this process is the blue area in figure 2.7.

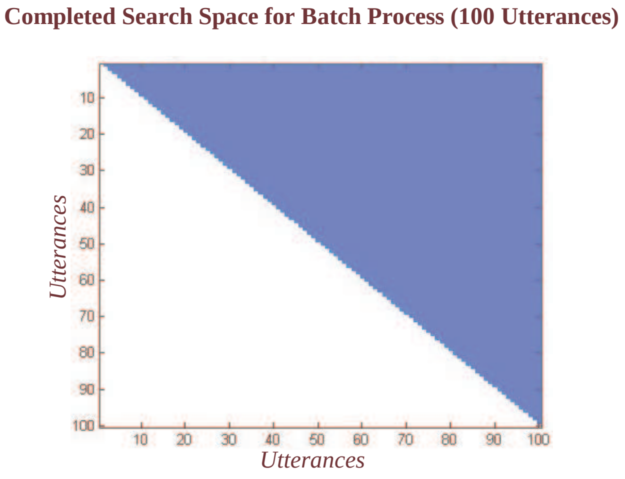

Figure 2.7: Plot of the completed search space for the DP-ngram as a batch process for 100 utterances.

As a batch process the DP-ngram method can only run on a limited number of utterances, with processing complexity increasing towards infinity. This approach does not lend itself to an online language and recognition process. It was necessary to design an implementation of the acoustic DP-ngram method that could potentially handle an infinite number of utterances.

2.2.2.2 Incremental DP-ngram

Running the acoustic DP-ngram method as a batch process on the ACORNS English corpus (4000 utterances) is not viable with current processing and memory capabilities. Therefore an incremental version of the acoustic DP-ngram method has been designed. Also, running the algorithm as an incremental process allows for a more cognitively plausible system [13].

The model is made incremental by restricting the number of past utterances to be compared with the incoming utterance(U_i) with an utterance window $\left(U_{j=UttWindow\to(i-1)}\right)$. This creates a search space as shown in figure 2.8 a) & b).

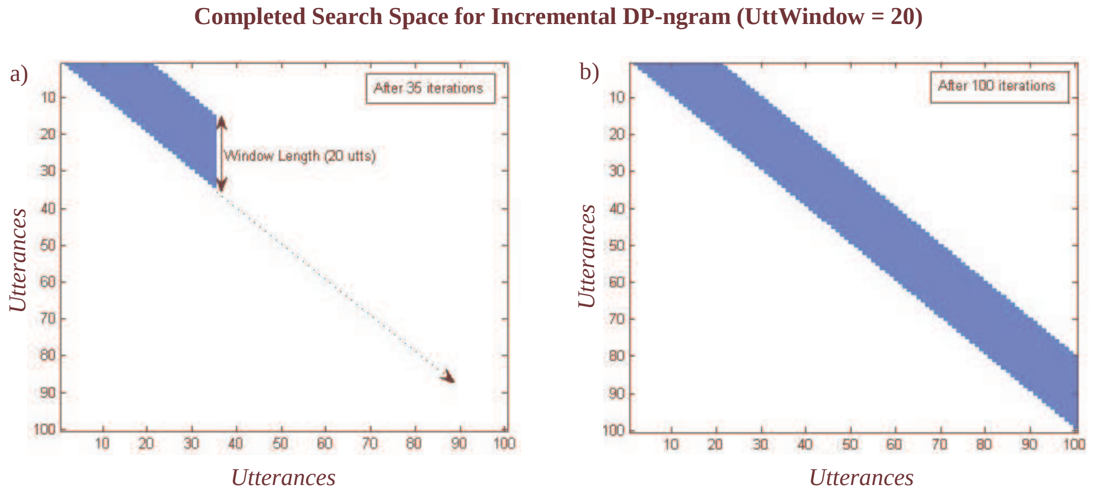

Figure 2.8: a) Plot of search space for the incremental DP-ngram after 35 utterances with a window length of 20 utternaces. b) Shows a plot of the completed search space after 100 utterances.

Decreasing *UttWindow* allows the system to run faster at the expense of reducing the search space, therefore running the risk of potentially missing important repetitions. It was necessary to find the minimum *UttWindow* length that will allow the system to find enough important repetitions in order to build accurate internal representations of the keywords.

2.2.3 Experiments

Accuracy of experiments within the ACORNS project is based on LA's response to its carer. The response the carer is looking for is if LA can predict the keyword tag associated with the current incoming utterance by only using the speech signal. The acoustic DP-ngram implementation attempts to solve this task using a method similar to traditional DP template based recognition. The recognition process is carried out by comparing exemplars, of discovered keywords, against the current incoming utterance using the DP-ngram method to calculate quality scores. Thus, the alignment that produces the highest quality score, by finding the longest alignment, is taken to be the match, with which we can predict its associated visual tag.

A number of different experiments have been carried out:

1. *Finding the optimal utterance window length for incremental DP-ngram*

For this experiment, varying values of the utterance window length (from 1 to 100) were used to obtain keyword recognition accuracy results across the same data set.

2. *Comparing DP-ngram as a batch and incremental process*

The optimal window length chosen for the incremental implementation is compared against the batch implementation of the DP-ngram algorithm.

3. *Keyword Discovery - Centroid vs Complete Exemplar List*

The KWD process stores a list of exemplars representing each keyword class. For the recognition task we can either use all the exemplars in each keyword list or a single exemplar that best represents the list, the 'centroid'. This experiment will compare these two methods for representing internal representations of the keywords.

4. *Speaker-dependency*

The algorithm is tested on its ability to handle the variation in speech from different speakers. Different feature vectors from the front end are fed to the system. The feature vectors used are listed below:

V1 - Default HTK 39-element MFCCs (no normalisation)
V2 - ACORNS 37-element MFCCs (no normalisation)
V3 - ACORNS 37-element MFCCs with Cepstral Mean Normalisation
V4 - ACORNS 37-element MFCCs with Cepstral Mean and Variance Normalisation

Using normalization methods will reduce the information within the feature vectors, removing some of the speaker variation. Therefore, accuracy results should be better for a data set of multiple speakers with normalization.

5. *Comparing DP-ngram against the NMF approach*

The NMF approach was the first and only end-to-end approach within the ACORNS project and is used as a baseline for ACORNS experiments. The NMF results are based on the ACORNS Y1 Dutch corpus with 4 different speakers. The data set contains 10 keywords and the utterances are fed to the system at random.

2.2.3.1 Test Data

The test data being used for experiments 1 and 2 is a sub-set of the ACORNS Y1 UK corpus – 100 different utterances from a single speaker. Each utterance contains one of the 10 keywords and each keyword is presented 10 times.
Experiment 3 uses 200 utterances from the ACORNS Y1 UK corpus, but includes all four speakers (2 male and 2 female) presented in a random order. There are still 10 keywords present but the number of times they occur within the data set is random.
Experiment 4 uses the same setup as the data set for experiment 3 but the utterances are from the ACORNS Y1 Dutch corpus. Table 1 shows the test data being used for all four experiments.

Table 1: Test data for DP-ngram experiments 1-4.

Experiment	Corpus	Utterances	Speakers	Keywords	Occurrences
1	UK Y1	100	1 m	10	10
2	UK Y1	100	1 m	10	10
3	UK Y1	200	1m	10	20
4	UK Y1	200	2 m - 2 f	10	Random
5	NL Y1	200	2 m - 2 f	10	Random

2.2.3.2 Keyword Recognition Results

1. *Finding the optimal utterance window length for incremental DP-ngram*

The DP-ngram algorithm was carried out on 100 utterances with varying utterance window lengths. The plot in figure 2.9 shows the total accuracy result for each window length used. The x-axis displays the utterance window lengths used (1–100) and the y-axis displays the total accuracy (%).

The results are as expected. Longer window lengths achieve more accurate results. This is because longer window lengths achieve a larger search space and therefore have more chance of capturing repeating events. Shorter window lengths are still able to build internal representations, but over a longer period. Accuracy results reach a maximum with an utterance window length of 21 utterances and then stabilize at around 58% (±1%). This shows us the minimum window length needed to build accurate internal representations of the words within the test set. The window length used for all proceeding experiments is 21 utterances.

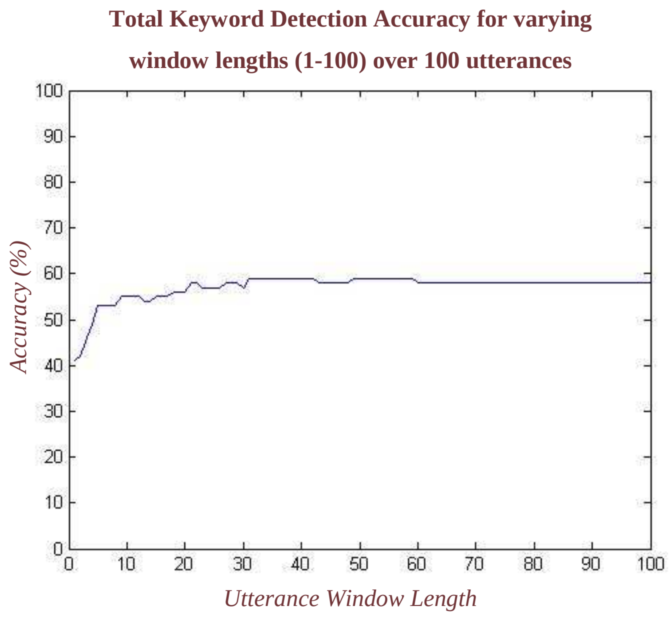

Figure 2.9: Plot of the total keyword accuracy using varying utterance window lengths of 1-100. Each trial has been carried out on a test set of 100 utterances from a single speaker.

2. *Comparing DP-ngram as a batch and incremental process*

The plot in figure 2.9 displayed the total accuracy result for the different utterance window lengths and does not show the gradual word acquisition process. Figure 2.10 compares the word detection accuracy of the system (y-axis) as a function of the number of utterances observed (x-axis). Accuracy is recorded as the percentage of correct replies for the last ten observations. The red plot shows the accuracy for randomly guessing the keyword.

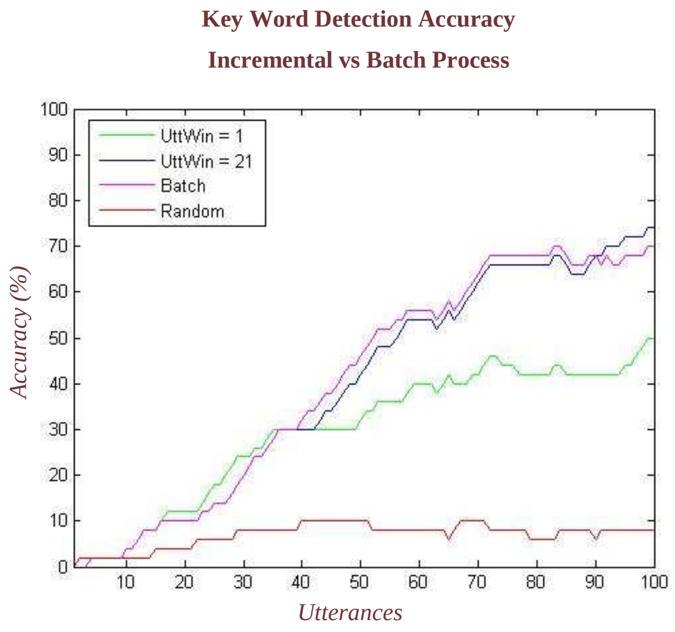

Figure 2.10: Keyword detection accuracy for the DP-ngram algorithm running as a batch and incremental process. Results are plotted as a function of the past 10 utterances observed.

It can be seen from the plot in figure 2.10 that the system begins life with no word representations. At the beginning, the system hypothesises new word units from which it can begin to bootstrap its internal representations. As an incremental process, with the optimal window length, the system is able to capture enough repeating patterns and even begins to outperform the batch process after 90 utterances. This is due to additional alignments discovered by the batch process that are temporarily distorting a word representation, but I believe the batch process would 'catch up' in time.

Another important result to take into account is that only comparing the last observed utterance is enough to build word representations. Although this is very efficient, the problem is that there is a greater possibility that some words will never be discovered if they are not present in adjacent utterances within the data set.

3. *Keyword Discovery - Centroid vs Complete Exemplar List*

Currently the recognition process uses all the discovered exemplars for each keyword class. This process causes the computational complexity to increase exponentially. It is also not suitable for an incremental process with the potential to run on an infinite data set.

Another method employed was to calculate the 'centroid' for each keyword class and use this single exemplar unit for recognition. Figure 2.11 shows the accuracy as a function of utterances observed for both methods.

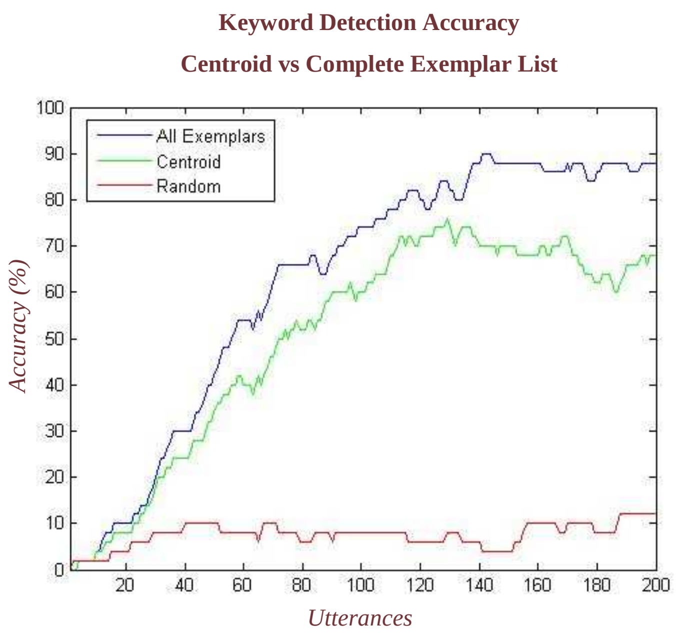

Figure 2.11: Comparison of keyword detection accuracy using centroids or complete exemplar list for recognition.

The results show that the 'centroid' method is quickly outperformed and that the accuracy gap increases with experience. After 120 utterances performance seems to gradually decline. This is because the 'centroid' method cannot handle the variation in the acoustic speech data. Using all the discovered units for recognition allows the system to reach accuracy results of 90% at around 140 utterances, where it then seems to stabilize at around 88%.

4. *Speaker-dependency*

This experiment has been carried out to test the speaker-dependency of the DP-ngram method. The addition of multiple speakers will add greater variation to the acoustic signal, distorting patterns of the same underlying unit. Over the 200 utterances observed, accuracy of the internal representations increases but at a much slower rate than the single speaker experiments.

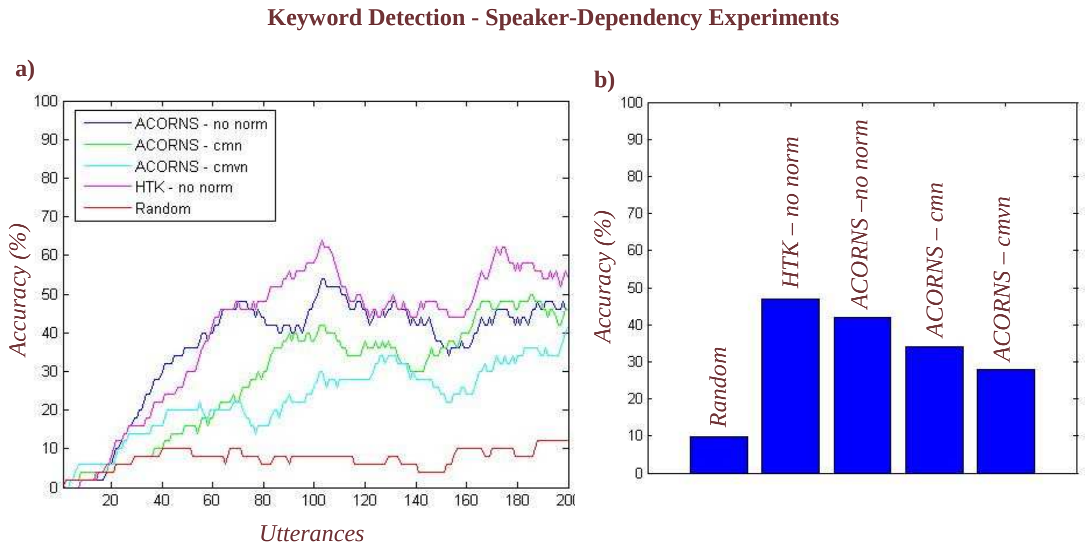

Figure 2.12: a) Accuracy of the system using different feature vectors as a function of utterances observed. b) Total accuracy after 200 utterances.

The assumption that using normalization methods would achieve greater word detection accuracy, by reducing speaker variation, does not hold true. On second thought it is not surprising, as the system will be collecting exemplar units for each speaker.

This brings up another issue; the optimal utterance window length for the incremental DP-ngram process was calculated for a single speaker. Increasing the search space will allow the model to find more repeating patterns from the same speaker. With this logic, it could be hypothesized that the optimal search space should be four times the size used for one speaker and that it will take four times as many observations to achieve the same accuracy results.

5. *Comparing DP-ngram against the NMF approach*

The NMF approach is currently the baseline for word discovery experiments within the ACORNS project. Figure 2.13 plots the keyword detection accuracy results for the two methods as a function of observed utterances. The test set consists of Dutch utterances from four different speakers (2 male and 2 female) in a random order.

Accuracy of the internal representations rise at a much faster rate for the NMF method compared to the DP-ngram method, although there is a sharp rise up to 40% after 170 utterances.

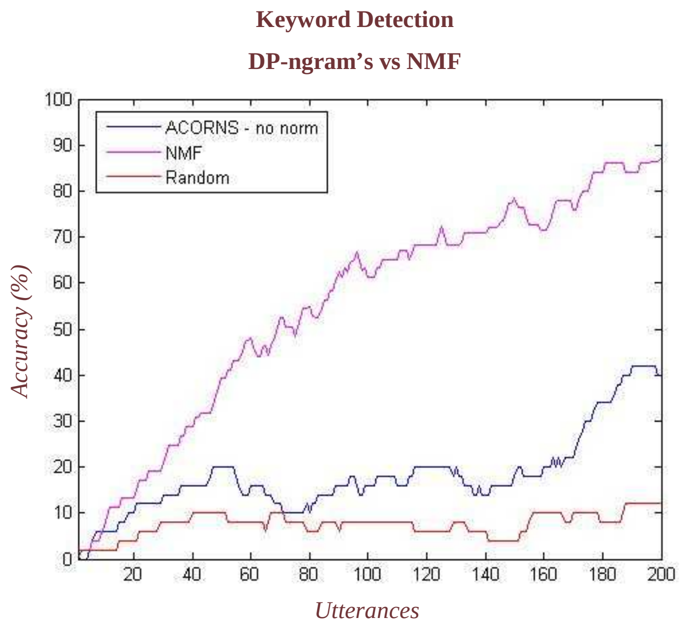

Figure 2.13: Comparison of keyword detection accuracy for NMF and DP-ngram methods over 200 utterances.

2.2.4 Conclusions

Preliminary results indicate that the environment is rich enough for word acquisition tasks. The pattern discovery and word learning algorithm implemented within the LA memory architecture has proven to be a successful approach for building stable internal representations of word-like units. The model approaches cognitive plausibility by employing statistical processes that are general across multiple modalities. The incremental approach also shows that the model is still able to learn correct word representations with a very limited working memory model.

Additionally to the acquisition of words and word-like units, the system is able to use the discovered tokens for speech recognition. An important property of this method, that differentiates it from conventional ASR systems, is that it does not rely on a pre-defined vocabulary, therefore reducing language-dependency and out-of-dictionary errors.

Another advantage of this system, compared to systems such as NMF, is that it is easier to give temporal information of the whereabouts of important repeating structure which can be used to code the acoustic signal as a lossless compression method.

2.2.5 Discussion & Future Work

A key question driving this research is whether modelling human language acquisition can help create a more robust speech recognition system. Therefore further development of the proposed architecture will continue to be limited to cognitively plausible approaches and should exhibit similar developmental properties as early human language learners. In its current state, the system is now fully operational and intended to be used as a platform for further development and experiments.

The experimental results are promising. However, it is clear that the model suffers from speaker-dependency issues. The problem can be split into two areas, front-end processing of the incoming acoustic signal and the representation of discovered lexical units in memory.

Development is being carried out on various clustering techniques that build constantly evolving internal representations of internal lexical classes attempting to model speech variation. Additionally, a secondary update process, implemented as a re-occurring 'sleeping phase' is being investigated. This phase is going to allow the memory organisation to re-structure itself by looking at events over a longer history as a batch process.

The processing of prosodic cues, such as speech rhythm and pitch intonation, will be incorporated within the algorithm in an attempt to increase accuracy and further exploit the richness of the learners surrounding environment. Adults, when speaking to infants, will highlight words of importance through infant directed speech (IDS). During IDS adults place more pitch variance on words that they want the infant to attend to. Development of an additional perception module that uses pitch variance as a simple attention mechanism has been initiated; the module gives a greater weight to instances within an utterance with more pitch variance in an attempt to discover units of greater importance.

Further experiments have been planned to see if the model exhibits similar patterns of learning behaviour as young multiple language learners. Experiments will be carried out with the multiple languages available in the ACORNS database (English, Finnish and Dutch).

2.3 Computational mechanics (CMM)

2.3.1 Causal states

Computational mechanics is a research programme with the aim of using automata theory to describe patterns and the complex, stochastic processes that generate them. Central to these descriptions is the concept of *causal states* [19, 20]. The causal states are equivalence classes defined over the possible *histories* (sequences of observations from negative infinity until the current time t) of a given stationary, discrete-time stochastic process. Two histories belong in the same causal state if and only if they give the exact same beliefs about the future, i.e., if they imply the same probability distribution over all *futures* (sequences of observations from t+1 and on to infinity). The causal states are thus a partitioning of the set of possible histories.

One can show that the causal state representation is a *minimal sufficient statistic* for the observation sequence; it retains precisely all information from past observations relevant for predicting the future, and nothing more. Moreover, appending a symbol to a history string gives a new history string that also belongs in some causal state. This way it is possible to define transitions between the states. Interestingly, the states and their transitions together constitute a Markov process—even if the original process is not Markovian. Unlike HMMs, the current state can, in this description, be uniquely identified from the available sequence of observations.

2.3.2 The CSSR algorithm

Causal states are a theoretical construct based on full knowledge of the underlying process. Fortunately, in practice an approximation of the causal states can be learned from one or more empirical data sequences using the so-called *causal state splitting reconstruction algorithm* (CSSR) by Shalizi, Klinkner, and Crutchfield [21, 22]. Given enough data, this procedure converges on the true causal states. However, the algorithm only operates on sequences of discrete symbols from a finite alphabet.

Unlike the theory, practical algorithms have to make do with histories of finite length. CSSR, in particular, only considers the $L_{\max}$ most recent symbols at any given point in the data, known as a *suffix* of the history string, where $L_{\max}$ is a user-set memory length parameter. Despite this limit on suffix length, CSSR can actually learn certain processes with a non-fixed, potentially infinite memory, such as the even process of Weiss. However, asymptotic convergence requires that the number of causal states is finite and that $L_{\max}$ is not set too low.

In brief, the CSSR algorithm consists of three main steps: parsing the data, homogenization, and determinization. These are outlined below (see also [21, 22]).

- To parse the data the algorithm simply counts the number of occurrences of all N-grams in the data with a length shorter than or equal to $L_{\max}$+1. These can be arranged into a tree, so CSSR belongs to the class of so-called *context tree* or *suffix tree* methods. This class also includes variable length Markov models like those discussed in [23], and compression algorithms such as [24].

- During homogenization CSSR iteratively looks at longer and longer suffixes, up until length $L_{\max}$, and collects these together into states based on what distribution they give for the next symbol. The assignment is based on a statistical test, for example two-sample chi-squared, comparing the distributions. The test is carried out at a level α, a second user parameter.

  There is a bias for placing suffixes in the same state as their *parent suffix* (the suffix obtained by removing the oldest symbol from a given suffix), to prevent unnecessary splits. The result of homogenization is known as *precausal states*, since they can predict optimally one step into the future.
- Since our beliefs about the future change in a deterministic way with each new symbol we observe, the transitions between the causal states must be deterministic given the next symbol. Consequently, precausal states where suffixes end up in more than one state after appending some positive-probability follower symbol cannot be true causal states, and are therefore split to be deterministic. This may cause other, previously deterministic states to become non-deterministic, but the procedure must eventually terminate. The deterministic transitions then achieved enables the resulting set of states to predict the entire future optimally, assuming the precausal states were correctly partitioned.

Before and after determinization, CSSR also identifies any *transient states*, which are states that typically only are visited a finite number of times even if the output automaton is run for an infinitely long time. These are not considered true causal states and are therefore removed.

Since the parse tree may be expanded fully to the depth $L_{\max}+1$, the worst-case requirements for data and computational power are exponential in $L_{\max}$. On the other hand, since the data is read sequentially once, computational complexity is linear in the data sequence length.

The output of CSSR is a so-called *deterministic finite automaton*, or DFA, which is a state representation similar to HMMs. However, unlike HMM training with the EM-algorithm, the states and their structure is recovered automatically. CSSR thus performs unsupervised pattern discovery, not just pattern recognition.

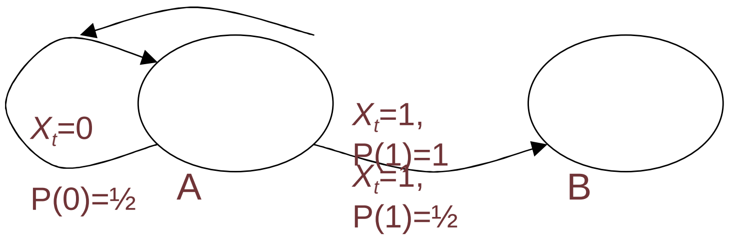

Figure 2.14: The even process, a so-called strictly sofic process with two causal states learnable by CSSR. The output generated by the process always has an even number of contiguous ones.

2.3.3 First year experiments with CSSR in ACORNS

The aim of the first year experiments with CSSR in ACORNS was to test the suitability of the algorithm for speech recognition and language acquisition. Since CSSR appears not to have been tested much in this context, the first experiments were to apply the procedure to a simple speech-related dataset in order to study the properties of the learned representations. Specifically, CSSR was applied to a symbolic representation of the recording protocol used in generating the first-year ACORNS Swedish speech data corpus. In this representation, each word was assigned a unique symbol. Word variants such as inflections were given different symbols if their spelling differed. The protocol was then converted to a single data stream by concatenating all

1,000 spoken lines with phrases separated by an additional symbol signifying inter-utterance silence. This yielded a data sequence of 4,295 symbols drawn from a 23-symbol alphabet.

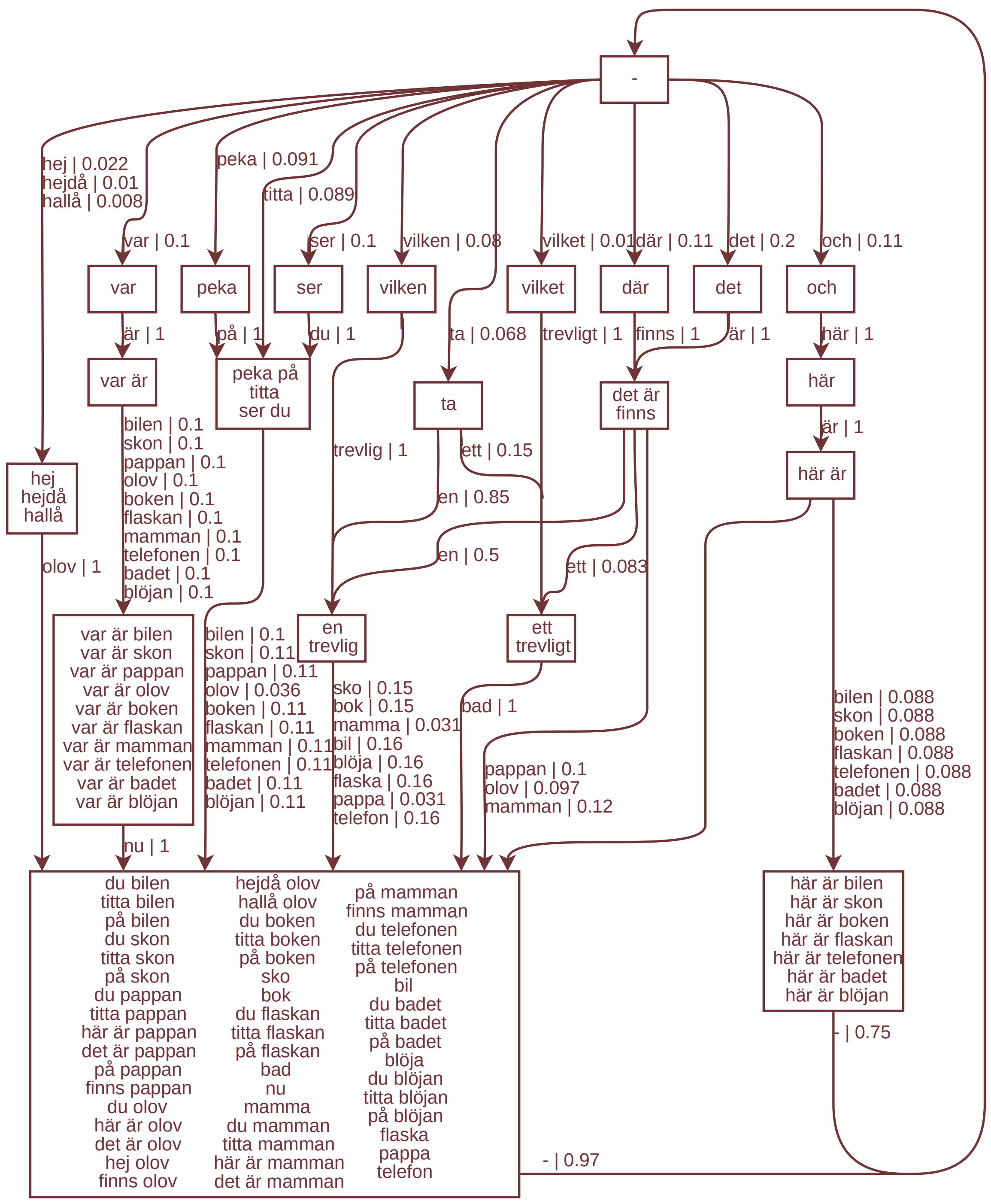

Figure 2.15: Reconstructed automaton with Y1 Swedish data (L_{max}=4, α=0.002). The existence of two separate end states indicates a likely problem with the original CSSR implementation obtained at [25], necessitating a reimplementation of the algorithm for ACORNS.

Results from these first experiments were encouraging. Despite the limited amount of data, CSSR learned a near-perfect automaton representation of a stationary stochastic process to generate the sentences in the observed data. Each state typically represented a specific word or position within one of the carrier sentences. However, getting the desired conclusion required setting the algorithm parameters just right; this problem diminished in importance if additional training data was randomly generated from the Y1 bag of sentences.

Following the first experiments above, a second goal was to assess the behavior of CSSR on more realistic data, including complications such as noise. To this end, another dataset was generated, similar in character to the symbolic representation of the ACORNS recordings but also featuring low-probability (P=0.05) symbol substitution noise. The data consisted of one million symbols from a simple model of speech as a sequence of randomly occurring words. Every word comprises a sequence of 'phones' (symbols) taken from a pre-generated random 'wordlist' of ten short symbol strings, each 4–8 symbols long. Eight different phones were used. Note that this stochastic process can also be considered as a model of ten different sentences repeated at random, similar to the material used in the first experiments, if each symbol is instead taken to represent a word.

In stark contrast to the result on noise free data, the algorithm here failed to converge on a limited set of causal states. Instead, the number of reconstructed states now grew steeply as larger and larger values for the memory length parameter L_{max} were considered, with no end in sight. The computing power requirements also increased prohibitively quickly. The same behavior persisted for similar language models with reduced word lengths, shorter word lists, and smaller alphabet sizes, as long as noise was present.

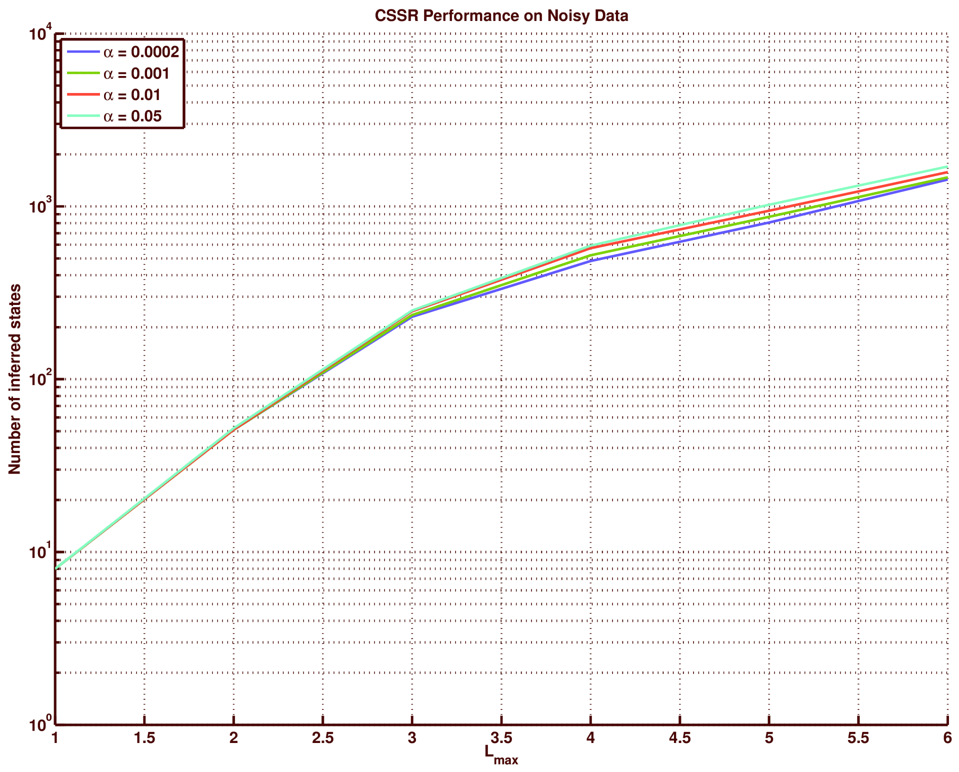

Figure 2.16: Unchecked growth in the number of reconstructed states for a noisy test dataset with one million symbols from a size eight alphabet.

Similar divergence with increasing L_{max} has also been observed by a group studying the applicability of CSSR for natural language processing tasks such as Named Entity Recognition, on a data corpus derived from Spanish text [26].

2.3.4 Causal states and noise

After the sensitivity to noise was uncovered, CSSR efforts in ACORNS have focused on understanding this behavior and exploring ways around it. Theoretical work within WP2 has established that many HMMs, also very simple ones, cannot be represented by a finite number of causal states, and are not learnable using CSSR. There are therefore strong indications that the explosive increase in the number of states identified seen in the experiments corresponds to a genuinely large number of causal states.

The essence of the problem appears to be that, the further back the CSSR algorithm looks at the data, the more information comes to light that affects future behavior, so memory length is infinite. As the algorithm is intended to capture all information relevant for prediction, these differences count in CSSR even if their influence is small enough that a human would label them as noise; the algorithm's definition of “pattern” versus “noise” is not the same as our own. Consequently, the algorithm discovers a very large number of distinct possible predictions for the future, each of which has to be represented by its own causal state.

While it is useful to have some knowledge about the characteristics of the noise in the process being studied, it seems that CSSR takes this too far. Aside from the obvious problem of storing such a large number of states in memory, there is a clear risk of overfitting as the information in the data is divided between all these states (although many of the causal states are typically quite similar, the base form of CSSR is not designed to take this into account). Results in [26] also show that, whereas the number of states increased rapidly in L_{max}, the best system performance was actually attained with a very small number of states at $L_{\mathrm{max}}=3$, the shortest memory length used.

It can be argued that learning and generalization must be a lossy process, where the goal is to identify what information in the data to discard, so that only the salient parts are kept. The causal states, in particular, are designed to retain precisely all information that is relevant for prediction. Since this can be unfeasibly much to learn, a natural question is if it is possible to discard additional, mostly unimportant information, and only keep those parts that significantly influence on our predictions, in order to obtain a more compact representation. After all, the results in [26] hint that a smaller, lossy representation can achieve better performance for realistic sample sizes.

2.3.5 CSSR with resolution

Following the reasoning above, WP2 is developing an extension of CSSR where the algorithm is modified to include a concept of *resolution*. The central idea is to create a state representation where any two casual states are not necessarily distinguished from each other if they give similar (but not identical) beliefs about the future. This representation should be smaller than that offered by the causal states, without sacrificing too much predictive power.

The extension would work in two ways, starting with homogenization. Like CSSR, history string suffixes should there be assigned to states using statistical hypothesis tests, but these may now test if two next-step distributions are similar (if the difference in some metric is likely to be less than some user-set resolution parameter δ), instead of checking if they are exactly the same or not. This should produce fewer states prior to determinization that still predict the next symbol well. We tentatively call these states *precausal clusters*.

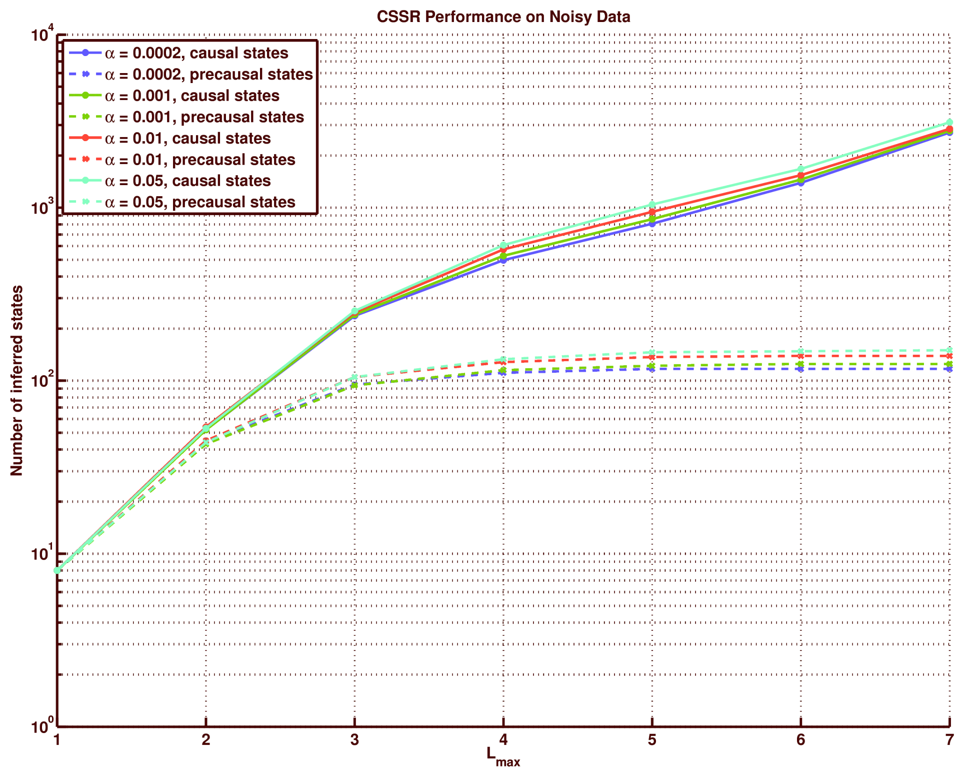

Figure 2.17: Recent result illustrating the number of recurrent causal and precausal states as a function of $L_{\max}$ for the same learning task as the previous figure.

Looking into the behavior of CSSR in an application, it seems like the number of precausal states largely stabilizes already at low $L_{\max}$, but that determinization then fractures these states into smaller and smaller pieces as $L_{\max}$ grows; the precausal states for a noisy problem do not determinize in a simple way. It is unlikely that this will change much by simply creating precausal clusters instead. A second, complementary change would then be to introduce the idea of resolution to the determinization stage as well.

One possibility currently under consideration would be to perform *incomplete determinization*, where the state at t+1, considered as a random function of the state at t and the observation at t+1, is not required to be fully deterministic when the algorithm is finished. Instead, determinization may stop whenever all transition functions satisfy $H(S_{t+1}(S_t,X_t))\leq H_{\max}$, where the maximum permissible entropy $H_{\max}$ is an additional parameter.

While the output after such a procedure can be a non-deterministic automaton, the amount of randomness in the transitions is limited, which should work to reduce how much information is lost. Furthermore, since most inference algorithms and other tools typically apply equally well also to non-deterministic processes like HMMs, there is no obvious reason to require an output automaton with deterministic transitions if only an approximation of the causal states is desired. Indeed, there are other algorithms based on CSSR, such as CCSA [27], that do not perform any determinization at all, a behavior that can be replicated here by choosing a high enough permissible entropy.

2.3.6 Plans for the final year

WP2 is currently preparing a C++ implementation of CSSR with and without resolution, to be released soon. This implementation will be used for experiments to explore the effects of incorporation resolution into the CSSR process. The test data for these investigations will include a symbol (speech) stream generated by the same BNF grammar used create the Y2 ACORNS speech database. Since CSSR performs best on symbol dictionaries of small size, this would likely be applied to a phonemic representation of the speech. The work on CSSR with resolution will result in a publication.

For the final year, the work on CMM/CSSR should also be integrated into the ACORNS framework. One proposal is to apply CSSR with resolution to parallel streams of phones and prosodic information. The learned automata may then be used with the Viterbi algorithm to reconstruct a prosodic stream from phones in cases where it is absent or weakened, such as ADS speech. The reconstructed prosodic information might aid tasks such as segmentation. Other learning tasks and ways of using the learned representations may also be considered.

Apart from endowing CSSR with a notion of resolution, a parallel line of research may be to integrate established N-gram techniques such as smoothing (pseudocounts) and back-off models into CSSR and algorithms derived from it, for enhanced performance.

There are also related Markov model learners where states are identified by strings of explicit observations [28, 29], or that otherwise have been developed with the power to solve similar learning tasks as CMM/CSSR, as discussed in [30]. These may be considered both as alternatives to and for purposes of comparison with the techniques currently being investigated.

2.4 Self-learning vector quantization (SLVQ)

2.4.1 Overview of the SLVQ

Motivation for developing a novel algorithm for speech quantization originated from the need of cognitively plausible classification mechanism that can incrementally learn to categorize sounds in an unsupervised manner. The idea of using neural networks for bottom-up speech sound classification was discarded from the beginning due to the difficulties of interpreting and adjusting their behavior. Therefore a novel algorithm was developed for spectral classification purposes. We shall call this new algorithm self-learning vector quantization (SLVQ), since it performs quantization based on incremental learning from the data in an unsupervised manner.

The self-learning vector quantization algorithm takes one feature vector at a time as input and compares it to the existing cluster structures. If no suitable match is found, a new cluster is created for the input. Radii of the clusters are defined adaptively by the amount of cluster “mass” in corresponding parts of the cluster space, high density of space leading to higher resolution (smaller cluster radii). The algorithm resembles k-means clustering in many aspects, but with a handful of important differences:

1) *The number of clusters is not specified* manually in advance. The spectral content of input determines the number of clusters that will form. However, other manually set parameters exist that have impact on the amount of clusters.

2) The algorithm works *incrementally*. The classification process starts already from the second input vector. There is no need to store massive amounts of vector data.

3) The algorithm *keeps track only of the cluster centroids*. New input vectors merging to a cluster only affect the cluster centroid and are then discarded immediately as separate entities.

4) Due to points 2) and 3), the algorithm is *memory efficient and computationally cheap*.

5) Due to points 2) and 3), the algorithm *does not obtain a global minimum in quantization error* of the input. However, this has also some positive aspects on non-uniformly distributed data such as speech (see further below).

6) Due to points 2) and 3), the algorithm can create new clusters for novel input if necessary without extensive computation (expanding the codebook) while the old clusters can be either adjusted accordingly or kept fixed.

2.4.2 The SLVQ-algorithm

Important aspect in the training of SLVQ space for speech recognition purposes is the type of input used for quantization. In experiments performed so far the training of the cluster space uses MFCC features extracted from phone-like speech segments, where in practice a segment is defined as an internally spectrally coherent unit. Segmentation is performed with a blind segmentation algorithm (Räsänen, 2007; Räsänen, submitted for publication). The use of segmental information enhances the information value of spectral vectors because the algorithm is able to extract spectral features from locations that are well aligned with phones. Velocity and acceleration information is taken from transition points between phone-like units, whereas the static spectrum is extracted from the middle parts of the segment. This reduces the variability in the vectors since spectrally complex transitions are not included in static spectra. However, it may also ignore small detail information from these transition points. In any case, the use of segmental information reduces the amount of data required for the clustering process significantly and helps to avoid groups of clusters that describe variations of ambiguous transitions from one phone to another in great detail. Since SLVQ was originally developed for use in association with this type of input, the use of segmental input is coupled with the description of the algorithm for the remainder of this report, although the possible input data type is in no way limited to segmental features in the algorithm.

During development of first versions of incremental clustering it turned out to be evident that quantization accuracy is very inefficient with clusters of equal radius unless the input data is also uniformly distributed in spectral space, which is not true for FFT or MFCC representations of speech frames. More resolution is required at those sections of feature space that contain large amounts of input data. Therefore cluster radiuses are defined adaptively: the radius of a cluster depends on how densely populated the vector space is in the neighborhood of the cluster. Density of the space from the viewpoint of cluster X_j is measured with radius weight factor w_j which is defined as:

$$w_j = \frac{\sum_i d_{i,j}^k n_i}{\sum_i n_i} \qquad (2.4)$$

where n_i is the number of vectors merged to cluster i and k is a constant that determines how the measure behaves as a function of distance (a linear distance weight k = 1 has been found effective in experiments performed so far). Distance $d_{i,j}$ between centroids i and j is computed using a chosen metric, so that the d increases as the similarity decreases. Cross-correlation of MFCC vectors was used in the development and experiments with the algorithm (note that cross-correlation increases with the similarity so its value has to be inverted in order to be compatible with notation used in this paper).

All cluster radiuses $\boldsymbol{t}$ can be then computed by first combining weight factors of all clusters into a single vector $\boldsymbol{w}$ and then operating it with function $\mathrm{M}(\boldsymbol{w})$ that maps the values of $\boldsymbol{w}$ linearly into range $[t_{min}, t_{max}]$, the largest value in $\boldsymbol{w}$ getting value of t_{max} and smallest value of $\boldsymbol{w}$ getting value of t_{min}.

$$\boldsymbol{t} = \mathrm{M}(\boldsymbol{w}) \qquad (2.5)$$

The steps of the clustering process can be written down as following:

```
1.    Take a new input vector $\boldsymbol{v_i}$ and compute its distance $d_{i,j}$ to all existing
      clusters.

2.    if $\forall d_{i,j} > t_j$, where $t_j$ is radius of cluster $X_j$
            create a  new cluster with centroid $\boldsymbol{v_i}$ and radius $t_0$. Increment s by $s_+$.
      else if $\exists d_{i,j} \leq t_j$,
            merge $\boldsymbol{v_i}$ to $X_c$, where c = $arg_j\ min\ d_{i,j}$, by having $\boldsymbol{x_c} = (\boldsymbol{x_c}+\boldsymbol{v_i})/(n_c+1)$,
            where $\boldsymbol{x_c}$ is the cluster centroid of $X_c$ and $n_c$ is the number of vectors
            already merged to the cluster $c$. Increment s by $s_+$.

3.    if $s > s_t$, where $s$ is sleep counter and $s_t$ sleep threshold
            3.1 update all cluster radiuses using eqs (1) and (2).
            3.2 merge all cluster pairs $X_i$ and $X_j$ that that satisfy $d_{i,j} < t_j$ or $d_{i,j}$
            $< t_i$ by having by having $\boldsymbol{x_{ij}} = (\boldsymbol{x_i}+\boldsymbol{x_j})/(n_i+n_j)$. Go to 3.1 until all
            clusters satisfy $\forall d_{i,j} > t_j$.
            3.3 reset $s$.
4.    Go to step 1.
```

Most central parameters are t_0, s_t, t_{min}, and t_{max}. Parameter t_0 defines the default radius (or merging threshold) for a new cluster before any adaptation takes place. Setting a very small default radius for clusters will first result in a cloud of small clusters that are later merged to form larger clusters when space densities are updated. Therefore, a small default radius maximizes the quantization accuracy but makes the process computationally expensive. Larger radiuses lead to a smaller number of larger clusters, which leads to increased resolution only if these clusters become large enough. A drawback here is that a cluster may become created to a location that is between two important input categories that should be distinguished from each other, subsuming input from both of these categories to itself. Since the algorithm does not currently support cluster splitting, it may be so that the large cluster will still gather tokens from both categories even after their radius is reduced, even though also new clusters will form in the neighborhood that gather some of the spectrally close variants.

The value of s_t determines how much data is collected before reorganization of the cluster space is performed. It is called a *sleep threshold*, since reorganization of the space resembles neural reorganization occurring in human brain while sleeping when external inputs are suppressed. Again, larger values lead to increased accuracy since probability that structures of the input are well represented in the gathered data increases. Use of larger values simultaneously increases memory requirements and computational complexity.

Finally, parameters t_{min} and t_{max} define the radius range of clusters, radiuses being computed using the equations (2.4) and (2.5). In other words, this ultimately defines the maximum resolution that becomes allocated to the densest part of the space and also the minimum resolution that becomes allocated to the sparsest part of the space.

2.4.3 Performance of SLVQ

The algorithm has been tested in word learning experiments with ACORNS Y1 and Y2 corpora in association with the concept matrix algorithm. The SLVQ algorithm performs relatively well compared to k-means quantization with Y1 material, both quantization methods leading to 100% word recognition rates with the concept matrix approach (section 2.5). Confidence margins to second best word-hypotheses are at approximately the same level with both quantization approaches. For the Y2 material, no direct comparison with k-means quantization has been performed yet.

One of the nice properties of SLVQ is that it performs very well in classifying silent portions of speech under a single label despite the fact that nearly half of the signal timeline (Y1 corpora) is silence. This seems to be due to the adaptive cluster threshold: because the (MFCC) feature vectors from silence frames are very distinct from any other speech content due to large (negative) values in coefficients c_0 and c_1, and therefore that part of the cluster space (dominated by low c_0 and c_1) is relatively sparse, the cluster weight factor w_j becomes small and usually only one major cluster centroid is formed for silence. This is somewhat different from the *k*-

means approach, where several clusters are often assigned for modeling silence if there is a sufficient amount of small variation in silence frames and silence is not limited by, e.g., a silence/speech detection algorithm.

A problem with SLVQ, as with other known clustering methods, is that the number of clusters is still controlled directly or indirectly by the user. The quality of a codebook for a specific purpose depends on its size, and therefore SLVQ also requires adjustment and experimentation with the parameters (see previous sub-section) in order to obtain suitable results. However, the behavior of the algorithm is extremely non-linear in relation to the parameter values, and the number of obtained clusters seems to converge to a very similar range of 60-160 clusters in a large range of parameter values (English speech, Y2 corpus, +9000 utterances). Reasons behind this effect are partially unclear and require more studying.

2.5 State transition (concept) matrices

2.5.1 General view

In order to create statistical models for the words embedded in quantized time-series, concept matrices were developed in WP2 for robust transition based analysis. Instead of traditional automata based approaches (see problems with N-gram type of approaches, sections 2.1, 2.2, 2.6.2, and CMM, section 2.3; also cf., Hidden Markov Models), the concept matrix approach does not make the assumption that subsequent discrete elements in the time-series are statistically independent of each other (that is, no Markov property assumption), but actually builds statistical model for dependencies at different temporal ranges. This helps to overcome the problem of input variability and distortion, as the input can be compared to internal representations in probabilistic terms over larger temporal windows and the best matching, previously learned, internal representation is chosen from the long-term memory as an explanation for the input. With a limited vocabulary, the system can learn quickly to recognize words that are sufficiently similar to training exemplars.

The concept-matrix approach shares some similarities with non-negative matrix factorization (NMF), most importantly in that it also tracks recurring discrete units that are separated by a number of non-defined units in between. However, the way that structures from these dependencies are discovered differs in many aspects. Transitional probabilities from discrete elements to others at a specific temporal distance are stored in concept related statistical models that associate time-series statistics with multimodal information (tags). The statistical model can be normalized in a way that cumulates several sources of structural information in order to have maximal information value when contrasted against other concepts.

2.5.2 Concept matrix algorithm

The concept matrix algorithm is a general-purpose pattern-discovery algorithm for discrete time-series and other data types that can be expressed as discrete sequences (e.g., images). Therefore the core of the algorithm is presented here as a generalized case, even though it is used for speech processing and especially for word learning in ACORNS.

Input to the system consists of a time series of discrete elements or spatial information sampled to form 1D-sequences, and in the training phase, tags specifying some event associated with the sequences. In some cases one information modality may provide a tag for another modality. The basic elements of the sequences are called *labels*. In the simplest case they may refer to items in a vector quantization codebook, or they can be produced by any kind of discretization of time-series or images. In a more complex case they may refer to some higher-level representation of information, e.g., events or items possibly reflecting clear qualitative properties. The other information source (possibly another modality source) is represented by a set of so-called *concept tags* *c*.

Tags are integer values that represent invariant outputs of another process that are being associated to the time-series input (e.g., a categorization process performed in another modality like visual or haptic perception in case of speech recognition, or some other group of manually defined events that should be associated with the time-series).

The mechanism may work also in the opposite direction; an acoustic event may serve as a tag to learn visual patterns. One modality may form tags to other modalities to help the learning. More generally, the method allows construction of statistical associations between different modalities. This is one of the key issues on the way to model and understand the formation and learning of *meanings* (by agents and humans).

When a concept is activated and a sequence represented, the algorithm starts to collect frequency data of the occurrences of label pairs in the sequence at a distance l. This data is stored in a histogram table or a matrix (**T**). The original labels can be used as pointers to **T** when the number of occurrences of the corresponding label pair is needed. The histogram collected in **T** is then used to produce another representation **P**.

The backbone of the algorithm is a matrix $\mathbf{P}_{\mathbf{l,c}}$ of size N_q x N_q, where N_q is size of the codebook, that resembles transition probability matrices but does not contain well-defined probabilities but cumulative probability sums instead (that is, $\sum p \neq 1$). The matrix keeps record of normalized transition probabilities from label a[t-l] to label a[t] in input sequence a when an external information source, called concept c, is activated, where $l,c,t \in \mathbb{Z}$, and l is a member of set $\boldsymbol{l} = \{l_1, l_2, l_3,\ldots, l_n\}$ and c is a member of set $\boldsymbol{c} = \{1, 2, 3,\ldots,N_c\}$. In other words, N_c is the total number of concepts introduced to the system. If we define $N_l = \|\boldsymbol{l}\|$, there are a total of $N_P = N_l * N_c$ instances of **P** matrices, one for each concept at a specific lag. In addition, there is a matrix $\mathbf{T}_{\mathbf{l,c}}$ that is otherwise similar to $\mathbf{P}_{\mathbf{l,c}}$ except that it keeps record of the transition frequencies instead of normalized probabilities from label a[t-l] to label a[t] in the presence of concept c.

Since values of **P** are not classical probabilities in the range [0, 1] due to a three-stage normalization process, values of P will be referred as activation values and **P** will be referred as activation matrix. Activation values stored in **P** will be computed by using the frequency information stored in **T**.

2.5.2.1 Training

For simplicity of the notation, elements of matrices $\mathbf{P}_{l,c}$ and $\mathbf{T}_{l,c}$ are denoted in the form $P(a_i,a_j|l,c)$ and $T(a_i,a_j|l,c)$, where the first two variables a_i and a_j define matrix element indices of the labels (transition from a_i to a_j or co-occurrence of a_i and a_j), whereas l defines the lag and c defines the concept.

The input consists of training sequences $\mathbf{S} = \{\boldsymbol{s_1},\boldsymbol{s_2},\ldots,\boldsymbol{s_n}\}$ and sequence related concepts $\mathbf{V} = \{\boldsymbol{v_1},\boldsymbol{v_2},\ldots,\boldsymbol{v_n}\}$, where each $\boldsymbol{v_i} = \{c_1,c_2,..,c_n\}$, $\boldsymbol{v} \in \boldsymbol{c}$. All transitions in the sequence $\mathbf{s_i}$ occurring at lags $\boldsymbol{l}$ are updated in the transition frequency matrices $\mathbf{T}_{l,\mathbf{c}}$, where c is a member of the $\boldsymbol{v_i}$ associated with $\boldsymbol{s_i}$**.** This process is repeated for all S in the training material. The following pseudo-code illustrates this process:

Table 2: A pseudo-code example illustrating the collection of transition frequencies.

```
for i = 1:length{S}
        s = S(i);
        v = V(i)
        for lag = 1:length(l)
                for t = 1:length(s)
                        for c = 1:length(v)
                                T(s[t-lag],s[t] | lag,c) = T(s[t-lag],s[t] | lag,c) +1;
                        end
                end
        end
end
```

After all transitions occurring in the training material are added to the transition frequency matrices **T**, the matrices are normalized to transition probability matrices P' by normalizing the transition probability from each label to all other labels ($\sum_x \Pr(a_i,a_x) = 1$) by having:

$$P'(a_i,a_j\,|\,l_d,c_k)=\frac{T(a_i,a_j\,|\,l_d,c_k)}{\sum_{x=1}^{N_q}T(a_i,a_x\,|\,l_d,c_k)} \tag{2.6}$$

where N_q is the codebook size, that is, the number of unique elements, in the time series. Next the probability that a specific transition occurs during the presence of a tag instead of all other transitions is added cumulatively to the $P'_{l,c}$:

$$P''(a_i,a_j\,|\,l_d,c_k)=P'(a_i,a_j\,|\,l_d,c_k)+\frac{T(a_i,a_j\,|\,l_d,c_k)}{\sum_{x=1}^{N_q}\sum_{y=1}^{N_q}T(a_x,a_y\,|\,l_d,c_k)} \tag{2.7}$$

This enhances the value of those transitions that are very common in presence of the concept. It should be noted that the matrix is now no longer a well-defined transition probability matrix in a sense that the next state probabilities do not sum up to one. Therefore values of **P** are from now on referred as (concept specific) activation values and outcomes of the recognition process are called concept activations.

Finally, the probability that a transition occurs during the presence of a concept c_k instead of any other concepts is incorporated in the final activation matrix **P** by having:

$$P(a_i,a_j\,|\,l_d,c_k)=\frac{P''(a_i,a_j\,|\,l_d,c_k)}{\sum_{z=1}^{N_c}P''(a_i,a_j\,|\,l_d,c_z)}-\frac{1}{N_c} \tag{2.8}$$

In other words, the cumulative probability of a transition from a_i to a_j in case of the tag c is divided by the sum of probabilities of the same transition occurring during all possible tags $\boldsymbol{c}$. If a transition becomes equally probable for all concepts, therefore containing no information value, it would have a probability of $1/N_c$. Therefore each element in all matrices has $1/N_c$ subtracted from its original value to have zero activation for fully random case and negative value for transitions that occur more often during other concepts.

2.5.2.2 Recognition

Activation level of a concept c_i at time t given input sequence $\boldsymbol{s}$ can be expressed as:

$$A(c_i,t)=\sum_{d=1}^{N_l}P(s[t-l_d],s[t]\,|\,l_d,c_i) \tag{2.9}$$

when only backward history of the input sequence is included. It is also possible to have a bidirectional recognition process by including $P(s[t],s[t+l]|l_d,c_i)$ activation values to the sum in equation (2.9) if the next labels up to largest lag max($\boldsymbol{l}$) in the sequence are known in advance. This enhances localization of the recognized event, as the peak value of the activation curve becomes centered to a point where there is most statistical support for the specific concept, distributed symmetrically around that point in terms of transitional probabilities.

Equation (2.9) provides a local activation estimate for each concept candidate but in many applications it is useful to examine the activation output in a larger temporal window since the events that are being recognized spread over several subsequent time frames. One possibility to do this is to first low-pass or median filter the activation curves in a larger temporal window. Then each of these concept-related temporal activation curves is searched for a subsequence of length $L_i \in [L_{min}, L_{max}]$ having a maximum cumulative sum of activation values. After these subsequences are found for each concept model c, the subsequence i with highest cumulative sum defines the concept hypothesis c_i.

The L_{min} sets a minimum temporal limit for the information that is included in the recognition process and should be at least as long as the shortest possible event being recognized. Similarly, L_{max} defines a temporal upper limit for information integration and should be at least as long as the longest possible event being recognized. However, having even larger values for L_{max} may be beneficial in several situation, as the context of an event contains often cues to the event itself and the statistics embedded in transition probability matrices take this information into account.

As the reader may have already noticed, it is also possible to run the entire algorithm in parallel for several synchronously quantized input streams in order to incorporate several sources of information. This transforms frequency and activation matrices into form $T_\psi(a_i, a_j \mid l, c)$ and $P_\psi(a_i, a_j \mid l, c)$, where ψ denotes the number of the input stream being processed. Training is performed similarly to the single stream condition in order to build separate concept matrices for each concept at each lag and for each stream. In the testing phase probability output from all streams is combined to have a probability of a concept c_i at time t of:

$$A(c_i, t) = \sum_{\psi=1}^{\|\psi\|} \left(\sum_{d=1}^{N_l} P_\psi(s[t-l_d], s[t] \mid l_d, c_i) \right) * \omega_\psi \qquad (2.10)$$

where ω_ψ is a weighting factor defined for each input stream.

2.5.3 ART – Association Response Table

One of the latest developments at TKK is to apply state transition probability matrices (or so called concept matrices) in order to discover similar structures in the quantized input stream as has been learned before in association with a visual (tagged) input. In the following we give a short description of the principle and the first, preliminary results obtained.

Every keyword has its own recognizer (see section 2.5.2.1 for training of the models) running in parallel with the others. Previously learned statistical structures, or internal representations, stored in each recognizer are being activated with the current index sequence occurring in the input. The output activity of the recognizer is high when the input sequence fits well its internal model. The activation is computed at every index and is called association response. The collection of all these responses creates a table called *Association Response Table* (ART).

Figure 2.18 gives an example of the ART representation of the Finnish sentence “*Hän antaa likaisen lehmän*”.

The present sentence includes three keywords and all produce a strong association with the learned model. Because words “*antaa*” and “*likainen*” share two first syllables of word “*talitintti*”, the talitintti-recognizer reacts also briefly. Note that the recognizers are trained for all different inflectional forms of the words occurring in the training data instead of base forms. This can be seen in, e.g., activation in the end of word “likainen” (baseform) which is pronounced “likai<u>sen</u>” in this context, where underlined part “…*sen*” is shared between several nouns and adjectives trained in the same inflectional form. The keyword probability frame (at the bottom) indicates clearly that with a high probability the sentence has four strong activations. However, after taking temporal overlap of activations into account with proper inhibition, only the correct keywords remain activated.

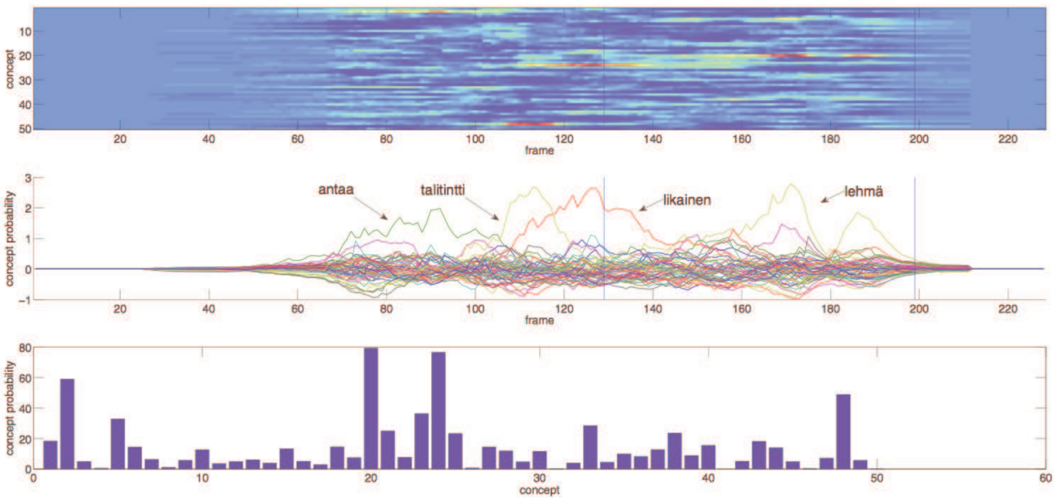

Figure 2.18: ART-representation of the sentence "*Hän antaa likaisen lehmän*". X-axes: time indices in steps of 10ms, y-axes: outputs of fifty keyword recognizers trained to the system. As can be seen, the keywords "*antaa*" "*likainen*" and "*lehmä*" are all reacting strongly to the utterance. Also word "*talitintti*" gains brief activation due to phonetic similarity with the end of the first and the beginning of the second word.

The activation curves have remarkable similarity to, e.g., activation processes occurring in well-known models of speech perception, e.g., TRACE [1] and Merge [7]. In this case the temporal order of activation processing is not taken into account and activated units do not inhibit other units before overall activity of all recognizers is computed. This means that recognition consists of internal representation (or lexical) competition where bottom-up (feed-forward) connections define the activation of the concepts stored in long-term memory. Ideally this leads to simultaneous activation of several words that share some syllables present in the input, no matter in what part of the words that syllable is located. In the next processing step, temporal integration of activations leads to inhibition of all but that word model that receives most activation in a larger temporal scale. Preliminary results indicate that temporal integration of 250 ms seems to give best results in terms of keyword recognition rate

100 % correct recognition has been achieved with this method by using UK-Y1 corpus (four speakers in random order, total 4000 utterances) and WP2 incremental SLVQ indices obtained every 10 ms. The results with Y2-corpus are also very promising. Preliminary experiments indicate over 94 % correct recognition rate with multiple speakers. It also seems that the algorithm is capable to produce very accurate word segmentation as long as the subsequent words are all known to the system. Segmentation to syllable-size units could be also possible, but the material used in the experiments is too limited for obtaining enough syllable pairs occurring in different keywords. Figure 2.19 displays characteristic learning curves for the process for UK-Y1 corpus with multiple speakers and Y2 FIN corpus with a single speaker.

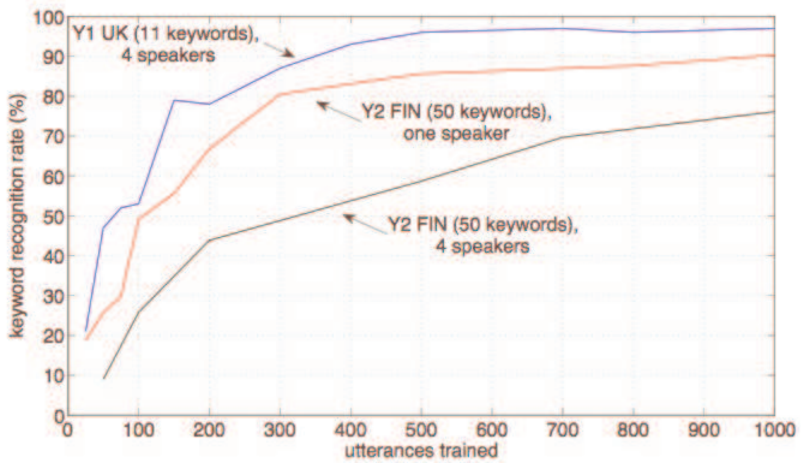

Figure 2.19: Learning curve for Y1 UK corpus with 11 different keywords by 4 speakers (blue) and Y2 FIN corpus with 50 different keywords and single speaker (red). Results are shown up to first 1000 training utterances.

2.5.4 Discussion and future work

The concept matrix approach is still a relatively new discovery and the literature does not seem to have comparable approaches to this type of associative structural description of time-series. It is distinct from classical automata and HMM thinking where a continuous path is being searched for through all elements in the series. The implementation is also different from neural networks, although both capture a sort of fuzzy statistical description for regularities in the input. The concept matrix method is not a black box but a relatively easily analyzable system where internal structures are explicitly available to direct observation.

The results with the Y1 and preliminary experiments with the Y2 corpus have been very promising. However, the same major drawback still exists that hinders all other reviewed approaches to pattern discovery and what concept matrices were partially intended to overcome: variability of the input means that this variability has to be captured in the statistics of each word model before recognition can be performed reliably. This restricts instance-based learning, since the algorithm is unable to generalize from a single prototype to all possible variations. As this variability problem originates from the quantization where produced labels change due to spectral changes in speech, overcoming this problem would require some sort of restructuring of the quantization outputs based on some criteria, e.g., linguistic level contextual information. Approaches for this type of clustering, or re-organizing, of bottom-up created codebooks are currently being explored in WP2 and the first findings are somewhat promising.

It should be also noted that the concept matrix approach requires that a new matrix is created for each new keyword occurring in the input. Therefore, while the transition analysis in combination with SLVQ may be justified as a cognitively plausible symbolic interpretation of an associative process with subsequent neural activations and Hebbian learning, as an implementation on digital Von Neumann computers it requires relatively large amounts of memory. With, e.g., codebook of 150 labels, vocabulary of 500 concepts stored with 8 byte floating-points, modeling of only single step transitions would already take $150^2*500*8 = 90$ megabytes. However, sparseness of the matrices and pruning of statistical information can be utilized in order to compress the representations without loss in recognition accuracy.

Due to its recent discovery, the concept matrix approach is still under intensive development and testing. Therefore the experimental results that were presented here are few and still preliminary. One topic in the future research will be to find ways to further accelerate the learning speed of the models so that the required amount of training data becomes less. In addition to changes in the algorithm itself, it may also require further studying of the properties of representations used in the pattern discovery process.

2.6 Other experiments and findings

2.6.1 Segmental dynamic programming (SDP)

In order to detect recurring units from sequences of segmental labels provided by SLVQ and blind segmentation, a segmental based utterance alignment method was explored. It shares many properties with DP-ngrams (section 2.2) but it uses distances between spectral centroids of segment-like units instead of distances between pure spectral vectors at signal level. The idea was to detect and align the same word occurring in two different utterances, or to collect smaller recurring combinations of phone-like units from speech material and use this for recoding of the original signal.

2.6.1.1 The algorithm

Pre-processing:

1. Speech material is segmented into phone-like units
2. SLVQ cluster space is created from these segments.
3. Speech signals are quantized into series of phone-like unit labels.
4. Spectral distances between labels are computed using cluster centroid information.

Segmental dynamic programming:

5. A distance matrix **D** is computed for a pair of utterances A and B having the same multimodal tag. Each element $\mathbf{D}(a_i,b_j)$ represents the spectral distance from the centroid of phone-like unit a_i in utterance A to the centroid of unit b_j in utterance B.
6. Distance matrix **D** is filtered with a non-linear filter to enhance salience of temporally continuous alignments (figure 2.20).
7. Distance matrix **D** is sliced into diagonal bands of width *w* and each band is searched for optimal (minimum distance) path using DTW (similarly to approach in [14]). The slice and the corresponding path yielding minimal overall distance is chosen as the alignment (figure 2.21; note that effective length of a slice has to be compensated in overall path distance computation since corners of the matrix have naturally shorter diagonals).
8. The sub-path of minimum length L_{min} and maximum length L_{max} having smallest cumulative distance is selected and corresponding sub-sequences S_A and S_B from utterances A and B are extracted.
9. Sequences S_A and S_B are stored in a sequence library.

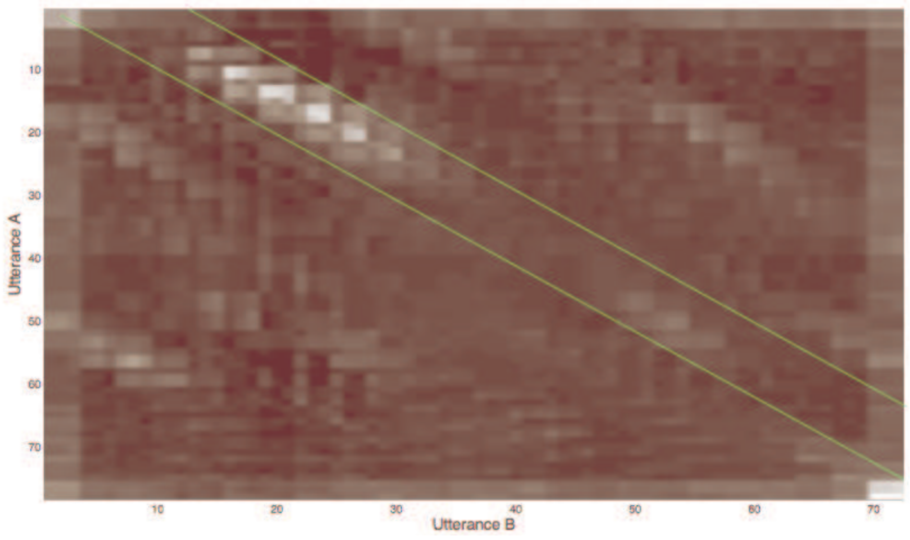

Figure 2.20: A distance matrix between two utterances after filtering. The same keyword occurring in both utterances can be clearly seen as a continuous good match along the diagonal direction. Green lines indicate slice boundaries.

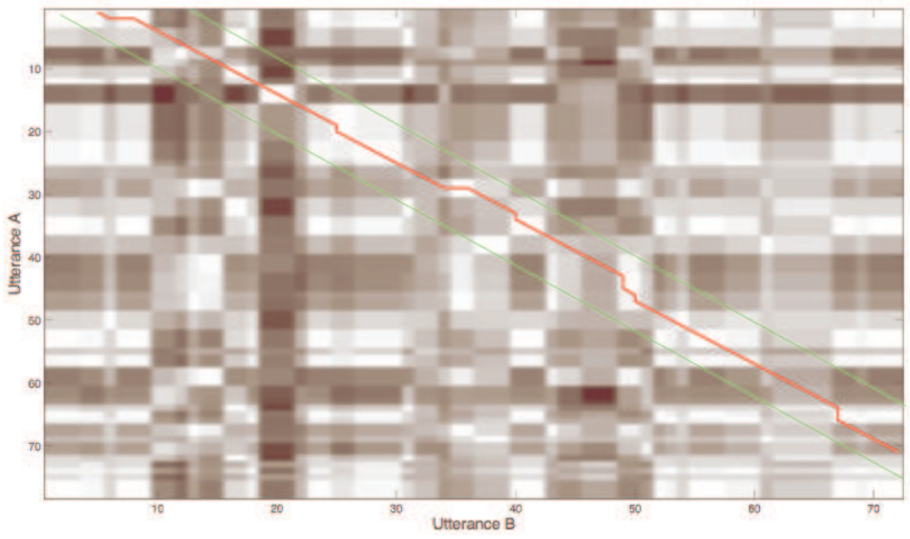

Figure 2.21: Segmental distance matrix of two utterances and the obtained best possible path aligning these two (red) in the best matrix slice (green boundaries).

2.6.1.2 Findings

The algorithm was briefly tested on detecting both word-like units and sub-word units. As the word-like units were collected to a list specified by multimodal tags, they could be used for word recognition. Attempts to re-align these word-like units in order to build a number of prototypical models for each keyword did not work out very well, since mutual distances of sequences were dominated by a sort of CVCV-type structure in which well matching segments (high energy, well defined spectrum) were followed by poorly-matching segments (low energy). Without the use of temporal knowledge about segments or more detailed information about acoustic landmarks, the alignment, and therefore building a prototypical spectral and/or spectral representation based on this alignment, led to poor recognition rates. By computing a mean distance from novel input to all entries in each keyword library led to more promising results but was found to be computationally (and cognitively) implausible.

One of the major flaws of the approach was that the segmental level resolution caused problems in the alignment because of the possible insertions and deletions in different realizations of the same words. Also, as the number of labels for each segment was increased from one to two, the alignments become significantly more accurate. In all cases, a possibility for misaligning two utterances with a similar keyword, but e.g., different speaker seemed to be relatively high as the algorithm picked up small portions of similarity from several parts of the utterances. Also the variability of the labels in different utterances caused combinatorial explosion (also discussed in sections 2.1, 2.2, and 2.6.2) of recurring units of different lengths even when the speech material (Y1 UK corpus) was relatively simple and contained a very limited amount of shared syllables between different keywords. The alignment process was also found to be computationally expensive even though segmental sequences are very compact as such. Slicing of the matrix and detection of the optimal path through each slice is computationally expensive, and the number of possible utterance pairs with, e.g., 4000 utterances and 11 keywords is already almost 1.5 million if only keywords sharing the same tag are aligned. This also raised questions about cognitive plausibility, since the algorithm needs to have all those several hundreds of entire utterances (as segmental sequences) stored in the memory for pair-wise comparison. Many of these issues have been addressed in the DP-ngram approach in section 2.2 of this paper.

Development of this method was discontinued in the early autumn 2009 due to superior performance of the concept matrix approach. This does not indicate, however, that the method would have been found to be conclusively poor for the word or sub-word unit learning task, but more like that the quality and detail of segmental representations and linguistic simplicity of the ACORNS Y1 corpus were found to be insufficiently rich for building reliable and extensive library of sub-word units for representation of the speech signal.

2.6.2 One more N-gram experiment

In September 2008 the TKK team performed one more N-gram experiment based on a “brute force” method where the VQ-indices of the first 300 sentences of the UK-Y1 corpus where searched for identical N-grams with $2 \le N \le 9$. Among others the following aspects were studied:

1° Size of the N-gram library when the number of sentences increased from 150 to 300 in steps of 50 sentences.
2° Histogram of the N-grams in the library.
3° Modeling of the keywords based on the N-gram library.
4° Usability of the keyword models in recognition.

In short, the simulations did not reveal any fundamentally new insights related to the usage of the N-grams in keyword recognition. Almost linear increase in the number of different N-grams in the library at each step of 50 sentences indicates that the VQ-index data with alphabet size of 150 is quite noisy which makes the derivation of a compact library very difficult or even impossible. This outcome reflects the general results obtained by the CMM-method where the number of causal states continues to increase due to the large variation in the VQ-index sequences.

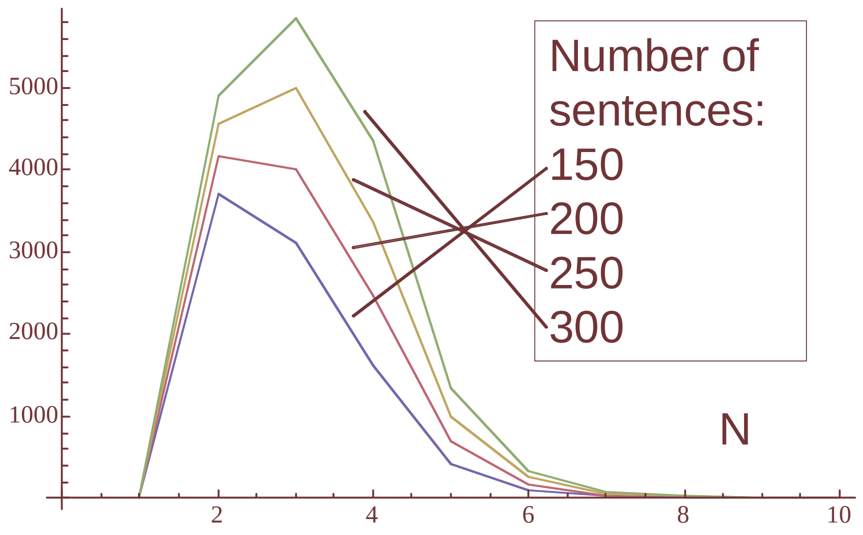

Figure 2.22: Number of different N-grams of different length (N) found in the 150 … 300 first sentences of the UK-Y1 corpus (VQ-indices obtained by k-means method).

The analyzed material consisted of over 34,000 VQ indices obtained by the k-means method. The total number of different N-grams of length 2 … 9 which occurred at least twice (also called primitives) was 11,597. The noisiness of the sequences is indicated by the fact that only few identical 9-grams were found.

One important view obtained with this analysis is that when new sequences flow in, the number of different N-grams grows constantly. There is always a large number of N-grams which have occurred only once or two times. There will probably never be data enough to construct sufficient statistical models for all these N-grams. There is a constant conflict between the need for new data and the increasing number of different N-grams.

The created N-gram library contains also knowledge about in which sentences and at which positions in the sentences each N-gram is located. This allows, e.g., studying of those N-grams located only in the sentences of the same keyword (same tag). We call these n-grams *pure primitives*. They form a special set of N-grams able to effectively discriminate between different keywords.

The simulation gave some hope that it is possible to model the keywords based on their elements (primitives) and the way they are organized, and to derive a system able to recognize many keywords in the same sentence. The recognition tests gave promising results, which in their turn activated the design of a novel concept matrix based architecture presently under active research and documentation (see; 2.5.2).

The results of the simulation helped to formulate also the preliminary view on the problem of multiple keywords presented in section 1.6.

3. General conclusions

Several different approaches for pattern discovery in speech have been undertaken in ACORNS. They were briefly reviewed in this report. These activities are summarized in the following table:

Table 3: ACORNS pattern discovery activities.

	Multigram	**DP-ngram**	**CMM/CSSR**	**SLVQ+Segm.**	**STM**
Activities, experiments	Presently inactive.	In progress.	In progress.	In progress.	Word recognition in combination with SLVQ.
Positive observations	Good performance on TIDIGITS using HMM acoustic model.	Promising results. Alignment already at signal level.	Learns states and deterministic structure.	Provides incrementally learning quantization.	Learns statistical structures that can be used for pattern discovery in speech.
Problems found	High number of word candidates with VQ data. Sensitivity to preset parameters. Not flexible.	Speaker dependent representations.	Basic method not robust against noise: does not converge to finite set of states with real speech input.	Codebook size depends on parameters. Segmental signal descriptions form too sparse statistics due to spectral variability.	Cannot generalize from exemplars.
Future plans	On hold state.	Improved front-end and repre-sentations. Sleeping mode. Attention mechanism.	Experiments with resolution.	Development and experiments continue alongside STM.	Refinement of the method. Transformation of signal representations using statistical information embedded in STM.
Other observations & notes	Hierarchical learning process with phone-like units needed.	Studies on learning in multi-lingual en-vironment coming	Comparison with related techniques.		Computationally extremely fast but requires much memory.

The pattern discovery studies made in ACORNS reflects the main methods and tools in the field. We could classify these in two groups, one utilizes *connected discrete elements* as the basic representation, the other *statistical dependencies or associations* between pairs of elements found at different distances. The first group of methods tries to model the sequence with fixed subsequences (connected elements) and their statistics, meanwhile the other methods try to describe the *structural properties* of the sequence (or sub-segments of it) by general statistical methods.

We could summarize these two lines by the following table:

Table 4: The two classes of pattern discovery methods studied in ACORNS

Method	**Connected Discrete Elements (CDE)**	**Structural Description by Statistical Associations (SA)**
Name	Multigrams N-grams DP-ngrams CMM/CSSR	NMF (base vectors) STM (transition matrices)
Theory	Markov Chain, Automata-theory	Relative Frequencies
Word Model	Relatively exact and rigid. Often lossless coding allows exact reconstruction.	Flexible. Leads to lossy compression. Exact reconstruction not possible.
Problems	Each keyword model needs a large amount of different sub-sequences that are difficult to compress and represent in a compact form. Sometimes memory and statistics explosion. Pruning needed.	Transition matrices need quite much memory if common units for the words are not extracted and modeled separately. Fuzzy structural descriptions.
Positive Observations	Provides common units for all word models.	Noise robust. Learns all statistically meaningful structures. Systematic.
Other Notes	**Both approaches** in their present forms have to learn to cope with the noise inherited from the quantization process due to spectral variability in speech in order to create flexible and noise tolerant word models. **Some** CDE methods may provide a connection to SA by coding the input sequence to another, higher-level representation for further processing. **Common Problem:** How to detect and code *the most relevant structures* out of the diverse set of input variations to create higher-level *generalized*, *invariant representations*.	

In addition to the pattern discovery algorithms themselves, a major cornerstone in pattern discovery from speech input seems to be the representation of the speech signals for different levels of processing, and is therefore a worthy topic for discussion. It seems that quantization of speech signals based on (linear) distances between spectral frames is unable to combine sufficient resolution to distinguish different phones and simultaneous robustness to variability inside phone classes. Speaker and context-dependent variability combined with different speaking rates (different number of frames for similar events in different situations) and possibly some other distortion sources (e.g., background noise) leads to quantized sequences that very rarely if ever re-occur in an exactly similar form, not even at a phone level, not to mention syllable or word levels.

Due to the noise and variability in the label sequences, it seems to be inefficient to apply exact sequential models where each word is represented by a fixed order deterministic series of elements. When the same sequential structure occurs very rarely in the exact same form, this hinders the matching between previous occurrences and new input. Different phonetic variations of a same word are often coded as separate sequences or sets of sub-sequences, and later recognition of these varying forms of the word requires that they are incorporated in the word models despite the fact that some of them may occur relatively rarely compared to the more common realizations. This also makes pruning of the models difficult, as these variations are extremely difficult to separate from non-meaningful noise items.

Models based on more fuzzy statistical description of co-occurrences of different events (labels) at different temporal delays cope better with large amounts of insertions, deletions and substitutions in the input sequences – as long as important details are not blurred in the statistical representation. This is because they attempt to determine the most likely explanation (association) for a novel input by integrating information from larger temporal windows into internal representations, instead of being dependent on well- defined transitions between adjacent labels. For example, the concept matrices provide associations over any reasonable number of elements in the sequence and in this way they are able to build important statistical bridges over corrupted and noisy sections. Therefore, they provide a much more generalized statistical picture of the sequence than a Markov model can produce since the latter is limited by the Markov property. Moreover, *speech is not markovian on any of its representational levels that range* from low-level signal samples up to words in sentences. The reason why Markov models are so widely applied to speech is that they considerably simplify the treatment of statistics. The problems caused by the Markov property limitation is partially solved by equipping the model with higher layers, e.g., language model, able to handle larger contexts. However, also these processes are limited by the Markov property.

The latest results obtained with the concept matrix method show clearly how important it is to model efficiently and accurately the strong statistical dependences (causal relations) in speech over wider contexts (time spans). The usage of the Markov property is a relatively strong limitation, which leads to a remarkable loss of useful structural information.

It is also of great importance that a common problem for all approaches reviewed in this report is the non-existent ability to generalize from a handful of word tokens to all possible realizations of the same word due to the variability in input discussed above. Without a link between all possible realizations of a same event, e.g., originating from a specific articulatory gesture, exemplar based learning is impossible and different variations of each keyword have to be trained separately into each word model instead of training only different variations of each speech sound category to a general system that then transforms the signal representations to a more invariant level. Ideally, signal quantization or representational transformation leading to phoneme-like categories would make the task of linguistic pattern discovery much easier, as the non-informative variability of the input would reduce significantly and the problem would turn into more traditional natural language processing task.

While the mapping from each variant to a separate unique lexical representation is something that has received behavioral evidence for babies under age of 8 months [36, 37], a more invariant representational level is needed to enable the fast growth of the lexicon, so-called word spurt, witnessed in babies during the second year of their lives. Findings in experimental psychology suggests that children are somehow able to acquire these native phonemic categories before efficient and rapid learning of words begins (see, e.g., [2] for a review), but this still remains as an unsolved problem in computational modeling although some promising results utilizing multimodal information have been achieved (e.g., [15]). Children are capable of tracking systematical behavior of speech sounds to a level where meaningful differences can be distinguished It is likely that temporal context of speech sounds and multimodal information plays important part in this discovery of refined linguistically meaningful units, and the pattern discovery methods developed so far in ACORNS are actually also modeling these aspects of speech. Therefore the change in learning strategy due to changes in internal signal representations is something that hopefully can be addressed in the future with the help of the ongoing pattern discovery approaches. More invariant representation level may also open new possibilities for the methods like CMM and multigrams currently suffering from the variability at the signal level.

References:

[1] McLelland J. L. & Elman J. L.: *The TRACE Model of Speech Perception*. Cognitive Psychology, Vol. 18, pp. 1-86, 1986.

[2] Kuhl P. K.: *Early language acquisition: Cracking the speech code*. Nature Reviews Neuroscience, Vol. 5, No. 11, pp. 831-843, 2004.

[3] Räsänen O. J.: *Speech Segmentation and Clustering Methods for a New Speech Recognition Architecture*. Master's thesis, Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing http://lib.tkk.fi/Dipl/2007/urn010123.pdf, 2007.

[4] Deligne S. & Bimbot F.: *Inference of variable-length linguistic and acoustic units by multigrams.* Speech Communication, Vol. 23, pp. 223-241, 1997

[5] Nowell P. & Moore R. K.: *The Application of Dynamic Programming Techniques to Non-Word Based Topic Spotting*. In Proc. Eurospeech '95, pp. 1355-1358, 1995.

[6] Sankoff D. & Kruskal J. B.: *Finding similar portions of two sequences*. In Sankoff D. & Kruskal J. B. (eds.) *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley Publishing Company, pp. 293-296, 1983.

[7] Norris D., McQueen J. M. & Cutler A.: *Merging information in speech recognition: Feedback is never necessary*. Behavioral and Brain Sciences, Vol. 23, pp. 299-325, 2000.

[8] Gärdenfors P.: *How Homo Became Sapiens: On the Evolution of Thinking*. Oxford University Press, 2003.

[9] Piaget J.: *Psychology of Intelligence*. Littlefield, Adams, and Company, 1966.

[10] Laver J.: *Principles of Phonetics*. Cambridge University Press, Cambridge, 1994.

[11] Rolls E. T.: *Invariant object and face recognition*. In Chalupa L. M. & Werner J. S. (eds.) *The Visual Neurosciences*. MIT Press, Cambridge, 2004.

[12] Dietrich E. & Markman A. B.: *Discrete thoughts: Why cognition must use discrete representations*. Mind and Language, Vol. 18, pp. 95-119, 2003.

[13] Brent M. R.: *Speech Segmentation and Word Discovery: A Computational Perspective*. Trends in Cognitive Sciences, Vol. 3, No. 8, pp. 294-301, 1999.

[14] Park A. & Glass J. R.: *Towards Unsupervised Pattern Discovery in Speech*. IEEE Workshop on Automatic Speech Recognition and Understanding 2005, pp. 53-58, 2005.

[15] Coen M.: *Self-supervised Acquisition of Vowels in American English*. In Proceedings of the Twenty First National Conference on Artificial Intelligence (AAAI'06). Boston, MA, 2006.

[16] Driesen J. & Van hamme H.: *Improving the multigram algorithm by using lattices as input*. In Proc. Interspeech 2008, pp. 2086-2089, 2008.

[17] Dromi E.: *Early Lexical Development*. Cambridge, MA: Cambridge University Press, 1987.

[18] Carey S.: *The child as word learner*. In Halle M., Bresnan J. & Miller A. (eds.): Linguistic theory and psychological reality, pp. 264-293. Cambridge, MA: MIT Press, 1978.

[19] Shalizi C. & Crutchfield J.: *Computational Mechanics: Pattern and Prediction, Structure and Simplicity*. Journal of Statistical Physics, Vol. 104, pp. 816–879, 2001.

[20] Shalizi C.: *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. Ph.D. thesis, University of Wisconsin, Madison, 2001.

[21] Shalizi C. & Shalizi K.: *Blind construction of optimal nonlinear recursive predictors for discrete sequences*. In Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference, pp. 504–511, 2004.

[22] Shalizi C., Klinkner K. & Crutchfield J.: *An Algorithm for Pattern Discovery in Time Series*. Santa Fe Institute Working Paper 02-10-060, 2006.

[23] Ron D., Singer Y. & Tishby N.: *The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length*. Machine Learning, Vol. 25, pp. 117–149, 1996.

[24] Kennel M. & Mees A.: *Context-tree modeling of observed symbolic dynamics*. Physical Review E, Vol. 66, pp. 056209.1-056209.11, 2002.

[25] Shalizi C.: *CSSR: An Algorithm for Building Markov Models from Time Series*. Web page, accessed 2008-11-21 at http://www.cscs.umich.edu/~crshalizi/CSSR/.

[26] Padró M. & Padró L.: *A Named Entity Recognition System Based on a Finite Automata Acquisition Algorithm*. Procesamiento del Lenguaje Natural, Vol. 35, pp. 319–326, 2005.

[27] Schmiedekamp M., Subbu A. & Phoha S.: *The Clustered Causal State Algorithm: Efficient Pattern Discovery for Lossy Data-Compression Applications*. Computing in Science and Engineering, Vol. 8, pp. 59–67, 2006.

[28] Gavalda R., Keller P., Pineau J. & Precup D.: *PAC-Learning of Markov Models with Hidden State*. In Proc. Machine Learning: ECML 2006, 17th European Conference on Machine Learning, pp. 150–161, 2006.

[29] Holmes M. and Isbell C. Jr.: *Looping suffix tree-based inference of partially observable hidden state*. In Proc. 23rd international conference on Machine learning, pp. 409–416, 2006.

[30] Shalizi C.: *Methods and Techniques of Complex Systems Science: An Overview*. In: T. Deisboeck and J.Y. Kresh (eds.), *Complex Systems Science in Biomedicine*, pp. 33–114, New-York: Springer-Verlag, 2006.

[31] Duda R. O., Hart P. E. & Stork G. E.: *Pattern classification*. John Wiley & Sons Inc, 2001

[32] Grenander U.: *General Pattern Theory: A Mathematical Study of Regular Structures*. Oxford University Press, 1994

[33] van Leuuwen C.: *Perception*. In Bechtel W. & Graham G. (eds.): *A Companion to Cognitive Science*. Blackwell publishing, Oxford, UK, 1998

[34] Bailly F. & Longo G.: *Space, Time and Cognition – From The Standpoint of Mathematics and Natural Science*. In Peruzzi A. (ed.): *Mind and Causality*, pp. 149-199, John Benjamins, Amsterdam, 2004

[35] Bar M., Kassam K. S., Ghuman A. S., Boshyan J., Schmidt A. M., Dale A. M., Hamalainen M. S., Marinkovic K., Schacter D. L., Rosen B. R. & Halgren E.: *Top-down facilitation of visual recognition*. Proceedings of the National Academy of Science, Vol. 103, pp. 449-454, 2006

[36] Singh L., White K. S. & Morgan J. L.: *Building a word-form lexicon in the face of variable input: Influences of pitch and amplitude on early spoken word recognition*. Language Learning and Development, Vol. 4, pp. 157-178, 2008

[37] Houston D. M. & Jusczyk P. W.: *The role of talker specific information in word segmentation by infants*. Journal of Experimental Psychology: Human Perception and Performance, Vol. 26, pp. 1570-1582, 2000