4-2005

Keyword Spotting Using Normalization of Posterior Probability Confidence Measures

Rachna Vijay Vargiya

Keyword Spotting Using Normalization of Posterior Probability Confidence Measures

by
Rachna Vijay Vargiya

Bachelor of Engineering
Computer Science and Engineering
Maulana Azad College of Technology
2002

A thesis
submitted to the Graduate School of
Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

Melbourne, Florida
April 2005

We the undersigned committee hereby recommends that the attached document be accepted as fulfilling in part the requirements for the degree of Master of Science in Computer Science.

"Keyword Spotting Using Normalization of Posterior Probability Confidence Measures "
a thesis by Rachna Vijay Vargiya

Marius C. Silaghi, Ph.D.
Assistant Professor, Computer Science
Major Advisor

, Ph.D.
Associate Professor, Computer Science
Committee Member

, Ph.D.
Associate Professor, Department
Committee Member

William D. Shoaff, Ph.D.
Associate Professor and Program Chair
Computer Science

ABSTRACT

Keyword Spotting Using Normalization of Posterior Probability Confidence Measures

by

Rachna Vijay Vargiya

Thesis Advisor: Marius C. Silaghi, Ph.D.

Keyword spotting techniques deal with recognition of predefined vocabulary keywords from a voice stream. This research uses HMM based keyword spotting algorithms for this purpose. The three most important componenets of a keyword detection system are confidence measure, pruning technique and evaluation of results. We suggest that best match for a keyword would be an alignment in which all constituent states have high emission probabilities. Therefore score of even the worst subsequence must also be better than a thershold and a path can be represented by the score of its worst subsequence match. This confidence measure is called Real Fitting. The harsher the pruning in a technique, the fewer paths survive. This increases the speed as well as the risk of pruning the best match. Three levels of pruning are explored and results and performance are compared. Since the proposed algorithms do no follow

the principle of optimality,possible optimal paths could be pruned. Therefore we propse a pruning that permits a set of paths to be retained in every frame instead of a single path. Finally the evaluation of these algorithms plays an important role in assessing the performance of these algorithms. since the choice of keywords can affect the results, an equal opportunity evaluation is proposed which evaluates the algorithms on their respective best keywords.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Speech recognition is the process of matching voice pattern to a vocabulary. Successful speech recognition systems which can be used as voice dialing for telephone systems, speech driven word processors or communication aid for the disabled, are a field of interest in today's world due to the multifarious areas of applicability. This chapter provides an introduction to the field of speech recognition and describes our goal and motivation.

1.1 Speech Recognition

Speech recognition can be broadly classified into three categories.

- Isolated word recognition: deals with recognizing predefined keywords when spoken in isolation. The word being recognized is neither preceded

nor followed by any other speech/sound.

- Keyword Spotting: This category also deals with recognizing predefined keywords. However, in this case the keyword is recognized in continuous speech (could be preceded or followed by other speech/sounds).

- Continuous Speech Recognition: This category deals with recognizing entire sequence[s] of words from a continuous utterance. It is usually followed by a syntactic and semantic analysis.

Isolated word recognition is the simplest of the three tasks. It is a quick and inexpensive form of speech recognition. However it has limited applicability since isolated occurance of words is rare. Continuous speech recognition, on the other hand, is a complex and computationally intensive task that requires a large amount of training. High computational complexity makes it hard to achieve and more expensive to implement.

Many applications have a requirement that lies between the two approaches described above. They do not require entire sequence of words to be recognized. These applications are covered by keyword spotting. Examples of applications using keyword spotting are: classification of speech messages, word-based commands etc.

Speech recognition systems are also classified based on the allowed number of speakers as follows:

- Speaker dependent recognizers can recognize words uttered only by a single speaker. The model is trained and can perform recognition only for that speaker.
- Multiple speaker recognizer can successfully recognize words spoken by predetermined set of multiple speakers.
- Speaker independent recognizers can recognize words spoken by any person. Training as well as recognition is independent of the speaker.

Clearly, speaker independent recognizers have the most applicability amongst all three categories. Such a system could perform recognition without the any user specific training.

1.2 Problem Statement

This research focusses on speaker independent keyword spotting. We propose novel keyword spotting algorithms to achieve a high rate of recognition at considerably low number of false recognitions.

It is also observed that the choice of keywords influences the performance of keyword spotting algorithms. Certain words are recognized more easily than

other words making the selection of keywords a difficult task. The properties that make words "good" or "bad" keywords are still vague. Comparison of two or more techniques on a set of keywords may be inaccurate if the keywords favor one algorithm. This research also investigates an evaluation which resolves this limitation.

1.3 Contributions

The major contributions of this research are summarised as follows.

- A novel confidence measure called Real Fitting is proposed. The measures results in high detection rates, outperforming existing confidence measures.

- Two different kinds of pruning are investigated for confidence measures: double normalization and real fitting. A novel pruning technique not only yields better results but also is more efficient.

- Three different prunings are investigated for Real Fitting. It is observed that with relaxation of pruning the performance of real fitting improves significantly.

- Limitations of standard evaluation of keyword spotting techniques are highlighted. A novel evaluation, equal opportunity evaluation, is pro-

posed.

- A few properties observed in the list of keywords being recognized easily are put forth.

1.4 Organisation of Thesis

The next chapter focusses on the different concepts and phases of speech recognition integral to the understanding of this thesis. It also discusses the models used for representation of speech in our experiments. The following chapter presents the different keyword spotting techniques – existing and novel techniques proposed in this research. The fourth chapter provides experimental results and their analysis using a novel evaluation. The final chapter presents our conclusions and the contributions of this research.

Chapter 2

Speech Recognition Concepts

In this chapter some relevant concepts in the field of speech recognition are presented. It describes the structure of speech, models used to represent keywords and gives an introduction to the important phases in speech recognition.

2.1 Keyword Modeling

It is essential to model speech for the purpose of recognition. Speech is a time-varying signal [1]. The selected represention must be able to model this attribute. Following sections describe a model which can model time-varying data, and hence is suitable for representing speech. The model is called Hidden Markov Model.

[1] HMM based recognizers assume that the speech signal over very small time frames is stationary.

2.1.1 Hidden Markov Models

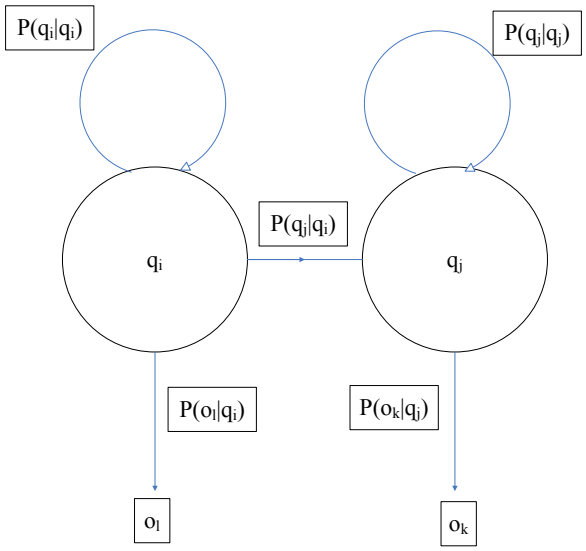

Figure 2.1: Example of a Hidden Markov Model

Hidden Markov Models, also referred to as HMMs, are variants of finite state machines. An HMM consists of a set of states, a set of output alphabet, transition probabilites and output probabilities. Transitions from one state to another are associated with transition probabilities. Every state can generate an output governed by a probability called the output probability. Formally, a HMM is defined by the triple (λ, B, Π) where,

- $\lambda = P(q_j^{t+1}|q_i^t),\ q_i,\ q_j\ \epsilon\ Q,$

  λ is the set of transition probabilities

  Q is the set of states in the HMM model

  t is the time ($t >= 1$).

- $B = P(o_l|q_i),\ o_l\ \epsilon\ O$

  B is the set of output probabilities

  O is the set of output alphabet

- $\Pi = P(q_0)$

  q_0 is the initial state.

Figure 2.1 shows a simple HMM with two states q_i and q_j generating the outputs o_l and o_k respectively. The states of an HMM can be divided into two categories: emitting and non emitting states. States that emit a phoneme are called emitting states while those that do not emit a phoneme are called non-emitting states. The entry and exit states are usually the non emitting states in an HMM. The outputs of emitting states as well as the transition among states are governed by probabilities ($P(q_j|q_i)$, $P(q_i|q_j)$ and $P(o_l|q_i)$, $P(o_k|q_j)$). The name of *hidden markov models* is derived from the fact that in an HMM, only the output sequence, O, is observable to us and not the underlying state sequence. Since the same output sequence (observation) may be produced by different underlying sequence of states, predicting the state sequence from an

output sequence (an utterance) is a bayesian inference. A word can be spoken differently by people with different accents, ages etc.

2.1.2 Words and Phonemes

Speech has a hierarchical structure. It consists of sentences which are composed of words. Each word can further be broken down into smaller units of sound, called *phonemes*[2]. Phonemes are the atomic units of sound. All words can be represented as a concatenation of some of these phonemes. For example, the keyword *abaisser* may be represented by the set of phonemes *a b E s e*. Since the number of words in a vocabulary can be much larger than the number of phonemes, modeling phonemes is simpler. Models of these phonemes can then be used to model words and hence model speech. Thus the first step in modeling speech is modeling phonemes.

2.1.3 Modeling Phonemes using HMM

We represent each phoneme by a single HMM. The HMM of a word is formed by the concatenation of the HMMs of its constituent phonemes.

[2]Since each sound may be spoken differently by individuals, a family of sounds which are considered the same in a language are represented by one phoneme.

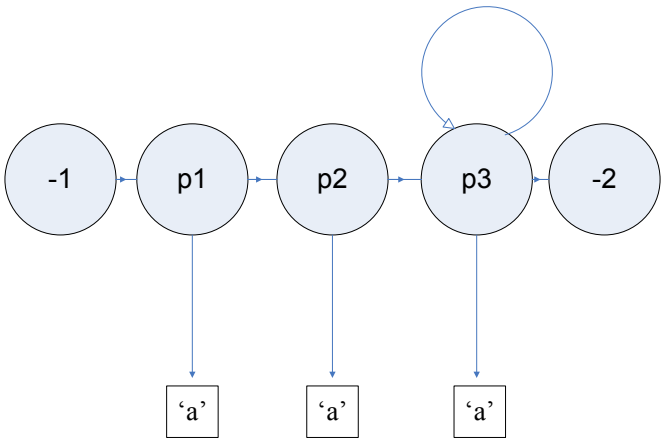

Figure 2.2: Hidden Markov Model for a single phoneme ’a’

For the database in this research, BREF [5] , the HMM of a phoneme has a simple left to right topology, as shown in Figure 2.2. This means the transitions are permitted to take place only from left to right. The selected model has three emitting states and two non emitting states. The three emitting states are included to impose a lower bound on the time required to pronounce a phoneme (a person can not speak faster than a limit imposed by nature). It is impossible to get to the final state without passing through at least three states

emitting the phoneme. Since there is no upper bound on the maximum number of times a phoneme can be emitted(there is no limit on how slowly a person may speak), the third phoneme is allowed to make a transition to itself which allows indefinite stay in the phoneme. The non emitting states are the entry and exit states. They are included for the purpose of concatenation of multiple HMMs. An example of the HMM of a phoneme 'a' is shown in Figure 2.2. All the three emitting states, $p_1,\ p_2,\ p_3$, emit the same phoneme 'a'. The last emitting state, p_3, can make a transition to itself so that the model can represent both *aaa* and *aaaaaaaaa*. Entry and exit states are represented by -1 and -2 respectively.

2.1.4 Modeling Words using HMM

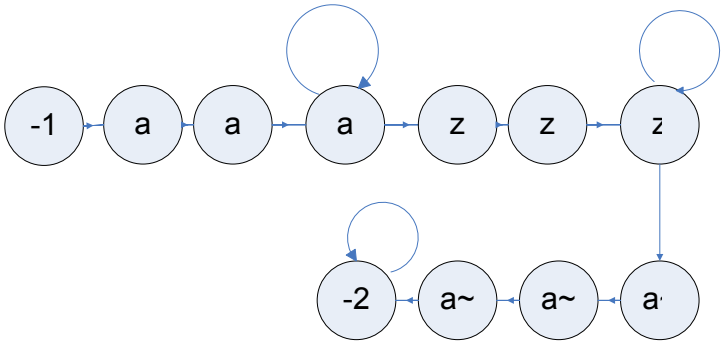

Figure 2.3: Hidden Markov Model for a keyword 'agent'

Word HMMs are formed by simple concatenation of phoneme models as shown in Figure 2.3. The entry and exit states of all but the first and the last HMM are replaced by the HMM of the phoneme preceding and following the current phoneme respectively. For the first and the last phoneme HMM, only the exit and entry states are replaced respectively. As an example, the french keyword *agent* is composed of the phonemes *a Z a* . The HMM of the keyword is formed by concatenating the HMMs of individual phonemes as shown in Figure 2.3. The models for phonemes 'a', 'Z', and 'a ' are concatenated by replacing the appropriate entry and exit states.

2.2 Feature Extraction

For the analysis of an utterance, an effective and compact representation of the speech signal is required. Feature Vectors are one such representation. The first component of a speech recognition system consists of a front end signal processor that can carry out such a conversion of speech signal into a representation that can be interpretted by a speech recognition algorithm, namely feature vectors. Each feature vector represents a speech spectrum of a short duration obtained by sampling the speech signal. This phase is called Feature Extraction. The feature vectors represent the signal for further processing.

2.3 Decoding

The goal of a keyword spotting algorithm is to search among a set of keywords, the keyword with the highest probability of producing a given utterance. To summarize two major topics discussed so far:

- As described in Section 2.2, a speech signal is converted to a set of feature vectors or a set of observations $X = x_1, x_2, \ldots x_n$.
- In an HMM (keyword model), only the output sequence is visible while the underlying state sequence is hidden.

Decoding is the process of searching the state sequence in a model (representing a keyword) most likely to produce the observation. I.e. given an observed utterance (represented by feature vectors), X, decoding is the process of computing the probability of a keyword (represented by an HMM), W, to produce the utterance, i.e $P(W|X)$. However, direct estimation of $P(W|X)$ is difficult in practice. So one decomposes the problem using Bayes' rule as below.

$$argmax_W P(W|X) = argmax_W \frac{P(W)P(X|W)}{P(X)} \tag{2.1}$$

$P(W)$ is the prior probability of observing the word independent of the signal. It is determined by the *language model*. The remaining part of the equation (2.1), $P(X|W)/P(X)$, denotes the probability of observing a signal (or feature vectors) given the occurance of a keyword. This probability is determined by the

acoustic model. These probabilities are calculated during the training phase. Decoding uses the language and acoustic models to search the most probable state sequence of a keyword model that produced the output sequence, W. Most probable state sequence of an HMM would be the one which would maximize equation (2.1). Decoding, like other search techniques has depth-first and breadth first approaches.

Decoding using depth-first search pursues one path (a state sequence) at a time. The most promising state sequence is explored first. The most common techniques using depth-first search are *stack-decoding* and *A*-decoding*. These techniques are more commonly used for continuous speech recognition.

Breadth first decoding pursues the search of all candidate paths (state sequences) simultaneously. Most common breadth-first decoding is *Viterbi decoding*.

Viterbi is a dynamic programming algorithm which exploits Bellman's optimality principle to pursue the candidate state sequences. An utterance is said to be produced by a model i if,

$$P(X|M_i) \; > \; P(X|M_j) \;\; \forall j \; \neq \; i \tag{2.2}$$

where X = utterance,

M_k = Model representing the word k

Viterbi being a time synchronous algorithm is particularly suited for time dependent speech signal. The computation complexity for Viterbi is linearly proportional to the length of the utterance and the number of states in the model. However, Viterbi cannot be used directly for the purpose of keyword recognition.

Details are discussed in the following chapters.

2.4 Confidence Measures

Speech recognition systems take as input an actual input utterance and output a sequence of words which produced the input utterance. The correctness of the system is nothing but the correspondance between the output of the recognizer and the input sequence. The goal of these systems is to maximize this correspondance. However, none of the existing systems have been able to achieve absolute correctness and there is a degree of uncertainty associated with the correctness of the result. It is, therefore, necessary to have a measure to indicate how well the output corresponds with the input sequence to affirm that the result of the system is correct or incorrect. This measure is called confidence

measure.

Chapter 3

Keyword Spotting Techniques

Continuous speech recognition requires complete decoding of speech signal and it outputs a completely decoded sentence. Keyword spotting only requires to output whether a keyword is present in a signal or not. This makes keyword spotting different from continuous speech recognition. Keywords are embedded in extraneous speech and may begin and end at any instant in the utterance making keyword spotting a non trivial task and as discussed in the previous chapter, Viterbi can not be applied directly to a keyword spotting system. This chapter describes in Section 3.1, some of the existing approaches that have been adopted to achieve keyword spotting. Some of the novel approaches in keyword spotting proposed by this report are described in Section 3.2.

3.1 Related Work

The existing work done in keyword spotting can be categorized under two major approaches. The first approach is the filler or garbage models approach. In this approache, all words other than the keywords assumed to be garbage and are represented by preassigned models called garbage or filler models. The second approach is the sliding model approach. This approach computes for every begin and corresponding end frame, the probability of the keyword occurance. Details of the two approaches are discussed in the following sections.

3.1.1 Garbage/Filler Models

For a keyword spotting algorithm, the keyword to be recognized is embedded in extraneous speech. Garbage model handles this extraneous speech by explicitly modeling the extraneous speech by non-keyword models. The input speech signal is represented by a sequence of extraneous speech embedded with vocabulary keywords. Assuming *keyword* is in the vocabulary, it is represented by the keyword model and the rest of the utterance is represented by filler (or garbage) models. Every keyword has its own unique model.

Garbage models for keyword spotting are used by Higgins and Wohlford in [3]. They used a template based connected speech recognition system with a set of "filler" templates. The system outputs a continuous stream of keyword and

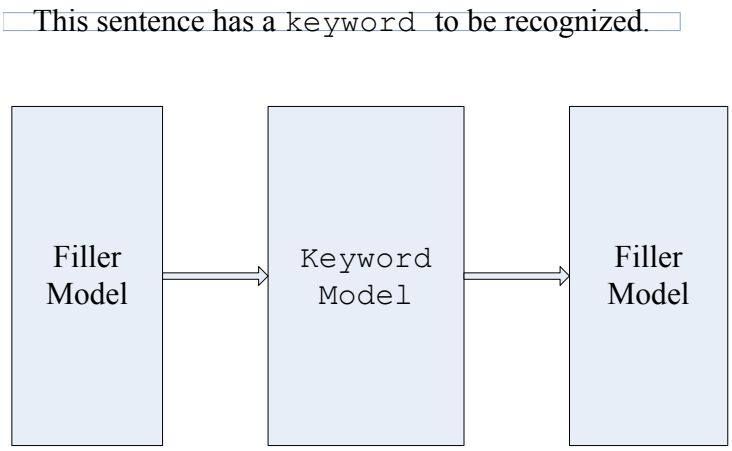

Figure 3.1: Keyword Spotting using Garbage Models

filler templates. The keyword template in this stream represents a putative hit. Rose and Paul in [7] also use a filler model approach. However they use HMM based filler models, filler models in combination with likelihood ratio scoring.

A syntax-derived, frame synchronous, network search algorithm is used in [11] to match frames against the stored HMMs. The syntax allows n number of

garbage models followed by the model of a keyword to be recognized, followed by another sequence of m garbage models, where $n, m >= 0$. Their algorithm achieves a recognition rate of about 95 percent. However, their vocabulary size was limited to five words. Also, their grammar was constrained to find exactly *one* keyword per utterance since they had prior knowledge that only one vocabulary words appears in any utterance. This paper proposes techniques which are free of any such assumptions. The number of keyword occurances is unconstrained and the vocabulary size is much larger.

3.1.2 Sliding Window Models

Dynamic programming techiques for keyword spotting, which are the basis of sliding window techniques, were originally proposed by Bridle [2]. Each dynamic path is considered as a possible keyword occurrence. Overlapping occurrences are removed and scores are normalized in the second stage. Thresholds are then established to detect true hits.

Unlike garbage models, sliding window models do not require explicit modeling of extraneous speech. Instead, a Viterbi alignment is computed beginning every frame (since a keyword may be present at any instant in the utterance). This may be interpreted as sliding a keyword model across the input speech signal to identify possible matches. The scores of all alignments, beginning

and ending in all frames, are computed. The confidence measure of a keyword model for a speech signal is the score of its best alignment. If an alignment of a keyword mode whose confidence measure indicates high probability of the keyword being present in the utterance for a keyword model exists, the keyword is assumed to be present.

The input to the algorithm is a list of probabilities for all states in each time frame. These probabilities are the emission probabilities of each state in the keyword HMM. A trellis of these probabilities is constructed. For every begin frame i, all candidate alignments are explored. The score of an alignment,$a(i, j)$, is computed at every frame j, from the score at frame $j-1$, $a(i, j-1)$. The local score(likelihood measure) at frame j is added to the score $a(i, j-1)$ to obtain $a(i, j)$.

Figure 3.2 shows an example trellis with nine frames. Four possible alignments are shown in the Figure: P_1, P_4, beginning in the first frame and P_2, P_3 beginning in the second frame. Such alignments are computed for each frame, $x_1, x_2, \ldots, x_n$. The best of these alignments is retained and the score of this alignment is assigned to the keyword model. The scores are a function of the product of emission probabilities of the states in the path (state sequence). This may cause the algorithm to be biased toward shorter alignments. Therefore it is important that the scores be normalized at the end. An obvious normalization

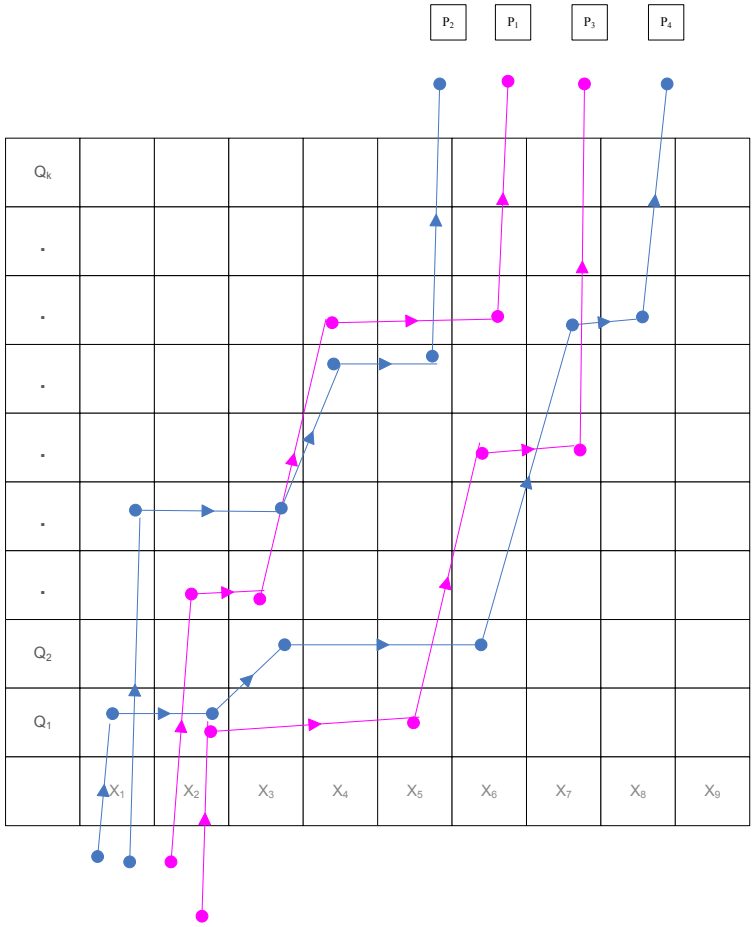

Figure 3.2: Keyword Spotting using Sliding Window Technique

factor would be the length of the path. Length of a path is the total number of frames or states in it. For example, the path P_1 traverses five frames (the path has five states), hence its length is five. The score of this path is normalized at the end by its length, which is five.

Wilpon et al [10] use sliding window model technique, as described above. For the five vocabulary keywords, *Collect, Calling-card, Third-number, Person, Operator*, a recognition rate of 86.1 percent is reported.

Junkawitsch et al also use a similar approach in [4]. Details of this aproach and some interesting work done by Silaghi and Bourlard in [8] are described later in Section 3.2.1.

The disadvantage of sliding window models is that the algorithm takes longer to run since all alignments, beginning every frame, are considered with little pruning.

3.2 Proposed Techniques

Both Garbage/Filler models and Sliding Window models suffer from certain limitations. While the former technique requires explicit modeling of extraneous speech, latter is computationally expensive. We present alternative one-pass technique that:

1) runs faster due to better pruning strategies and

2)does not require filler models. It does not require scoring for all possible begin end points, hence space and time complexities are lower. It also achieves higher recognition rates for larger vocabularies.

3.2.1 One Pass Algorithm

One pass algorithm searches the most probable alignment of a keyword HMM in a speech signal. Every path (state sequence), is associated with a score or confidence measure. This confidence measure represents the probability of the alignment to have produced the utterance. The alignment with highest confidence measures is retained while the rest are pruned. Proposed confidence measures are discussed in 3.2.2 and 3.2.3.

The algorithm is similar to sliding window technique. A trellis of emission probabilities of all states in the hypothesized keyword HMM at all frames is constructed as shown in Figure 3.3. At any frame, N new paths are permitted to enter the trellis(since a keyword may begin at any frame), where N is the number of distinct states to which transition from the entry state (-1) is possible. Each of these N paths enters one of the distinct states. Score for a new path entering a frame is equal to the local score of the state since a new path bears no history (the score of the path so far is 0). All existing paths (which entered the trellis at some previous frame) do bear a history and have a nonzero score accumulated so far. In every frame, paths ending in states emitting the same phoneme are compared. Only the best path per phoneme (set of states emitting the same phoneme) is retained while the rest are pruned. This means that at any time, total number of live paths in the algorithm is no more than the total

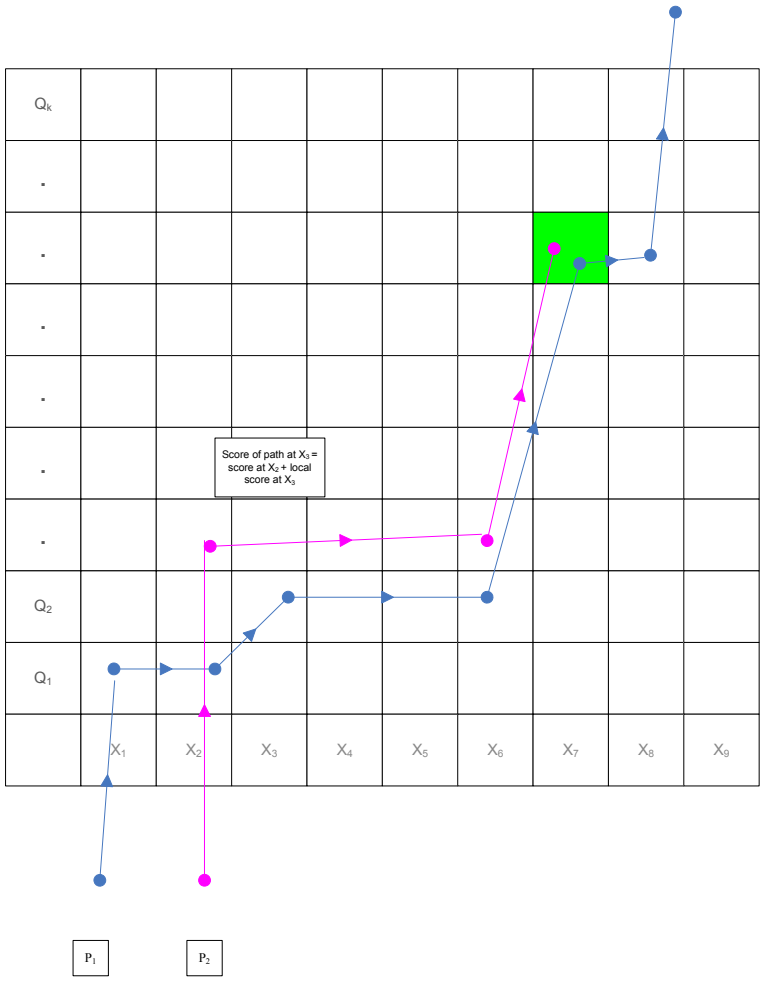

Figure 3.3: One Pass Algorithm

number of phonemes (states emitting the same phoneme can be considered same). This keeps the space requirement of the algorithm limited. Every path propagates from a frame say n, to the subsequent frame, $n+1$, by making a transition. The score of the path is updated by accumulating the local score at frame $n+1$ into the score of the path till frame n. Finally, a set of paths that

were not pruned remain. Among those, the best alignment (one with the best score or confidence measure) is reported as the most probable alignment of the model.

Figure 3.3 illustrates the comparison between two candidate paths P_1 and P_2 for a keyword model. Each path traverses the trellis updating its score at each frame. At frame seven, they enter the same state. Assuming the score of path $P_1 >$ score of path P_2, the latter gets pruned, leaving P_1 as the surviving alignment.

One pass algorithms are considerably faster than the sliding window model technique because all paths entering the trellis are evaluated in the same pass of the algorithm. The need to repeatedly compute alignments beginning every frame is eliminated.

A significant issue in keyword spotting is the choice of the confidence measures in a detection by a path in the HMM. If the confidence measure is poor, the most likely paths may get pruned leading to poor results. Effective confidence measures resulting in high detection rates are proposed in the following sections.

Sliding window approach described in the previous sections has been used by Junkawitsch et. al in [4] and Wilpon in [10].The difference between their approaches is in the way local scores are computed. [4] compute the difference

between the negative logarithm of the emission probability of a state and the value of the best state for that frame and assign it to the local score. Like [10] alignments are permitted to start at any frame, and the score of a path entering a state in frame j is computed by adding the local score of the state in the current frame j to the score of the path in frame $j-1$. Again, scores of paths of different lengths are normalized by the length of the paths, that is the number of frames the path has traversed. Junkawitsch et al used keyword specific thresholds and trained context dependent as well as context independent HMMs on 12 hours of speech data. The experiments were performed on twenty five keywords and obtained the Figure of merit of 58.5 percent for context independent and 73.9 percent for context dependent HMMs. The maximum detection was 80 percent at the rate of 10 false detections/keyword/hour for context dependent HMMs.

An interesting keyword spotting approach without using filler models is presented by Silaghi and Bourlard in [8]. They investigate the confidence measure of Double Normalization which is described and extended in this report. Their paper proposes Iterative Viterbi Decoding which aims at mathematically computing the probability of a state being a garbage state given an utterance i.e. $P(q_G|x_n)$ instead of using explicit garbage models. IVD achieves about 95

3.2.2 Double Normalization

As described in Section 2.4, confidence measures are an indicator of the probability of a path (or state sequence or an alignment) producing an utterance, and are therefore used for the comparison of different candidate paths. Among the paths being compared, one with better confidence measure is retained while the other is pruned. As mentioned before, a path is allowed to enter at any frame and it may make a transition to a state producing the same phoneme any number of times. Since one pass algorithm compares paths entering in different frames, paths in the same state may differ in their length[1]. This difference in length can cause a bias in the scores of the paths, favoring paths that are shorter (that entered later in the trellis).

As a path enters a state in the trellis, the emission probability of the state in the current frame is added to the confidence measure of the path. Since probability is always less than one, this lowers the confidence measure of longer paths. A good normalization technique is thus required to counter this bias in confidence measures. Wilpon et al explore one such normalization in [10].

We propose the confidence measure of Double Normalization which proposes normalizing the score twice. The fact that emphasis on a phoneme may cause a

[1]length of the path is equal to the number of frames the path has traversed

path to remain in one state[2] gives rise to the first normalization. This normalization is done over the length of the phoneme. Visiting large number of distinct phonemes may also lower the confidence measure of a longer path. This gives rise to the second normalization, which is over the number of distinct phonemes visited by the path. The resulting confidence measure is given as:

$$\frac{1}{NVP} \sum_{p_i \epsilon VP} \frac{\sum_{pst_i \epsilon PST} - \log(pst_i)}{length(p_i)} \tag{3.1}$$

where, NVP: the number of phonemes visited by the path

p_i: a phoneme in the path

VP: the set of all the phonemes

pst_i: the emission probability of the phoneme p_i in the current frame

PST: set of emission probabilities of all the phonemes in VP

length(p_i): returns the length of the phoneme (number of consecutive states emitting this phoneme)

One pass algorithm using Double Normalization is presented in Algorithm 1. Γ represents the set of all frames[3] in the trellis (or utterance), Π is the set of all candidate paths or alignments of the keyword model. A phoneme emitted by a state s is represented by $P[s]$.

[2]States represent emitted phonemes

[3]a frame is a period of 10ms in the utterance

For every frame in the utterance, candidate paths of the keyword model are computed by exploring transitions from the paths in the previous frame as explained in Section 3.2.1. A path ending in a state σ in frame $\tau - 1$ could make a transition in frame τ to a state emitting the same phoneme as shown in step 0.1 or to a state emitting a different phoneme as shown in step 0.2; the score of the path is updated by accumulating the score of current state accordingly. All paths ending in states emitting the same phoneme in the current frame are compared. Only the best path per state is retained as shown in step 0.3.

Based on the criteria of comparing two paths, two versions of Double Normalization were studied: Double Normalization 1 (DN1) and Double Normalization 2 (DN2). Both of these vary only in the criteria that must be met for two paths to be compared and thus pruned. Details are discussed below.

Double Normalization 1

As discussed before, the hmm of a word is formed by the concatenation of the hmms of its constituent phonemes. Consider a word which consists of phonemes *a, b, z* in the order *a b a z*. The HMM of the word would consist of the following states:

$$-1, A_1, A_2, A_3, B_1, B_2, B_3, A_1, A_2, A_3, Z_1, Z_2, Z_3, -2$$

```
for each time frame τ ∈ Γ do
    for each path ρ ∈ Π do
        for each possible transition to state σ from the current state ς do
            if (P[σ] == P[ς]) then
0.1             Increment length of phoneme P[σ];
                Update posterior pst for phoneme P[σ];
                Update score for path ρ;
            else
0.2             Set length of phoneme P[ς] to 1;
                Increment number of visited phonemes, NVP, for path ρ;
                Initialize posterior pst for phoneme P[ς];
                Update score for path ρ;
            for any path ϱ ∈ Π ending in state emitting P[ς] and frame τ do
                if (ρ < ϱ) then
0.3                 Add ρ to Π;
                    Delete ϱ from Π;
```

Algorithm 1: Double Normalization

where -1 and -2 are the non emitting entry and exit states respectively and $\{A_1,\ A_2,\ A_3\}$, $\{B_1,\ B_2,\ B_3\}$, $\{Z_1,\ Z_2,\ Z_3\}$ are the three emitting states in the HMMs of phonemes *a, b* and *z* respectively.

If HMM of each phoneme that is concatenated for the formation of the HMM of a word is considered a level, it can be said that the HMM in the example above consists of 4 levels: first level consisting of states A_1, A_2 and A_3, second level consisting of states B_1, B_2 and B_3, third level again consisting of the same states as the first level (A_1, A_2 and A_3) and finally the forth level consisting of states Z_1, Z_2 and Z_3. If an alignment of this HMM is being computed in an utterance, two candidate alignments being evaluated in the trellis could be:

$$-1\ A_1\ A_2\ A_3\ A_3\ A_3\ A_3\ A_3\ A_3\ B_1\ B_2\ B_3\ B_3\ A_1\ A_2$$

$$-1\ A_1\ A_2\ A_3\ A_3\ A_3\ A_3$$

As shown, path A has visited 16 states (or frames). If path A enters the trellis in frame 1, the path would currently end in frame 16. As path B consists of 6 states, if path B enters the trellis[4] in frame 10, last state of path B would also be in frame 16. Assuming this scenario, both paths A and B end in frame 16. Also, the last state in both the paths, A_2 and A_3 emit the same phoneme, *a*. As mentioned in Section 3.2.1, the algorithm compares paths ending in the

[4]A path is allowed to enter in any frame

states emitting the same phoneme in a frame and only the best is retained. This means that paths A and B should be compared and the inferior path must be pruned. However, based on the criteria of DN1, this is not so.

Based on the idea of levels in a word HMM, it can be said that alignment A is currently at the third level of the HMM while alignment B is at the first level of the HMM. DN1 allows paths to be compared only if their current states emit the same phoneme and are at the same level in the HMM. Because Paths A and B end at different levels in the HMM, DN1 does not allow the comparison between paths A and B even though they end in the same state (states emitting the same phoneme, a).

Double Normalization 2

The difference between DN2 and DN1 is in the constraints that need to be satisfied for two paths to be compared. As discussed above, for two paths to be compared, DN1 requires that not only the two paths end in the states emitting the same phoneme, but also are at the same level in the current frame. DN2 relaxes this constraint and allows the paths to be compared if the paths end in states emitting the same phoneme in a frame, irrespective of the level at which the alignments currently end. Taking the example from 3.2.2, paths A and B would be compared at frame 16, the inferior being pruned. This leads to a

harsher pruning in DN2, causing fewer paths to survive. The total number of paths in DN2 would never exceed the total number of unique phonemes in the word model. In DN1 it would never exceed the total number of phonemes in the word model (which would be greater than the former if a phoneme is repeated in the model).

3.2.3 Real Fitting

Like DN, in Real Fitting also every state is associated with its emission probability in every frame and the confidence measure is a function of the probability of all states in an alignment. The best alignment would be one in which all constituent states have high emission probabilities in their respective frames. This means that for the best alignment, every subsequence of the alignment must also be a good match and the score of even the worst subsequence must also be better than a thershold. Based on this fundamental, Real Fitting suggests that a path can be represented by the score of its worst subsequence match. Therefore, instead of a confidence measure based on the posterior of the entire state sequence, one based the posterior of the worst phoneme match subsequence should suffice. This confidence measure represents the worst subsequence alignment for the model. It can be represented mathematically as,

$$MAX_{p_i \epsilon VP} \frac{\sum_{p_i} - \log(pst_i)}{length(p_i)} \tag{3.2}$$

Where, p_i represents the phoneme with worst subsequence match

VP is the set of all the phonemes

pst_i is the emission probability of the phoneme p_i in the given frame

length(p_i) returns the length of the phoneme[5].

A one pass algorithm using real fitting confidence measure is presented in Algorithm 2. Γ represents the set of all frames in the trellis (or utterance), Π is the set of all candidate paths. A phoneme emitted by a state s is represented by $P[s]$. The algorithm is similar to Double Normalization: For every frame in the utterance, candidate paths of the keyword model are computed by exploring transitions from the paths in the previous frame. A path ending in a state σ in frame $\tau - 1$ could make a transition in frame τ to a state emitting the same phoneme as shown in step 0.1 or to a state emitting a different phoneme as shown in step 0.2; the score of the path is updated by accumulating the score of current state accordingly. All paths ending in states emitting the same phoneme in the current frame are compared, pruning the inferior path and retaining only the best path per state as shown in step 0.3.

The difference between one pass algorithm for RF and DN is when the score of a path is updated. As opposed to DN, the score of a path remains unchanged in RF as long as the path makes a transition to the same state (state emitting

[5]length of a phoneme is the total number of frames in which it occurs consecutively

the same phoneme, say $P[\sigma]$). When a transition to a state emitting a different phoneme is made, the average posterior of the set of consecutive states emitting the same phoneme, $P[\sigma]$ is computed. If the real fitting score of the subsequence of consecutive states emitting the same phoneme is lower than the current score of the path, this is the worst subsequence encountered so far. The score of the alignment is thus updated with the score of this subsequence. This way, score of an alignment is always equal to the score of its worst subsequence match found so far.

Like Double Normalization, different versions of pruning in Real Fitting are proposed, namely Real Fitting 1 (RF1), Real Fitting 2 (RF2) and Extended Real Fitting (XRF). The detailed description of each is presented below.

Real Fitting 1

The pruning in RF1 is similar to DF1. As described earlier under Double Normalization 1, if each HMM being concatenated to form a word HMM is considered a level, DN1 imposes the constraint that only those paths can be compared for pruning which end in states emitting the same phoneme and are at the same level in the word HMM. The same constraint applies in RF1. In the example from DN1, RF1 would also forbid paths A and B to be compared although they end in states emitting the same phoneme in a frame because they

for *each time frame* $\tau \in \Gamma$ **do**

  for *each path* $\rho \in \Pi$ **do**

    for *each possible transition to state* σ *from the current state* ς **do**

      if $(P[\sigma] == P[\varsigma])$ **then**

        Increment length of phoneme $P[\sigma]$;

        Update posterior *pst* for phoneme $P[\sigma]$;

        Update score for phoneme $P[\sigma]$;

      else

        Set length of phoneme $P[\varsigma]$ to 1;

        Increment number of visited phonemes, NVP, for path ρ;

        Initialize posterior *pst* for phoneme $P[\varsigma]$;

        Compute average posterior of the last phoneme $P[\sigma]$;

        if *(*$P[\sigma]$ *score* $<$ ρ *score)* **then**

          Set ρ score = $P[\sigma]$ score;

      for *any path* $\varrho \in \Pi$ *ending in state emitting* $P[\varsigma]$ *and frame* τ **do**

        if $(\rho < \varrho)$ **then**

1.1   Add ρ to Π;

          Delete ϱ from Π;

Algorithm 2: Real Fitting

are at different levels in the word HMM.

Real Fitting 2

Real Fitting 2 is different from Real Fitting 1, the same way that Double Normalization 2 differs from Double Normalization 1 - it allows paths ending in states emitting the same phoneme in a frame to be compared, irrespective of the level they are at in the word HMM. Thus Real Fitting 2 would allow paths A and B to be compared, pruning the inferior path and retaining the superior one. Like in DN1 and DN2, RF1 relaxes the pruning of paths as compared to RF2, allowing more paths to survive. Like DN, the total number of paths in RF2 would never exceed the total number of unique phonemes in the word model, whereas in RF1, it would never exceed the total number of phonemes in the word model (which would be more than the former if a phoneme is repeated in the model).

Extended Real Fitting

Real Fitting is not an optimal algorithm. Two paths are compared and the inferior path (one with a worse confidence measure) is pruned before the paths reach their final states. The proposed algorithms do no follow the principle of optimality. This could lead to pruning of possible optimal paths since there is no way to guarantee that the alignments measures would maintain the same

order of precedence of scores in future. Both RF1 and RF2 suffer from this limitation.

It has been described earlier that RF1 relaxes the pruning of paths from RF2 by allowing fewer paths to be compared. Extended Real Fitting (XRF) goes one step further from RF1 in the relaxation of path pruning. Extended Real Fitting does not retain just one path per phoneme in the trellis. Instead, it retains a set[6] of paths per phoneme satisfying some constraints independent of the other paths. We define:

- Φ: maximum acceptable values of the negative logarithm of the phoneme posterior(emission probability of state producing the phoneme)
- Λ: length of the phoneme
- φ: negative logarithm of posterior of a phoneme in the current frame for a path
- ϕ: cumulated posterior of a phoneme over frames in which the phoneme repeats itself consecutively
- λ: length of the phoneme or the number of frames over which the phoneme repeats itself consecutively

Consider a word with phonemes *a, b* and a candidate path:

[6]Our experiments retained 3 paths per trellis

$$-1, A_1, A_2, A_3, A_3, B_1, B_2, B_3$$

States A_1, A_2, A_3 and B_1, B_2, B_3 are the states emitting to phonemes a and b respectively. φ_i represents the negative logarithm of the emission probability of state i in the frame of occurrance. E.g. φ_1 is the negative logarithm of probability of state A_1 to emit phoneme a in frame 1. Every state is associated with an emission probability, φ in its frame of occurrance. ϕ is the cumulated posterior for the states emitting a phoneme over frames in which the phoneme appears consecutively . It can be defined as:

$$\phi = \sum_{i=1}^{N} \varphi_i$$

where states (1 ... N) emit the same phoneme, i.e. the phoneme for which the average posterior is being computed, say a. In our example, ϕ for $a = \varphi_1 + \varphi_2 + \varphi_3 + \varphi_4$

where: φ_k is the emission probability of state A_k in frame k.

This makes ϕ/λ is the average posterior for a phoneme.

Constraints imposed by XRF for retention of paths are as follows.

- $\varphi > \Lambda\Phi$
- $\phi/\lambda > \Phi$

The first constraint ensures that the emission probability of every state in a candidate path is greater than a maximum allowed value, $\Lambda\Phi$. The second

constraint verifies that the average posterior of every phoneme is also greater than a threshold, Φ. The values of Λ and Φ are predetermined.

Constraints imposed by XRF could be satisfied by a large number of paths for large values of Λ and Φ. This leaves a considerable number of paths in the algorithm unpruned, thus preventing the pruning of a possibly optimal path. Defining a maximum number of paths (MAX) in each state in a frame that can be permitted to exist at any time ensures that the number of paths does not grow too large thereby slowing the algorithms significantly. The best MAX paths, satisfying the constraints described above, are retained. The best paths are the ones with best confidence measure described in equation 3.2

Chapter 4

Experimental Evaluation

This chapter presents the evaluation of the keyword spotting algorithms discussed in Chapter 3. The first section is about the database used for conducting the experiments. The next section describes the existing evaluation criteria for the keyword spotting algorithms. Then we present the results and discuss the drawbacks of the evaluation procedure in Section 4.3. A procedure to overcome this drawback is proposed and the results using this procedure are presented in the final section.

4.1 Experimental Dataset

The algorithms proposed in this report were evaluated on a part of the BREF database. BREF is a large French speech corpus containing about 100 hours of

speech data gathered from about 120 speakers selected from Paris area. The text to be read was selected from a french newspaper *Le Monde*. It consists of 3300 short sentences and 3800 long sentences (12.4 words/sentence and 21 words/sentence respectively) with 35 distinct phones and 14,089 distinct words. About 46 percent speakers were male and 54 percent were female of ages ranging from 18–73 years. Details on the database can be found in [5].

We did our experiments on 242 sentences of the database using the same 35 distinct phones and 1221 distinct words.

4.2 Evaluation Criteria

The algorithms proposed in this report, like most other keyword spotting algorithms, output a confidence measure for each keyword for an utterance. This confidence measure indicates how strongly the algorithm believes the keyword is present in the utterance. A threshold is set and keywords with better[1] confidence measures than the threshold are considered present in the utterance. Thus the output of the algorithms for a keyword in an utterance is *true*(keyword is present) or *false* (keyword is not present).

[1]Since the confidence measures being evaluated are functions of the negative logarithm of the posterior probability, lower values of confidence measures mean higher probability and thus better confidence measures

We count *true positives*[2], *false positives*[3] and *false negatives*[4] for the evaluation of results. Receiver Operating Characteristic curves are plotted to compare and assess the performance of two or more algorithms.

4.2.1 Receiver Operating Characteristics

Receiver Operating Characteristics, ROC curves were initially used to interpret radio signals containing noise. They are used for evaluating decision making applications which produce a *true* or *false* result. Such results involve a threshold which determines whether the output should be true or false. If values lower than the threshold are considered true, more values will be classified as true with higher threshold. This increases the true positive rate as well as the false positive rate and lowers the false negative rate. The curve of true positives (TP) or false negatives (FN) versus false positives (FP) is called an ROC curve. Good results have points on the graph with high true positive rates or low false negative rates and low false positive rates. An ROC curve and thereby the performance of an application is evaluated by the area under the ROC curve. If the curve is plotted between FN and FP, it should resemble the curve shown in Figure 4.1

With low thresholds, the number of FN is higher while that of FP is lower.

[2]results correctly identified as true by the application

[3]results incorrectly identified as true (they were actually false)

[4]results incorrectly identified as false (they were actually true)

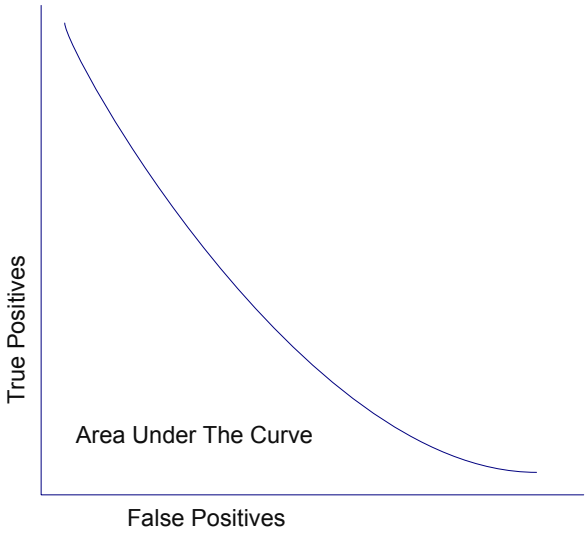

Figure 4.1: ROC curve made by plotting False Negatives versus False Positives

As the threshold is raised, FNs decrease while FPs increase. For an application with good results, there should be a point in the curve where both FNs and FPs are low and neither increases abruptly. This is represented by the area under the curve: lower the area under the curve, better the performance.

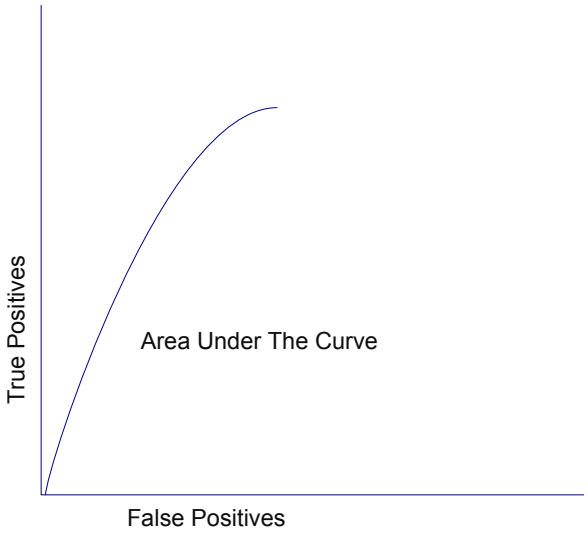

Figure 4.2: ROC curve made by plotting Truep Positives versus False Positives

However if the curve is plotted between TP and FP, the curve would resemble the curve shown in Figure 4.2. With low thresholds, the number of TP as well as the number of FP is low. As the threshold is raised, both show an increase. For good results, the curve now should have a point with high number of TPs

and low FPs. Larger area under the curve indicates better results. Thus the area under an ROC curve reflects the performance of the application.

ROC curves and the area under the curves have been used for the evaluation of the algorithms proposed in this report.

4.3 Results and Analysis

This section presents the experimental procedure and the results obtained. Two different experimental procedures were followed: the first procedure, *Procedure 1* or *Standard Evaluation*, is the standard procedure followed to compare two keyword spotting algorithms. We then discuss the limitations of Procedure 1 when used for keyword spotting algorithms and present a novel procedure, *Procedure 2* or *Equal Opportunity Evaluation*, to overcome those limitations. Each of the procedures is discussed in detail in the following subsections.

4.3.1 Standard Evaluation - Procedure 1

This is a simple procedure used to compare any two keyword spotting algorithms using ROC curves. The first step is to select a set of words and the utterances over which the techniques are tested. For each utterance, the techniques are

then allowed to present their confidence measures[5] for each word. Based on these confidence measures, the number of TP (or FN) and FP belonging to each technique are computed for a set of thresholds. An ROC curve is then plotted. The performance of the technique whose ROC has higher curvature is better.

Using this procedure of standard evaluation, two of the techniques - XRF and DN2 - were evaluated on two different sets of keywords.

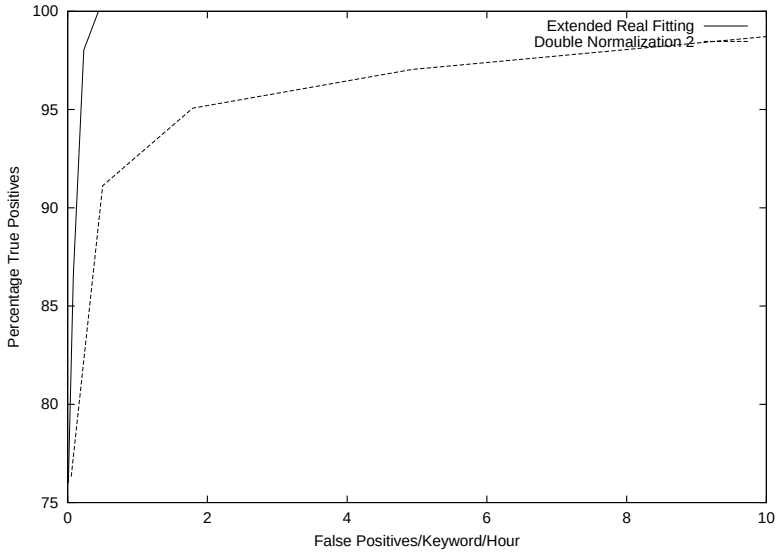

Figure 4.3: ROC curve comparing XRF and DN2 for the first set of 100 keywords

Each set of keyword consisted of 100 words selected from the 242 utterances in the database. All algorithms were made to calculate the confidence measures

[5]Since the confidence measures proposed in this report are based on the negative logarithm of posterior probability, lower values for confidence measures are better than higher ones.

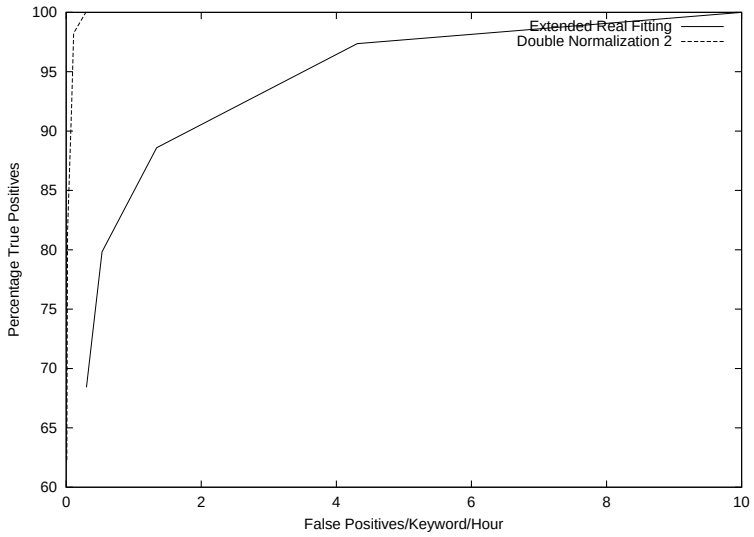

Figure 4.4: ROC curve comparing XRF and DN2 for the second set of 100 keywords

for each keyword. TPs and FPs were computed from these confidence measures for three different thresholds and an ROC curve was plotted between TPs and FNs for the three different thresholds. The resulting graph is shown in Figure 4.3. Same experiments were repeated with the second set of 100 keywords. Another ROC was plotted between the TPs and FPs of the techniques for the same thresholds. The results are shown in igure 4.4. The x-axis represents the detection rate, or the percentage of keywords correctly detected(percentage true positives). The y-axis represents the total number of false positives per keyword per hour of the speech data.

Clearly, the two curves in Figure 4.4 and Figure 4.3 show contradicting results. In Figure 4.3, XRF clearly outperforms DN2. XRF starts with a detection rate of about 78 percent at 0 FPs. As the threshold is increased, XRF achieves 100 percent detection at 0.5 false positives/keyword/hour. DN2 on the other hand achieves only 97 percent detection at about 5 false positives/keyword/hour. Increase in the detection rate after this point comes at a cost of very high false positive rate. Based on this graph, we could easily conclude that XRF achieves higher detection rate than DN2 on this database. However, the experiments performed on the second set of words prove this conclusion false. In Figure 4.4, DN2 starts with a detection rate of about 80 percent at 0 FPs. With the increase in threshold, it goes on to achieve 100 percent detection rate at 0.35 false positives/keyword/hour. Whereas, XRF achieves 98 percent detection at 4 false positives/keyword/hour, after which, the detection rate increases at a cost of very high false alarms rates. As mentioned before, area under the curve reflects the performance of the technique. Since the curve is plotted between TPs and FPs, higher the area under the curve, better the performance. In Figure 4.3, area under XRF is higher than DN2 whereas in Figure 4.4, area under DN2 is higher than XRF by about the same amount.

Hence, the results in Figures 4.3 and 4.4 are inconclusive. It is clear that the performance of a technique is strongly influenced by the choice of keywords.

Thus, to evaluate two techniques on a set of randomly selected keywords is not fair. To overcome this issue, we propose a novel evaluation, *Equal Opportunity Evaluation* discussed in Section 4.3.2.

4.3.2 Equal Opportunity Evaluation - Procedure 2

For a fair evaluation of keyword spotting techniques, it is important to remove the bias caused by the choice of keywords as discussed in Section 4.3.1. Equal Opportunity Evaluation overcomes this limitation by allowing each technique to achieve its best possible result. The best performances of the techniques are then compared. So each technique has an equal opportunity to display its best performance.

As discussed earlier, performance of a technique is strongly influenced by the choice of keywords. To allow the techniques to perform their best, the keywords on which the techniques perform the best should be selected. Equal opportunity evaluation selects the best keywords for each technique and then compares the performances of each technique on its respective list of best keywords.

To begin with, the first question that rises is how is the list of best keywords for each algorithm computed. Details of the procedure are described in the next few sections.

Initial Computation

In this phase, the results are computed for all the words (1221 words) in the database. All techniques are made to compute confidence measures of all words for each utterance in the database. To evaluate a word for a technique, its FPs and FNs are computed for a set of different thresholds. So at the end of this phase, a list of words and its FPs and FNs is available for each technique to be evaluated.

In our experiments, techniques using double normalization confidence measure, namely DN1, DN2, were evaluated at threshold values 1, 1.5 and 2 while the techniques using real fitting confidence measure, namely RF1, RF2 and XRF were evaluated at threshold values 3, 3.5 and 4.

Best Keywords Selection

With the list of confidence measures obtained in the previous section, an ROC curve is plotted between the FPs and FNs at different thresholds for all the words produced in the previous step. The area under the ROC for each word is computed and the words with lowest area[6] under the ROC are selected.

Figure 4.5 shows an example of an ROC of a word. If the curve is represented by $y = f(x)$, the area under the curve can be evaluated by :

[6]since ROC is plotted between FNs and FPs, lower area means better performance

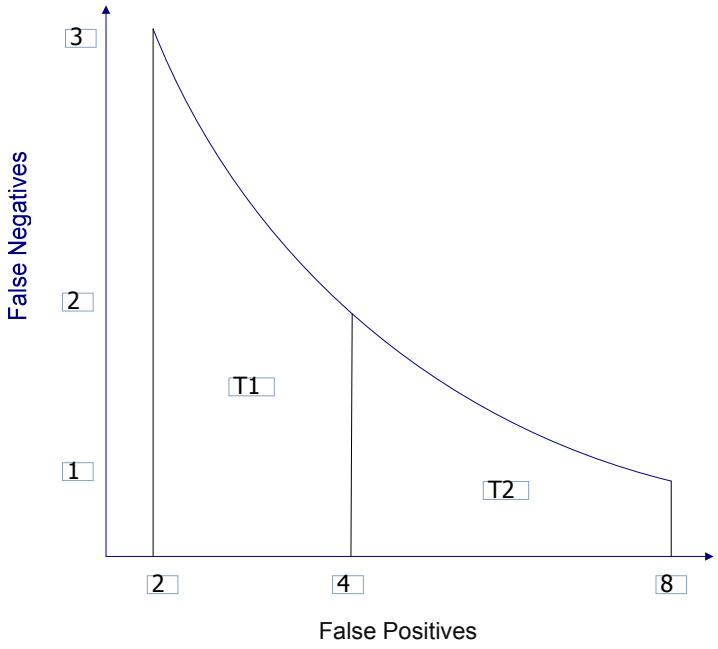

Figure 4.5: ROC curve of a word

$$\int f(x)dx$$

In Figure 4.5 the area can be evaluated by $\int_4^{10} f(x)dx$. It can also be approximated by computing the area under the trapeziums formed by joining the consecutive points, T1 and T2. This evaluates to: $(3+2)*(4-2)+(2+1)(8-4) =$

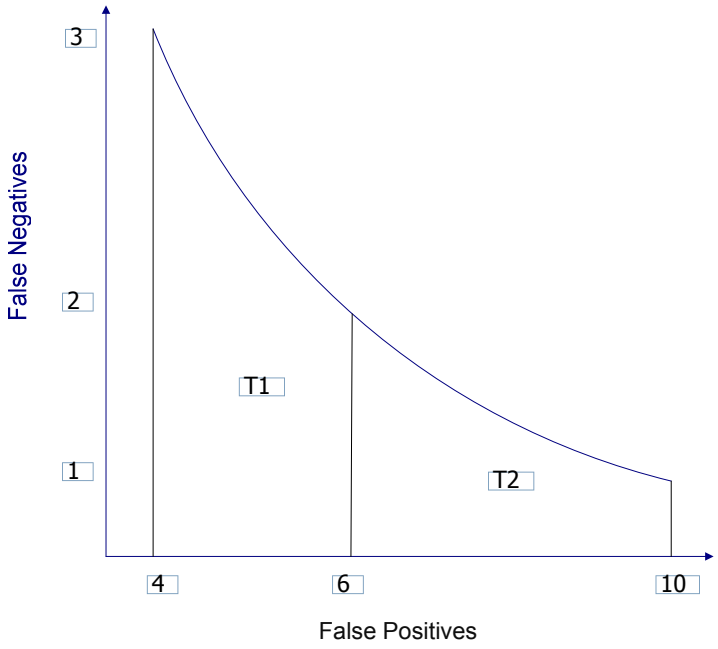

Figure 4.6: ROC curve of another word

22. However in certain scenarios, this computation fails. E.g. when another word has a curve as shown in Figure 4.6. In such a case, although the word with ROC in Figure 4.5 should be considered better (because the FPs are lower for same FNs), their areas would evaluate to the same value,

$(3+2)*(6-4)+(2+1)*(10-6) = (3+2)*(4-2)+(2+1)*(8-4) = 22$. To overcome this, we present an alternate method of computing area under ROC: The area under the curve is approximated by computing the area under the rectangles formed by joining the points with the x and y axes as shown in Figure 4.7.

With the latter method, the area under the curves in Figures 2 and 3 evaluate to $(2*3)+(4*2)+(8*1) = 26$ and $(4*3)+(6*2)+(10*1) = 34$ respectively, indicating the ROC of the former word in Figure 4.7 is better, as required. This method of computing area under ROC computes the area under the curve and also takes into account the position of the curve on the axes, giving a better evaluation for a curve closer to the axes, hence a curve with lower FPs. Therefore, we used this method, to compute the area under ROC for each word for each of the techniques.

Final Evaluation

From the list of words and the corresponding area under the ROCs produced in the previous step, 100 words with lowest area under the ROC are selected. For each of these techniques, ROCs are then plotted using the selected best 100 words. We plotted the ROCs using total number of TPs and FPs of the best 100 words of the technique for a set of different thresholds, and then compared these ROCs.

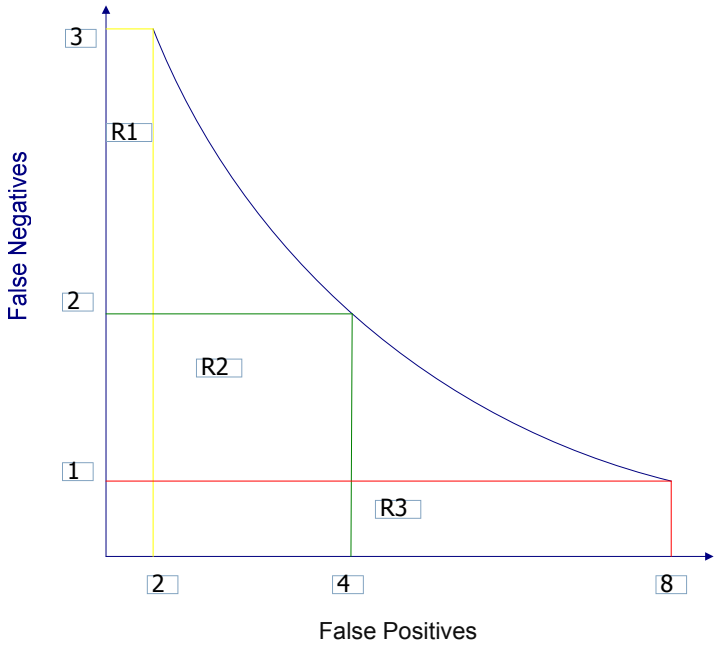

Figure 4.7: ROC curve of another word, area being computed by alternate method

The first comparison was made between the different types of prunings discussed in chapter 3 under sections 3.2.2 and 3.2.3. The two prunings under Double Normalization were DN1 and DN2. The ROC comparing these two is

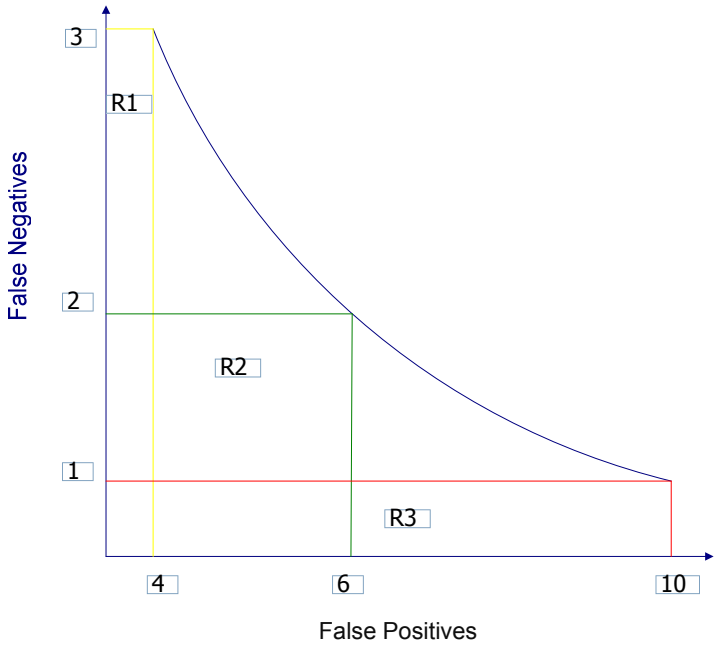

Figure 4.8: ROC curve of another word, area being computed by alternate method

presented in Figure 4.9.

In the Figure, DN1 starts at a higher detection rate at 0 FPs however, DN2 goes on to achieve 100 percent detection at 0.3 FPs/keyword/hour as opposed

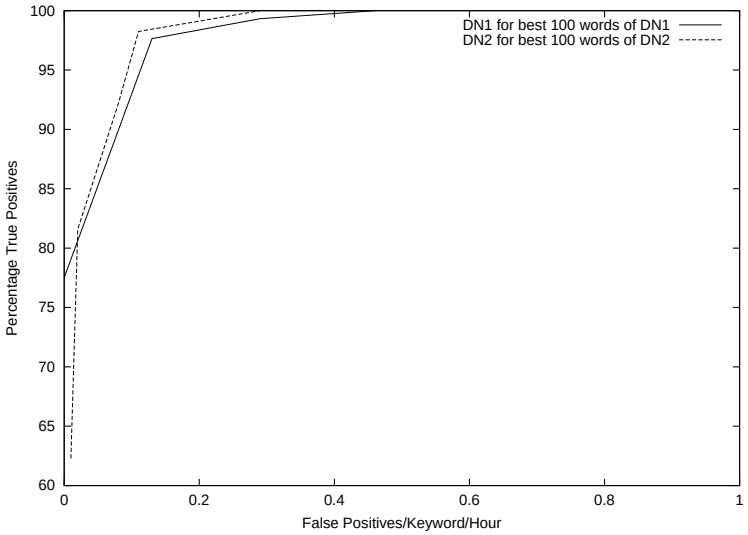

Figure 4.9: ROC curve comparing DN1 and DN2 for their respective best 100 keywords

to DN1 which achieves 100 percent detection rate at 0.5 FPs/keyword/hour. It can be said that as the threshold is increased, DN2 performs better than DN1. Since the pruning is harsher in DN2, fewer paths survive during the search. This also makes DN2 faster than DN1. The average time required by DN2 to evaluate the confidence measure of a keyword in an utterance of 740 frames (7.4 seconds of data) is 5.82 seconds/keyword while that of DN1 is 6.45 seconds/keyword.

The next comparison is between the three prunings under Real Fitting - RF1, RF2 and XRF. As discuessed earlier, XRF retains a set of path per column in trellis. We used a predetermined number - three paths per column -

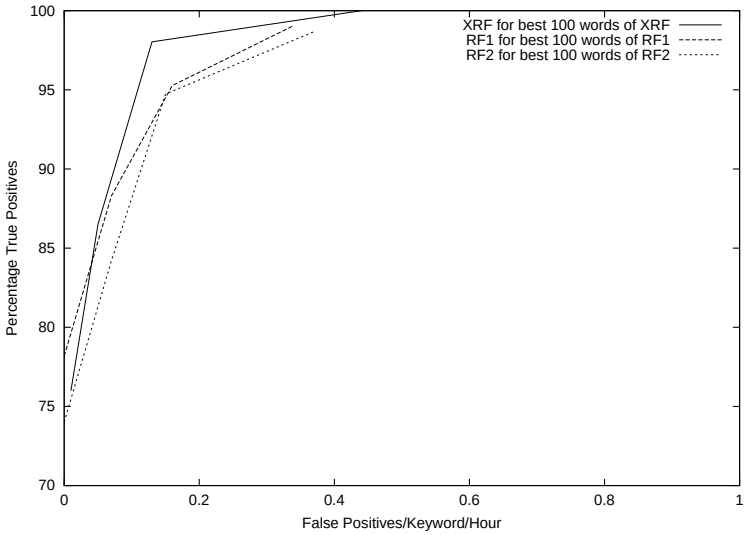

Figure 4.10: ROC curve comparing RF1, RF2 and XRF for their respective best 100 keywords

for our experiments. The ROC is shown in Figure 4.10. As the pruning here gets harsher, the detection rate is lowered. The difference is more pronounced at higher thresholds when the detection rates are higher. As discussed in the previous chapter, pruning in RF2 is the toughest, followed by RF1 and is the least in XRF allowing more paths to survive. As predicted from the pruning, detection in XRF is the highest followed by RF1 and RF2. Thus, we can say that if the pruning in the algorithm is further relaxed (to allow more than three paths in a frame), the performance should improve. However, this improvement comes at a cost of the time required by each algorithm to compute the confidence measure of a word in an utterance. The average time taken by RF1 to

detect a keyword in an utterance of 740 frames (7.4 seconds of data) is 5.06 seconds/keyword, while that taken by RF2 is 4.23 seconds/keyword and XRF (with 3 paths per state per frame) takes 5.22 seconds/keyword.

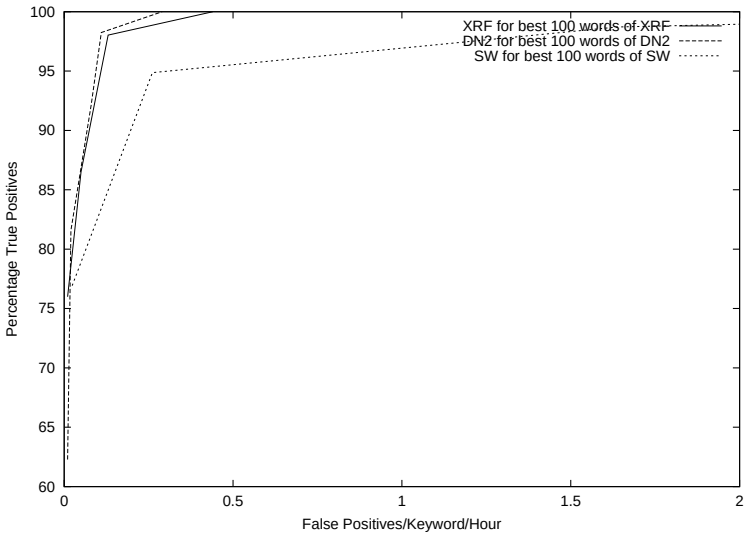

Figure 4.11: ROC curve comparing XRF and DN2 for their respective best 100 keywords

The next step is to compare the two confidence measures - Double Normalization and Real Fitting. We picked the best prunings in each confidence measure, DN2 and XRF, and compared their results. The ROC is shown in Figure 4.11. As shown, although XRF starts at higher detection for lower FPs, DN2 goes on to achieve 100 percent detection before XRF. Depending on the threshold, one technique is better than the other. Also, if more number of paths are per-

mitted per frame in XRF, its detection rate could surpass the detection rate of DN2. We also compared the performance of our techniques with Sliding Window (SW) Technique described in chapter 3, Section 3.1. DN2 and XRF exceeded the performance of SW as suggested by the results in Figure 4.11. We also compared the time taken by the three techniques and XRF took the least time followed by DN2. As mentioned before, on an utterance of 740 frames, XRF (with 3 paths per state per frame) takes 5.22 seconds/keyword while DN2 takes 5.82 seconds/keyword. For the same utterance, SW takes 20 seconds/keyword! Clearly, not only do DN2 and XRF have better detection rates, they have a much lower time complexity as compared to Sliding Window technique.

4.4 Phonemes in Best Words

We also conducted some experiments to analyze the performance of individual phonemes. Tables 4.1 and 4.2 list all the 26 phonemes in the Bref database and the number of times they occurred in list of best 100 words for each algorithm. An interesting fact observed is that inspite of the fact that the words constituting the best 100 words for each algorithm were different, the frequency of the phonemes is very similar across algorithms. Some of the examples of frequently occurring phonemes in the table are phonemes E, R, i. This means that not only are some words more easily spotted by algorithms, some phonemes are

easier to recognize also. Another such analysis was done on the length of the most easily recognized words.

Table **??** lists the lengths and the number of words of those lengths found in the list of best 100 words for each algorithm. It can be observed the all algorithms perform well on keywords with lengths 6 and 7. Thus it can be said that keywords of length about 6-7 form good keywords. Another interesting observation is that algorithms using RF confidence measure detect more shorter keywords than the algorithms using DN confidence measure while the latter spot more longer keywords than the former. A system comprising of both kinds of algorithms would thus perform well on a wider range of lengths of keywords.

The purpose of these results is to be able to find better keywords for keyword spotting systems to enhance their performance.

Table 4.1: Frequency of phonemes in the best 100 words

Phoneme	*RF1*	*RF2*	*XRF*	*DN1*	*DN2*
2	2	2	1	2	2
9	4	3	2	4	5
@	9	7	4	13	11
E	53	50	71	59	66
H	2	2	3?	2	5
N	3	3	2	1	1
O	7	7	9	6	9
R	45	46	61	61	63
S	5	7	7	9	9
Z	9	12	8	10	9
a	46	44	45	53	56
a	30	27	20	27	29
b	10	10	6	12	16
d	16	25	22	19	17
e	7	9	9	22	19
e	10	11	9	10	8
f	17	15	16	19	15
g	2	5	4	9	9
i	47	44	45	62	61

Table 4.2: Frequency of phonemes in the best 100 words

Phoneme	*RF1*	*RF2*	*XRF*	*DN1*	*DN2*
j	22	24	19	25	23
k	36	31	22	36	36
l	19	16	18	36	35
m	23	20	20	24	27
n	18	19	14	18	20
o	10	11	25	22	20
o	19	18	11	23	21
p	23	24	38	43	40
s	58	43	45	49	51
t	34	28	39	43	43
u	3	7	1	5	2
v	13	13	16	15	13
w	2	4	5	6	4
y	10	9	14	15	14
z	10	11	15	13	12

Table 4.3: Frequency of keyword lengths in the best 100 words

Length	*RF1*	*RF2*	*XRF*	*DN1*	*DN2*
2	1	1	1	1	0
3	2	3	1	0	0
4	2	2	8	1	1
5	27	32	19	7	4
6	31	31	35	20	29
7	24	21	14	21	21
8	6	4	8	16	15
9	3	2	7	16	13
10	2	1	2	10	6
11	0	1	4	6	6
12	2	2	1	0	3
13	0	0	0	1	1
14	0	0	0	1	1

Chapter 5

Conclusions

Keyword spotting techniques are a leading class of speech recognition techniques. They deal with recognition of known vocabulary words in a stream of voice signal. Three concepts of keyword spotting techniques that this research focusses on are confidence measures, pruning techniques and evaluation criteria.

Confidence measures represent the correspondance between the output and the input sequence. It is therefore important to have effective confidence measures to attain accurate results. Based on the logic that the best alignment for a keyword would be one in which all constituent states have high emission probabilities in their respective frames, the confidence meausre of Real Fitting is proposed. Real Fitting represents the score of a path by its worst subsequence match. Algorithm using this confidence measure achieves 100 percent

detection rate at 0.5 false positives/keyword/hour. For certain thresholds, it also outperforms algorithm based on Double Normalization. Results also show that algorithms based on Real Fitting are faster than those based on Double Normalization.

Pruning techniques determine which paths are eliminated and hence play a crucial role in keyword spotting. We introduce different pruning techniques for the One pass algorithm used by Double normalization and Real Fitting techniques. DN technique is tested with two different prunings, DN1 and DN2. The former requires the paths to be at the same level for comparison while the latter does not. This makes fewer paths survive in the latter leading to faster computation. DN2 achieves 100 percent detection at 0.3 false alarms/keyword/hour in 5.82 seconds/keyword while DN2 achieves 100 percent detection at 0.5 false alarms/keyword/hour in 6.45 seconds/keyword. RF1 and RF2 differ in pruning like their DN counterparts, DN1 and DN2. RF2 proves to be the fastest technique among all the techniques explored in this research, achieving 97 percent detection rate 0.4 false alarms/keyword/hour in 4.23 seonds/keyword which is only slightly lower than RF1 detection 98 percent detection rate at the same false alarm rate taking 5.06 seconds/keyword. XRF is a new pruning idea that maintains a set of paths per frame in place of a single path per frame. It has a detection rate of 0.5 false positives/keyword/hour, outperforming the rest, and

takes 5.22 seconds/keyword. An interesting point to note here is that all Real Fitting techniques are faster than Double Normalization techniques.

The results of a keyword spotting algorithm can be strongly influenced by the choice of keywords on which the experiments are run. This assumption is supported by the ROCs drawn to compare DN2 and XRF on two sets of keywords. Another contribution of this research is an evaluation technique that allows different algorithms to be compared without this influence. Equal opportunity evaluation selects the best keywords for each technique and compares the performance of the techniques on their respective best keywords. Although the best keywords for each technique differ, the phonemes constituting the best keywords across techniques are consistent, which leads as to conclude that some phonemes are easier to recognize. Most keywords which are recognized well by algorithms vary in length between 6-7 characters long. Another interesting fact to note is that RF techniques are better are recognizing shorter keywords than DN while the latter recognize longer keywords more easily than the former. Hence a system that combines the two is expected to have a broad range of detection.

Bibliography

[1] Herve Bourlard, Bart D'hoore, and Jean-Marc Boite. Optimizing recognition and rejection performance in wordspotting systems. In *yet to find out*.

[2] J. S. Bridle. An efficient elastic template method for detecting given words in running speech. In *Brit. Acoust. Soc. Meeting*.

[3] A. L. Higgins and R. E. Wohlford. Keyword recognition using template concatenation. In *Proc. Int. Conf. on Acoust. Speech and Sig. Processing*.

[4] J. Junkawitsch, L. Neubauer, H. Hoge, and G. Ruske. A new keyword spotting algorithm with pre-calculated optimal thresholds. In *Yet to find out*.

[5] Lori F. Lamel, Jean-Luc Gauvain, and Maxine Eskenazi. Bref, a large vocabulary spoken corpus for french. In *to be filled*.

[6] Steve Renals and Nelson Morgan. Connectionist probability estimation in hmm speech recognition. Technical report, International Computer Science Institute, 1992.

[7] Richard C. Rose and Douglas b. Paul. A hidden markov model based keyword recognition system.

[8] Marius C. Silaghi and Herve Bourlard. Posterior based keyword spotting approaches without filler models. Technical report, Laboratoire d'Intelligence Artificielle, 1992.

[9] sue johnson. *describe what is meant by the term keyword spotting.* PhD thesis, cambridge, 1997.

[10] J. G. Wilpon, C. H. Lee, and L. R. Rabiner. Application of hidden markov models for recognition ofa limited set of words in unconstrained speech. In *Yet to find out.*

[11] Jay G. Wilpon, Lawrence R. Rabiner, Chin-Hui Lee, and E. R. Goldman. Automatic recgonition of keywords in unconstrained speech using hidden markov models. In *Yet to find out.*