Improving Performance of a Keyword Spotting System by Using a New Confidence Measure

Luciana Ferrer, Claudio Estienne

School of Engineering
University of Buenos Aires, Argentina
lferrer,cestien@fi.uba.ar

Abstract

This work describes a HMM-based keyword spotting system. In this system, keywords are modeled as concatenations of the corresponding phoneme models, consequently, no specific databases are needed to train the system. In addition no filler models are required, therefore small computational requirements are necessary.

Two main stages define the whole system. The first stage is based on a previous work of Junkawitsch *et al.* It calculates, for each keyword, a score signal that measures the match between the keyword model and the utterance and extracts from that signal those segments where the match is good. The segments corresponding to possible keywords are used as input hypotheses for the second stage in order to get a new confidence measure. This second score is determined based on a comparison between the vector of emission probabilities for an hypothesis over the keyword model and the vector of emission probabilities for the best sequence of phonemes, in the segment where the hypothesis was detected. The first score is linearly combined with the second one resulting in a new score which performs significatively better than that one.

1. Introduction

Keyword spotting is a speech recognition branch consisting on detecting a small set of keywords from a speech stream. At least three approaches can be found in literature. The most obvious is to use a large vocabulary continuous speech recognition system (LVCSR) to produce a word string, and then search for keywords in that string ([1]). Other common approach is the use of filler models which represent the non-keyword intervals of the utterance ([2]). In this method the recognition is made with a continuous speech recognition system, using a grammar formed by the filler and the keyword models. Finally, a third method proposed by Junkawitsch *et al.* in [3] and [4], is based on a confidence measure whose maximization provides a probable position of a keyword in the utterance.

Although LVCSR approach seems to provide better performance (e.g., [2] or [5]), it requires knowledge of the specific task for which it will be used in order to select the words in the vocabulary, consequently, applications will be domain dependent. Also LVCSR word-spotters will require large body of training data, which gives rise to systems very expensive in computational terms. On the other hand, the use of filler models to represent non-keyword intervals seems to be an appealing approach to reduce computational cost and to maintain domain independence. However, as shown in [2], detailed filler models are needed in order to obtain higher performances. In doing that, most of the advantages of this system are lost.

The third method does not requires neither large training data bases nor filler models. It is based on a modification of the Viterbi search to compute normalized scores which indicate matching of the keywords at distinct time positions of the utterance. In order to decide if a keyword was or was not spoken, scores of the keywords are compared with decision thresholds.

In this work we propose a new scheme in which keywords obtained with the scheme presented in [3] are used as input hypotheses for a post-processor stage. This compound scheme produce significatively better performance than the original one.

The rest of this paper is divided as follows. First we will make a brief overview of the system presented in [3] in order to understand our proposal. Then we will describe our post processing system and discuss the main components. Finally we will report some experimental results comparing the performance of our system with the original scheme of [3] and also with a medium size vocabulary continuous speech recognition system .

2. Hypotheses generator overview

As already mentioned, the aim of the first stage of our keyword spotting system is to generate hypotheses that will be post processed on a second stage.

Figure 1 shows a block diagram of the hypotheses generator. The core of this system is a search algorithm, developed by Junkawitsch *et al.*, similar to the Viterbi one, that we will call score signals generator. This block receives as input the feature sequence vector corresponding to an utterance, and the set of HMMs corresponding to the keywords. Output of the block consists on a set of score signals, each signal associated to each keyword. This signal, called by us $Sc_1(t)$ (where t is measured in units of frames), can be interpreted as a distance measure from the optimum sequence of states over the keyword model between an initial time t_i and time t, to the sequence of *best states* in that same interval. The best state for each time is the one which has the maximum emission probability for the feature vector observed at that time. The initial time, t_i, and the sequence of states over the keyword model are both optimized by the algorithm to produce the maximum score.

From the definition of the score in [3] we can deduce that the signal $Sc_1(t)$ will always be greater than zero and will grow when the match between the utterance and the keyword model gets worst.

Whenever a keyword occurs in the utterance the score signal associated to it will show a maximum at the corresponding final time of emission of the keyword. The decision about that maximum corresponding effectively to a keyword, is taken

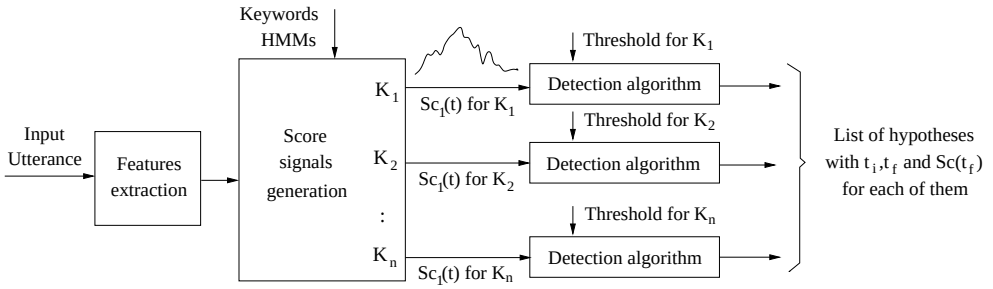

Figure 1: *Hypotheses generator.*

based on a certain threshold. The detection algorithm essentially consists on generating a keyword detection each time the score signal remains under the threshold for that keyword. Those thresholds are specified based on the desired percentage of detection.

As we are using the output of this system as input for the post-processing stage, we choose the decision threshold in order to allow all the really emitted keywords to be detected. Therefore, we use a desire percentage of detection of 99% to find the threshold for each keyword. Of course, for that decision threshold we also obtain a lot of false alarms. In practice, a second decision threshold has to be chosen after the post-processing system as will be explained in the discussion.

Output of the hypotheses generator for each utterance will consist on the detected keywords and, associated to each of them, the corresponding initial and final time of occurrence, t_i and t_f, and the value of the score signal at the final frame, $Sc_1(t_f)$, where the local maximum occur. These three values are obtained from the score signal by the detection algorithm.

It is important to notice that keywords HMMs are obtained as a concatenation of phoneme HMMs, so no special training data are needed to model keywords.

The reader is referred to [3] for further details about score generator and to [3] and [4] for details about thresholds determination.

3. Post-processing system

3.1. Second score generator

As output from the first system we have a list of hypothesized keywords, with a score Sc_1 associated to each of them (for simplicity, from now on, we will call $Sc_1(t_f)$ just Sc_1). The main goal of the post-processing stage is to get a better score than the score generated with the first stage. This new score combined with Sc_1, in turn will define the final score of the whole system.

As the first stage also provide us with initial and final frames for each hypothesized keyword, we can extract from the utterance the segment $y_i, \ldots, y_f$, of length L, where each hypothesis was founded. So, we can find the vector $Z = z_1, \ldots, z_L$ of emission probabilities for the optimal state sequence obtained by maximum likelihood with the keyword model for each hypothesized segment. In the same way we can also find vector $Z' = z'_1, \ldots, z'_L$ using a parallel phoneme model instead of the keyword model. These two vectors are easily obtained using the Viterbi algorithm to find the optimal state sequence, $s_1, \ldots, s_L$, over the corresponding model and then just printing the sequence of emission probabilities $b_{s_1}(y_i), \ldots, b_{s_L}(y_f)$. If the model input to the Viterbi algorithm is the keyword model, this emission probabilities vector will be Z, if the input is the parallel phoneme model this vector will be Z'.

Figure 2 shows a block diagram of the whole post-processing system.

Whenever hypothesized segment have a good match with keyword model, both vectors, Z and Z' will tend to be *similar*. A good measure of this similarity could be the mean square error between both vector sequences. So we call Sc_2:

$$Sc_2 = \frac{1}{L}\sum_{i=1}^{L}(z_i - z'_i)^2 \tag{1}$$

So, a "perfect" match will produce a null Sc_2, because the sequence of phonemes detected with the parallel phoneme model will be the same sequence that forms the keyword, giving the same sequence of emission probabilities.

Otherwise, if the match between the keyword model and the hypothesized segment is not perfect, there will be a better sequence of phonemes than that of the keyword. In that case the emission probability vectors will not have the same values and Sc_2 will be greater than zero.

3.2. Final score generator

We choose as the final score, Sc_{tot}, a linear combination of both Sc_1 and Sc_2 scores:

$$Sc_{tot} = \alpha Sc_1 + \beta Sc_2 \tag{2}$$

with $\beta = 1 - \alpha$, both between 0 and 1.

The value α is obtained experimentally in order to optimized the performance (see bellow).

Scores Sc_1 and Sc_2, grow when the match between the utterance and the keyword model gets worst. Being Sc_{tot} a linear combination (with positive coefficients) of those two values, it will have the same behavior. So, we can use this final score as a confidence measure to decide if the keyword was or was not spoken in that segment of the utterance.

4. Experiments

4.1. Evaluation conditions

The whole system was evaluated with a Spanish speech database borrowed from Star Lab at SRI International. Database was recorded in quiet room with a high quality microphone. The training set consists on 25725 utterances spoken by 102 different speakers. The test set is composed of 3049 sentences spoken by 12 different speakers. The duration of this set is approximately four hours.

We trained context-independent 3-states phonetics HMMs with the whole training set. Each state was modeled with a

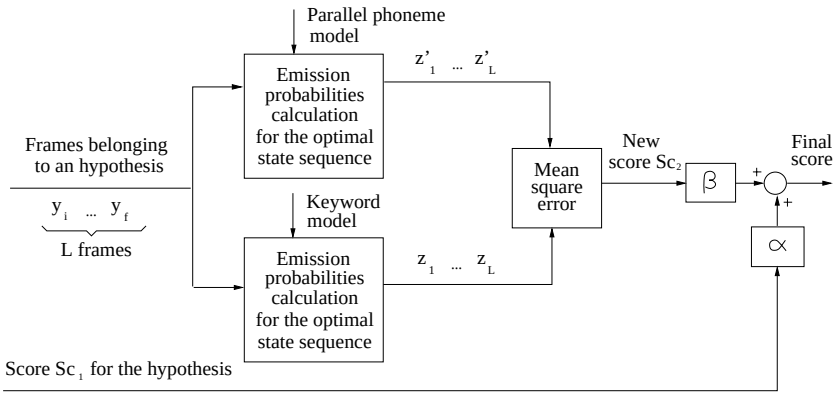

Figure 2: *Post-processing system.*

50 Gaussian mixture probability density function. A total of 12 mel-cepstral coefficients plus one energy coefficient plus 12 delta coefficients plus one delta energy coefficient conform a 26 components feature vector. 18 frequent short and long words were chosen as keywords from the test set, with 882 total occurrences.

The whole system was developed under Cambridge HTK (Hidden Markov Models Toolkit) frame. Scores were obtained by modifying Viterbi algorithm implementation of the toolkit.

4.2. Results on keyword spotting

In order to evaluate system performance we used the NIST factor of merit FOM (e.g., [6]) which is defined as the average detection rate from 0 to 10 false alarms per keyword per hour.

We process all the test set utterances with our system. For the calculation of the FOM a list of hypotheses sorted by the score Sc_{tot} is made. For each hypothesis it can be known whether it was a real detection or a false alarm by the use of the transcription of the corresponding utterance. With this information and the sorted list, the percentage of detection up to the appearance of any number of false alarms can be obtained. The FOM is to be calculated using those values of percentage of detection between 0 and 10 false alarms per keyword per hour. As can be seen, the FOM calculation is done without the need to set any threshold value.

α	1	0.8	0.65	0.6	0.4	0
1	22.4	24.9	26.9	27.5	29.8	29.2
2	36.1	40.2	41.0	40.9	40.5	38.7
5	48.0	51.3	51.7	50.9	50.3	46.2
10	56.5	60.2	59.7	59.4	58.9	56.0
14	59.8	63.3	64.2	64.0	63.1	60.7

Table 1: *Table of detection percentage for different number of false alarms and different values of* α.

Table 1 shows percentage of detection as a function of the number of false alarms and the value of α. Figure 3 represents FOM as a function of α. We can see a maximum at $\alpha = 0.65$, with a FOM of 65.44%. It also can be seen that for $\alpha = 1$ (which corresponds to the case of using only Sc_1) we have a FOM of 60.75% and for $\alpha = 0$ (which corresponds to the case of using only Sc_2) the value of the FOM is 61.40%

Analysis of results shows that:

- Score Sc_2 performs better than score Sc_1, but the best option is a combined score with $\alpha = 0.65$.
- Combined score introduces an increase on the performance of nearly 5% with respect to Sc_1 score.

4.3. Comparison to a continuous speech recognition system

In order to compare our system to a continuous speech recognition system we designed a medium vocabulary continuous speech recognition system with 700 words using parallel grammar. This set of words also included our original 18 keyword set. As a result we obtained a percentage of detection of 73.5% with an average of 14 false alarms per keyword. From table 1 we can see that the best percentage of detection obtained with our system ($\alpha = 0.65$) for the case of 14 false alarms is 64.2% which is clearly inferior.

It has to be noticed that in the case of the table 1 the number of false alarms are exactly 14 for each keyword, while in the continuous speech recognition system the 14 false alarms are an average over the 18 keywords.

5. Discussion

Increase in the performance of our score Sc_2 over original score Sc_1 can be predicted if we analyze both generators. In the first case, the state sequence we use to make the comparison which gives rise to Sc_1, is locally optimum. Each state is calculated in order to maximize the emission probability at the corresponding frame. So that could be a state sequence that cannot occur at all or one that has a very small probability of occurrence (depending on the transition probabilities). On the other hand, the state sequence that we use to make the comparison which gives rise to Sc_2, is based on a search over permitted transitions between phoneme states given by phoneme models. Therefore, more information about possible utterances is used in Sc_2 calculus than in Sc_1. That explains why this second score is able to outperform the first one.

Although it is not necessary for the evaluation of our system which is based on the FOM, any practical application requires the definition of a final threshold associated to each keyword, as a function of the maximum number of false alarms per keyword per hour that can be accepted. A hypothesized keyword will be considered a detection if the corresponding final score Sc_{tot} is less than the threshold for that keyword. A way of estimating this threshold set consists on getting all keyword hypotheses by processing a whole training set with our system. The list of hypotheses for each keyword can be ordered by increasing

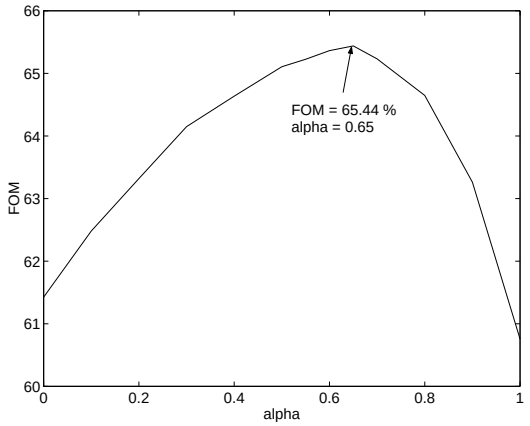

Figure 3: *FOM as a function of* α.

score, as it is done for the calculation of the FOM. Using this list we can fix the threshold for each keyword in the score value that corresponds to the maximum number of false alarms permitted for that keyword for the duration of the training set. It must be considered that the use of this method for the calculation of the threshold set gives rise to a big computational requirement in order to process all the training set with the system. On the other hand, this requirement will only be necessary during the training stage of the system but not during usage.

As shown in the experiments, the continuous speech recognition system performs better than our system. However, as mentioned above, keywords are modeled as phoneme concatenations so no specific task data bases are needed, also non-keyword parts of an utterance are not modeled in our system, for that reason computational cost is considerably low.

6. Acknowledgment

Authors want to thank to Star Lab at SRI International and specially to Dr. Horacio Franco for permitting to us the use of their Spanish speech database.

7. References

[1] Weintraub, M., “Keyword-spotting using SRI’s DECIPHER large-vocabulary speech recognition system”, Proc. EUROSPEECH’93, pp.1265-1268,1993

[2] Manos, A. S.,Zue, V. W., “A segment-based wordspotter using phonetic filler models” Proc. ICASSP’97, pp.899-902, 1997

[3] Junkawitsch, J., Neubauer, L., Hge, H., and Ruske, G., “A new keyword spotting algorithm with pre-calculated optimal thresholds” Proc. ICSLP’96, 1996

[4] Junkawitsch, J., Ruske, G., and Hge, H., “Efficient methods for detecting keywords in continuous speech” Proc. EUROSPEECH’97, 1997

[5] Rahim, M., “Recognizing connected digits in a natural spoken dialog” Proc. ICASSP’99, 1999

[6] Knill, K. M., Young, S. J., “Speaker dependent keyword spotting for accessing stored speech” CUED Technical Report F-INFENG/TR 193,Cambridge University, 1994