Strategies for Developing a Real-Time Continuous Speech Recognition System for Czech Language

Jan Nouza

SpeechLab, Technical University of Liberec
Halkova 6, 461 17 Liberec, Czechia
jan.nouza@vslib.cz

Abstract. This paper presents a set of 'strategies' that enabled the development of a real-time continuous speech recognition system for Czech language. The optimization strategies include efficient computation of HMM probability densities, pruning schemes applied to HMM states, words and word hypotheses, a bigram compression technique as well as parallel implementation of the real recognition system. In a series of off-line speaker-independent tests done with 1600 Czech sentences based on 7033-word lexicon we got 65 % recognition rate. Several on-line tests proved that similar rates can be achieved under real conditions and with response time that is shorter than 1 second.

1 Introduction

In this paper we present our approach to developing a continuous speech recognition system that is capable of real-time operation with lexicons containing thousands of Czech words. The system is the result of a long-term research and was completely built in our lab. It inherited many features from its predecessor, a discrete-utterance recognition system [1], which found its main application field in telephony services [2].

The system is based on the one-pass strategy [3] that processes a speech signal in time-synchronous way by efficiently combining acoustic scores of word models with language model probabilities. The words are represented by HMMs (Hidden Markov Models) that are constructed from a small set of elementary phoneme HMMs. The language model employs bigram probabilities estimated from a large text corpus. The short response time (< 1 s) has been made possible by the optimization of the search procedure, efficient bigram handling due parallel multi-thread implementation. The latter can be exploited on a PC with the MS Windows NT/2000 operating system.

The paper is structured as follows. In the next section we summarize the basic principles of the one-pass Viterbi search with a special focus on those parts and equations that are subject of the optimization procedures mentioned later. The third section gives some details on the acoustic and language model training. The proposed optimization strategies are presented in section 4. Experimental results achieved in both off-line and on-line tests appear in section 5, which is followed by conclusions.

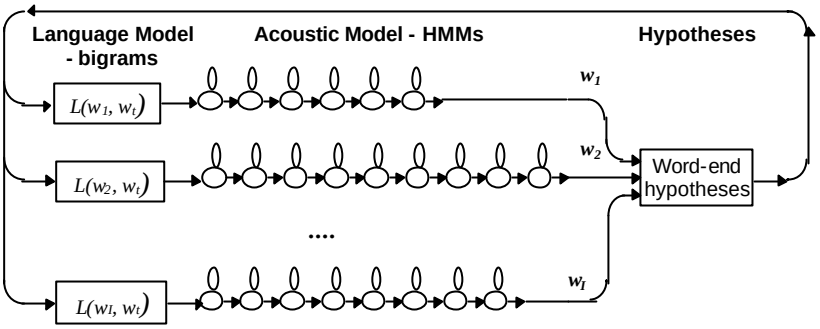

Fig. 1. Illustration of the one-pass speech recognition procedure as it is applied in the described system. It combines acoustic and language models to get the most likely hypotheses about the lexical content of the speech, within a computation optimized scheme

2 Time-Synchronous Speech Decoding Procedure

The task we want to solve can be described in the following way: Let us have a speech signal, an utterance, parameterized into sequence **X** of T frame vectors **x**(1), **x**(t) ... **x**(T). The goal is to find its lexical content, i.e. sequence **W** of words w_1, w_2 ... w_N, with a priori unknown length N. We will do it by exploiting acoustic, phonetic, lexical and syntactic knowledge embedded in two statistic models, the acoustic one (AM) and the language one (LM). They allow us to search for the most probable sequence **W** given the **X** by maximizing log posterior probability $P(\mathbf{W}|\mathbf{X})$:

$$\ln P(\mathbf{W}|\mathbf{X}) = \max_{w,N}\left[\sum_{n=1}^{N}\left(A(w_n|\mathbf{x}(t_n)\ldots\mathbf{x}(t_{n+1}-1)) + L(w_n, w_{n-1})\right)\right] \qquad (1)$$

Equation (1) is presented in the way so that the sum of logarithmic contributions of the acoustic and language models are explicitly shown.

The first term in Eq. (1) is the contribution of the AM of word w_n, which has been aligned with the partial sequence of frame vectors **x** starting at time t_n. If we use the classic left-to-right HMM for the word AM, we can write:

$$A(w_n|\mathbf{x}(t_n)\ldots\mathbf{x}(t_{n+1}-1)) = \max_{r,s}\left[\sum_{t=t_n}^{t_n-1}(\ln a_{rs} + \ln b_s(\mathbf{x}(t)))\right] \qquad (2)$$

where a_{rs} is the probability of transition from previous state r to current state s and b_s(**x**(t)) is the output probability of state s given frame vector **x** in time t.

The second term in Eq. (1) is the contribution of the LM. We employ a weighted bigram model:

$$L(w_n, w_m) = \ln g(w_n, w_m) \cdot C_{LM} + C_{IP} \quad (3)$$

where $g(w_n, w_m)$ is an estimate of the probability that in a sentence word w_n follows word w_m. Constant C_{LM} is a factor enhancing the contribution of the LM and C_{IP} is an insertion penalty. Optimal values for both the constants must be determined empirically on an evaluation database [6].

The well-known Viterbi algorithm searches for the solution of Eq. (1) by applying the dynamic programming method. It consists in time-synchronous evaluation of cumulative scores Q defined for each word w, its model state s and time t. Inside a word model the score is calculated according to equation (4):

$$Q(t, s, w) = \underset{r}{Max}[Q(t-1, r, w) + \ln a_{rs} + \ln b_s(\mathbf{x}(t)] \quad (4)$$

and at word boundaries according to equation (5):

$$Q(t, 0, w) = \underset{v}{Max}[Q(t-1, S, v) + L(w, v)] \quad (5)$$

where the value in the initial (0-th) state of word w is determined as the best combination of the score in the final state S of previous word v and the corresponding bigram value. Word v maximizing the right side of Eq. (5) is stored as the best predecessor of w in time t.

In this way we get the most likely solution of equation (1) by finding the word sequence whose last word ends up in time T with the highest score:

$$\ln P(\mathbf{W}|\mathbf{X}) = \underset{i}{Max}\, Q(T, S, w_i) \quad (6)$$

After identifying the last word w_N through equation (6) we get the complete sequence W by tracking back the stored predecessors.

3 Acoustic and Language Models

In our system the acoustic model is based on the application of the Hidden Markov Model (HMM) technique. Three-state left-to-right continuous HMMs model spectral and temporal characteristics of 41 Czech phonemes and background noise [4]. Both context-independent (monophones) as well as context-dependent (triphones) models are available. However, due to the efficiency reasons discussed in the next section we give preference to the former ones. The models have been trained on the database containing about 20 hours of annotated speech recordings provided by almost 100 speakers. The recordings were sampled at 16 bit/8 kHz rate and pre-processed to get 26-feature frame vectors composed of 12 cepstral and 12 delta-cepstral coefficients complemented by delta energy and delta-delta energy. (The 2nd derivatives of the cepstrum were not included since their addition did not exhibit any further improvement of the recognition rate.)

Each word in the given lexicon is represented by concatenated phoneme models according to the phonetic transcription rules that are automatically applied during the lexicon generation [5]. Recently a single model per word is used, however, the system can handle multiple pronunciation variants as well.

The language model is based on word bigrams. They have been derived from a large corpus of printed text available in electronic form. The corpus is made mainly of newspaper articles but it includes also several novels. It contains 55,841,099 Czech words, with 856,288 distinct word forms. The total number of 18,458,806 different word pairs have been found in the corpus. These served as a base for estimating various types of smoothed bigram models and for optimizing the LM constants that occur in equation (3). The details are presented in the complementary paper [6]. The recent version of the real speech recognition system employs the bigram matrix smoothed according to the Witten-Bell discounting scheme, which yielded the best performance in a series of extensive evaluation tests.

4 Strategies Applied for Real-Time Processing

Even though the dynamic programming (DP) technique reduces significantly the complexity of the search task, the full evaluation of Eq. (4-6) in the whole three-dimensional space of w, s and t cannot be managed in real-time. Instead some heuristics and data manipulating tricks must be introduced. In the following subsections we describe those adopted in our system.

4.1 Acoustic score caching

The evaluation of the above mentioned DP equations seem to be quite simple, because they include mostly addition and maximization operations. The only CPU-power demanding part is the last term in Eq. (4). It represents the probability density function (pdf) defined as a mixture of multiple (8 - 32) gaussians in 26-dimensional space. However, since the evaluation of Eq. (4) is done synchronously with time, the pdf values computed for given frame vector $\mathbf{x}(t)$ can be stored in cache and reused whenever the same state s appears in another word (or even in the same word). This caching scheme is extremely efficient in case of monophones because there are only 41 x 3 different states to be matched with vector $\mathbf{x}(t)$. Thus, for example, in a 10K word lexicon almost 99,94 % pdf calculations can be omitted and replaced by a much faster access to the cache. This is the main reason why we have built our system on the use of the monophones. Moreover, our choice has been justified by the fact that 32-mixture monophones yielded the same performance as the available 8-mixture triphones.

4.2 State and word pruning

During the search within the *wst* space many word hypotheses become unlikely. Due to the time-synchronous evaluation of Eq. (4-6) we can identify them according to their scores which are much lower compared with the currently best ones. Hence, at each time step we find the hypothesis with the best score Q_{best} and in the next time step we remove from the further consideration all HMM states s if:

$$Q(t-1,s,w) < Q_{best}(t-1) - C_{PT} \qquad (7)$$

where C_{PT} is a state pruning threshold. Its value must be optimized to get the largest computation reduction with minimum lost of recognition accuracy [6].

Besides the above state pruning scheme we apply also a word pruning option. It is based on keeping track of the surviving states. We do it for each word by storing the highest index from all unpruned states. If there is no surviving state in the current word, that word is temporarily removed from the searched space.

4.3 Word-end hypotheses pruning

To start a word hypothesis for word w according to Eq. (5), the maximization over all predecessor words v should be done. For a lexicon with M items this means that at each time step the total number of $M \times M$ summations of word-end scores and bigrams should be performed. However, practical experiments showed that it was not necessary to consider all existing word-end hypotheses. Only a small number of the preceding words v - those with the highest scores - will play a dominant role in Eq. (5). Hence, at each time step t we order the word-end hypotheses according to their scores and keep just a short list of the best ones. If the list has C_{WE} ($C_{WE} << M$) members, we save a large portion of the computing load. Surprisingly, the C_{WE} can be set in range 5 - 20 even for large lexicons without any significant performance degradation. More details can be found again in the complementary paper [6].

4.4 Handling bigrams

As the size of the application vocabulary increases, the memory space needed for storing the bigrams becomes prohibitively large. (For example, the full bigram matrix for a 10K word task would occupy 10K x 10K x 4B = 400 MB.) However, it is well known, that the number of distinct bigram values is much smaller. In particular, there is a considerably large amount of values that are same, which is the result of the smoothing technique. The natural solution to this problem is to compress the bigram matrix and store its values in the way that allows for efficient access.

In our system the bigrams $g(w_r, w_p)$ are stored as vectors $h(w_p)$ that share the common previous word w_p:

$$\left[g(w_r,w_p)\right] = h(w_1),h(w_p),\ldots h(w_M) \qquad (8)$$

The vectors $h(w_p)$ can be efficiently compressed because they contain smaller or larger groups of the same values. Moreover, we arrange the bigrams in the $h(w_p)$ not in the natural order (by index) but according to their values, from the higher to the lower ones. When evaluating the Eq. (5) we first initialize scores $Q(t, 0, w_r)$ by an appropriate default value and than we fill the scores using the ordered values taken from the vector $h(w_p)$. This arrangement together with the maximization principle embedded in Eq. (5) offers another computation saving. Thus only a small fraction from all the C_{WE} x M bigram combinations must be handled at each time step.

The reduction of memory requirements is even more significant. A bigram matrix for a typical 10K word lexicon can be stored in the memory space smaller than 5 MB. Yet another reduction is possible if we allow the bigrams to be quantized and approximated by a limited set of values.

4.5 Parallel processing

When speech recognition experiments are performed off-line, the signal is parametrized first and after that it is sent to the classifier. The same approach is often used also in on-line systems.

Modern computers and operation systems, however, enable the programmers to decompose complex tasks into separate subtasks that can run in parallel. The speech recognition task is quite suitable for such a decomposition. The most natural arrangement employs a three-level hierarchy. While the lowest level unit cares about the speech signal sampling, the middle level unit segments signal into frames and computes feature vectors, and the upper level unit focuses entirely on the classification procedure. The only critical issue is the correct synchronization of the units and the safe data transfer between them.

The demonstration version of our system employs the above parallel scheme. It runs on a PC with the Windows2000 operating system, which gives support to multi-thread implementation. This allows the system to start the recognition procedure in the moment when speech is detected and finish it very soon after the end of the utterance is approved.

5 Performance Evaluation

The individual procedures and their parameters were evaluated mostly in off-line experiments. These consisted in batch-organized recognition tests performed on a large evaluation database. The database contains 1600 utterances recorded by 40 people (23 men + 17 women) of various ages. The speakers were asked to read sentences that appeared on the PC screen. A common head-mounted microphone set was used for acquiring the signal. The recordings were done on different places (mostly at homes, in student rooms, occasionally in a computer lab.)

The sentences were drawn from newspaper articles on various topics: home and international news, sport events, culture articles, weather reports, etc. The 1600

utterances represent 85 minutes of speech and contain 16,027 words (in average 10 per utterance). The lexicon was made of 7,033 different Czech words that covered all the sentences.

An initial series of experiments was aimed at establishing the optimal values of the search procedure parameters, namely the language model weighting factor C_{LM}, the word insertion penalty constant C_{IP}, the state pruning threshold C_{PT}, and the length of the hypotheses list C_{WE}. These experiments were part of the language model evaluation project and they are discussed in more details in paper [6]. Here we want to focus on the global performance assessment.

The measure we used for the system evaluation was the accuracy rate defined as:

$$Acc = \frac{N - S - I - D}{N} \times 100\% \qquad (9)$$

The accuracy - as opposed to the frequently used Correctness rate - is considered as the best measure representing the practical usage of a recognition system because it takes into account all kinds of errors, including the substitutions S, deletions D, insertions I, and relates them to the true number of words N in the correct transcription [7].

Table 1 summarizes some of the most relevant results and compares the impact of different acoustic and language models. The best-performing language model referred to as the WB was produced by the Witten-Bell smoothing technique applied on the independent training corpus mentioned in section 3. The recognition times were measured on a PC (Athlon 1,3 GHz, 512 MB RAM).

Table 1. Results from off-line speech recognition tests done with 1600 Czech utterances

Acoustic model	**Language model**	**Accuracy [%]**	**Recog. time per sentence/word [ms]**
16-mixture HMM	zerogram, C_{IP} = -35	39.25	7213/720
32-mixture HMM	zerogram, C_{IP} = -35	49.15	7318/760
16-mixture HMM	WB, C_{LM} = 6, C_{IP} = -6	55.87	6265/625
32-mixture HMM	WB, C_{LM} = 6, C_{IP} = -6	65.42	6693/668

The on-line tests could not be so extensive since it is more difficult to arrange them. Up to now, two people volunteered to participate in an experiment, in which the subject had to read 100 sentences directly to the PC using a common head-mounted microphone set. The text was randomly drawn from the same list of 1600 sentences. The system automatically detected the speech, did the classification, displayed the resulting word sequence and counted the errors according to Eq. (9). The response of the recogniser was really instantaneous. The output text appeared on the screen always in time shorter than 1 second after the speaker finished the utterance. This fast response was possible due to the parallel implementation of the speech processing and classification routines. For the results see Table 2.

Table 2. Online speech recognition tests (100 sentences per speaker, 7033-word lexicon)

Speaker	Acoustic and Lang. Model	Accuracy [%]	Æ response time [ms]
MJN	32-mixture HMM, WB	72.81	654
ZVS	32-mixture HMM, WB	66.23	583

6 Conclusions

In the paper we present several strategies that allowed us to implement one of the first continuous speech recognition systems applicable for the Czech language. So far the system has been tested on a middle-size lexicon with fairly promising results. We believe that the system capabilities could be extended for vocabularies containing tens of thousands words. Our future work is going to be focused on the improvements of the acoustic model (increasing the HMM training database, evaluating other features sets and testing different HMM arrangements) and on further development of the language model that should take into account some specific characteristic of Czech (inflections, case and gender agreement in noun and verbal phrases, etc.)

Acknowledgments. This work was supported by the Grant Agency of the Czech Republic (grant no.102/02/0124) and project MSM 242200001. The author wants to thank Tomáš Nouza for his invaluable assistance in multi-thread programming.

References

1. Nouza J.: A Czech Large Vocabulary Recognition System for Real-Time Applications. In Text, Speech and Dialogue: Proceedings of the Third International Workshop on Text, Speech, Dialogue. Springer-Verlag, Heidelberg, 2000, pp. 217-222.
2. Nouza J., Holada M.: A Voice-Operated Multi-Domain Telephone Information System. Proc. of 25th Int. Conference on Acoustics, Speech and Signal Processing (ICASSP2000), Istanbul, June 2000, vol.VI, pp.3755-3758
3. Ney H., Ortmanns S.: Dynamic Programming Search for Continuous Speech Recognition IEEE Signal Processing Magazine, vol.16, no.5, Sept. 1999, pp.64-83
4. Nouza J., Psutka J., Uhlír J.: Phonetic Alphabet for Speech Recognition of Czech. Radioengineering, vol.6, no.4, Dec.1997, pp.16-20.
5. Nejedlová D., Volejník M.: Transkripce psaného ceského textu do fonetické podoby (Phonetic transcription of printed Czech text). In Pocítacové zpracování reci (ed. Nouza). Technical University of Liberec, 2001, pp.10-22.
6. Nejedlová D.: Comparative Study on Bigram Language Models for Spoken Czech Recognition. In this volume.
7. Huang X., Acero A., Hon H.-W.: Spoken Language Processing. A Guide to Theory, Algorithm and System Development. Prentice Hall. New Jersey 2001.