Large vocabulary continuous speech recognition for Bulgarian

Petar Mitankin
Institute for Parallel Processing
Bulgarian Academy of Sciences
petar@lml.bas.bg

Stoyan Mihov
Institute for Parallel Processing
Bulgarian Academy of Sciences
stoyan@lml.bas.bg

Tinko Tinchev
Faculty of Mathematics and Informatics
Sofia University
tinko@fmi.uni-sofia.bg

Abstract

The paper presents the results of a project completed by the authors for realizing a continuous speech recognition system for Bulgarian. The state-of-the-art speech recognition technology used in the system is discussed. Special attention is given to the problems with some specifics of the Bulgarian language namely the large vocabulary (450000 wordforms). Some implementation details of the language module are given. At the end the paper provides evaluation of the accuracy of recognition.

Keywords

Speech recognition, language model

1 Introduction

Speech is a preferable medium for enabling comfortable and effective human-computer interface. This explains the wide interest in speech technology. There are a number of implementations of computer speech synthesis systems, which deliver intelligible and naturally sounding voices. For Bulgarian the SpeechLab system [1] is widely used by the visually impaired people.

Speech recognition is a considerably more difficult problem especially in the case of continuous speech and large vocabulary. There exist sophisticated implementations for English and some other major European and Asian languages. There are no reports for a large vocabulary continuous speech recognition (LVCSR) systems for Bulgarian. This paper will present the result of a project for realizing a LVCSR system for Bulgarian, which has been executed by the authors.

After some background information given in the next section we will describe the basic modules of our system. Section 3 and Section 4 will provide technical details of the Bulgarian acoustic and language models correspondingly. The recognition process is discussed in Section 5. In Section 6 we present the implementation details of the language model and Section 7 shows our accuracy evaluation. The conclusion presents some general discussions and further development directions.

2 Background

The problem of speech recognition can be considered as the task to find the set of possible word sequences which sound close to the observed speech signal and are admissible according to a given language model. The recognition of isolated words is significantly simpler. This task can be solved using various techniques with modest computational efforts. The computational complexity might be linear in regards of number of words in the vocabulary and no language model is required. A LVCSR system has to recognize a sequence of words from a large vocabulary without information of the word boundaries. A naive method would require an exponential number of word permutations to be considered.

The state-of-the-art approach to this problem is the Hidden Markov Model (HMM) framework. An HMM represent a statistical process, where the elements of a sequence of observed variables are regarded as emissions of the state variables in a Markov chain. The observed variables in our case represent the characteristics of a short slice (frame) of the speech signal. The states in the Markov chain represent the phases of the phonemes, which we assume to be pronounced.

3 Acoustic model

Our implementation of the acoustic model consists of two parts. The first part is the digital signal processing (DSP) module, which extracts a vector of features for each frame of the speech signal. The second part consists of the phoneme models.

The audio input is sampled at 16 KHz. First an emphasis filter is applied for emphasizing the higher frequencies. Afterwards, the signal is sliced into frames using a 30 ms Hamming window in 10 ms steps rolling. The next step is extracting the most characteristic, in respect to the human perception, features of the signal. For that purpose we extract the Mel-frequency

International Conference RANLP 2009 - Borovets, Bulgaria, pages 246–250

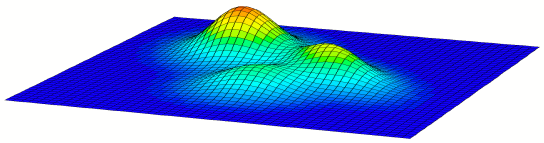

Fig. 1: *A two-dimensional Gaussian mixture with three components.*

cepstrum coeffiecients. The idea behind is to compensate for some specifics of the human ear like logarithmic energy and frequency perception, and to be able to separate the source of the signal from the articulation configuration (the filter). More details on the articulation model and the DSP part might be found in [7, 5]. The characteristic feature vector we obtain from each frame consists of the first and second derivative of the frame energy, the 16 cepstrum coefficients and their first and second derivatives – 50 parameters in total.

We use Bayesian classifiers for classifying the frames based on the extracted feature vectors. The Bayesian classifiers are implemented as Gaussian mixtures.

$$f(x; c_k, \mu_k, \Sigma_k) = \sum_{k=1}^{K} c_k \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^t \Sigma (x-\mu)};$$

$$c_k \geq 0; \sum_{k=1}^{K} c_k = 1$$

Clearly each Gaussian mixture is a n-dimensional continuous probability distribution (see Figure 1). The Gaussian mixture is used to define the probability for every feature vector to belong to a given class. In our implementation we have 50 dimensions and we use mixtures with up to 16 components.

Our phoneme models are based on continuous 3-state left-to-right HMM (Figure 2). More formally a continuous 3-state left-to-right first-order HMM is a tuple consisting of a sequence of states $\langle 1, 2, 3 \rangle$, transition probabilities $\langle a_{0,1}, a_{1,1}, a_{1,2}, a_{2,2}, a_{2,3}, a_{3,3} \rangle; a_{0,1} = 1; a_{1,1} + a_{1,2} = 1; a_{2,2} + a_{2,3} = 1; a_{3,3} = 1$ and emissions $\langle b_1, b_2, b_3 \rangle$, such that every emission b_i is a n-dimensional Gaussian mixture. In that case if a sequence of T feature vectors $O_1, O_2, \ldots, O_T \in \mathbb{R}^n$ is given, then the probability this sequence to be observed along the sequence of hidden states $s_1, s_2, \ldots, s_T \in \{1, 2, 3\}$ is:

$$P(O_1, O_2, \ldots, O_T | s_1, s_2, \ldots, s_T) = \prod_{i=1}^{T} a_{i-1,i} b_i(O_i)$$

The probability the sequence $O_1, O_2, \ldots, O_T \in \mathbb{R}^n$ to be observed by the HMM model is the sum of the probabilities this sequence to be observed along all the sequences of hidden states:

$$P(O_1, O_2, \ldots, O_T) = \sum_{s_1, s_2, \ldots, s_T} P(O_1, O_2, \ldots, O_T | s_1, s_2, \ldots, s_T)$$

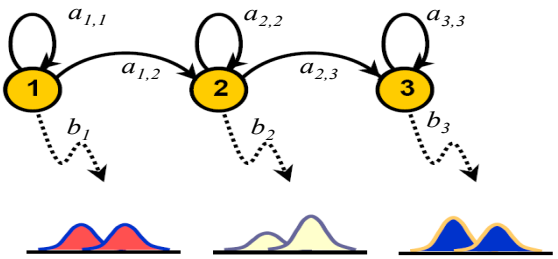

Fig. 2: *A continuous 3-state left-to-right first-order HMM.*

A	мерак	l	лек, лампа	g	град
a	мазе, кедър	m	мама	d	дар
e	тел, перо	n	нар	v	жар
i	бик, пирон	p	пек	z	зар
O	конче	r	ръка	k	кана
o	борба, кичур	s	син	4	чар
U	Тунис	t	тих	6	шах
Y	катър	f	фар	j	край, Кольо
b	баба	h	хол	0	чорбаджия
w	Варна	c	цар	9	годзила

Table 1: *Phonemes*

The acoustic model consists of 30 continuous 3-state left-to-right HMMs – one for each phoneme. The training of the HMMs is performed using a variant of the Baum-Welch algorithm [10, 5].

Our phonetic system uses different phonemes for the stressed and unstressed vowels "а", "о", "у" and "ъ". The unstressed vowels "а" and "ъ" are merged into one phoneme. The same applies to the unstressed "о" and "у". The palatalized consonants are considered as a pair of the corresponding not palatalized consonant followed by the semivowel "й". Table 1 presents the 30 phonemes used in our system.

4 Recognition model

The Recognition model has to provide a mapping between an audio signal represented as a sequence of feature vectors and the sequence of the pronounced words. This mapping is ambiguous because of the ambiguous positions of the word boundaries and the homophony phenomenon. For example there is no difference in the pronunciation of the Bulgarian phrases *Петко рита* and *Пет кориа*. Because of that the Recognition model has to provide an evaluation (probability) of the correspondence of the audio signal and a sequence of words in regards to the following criteria:

1. Acoustic similarity between the signal and the pronunciation of the words;
2. The syntactic correctness of the word sequence;
3. The semantic correctness of the word sequence in respect to the context.

The speech recognition task is then reduced to finding the most probable sequence of words for a given signal

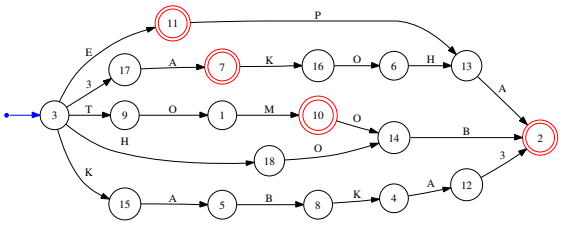

Fig. 3: *Dictionary if the wordforms* E, EPA, ЗА, ЗАКОНА, КАВКАЗ, НОВ, ТОМ, ТОМОВ *represented as a deterministic finite-state automaton.*

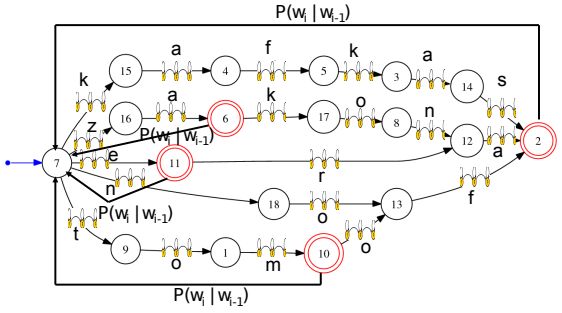

Fig. 4: *The Recognition model.*

in respect to the Recognition model.

The Recognition model is built in several steps. First a deterministic finite-state automaton D recognizing all the wordforms from the vocabulary is constructed using the methods presented e.g. in [2]. On Figure 3 the minimal deterministic finite-state automaton for 8 wordforms is shown. On the next step the vocabulary of phonetized wordforms is built. For this task we have used the phonetization rules for Bulgarian developed in the framework of the SpeechLab project [1]. The ruleset is represented as a composition of replace rules. For example the rule for de-voicing a sequence of consonants ending with an unvoiced consonant is (refer to [6]):

```
(w→f)|(z→s)|(g→k).../
_ (voiced|unvoiced)*.unvoiced.(vowel|sonor)
```

All the phonetization rules are represented as regular relations, which are composed into one relation R using the techniques presented in [6, 4]. The finite-state automaton representing the phonetized vocabulary $R(D)$ is derived by the mapping of D through R using the following regular operations: $R(D) = range(id(D) \circ R)$.

In the next steps the phonemes in the network $R(D)$ are substituted with the corresponding acoustic models – the 3-state left-to-right HMMs discussed in the previous section. In this way we derive a huge HMM which models the acoustic probability a sequence of feature vectors to be mapped against a given word.

Finally the Recognition model has to incorporate a language model which evaluates the possible sequences of words. The most common language model for a task of this scope is based on bigram probability estimates, relating the likelihood of co-occurrence for two consecutive words in a sequence.

The final recognition network is constructed from the phonetized vocabulary HMM by creating bigram-weighted transitions from the end of each word (the final states) to the beginning of every other word (the starting state). The bigram-weights are calculated dynamically as explained in section 6. The Recognition model is illustrated on Figure 4.

5 Recognition process

As discussed above, the recognition process is reduced to finding the most probable sequence of states in the Recognition model in regards to the feature vectors derived from the audio signal. The general scheme of the recognition process is given on Figure 5.

The computational complexity of the naive approach, which considers all possible sequences of hidden states would be $O(M^T)$, where M is the number of states in the language model HMM and T is the length of the sequence of feature vectors. The Viterby algorithm [10] might be used for reducing the complexity to $O(M^2T)$. In the case of LVCSR M is in the order of millions and T is in the order of a thousand. Therefore a full Viterby search can hardly be performed in real time. To reduce the computations a common approach is to employ beam search. In the Viterby algorithm at each step only a limited number – the best scoring paths are regarded. I.e. the dynamic programming traversal of the network is a partial breadth-first traversal, where the lower scoring paths are discarded. Only the N-best continuation "active" states are considered. In our evaluation we have experimented with the following values for N: 1000, 1500, 2000, 2500 and 3000.

The number of word sequences, which are derived using a full Viterby search is in the order of $10^{60}-10^{70}$. The beam search approach reduces the number of word sequences for a given utterance to about 10^{10}. The result of the traversal is stored into a weighted acyclic word lattice. An example of the word lattice for the utterance "петкорита" is shown on Figure 6. The weights correspond to the acoustic and word bigram probabilities, summed with particular weights. The acoustic probability is calculated from the acoustic models of the phonemes in the sequence and the bigram probabilities are derived from the bigram language model. On a second step the acoustic and bigram probabilities are re-scored using other weights. The system returns the best sequences in the word lattice in respect to the new weights. More details of this approach are presented in [8, 5].

Our system implements the Microsoft Speech API version 5.1. In this way the Bulgarian speech recognition is accessible under the Microsoft Windows operating system from all applications utilizing the Speech API like Microsoft Office. The memory used by the system is below 120 MB. The speed of the processing is under 0.9 times the real time on a modest computer in case of 2000 active states. This means that the user can use the system without the need to awaiting the processing.

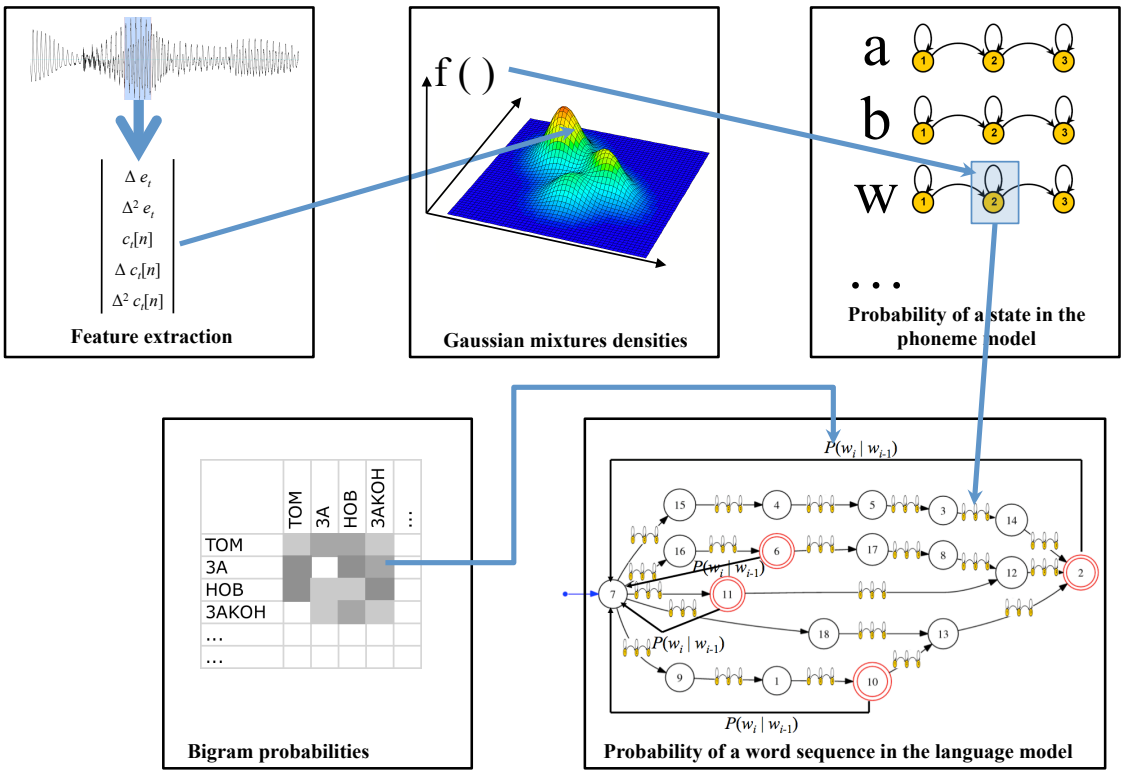

Fig. 5: *General scheme of the speech recognition process.*

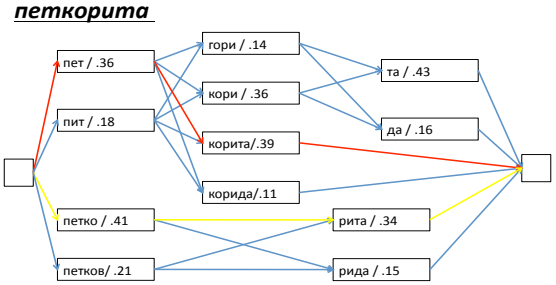

Fig. 6: *Word lattice.*

6 Implementation details of the Language model

Consider the sequence of n words $v_1, v_2, \ldots, v_n$. Its probability is approximated as $P(v_1) \cdot \prod_{i=1}^{n-1} P(v_{i+1} \mid v_i)$. A bigram language model can be considered as a data structure used for solving the following problem: given a bigram (pair of words) $w_1 w_2$, find $P(w_2 \mid w_1)$.

We use a corpus of legal texts to evaluate a bigram language model. The corpus consists of $200 \cdot 10^6$ words. 90% of the sentences in the corpus are used for the computation of the bigrams, while the other 10% are used for the calculation and minimization of the cross-entropy. The number of the words (monograms) in our dictionary is 442,501. In our implementation we use a bigram cut-off threshold: we take into account only the bigrams with at least two occurrences in the corpus. The number of these bigrams is 4,108,409. To smooth the monogram and bigram probabilities, we apply the absolute discounting algorithm, [9], in backing-off variant.

In [3] Jan Daciuk and Gertjan van Noord present a technique for very compact representation of language models. However their method requires binary search in sorted arrays. Since the time complexity is crucial for a good speech recognition system, we choose an approach that requires constant time for a bigram probability evaluation. The constant is small and does not depend on the number of the monograms and bigrams.

Every word is represented as a 32-bit integer. Its value is the position of the word in the lexicographical order of all words. Every byte of this 32-bit integer is considered as a separate symbol. Thus every word has exactly four symbols and every bigram is a string of eight symbols. The number of all symbols is 256. We keep all such eight-symbol strings in a perfect hash. The perfect hash is represented as a minimal acyclic deterministic finite state automaton $A = \langle \Sigma, Q, i, \delta, \sigma, f \rangle$ over alphabet Σ, where $|\Sigma| = 256$, Q is the set of states, $|Q| = 631,501$, $\delta : Q \times \Sigma \to Q$ is a partial transition function, the number of the transitions is 4,264,244, i is the initial state and f is the only one final state. Every transition of A is associated with a nonnegative whole number by the

Active states	1000	1500	2000	2500	3000
Juridical texts					
1-LER	93.72%	95.32%	96.05%	96.16%	96.51%
1-WER	81.46%	84.85%	87.10%	87.41%	88.08%
Speed (x RT)	0.32	0.55	0.82	1.21	1.67
Memory (MB)	100.73	108.3	111.54	116.0	137.21
Common texts					
1-LER	94.13%	94.83%	95.30%	95.50%	95.64%
1-WER	80.41%	82.37%	83.53%	84.07%	84.42%
Speed (x RT)	0.3	0.52	0.77	1.13	1.55
Memory (MB)	103.94	110.59	119.17	120.6	139.31

Table 2: *Evaluation table.*

function $\sigma : Q \times \Sigma \to \mathbb{N}$. A has the property that if the bigram $b = b_1 b_2 \dots b_8$ is accepted by A, then $\sum_{k=1}^{8} \sigma(\delta^*(i, b_1 b_2 \dots b_{k-1}), b_k)$ is the position of the bigram b in the lexicographical order of all $4,108,409$ bigrams. Here δ^* is the extended transition function. The function δ induces a sparse transition matrix and we use Tarjan's table, [11], to represent this matrix. The table has $4,895,828$ entries. Every entry represents a transition. The size of one entry is 12 bytes: 4 bytes encode a symbol, 4 - a value of σ and 4 - a destination state. We have achieved $87,1\%$ utilization of the Tarjan's table leaving $12,9\%$ empty entries. Using this table, given a state s and a symbol $a \in \Sigma$, we can compute in constant time $\delta(s,a)$ and $\sigma(s,a)$. We have another table Pr of $4,108,409$ four-byte floating-point numbers. In Pr we keep the $-log$ probabilities of the bigrams. Let us explain how our language model solves the following problem: given a bigram $b = b_1 b_2 \dots b_8$, find $P(b)$. We start a traversal of A with b. If b is accepted by A then $P(b) = Pr[\sigma^*(i,b)]$, where σ^* is the extended version of σ. If the traversal fails after b_4 then we use the smoothing formulae and the monogram model to compute $P(b)$. If the traversal fails before b_4 then $P(b) = P(b_5 b_6 b_7 b_8)$ and we use the monogram model to compute it. The representation of the monogram model is trivial: it is a table of $442,501$ four-byte floating-point numbers - one number for every word.

The size of the whole language model is $82,3$ MB. The computation of a bigram probability $P(w_2 \mid w_1)$ requires constant time.

7 Evaluation

We performed our experiments on a notebook with a Mobile AMD Sempron 3400+ processor and 2 GB RAM. The acoustic model was trained by adapting a speaker independent model with one hour of speaker's speech data. The vocabulary consists of about 450000 wordforms.

The test set consists of 63 utterances from juridical texts and 1276 utterances from common texts. We have evaluated the letter accuracy defined as $1 -$ letter error rate (1-LER) and the word accuracy defined as $1 -$ word error rate (1-WER). The speed indications are in respect to the real time. Table 2 summarizes the evaluation results.

A manual review of the recognition results revealed some specific classes of mistakes. As expected the complex clitics grammar in Bulgarian resulted in many cases of wrong word boundary segmentation e.g.:

да ли $\leftrightarrow$ *дали, би ли* $\leftrightarrow$ *били, за кон и* $\leftrightarrow$ *закони.*

The full form of the definite article inflection in masculine nouns was another major source of recognition errors. Since the "т" in the full form article inflections "ът", "ят" is often pronounced without an explosion the system confused it often with the wordform with the non-full form of the definite article e.g.:
конят $\leftrightarrow$ *коня, мъжът* $\leftrightarrow$ *мъжа.*

The single most frequent source of errors are the proposition "в" and "с". Their unvoiced phonetizations are the phonemes "f" and "s", which are very easily confused with any noise like speaker's aspiration especially at the utterance start. Since those are between the most frequent words in Bulgarian this resulted some times in the wrong insertion of an additional proposition in front of the utterance, especially in case of more noisy environment.

8 Conclusion

This paper presented the result of the project for building a LVCSR system for Bulgarian. Sophisticated techniques were applied to cope with the very large Bulgarian vocabulary. The evaluation results showed less than 4% letter error rate and 13% word error rate on speech recognition of juridical texts using real time processing speed on a modest notebook computer. The system has been packaged as a software product implementing the Microsoft Speech API 5.1.

An informal user feasibility test has been conducted in a company specialized in providing law information services. This test proved that although a user adaptation period is required, the system is already in a state permitting daily use for text dictation.

References

[1] M. Andreeva, I. Marinov, and S. Mihov. Speechlab 2.0 – a high-quality text-to-speech system for bulgarian. In *Proceedings of the RANLP 2005*, pages 52 – 58, Borovets, 2005.

[2] J. Daciuk, S. Mihov, B. Watson, and R. Watson. Incremental construction of minimal acyclic finite state automata. *Computational Linguistics*, 26(1), 2000.

[3] J. Daciuk and G. van Noord. Finite automata for compact representation of tuple dictionaries. *Theor. Comput. Sci.*, 313(1):45–56, 2004.

[4] H. Ganchev, S. Mihov, and K. U. Schulz. One-letter automata: How to reduce k tapes to one. In F. Hamm and S. Kepser, editors, *Logics for Linguistic Structures*, number 201 in Trends in Linguistics, pages 35–56. Mouton de Gruyter, Hillsdale, New Jersey, 2008.

[5] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing.* Prentice-Hall, 2001.

[6] R. M. Kaplan and M. T. Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, 1994.

[7] S. E. Levinson. *Mathematical models for speech technology.* Wiley, 2005.

[8] R. Mosur. *Efficient Algorithms for Speech Recognition.* PhD thesis, Carnegie Mellon University, 1996.

[9] Ney, Hermann, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8:1–38, June 1994.

[10] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77, 1989.

[11] R. E. Tarjan and A. C. Yao. Sorting a sparse table. *Communications of the ACM*, 22(11):606–611, November 1979.