THE WATSON SPEECH RECOGNITION ENGINE

R. Douglas Sharp, Enrico Bocchieri, Cecilia Castillo,
S. Parthasarathy, Chris Rath, Michael Riley, James Rowland

AT&T Laboratories
600 Mountain Avenue,
Murray Hill, NJ 07974

Abstract
In 1995, AT&T Research (then within Bell Labs) began work on a software-only automated speech recognition system named Watson™. The goal was ambitious; Watson was to serve as a single code base supporting applications ranging from PC-desktop command and control through to scaleable telephony interactive voice services. Furthermore, the software was to be the new code base for the research group, allowing fast deployment of new algorithmic advances from the lab into the field. A set of C++ objects has been developed which support these objectives. This paper gives an overview of the Watson Automatic Speech Recognizer software architecture, describes the algorithms employed, and provides performance numbers for some sample tasks.

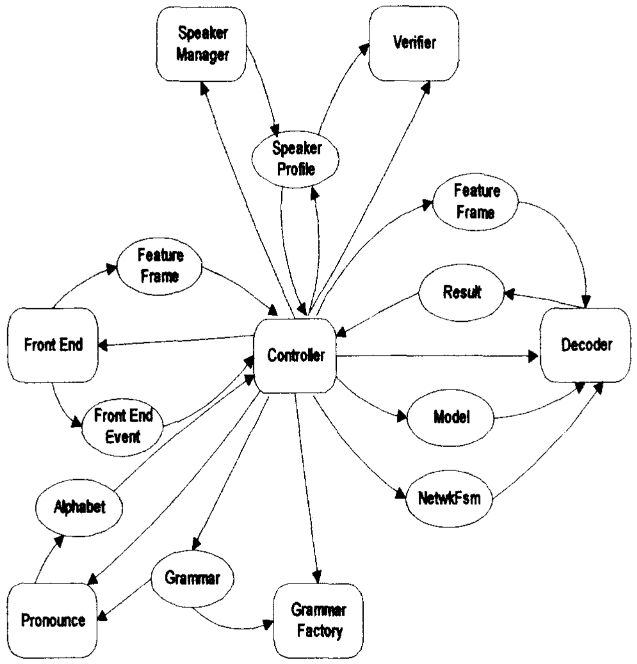
Introduction

Watson is a software product that supports speech recognition, text to speech synthesis, and speaker verification. It is licensed to applications developers by AT&T's Advanced Speech Products Group. The functionality in the initial versions of the Watson ASR product was obtained through bringing together the best features from a number of existing research ASR systems. The first such system was the Bell Labs Automatic Speech Recognizer (BLASR), which implements an efficient Viterbi beam-search for large vocabularies and supports context-dependent modeling[1]. Another was the OneBLASR recognizer[2], which was used for recognition experiments on the ARPA North American Business task[2].

For the Watson recognizer to bring value to systems ranging from a single processor PC through to scaleable platforms supporting high-end telephone network applications, the system had to be based upon a general, extensible, and object-oriented architecture. The object-oriented discipline of creating an encapsulation between interface and implementation was required in order that the features required for one application interacted minimally with those required for another. For example, the Intel-PC version of Watson is the only one that need be concerned with the details of the Intel MMX™ instruction set. On the other hand, some functions of Watson are used in many different ways. For example, the code which implements the Viterbi decoding algorithm supports training, real-time small-vocabulary recognition, large-vocabulary recognition, and speaker verification, each of which require specific features from the decoder which the other applications do not. Watson avoids the complexity associated with this profusion of features by making use of a simple *base class* for each major object in the system along with a specialized *derived class* per specific application of the object.

Watson Software Architecture Overview

The Watson ASR Engine is comprised of a set of C++ objects which collaborate to perform speech recognition and speaker verification. In the following diagram, the boxes represent the major subsystems, and the ovals represent the data objects which flow between them.



The *FrontEnd* subsystem accepts audio samples and produces *FeatureFrames* along with *FrontEndEvents* including *UtteranceStart* and *UtteranceEnd*. The *GrammarFactory* subsystem manages the *Grammars* loaded by an application and constructs the composite *NetworkFsm* used by the Decoder from the active rules of all loaded grammars. The *Decoder* subsystem performs a frame-synchronous Viterbi beam search on a *NetworkFsm*, consumes *FeatureFrames*, and returns the set of maximum likelihood paths subject to phone duration constraints. Partial path information required for stopping prompt playback on barge-in can also be returned.

The *Pronounce* subsystem maintains a current *Phonetic Alphabet* and provides dynamic translation from orthographic text strings to subword transcriptions.
The *SpeakerManager* subsystem maintains a set of *SpeakerProfiles* in memory and/or on disk, manages *SpeakerProfiles* for all known speakers and keeps track of the current speaker name and group. The *Verifier* subsystem is responsible for verifying that a set of utterances comes from a particular speaker.

Finally, the *Controller* subsystem coordinates the activities of the other subsystems and communicates with the application through the standard Microsoft Speech application programmer interface[3] (SAPI). Applications also have access to advanced capabilities beyond those covered in SAPI (such as speaker verification) through AT&T's Advanced Speech API (ASAPI)[4], a platform-independent API which extends SAPI.

Overview of the Algorithms

Front-end
The front-end can be configured to generate either Mel filter-bank[5] or LPC cepstrum[6]. Mel is currently used for microphone speech, while LPC is used for telephone. In either case, twelve cepstral coefficients as well as normalized energy are produced. With delta and delta-delta terms, 39 dimensional feature vectors are produced with a ten millisecond frame advance. Cepstral-mean subtraction (CMS)[7] is performed in order to make the front-end robust to channel and microphone variations. Silence and speech segments are identified using an energy-based heuristic and are normalized separately using leaky-integration. For high accuracy in telephone applications where limited adaptation data is available per call, the leaky algorithm supports a time-varying integration coefficient, allowing rapid initial adaptation based on a small initial look-ahead, with a longer effective look-behind as the number of frames available in the past increases. In order to preserve real-time response, the look-ahead buffer is collapsed when sufficient frames are available in look-behind, which allows the decoder to catch up if running in faster than real time.

Grammars
The Watson engine accepts either finite state networks[8] or grammars in Microsoft SGF format[3]. While SGF can be used to specify general recursive grammars, Watson supports only right-recursion. SGF grammars are compiled at load time into finite state networks, which are also used as the Watson SAPI native grammar format.

Decoding Network
The Watson network representation allows any finite-state model of context to be used in a very general class of decoding cascades, without requiring specialized decoders or full network expansion. The approach is based on two fundamental ingredients: a simple generalization, *weighted finite-state transducers*, of existing network models, and *on-demand composition*, a novel "lazy" execution technique for network combination. Weighted finite-state transducers generalize weighted probabilistic automata (e.g., Hidden Markov models or n-gram language models) by replacing the single label of an arc by a *pair* I:O of an input symbol I and an output symbol O. Tranducers permit us to explicitly represent mappings between representational levels. For example, a lexicon maps between words and their phonetic pronunciations and is readily encoded in a transducer. More interestingly, we can represent the mapping between context-independent phones and their appropriate context-dependent (e.g., triphonic) expansion directly as a transducer, without requiring specialized context-dependency code in the recognizer. Any finite-state language model (e.g, n-grams) can be trivially represented as an identity transducer restricted to the finite-state language.

Appropriate versions of the above transducers can then be *composed*, yielding a transducer from context-dependent models to word sequences that is guaranteed to represent all the required cross-word context dependencies. The crucial algorithmic advantage of transducer composition is that it can be easily computed on-the-fly. We developed a fully general lazy composition algorithm, which creates on demand, and optionally saves for reuse, just those states and arcs of the composed transducer that are required in a particular decoding run, for instance, those required for paths within the given beam width from the best path.

We can thus use the lazy composition algorithm as a subroutine in a standard Viterbi decoder to combine on-the-fly a language model, a multi-pronunciation lexicon with corpus-derived pronunciation probabilities, and a context-dependency transducer. The external interface to composed transducers does not distinguish between lazy and pre-computed compositions, so the decoder algorithm is the same as for an explicit network.

In recognition experiments with the DARPA Resource Management (RM), Airline Travel Information Services (ATIS), and North American Business News (NAB), we have found that only 1-2% of the full network needs to be expanded for a particular utterance. This significant savings in memory comes with little cost in time (<5% total CPU recognition time)[8],[9].

Search
Watson performs a fairly standard Viterbi beam-search[10,1]. At each frame only those paths are extended for which the difference between their score and the best score is less than a given threshold. One slight twist in the Watson implementation is the way in which multiple (n-best) paths are optimally maintained in the decoder[11]. Just as a single token per state in the decoding network is sufficient to maintain the information required to optimally return the best Viterbi path, so n tokens per state are sufficient to maintain n-best paths optimally, if each token can be guaranteed to correspond to a significantly different path[12]. To allow paths to be distinguished from one another with a single pointer comparison, a hash table is maintained where a path through the network corresponds to a series of hash table entries, and each entry contains a pair of values. The first is the symbol corresponding to the most recent

significant label in the path, and the second is a pointer to the preceding table entry on the path. This is illustrated in the following diagram[13].

Symbol Table

W1	The
W2	Quick
W3	Slow
W4	Red
W6	Fox

Hash Table

P1	W1	NIL
P2	W2	P1
P3	W3	P1
P4	W4	P2
P5	W4	P3
P6	W6	P4

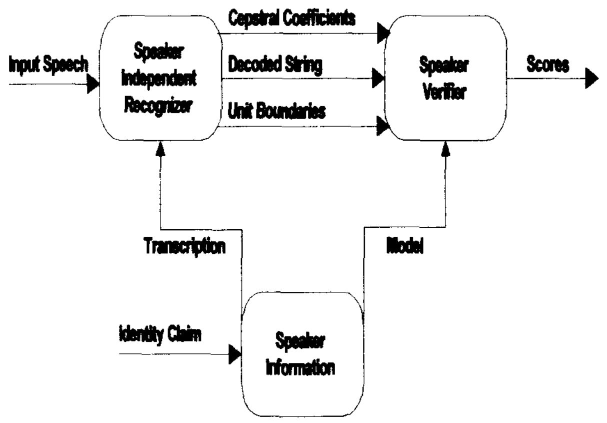
The illustrative tables above currently contain two paths through the network, corresponding to "The quick red fox" and "The slow red", along with other partial paths. Using such a table, maintaining the sorted list of n-best tokens at each state in the network given the previous lists of the m states which feed the current state takes only O(n*m) operations. Since most of the token merging is done within the HMM states and these states have only a self-loop and a transition to the next state, in practice we find that the Watson search has complexity O(n) for n-best candidates. Adding a new path into the hash table when crossing a significant word boundary (or verifying that this is a new path) takes roughly constant time (based on hashing with chaining).

Models

Units which are supported in the Watson recognizer range from whole-word context-independent digit models through to context dependent head-body-tail subword models. This allows a great deal of flexibility in the trade-off made between memory, speed, and accuracy. Continuous density HMM's are used to represent each unit. For each state of an HMM model, the spectral observation density is represented by a weighted mixture of multivariate normal density functions. General tying is supported at the mixture or state level. Each unit can have an associated duration penalty and insertion penalty.

Speaker-verification

Watson supports text-dependent speaker verification using digit-string or general phrase passwords. The block diagram of the verification system is shown below.



A user makes an identity claim by speaking a digit string or a phrase unique to that user. The input utterance is recognized and segmented into a sequence of units (either digits or phones) by the Watson speaker-independent recognizer. The label of the recognized utterance functions as an identity claim and is used to retrieve a set of speaker-dependent models (created at enrollment), which is then transmitted to the verifier along with the segmented utterance. The verifier scores the password utterance using the user's model for the phrase as well as a speaker background model [14,15]. The test statistic is the ratio of the target model score and the background model score.

The system requires limited enrollment data (usually a few repetitions of the password phrase) in a single session and yet provides robust performance under a variety of test conditions that are mismatched with training. This is achieved by using likelihood ratio scoring and also by performing ongoing adaptation of speaker dependent models and thresholds using all accepted in-service verification utterances.

Performance

The following results are for speaker-independent ASR on a variety of sample tasks. The real time factor is the processing time (on a 100Mhz Pentium) divided by duration of the recognized utterance.

Task	Accuracy	Real-Time
Microphone Digits[16]	95.3 / 98.7	0.23
Microphone voice-labels[17]	87.1	0.25
Telephone Digits[18]	86.0 / 98.5	0.16
Telephone easy NJ towns[19]	90.4	1.66
Telephone hard NJ towns[19]	67.5	1.65

Summary

From the conception of the Watson ASR system in January 1995 until the present time, we have made great progress towards our vision of a single stream of software which would serve as a portable, scaleable basis for interactive voice services in a variety of target architectures, and at the same time serve as the software base for ASR research within AT&T labs .

Interactive voice interfaces will become a ubiquitous feature of future applications, whether accessed over the telephone network, the Internet, or running on a local PC. By virtue of its modular, scaleable architecture and leading-edge technology, the Watson ASR engine has been designed to address each of these application domains. Ongoing algorithmic improvements from AT&T labs will be made available in the Watson product with the lowest possible delay due to the shared Watson code base.

References

[1] E. Giachin, C.-H. Lee, R. Pieraccini, L.R. Rabiner, "Implementation aspects of large vocabulary recognition based on intraword and interword phonetic units," CSELT Technical Reports - Vol. XIX - No. 1 - February 1991.

[2] Michael Riley, Fernando Pereira Emerald Chung, "Lazy Transducer Composition: a Flexible Method for On-the-Fly Expansion of Context-Dependent Grammar Networks" Snowbird '95: Snowbird, Utah. Dec 1995."Lazy Transducer Composition: a Flexible Method for On-the-Fly Expansion of Context-Dependent Grammar Networks" Snowbird '95: Snowbird, Utah. Dec 1995.

[3] Microsoft Corporation, "Speech API Developers Guide, Windows™ 95 Edition," Version 1.0 ©1995 Microsoft Corporation.

[4] AT&T Corporation, "Advanced Speech API Developers Guide", Version 1.0 ©1996 AT&T Corporation.

[5] Steven B. Davis, Paul Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentances," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-28, No.4, August 1980.

[6] B. S. Atal, Suzanne L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," The Journal of the Acoustical Society of America, 21 April 1971.

[7] F. Liu, R. Stern, X. Huang, A. Acero, "Efficient Cepstral Normalization for Robust Speech Recognition," Proceedings of ARPA Human Language Technology Workshop, March 1993.

[8] Pereira Fernando, Michael Riley, Richard Sproat, "Weighted Rational Transductions and their Application to Human Language Processing,", ARPA Workshop on Human Language Technology, Mar 1994, page 249.

[9] M. Mohri, F. Pereira, M. Riley, "Weighted Automata in Text and Speech Processing," ECAI 96: 12th European Conference on Artificial Intelligence, August 1996, Budapest, Hungary.

[10] B. Lowerre, D. R. Reddy, "The HARPY speech understanding system," Trends in Speech Recognition (W. Lee, ed.) Prentice-Hall Inc., New York, pp.340-346.

[11] Optimally subject to the Viterbi beam heuristic, that is.

[12] Significance is determined by the task at hand. For example, if only the word level transcription is desired, then each word can be held significant. If only certain words are significant, then the others can be made equivalent for the purposes of the n-best algorithm.

[13] Guy L. Steele Jr, "Common LISP : The Language," Digital Press, 1980.

[14] A.E. Rosenberg, S. Parthasarathy, "Speaker background models for connected digit password speaker verification," ICASSP96.

[15] S. Parthasarathy, A.E. Rosenberg, "General phrase speaker verification using sub-word background models and likelihood-ratio scoring" ICSLP96.

[16] String and digit forced-choice accuracy obtained on a data set of 299 1, 7, and 10 digit strings collected from NCR customers in their homes. Average string length is 4.36 digits. The recognition grammar allowed any number of digits to be recognized.

[17] In this subword recognition task, NCR customers spoke voice-dialing sentences such as "Call Jim at the office". Results are obtained on 149 test sentences run against a recognition grammar containing 182 distinct phrases.

[18] String and digit forced-choice obtained on a data set of 651 10, 14 and 15 digit strings collected from calls made over the AT&T network from mall payphones. The average string length is 13.0 digits. The recognition grammar allowed any number of digits to be recognized.

[19] In this subword recognition task, callers spoke words drawn from a list of 1000 New Jersey town names over the telephone. Utterances of the 100 names judged to be the least confusable and the 100 names judged to be the most confusable were recognized against a vocabulary containing all 1000 town names.