MICROSOFT WINDOWS HIGHLY INTELLIGENT SPEECH RECOGNIZER: WHISPER

Xuedong Huang, Alex Acero, Fil Alleva, Mei-Yuh Hwang, Li Jiang and Milind Mahajan
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052, USA

ABSTRACT

Since January 1993, we have been working to refine and extend Sphinx-II technologies in order to develop practical speech recognition at Microsoft. The result of that work has been the Whisper (Windows Highly Intelligent Speech Recognizer). Whisper represents significantly improved recognition efficiency, usability, and accuracy, when compared with the Sphinx-II system. In addition Whisper offers speech input capabilities for Microsoft Windows® and can be scaled to meet different PC platform configurations. It provides features such as continuous speech recognition, speaker-independence, on-line adaptation, noise robustness, dynamic vocabularies and grammars. For typical Windows® Command-and-Control applications (less than 1,000 words), Whisper provides a software only solution on PCs equipped with a 486DX, 4MB of memory, and a standard sound card and a desk-top microphone.

1. INTRODUCTION

To make Sphinx-II [10,6] usable in a PC environment, we need to tackle issues of recognition accuracy, computational efficiency, and usability simultaneously. A large amount of RAM and high-end workstation are unrealistic for today's popular PC environments where low-cost implementations are critical. The system must also be speaker adaptive, because there will always be some speakers for which the recognition error rate will be much higher than average due to variation in dialect, accent, cultural background, or simply vocal tract shape. The ability of the system to reject noise is also crucial to the success of commercial speech applications. Noises include not only environmental noises such as phone rings, key clicks, air conditioning noise, etc. but also vocal noises such as coughs; ungrammatical utterances, and Out Of Vocabulary (OOV) words. For a 20000-word dictation system, on average more than 3% of the words in an unconstrained test set are missing from the dictionary, and even when we increase the vocabulary size to 64,000 words, the OOV rate still remains higher than 1.5%. Lastly, recognition accuracy remains one of the most important challenges. Even if we exclude utterances containing OOV words, the word error rate of the best research systems remains to be higher than 9% for a 20,000-word continuous dictation task [6].

Whisper [8] not only inherited all the major features of state-of-the-art research system Sphinx-II, but it also incorporates context-free grammar decoding, noise rejection, improved channel normalization, and on-line speaker adaptation. Whisper supports Windows 95® and Windows NT®, and offers speaker-independent continuous speech recognition for typical Windows command and control applications. In this paper, we will selectively describe several strategies we used in Whisper to tackle efficiency and usability problems for command and control applications.

2. EFFICIENCY ISSUES

We have dramatically improved Whisper's computational and memory requirements. In comparison with Sphinx-II (under the same accuracy constraints), the necessary RAM was reduced by a factor of 20, and the speed was improved by a factor of 5. Efficiency issues are largely related to the size of models and search architecture [3], which is closely related to data structures and algorithm design as well as appropriate acoustic and language modeling technologies. In this section we discuss two most important improvements, namely acoustic model compression and context-free grammar search architecture.

By using the techniques described in the following sections, Whisper can run real-time on a 486DX PC offering speaker-independent continuous-speech recognition with a CFG required in a typical command-and-control application within 800KB of RAM, including code and all the data. In a command-and-control task with 260 words, Whisper offered a word error rate of 1.4%. For large-vocabulary continuous-speech dictation, more computational power (CPU and memory) is required in a trade-off with Whisper's error rate.

2.1 Acoustic model compression

The acoustic model typically requires a large amount of memory. In addition, likelihood computations for the acoustic model is a major factor in determining the final speed of the system. In order to accommodate both command-and-control and more demanding dictation applications, Whisper is scaleable, allowing several configurations of number of codebooks and number of *senones* [9]. In addition, to reduce the memory required by the acoustic model, Whisper uses a compression scheme that provides a good compression ratio while avoiding significant computational overhead for model decompression. The compression also offers improved memory locality and cache performance which resulted in a small improvement in speed.

Like discrete HMMs, semi-continuous HMMs (SCHMM) use common codebooks for every output distribution. Since the common codebooks are used for every senone output distribution, the output probability value for the same codeword entry is often

0-7803-2431-5/95 $4.00

identical for similar senones. For example, the context-independent phone *AA* uses about 260 senones for the top-of-the line 7000-senone configuration. These senones describe different context variations for the phone *AA*. We arranged the output probabilities according to the codeword index instead of according to senones, as is conventionally done. We observed a very strong output probability correlation within similar context-dependent senones. This suggested to us that compressing output probabilities across senones may lead to some savings. We compressed all the output probabilities with run-length encoding. The run-length encoding is lossless and extremely efficient for decoding. To illustrate the basic idea, we display in Table 1 the output probabilities of senones 1 through 260 for phone *AA*.

	Sen 1	Sen 2	Sen 3	Sen 4	...
Codeword 1	0.020	0.020	0.020	0.0	...
Codeword 2	0.28	0.30	0.020	0.0	...
Codeword 3	0.035	0.035	0.035	0.035	...
Codeword 4	0.0	0.0	0.0	0.0	...
Codeword 5	0.0	0.0	0.0	0.0	...
Codeword 6	0.0	0.0	0.0	0.076	...
Codeword 7	0.0	0.0	0.0	0.070	...
Codeword 8	0.057	0.051	0.055	0.054	...
Codeword 9	0.057	0.051	0.054	0.051	...
...	...	...	...	...	...
Codeword 256	0.0	0.0	0.0	0.080	...

Table 1. Uncompressed acoustic output probabilities for a 7000-senone Whisper configuration with 4 codebooks. We show the probabilities for one of the codebooks.

In Table 1, the sum of each column equals 1.0, which corresponds to the senone-dependent output probability distribution. For the run-length encoding, we choose to compress each row instead of each column. This allows us to make full use of correlation among different senones. The compressed form is illustrated in Table 2, where multiple identical probabilities are encoded with only one value and repeat count. For example, in codeword 1, probability 0.020 appears successively in senones 1, 2, and 3 so, we encoded them with (0.020, 3) as illustrated in Table 2.

Codeword 1	(0.020,3), 0.0, ...
Codeword 2	0.28,0.30, 0.020, 0.0,...
Codeword 3	(0.035, 4), ...
Codeword 4	(0.0,4), ...
....	.....
Codeword 256	(0.0,3), 0.08,...

Table 2. Run-length compressed acoustic output probabilities for a 7000-senone Whisper configuration with 4 codebooks. We show the probabilities for one of the codebooks after run-length encoding of the values in Table 1.

The proposed compression scheme reduced the acoustic model size by more than 35% in comparison with the baseline [7]. It is not only a loss-less compression but also enables us to measurably speed up acoustic model evaluation in the decoder. This is because identical output probabilities no longer need to be evaluated in computing semi-continuous output probabilities. As such they can be precomputed before evaluating Viterbi paths.

2.2 Search architecture

Statistical language models based on bigrams and trigrams [12] have long been used for large-vocabulary speech recognition because they provide the best accuracy, and Whisper uses them for large-vocabulary recognition. However, when designing a command-and-control version of Whisper, we decided to use context-free grammars (CFG). Although they have the disadvantage of being restrictive and unforgiving, particularly with novice users, we use it as our preferred language model because it has advantages like (1) compact representation; (2) efficient operation; and (3) ease of grammar creation and modification for new tasks. Users can easily modify the CFG and add new words to the system. Whenever a new word is added for a non-terminal node in the CFG, a spelling-to-pronunciation component is activated to augment the lexicon.

The CFG grammar consists of a set of productions or rules that expand non-terminals into a sequence of terminals and non-terminals. Non-terminals in the grammar would tend to refer to high-level task specific concepts such as dates, font-names, etc. Terminals are words in the vocabulary. We allow some regular expression operators on the right hand side of the production as a notational convenience. We disallow left recursion for ease of implementation. The grammar format achieves sharing of sub-grammars through the use of shared non-terminal definition rules.

During decoding, the search engine pursues several paths through the CFG at the same time. Associated with each of the paths is a grammar state that describes completely how the path can be extended further. When the decoder hypothesizes the end of a word, it asks the grammar module for all possible one word extensions of the grammar state associated with the word just completed. A grammar state consists of a stack of production rules. Each element of the stack also contains the position within the production rule of the symbol that is currently being explored. The grammar state stack starts with the production rule for the grammar start non-terminal at its first symbol. When the path needs to be extended, we look at the next symbol in the production. If it is a terminal, the path is extended with the terminal and the search engine tries to match it against the acoustic data. If it is a non-terminal, the production rule that defines it, is pushed on the stack and we start scanning the new rule from its first symbol instead. When we reach the end of the production rule, we pop the ending rule off the stack and advance the rule below it by one position, over the non-terminal symbol, which we have just completed exploring. When we reach the end of the production rule at the very bottom of the stack, we have reached an accepting state in which we have seen a complete grammatical sentence.

In the sake of efficiency, the decoder does not actually pursue all possible paths. When a particular path is no longer promising, it is pruned. Pruning is a source of additional errors since the correct path, which looks unpromising now, may prove to be the best when all the data is considered. To relax our pruning heuristic we use a strategy that we have dubbed the *"Rich Get Richer"* (RGR). RGR enables us to focus on the most promising paths and treat them with detailed acoustic evaluations and relaxed path pruning thresholds. On the other hand, poor (less promising paths) will be extended but probably with less expensive acoustic evaluations and less forgiving path pruning thresholds. In this way locally

optimal candidates continue to receive the maximum attention while less optimal candidates are retained but evaluated using less precise (computationally expensive) acoustic and/or linguistic models. The RGR strategy gives us finer control over the creation of new paths. This is particularly important for CFG based grammars since the number of potential paths is exponential. Furthermore, RGR gives us control over the working memory size which is important for relatively small PC platforms.

One instance of RGR used in Whisper is the control over the level of acoustic detail used in the search. Our goal is to reduce the number of context dependent senone probability computations required. Let's define $\alpha(p,t)$ as the best accumulated score at frame t for all instances of phone p in the beam, and $b(p,t+1)$ as the output probability of the context-independent model for phone p at frame $t+1$. Then, the context-dependent senones associated with a phone p are evaluated for frame $t+1$ if

$$a \cdot \alpha(p,t) + b(p,t+1) > \lambda(t) - K$$

where $\lambda(t)$ is the accumulated score for the best hypothesis at frame t, and a and K are constants.

In the event that p does not fall with in the threshold, the senone probabilities corresponding to p are estimated using the context independent senones corresponding to p. In Table 3 we show the improvements of this technique for Whisper 20,000 word dictation applications.

Reduction in senone computation	80%	95%
Error Rate Increase	1%	15%

Table 3. Reduction in senone computation vs. error rate increase by using RGR strategy in a 7000-senone 20000-word configuration for Whisper.

3. USABILITY ISSUES

To make Whisper more usable, we to tackle problems such as environmental and speaker variations, ill-formed grammatical speech input, and sounds not intended for the system, and speaker adaptation.

3.1 Improved channel normalization

Cepstral mean normalization [11] plays an important role in robust speech recognition due to variations of channel, microphone, and speaker. However, mean normalization does not discriminate between silence and voice when computing the utterance mean, and therefore the mean is affected by the amount of noise included in the calculation. For improved speech recognition accuracy, we propose a new efficient normalization procedure that differentiates noise and speech during normalization, and computes a different mean for each one. The new normalization procedure reduced the error rate slightly for the case of same-environment testing, and significantly reduced the error rate by 25% when an environmental mismatch exists [2] over the case of standard mean normalization.

The proposed technique consists of subtracting a correction vector $\mathbf{r}_i$ to each incoming cepstrum vector $\mathbf{x}_i$:

$$\mathbf{z}_i = \mathbf{x}_i - \mathbf{r}_i$$

where the correction vector $\mathbf{r}_i$ is given by

$$\mathbf{r}_i = p_i(\mathbf{n} - \mathbf{n}_{avg}) + (1 - p_i)(\mathbf{s} - \mathbf{s}_{avg})$$

with p_i being the *a posteriori* probability of frame i being noise, $\mathbf{n}$ and $\mathbf{s}$ being the average noise and average speech cepstral vectors for the current utterance, and $\mathbf{n}_{avg}$ and $\mathbf{s}_{avg}$ the average noise and speech cepstral vectors for the database used to train the system. Since this normalization will be applied to then training utterances as well, we see that after compensation, the average noise cepstral vector for all utterances will be $\mathbf{n}_{avg}$, and the average speech vector for all utterances will be $\mathbf{s}_{avg}$. The use of the *a posteriori* probability allows a smooth interpolation between noise and speech, much like the SDCN and ISDCN algorithms [1].

Although a more sophisticated modeling could be used to estimate p_i, we made the approximation that it can be obtained exclusively from the energy of the current frame. A threshold separating speech from noise is constantly updated based on a histogram of log-energies. This results in a very simple implementation that is also very effective [2].

3.2 Noise rejection

The ability to detect and notify the user of utterances containing out-of-vocabulary words; ungrammatical utterances; and non-utterances such as phone rings, is essential to the usability of a recognizer. This is particularly true when the language model is a tight context free grammar, as users may initially have difficulty confining their speech to such a model. We have added rejection functionality to Whisper that assigns a confidence level to each segment in a recognition result, as well as to the whole utterance, which can be used for an improved user interface.

Previous work on detecting noise words includes an all-phone representation of the input [4], and use of noise-specific models [13,14]. We have observed for continuous small-vocabulary tasks that the path determined by the best context-independent senone score per frame is a relatively reliable rejection path. We use the output of a fully connected network of context-independent phones, evaluated by using a Viterbi beam search with a separate beam width that may be adjusted to trade off speed for rejection accuracy. Transitions between phones are weighted by phonetic bigram probabilities that are trained using a 60,000 word dictionary and language model.

We used one noise model and one garbage model in the system. The noise model is like a phonetic HMM; its parameters are estimated using noise-specific data such as phone rings and coughs. The garbage model is a one-state Markov model whose output probability is guided by the rejection path. A garbage word based on this model may be placed anywhere in the grammar as a kind of phonetic wildcard, absorbing or alerting to the user ungrammatical phonetic segments after recognition. The noise model, in turn, absorbs non-speech noise data.

We measure the rejection accuracy using a multi-speaker data set with a mixture of grammatical and ungrammatical utterances as well as noise. With our rejection models, Whisper rejects 76% of utterances that are ungrammatical or noise and 20% of misrecognized grammatical utterances, while falsely rejecting fewer than 3% of correctly recognized grammatical utterances. Feedback supplied by the user is used to train the confidence threshold; this increases per-speaker rejection accuracy, especially for non-native speakers.

One interesting result is that our confidence measures used for noise rejection can be used to improve recognition accuracy. Here, word transitions are penalized by a function of the confidence measure. So the less confident theories in the search beam are penalized more than theories that have higher confidence intervals, which provides us with different information than the accumulated probability for each path in the beam. This is in the same spirit of our general RGR strategy used throughout the system. We found that the error rate for our command and control task was reduced by more than 20% by incorporating this penalty [5].

3.3 Speaker Adaptation

To bridge the gap between speaker-dependent and speaker-independent speech recognition, we incorporated speaker adaptation as part of the Whisper system. We modify the two most important parameter sets for each speaker, i.e. the vector quantization codebooks (or the SCHMM mixture components) and the output distributions (or the SCHMM mixing coefficients) in the framework of semi-continuous models. We are interested in developing adaptation algorithms that are consistent with the estimation criterion used in either speaker-independent or speaker-dependent systems. We observed in general a 50% error reduction when a small amount of enrollment data is used. The adaptation is particularly important for non-native English speakers

4. SUMMARY

We have significantly improved Whisper's accuracy, efficiency and usability over the past two years. On a 260-word Windows continuous command-and-control task and with 800KB working memory configuration (all the RAM required, including code and data), the average speaker-independent word recognition error rate was 1.4% on a 1160 utterance testing set. The system runs real-time on a PC equipped with a 486DX and 4MB of memory.

The emergence of an advanced speech interface is a significant event that will change today's dominant GUI-based computing paradigm. It is obvious that the paradigm shift will require not only accurate speech recognition, but also integrated natural language understanding as well as a new model for building application user interfaces. The speech interface cannot be considered highly intelligent until we make it transparent, natural, and easy to use. Through our ongoing research efforts, we believe that we can continue to push the quality of our system above and beyond what is implied by its acronym.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to Douglas Beeferman, Jack McLaughlin, Rick Rashid, and Shenzhi Zhang for their help in Whisper development.

REFERENCES

[1] Acero, A. "Acoustical and Environmental Robustness in Automatic Speech Recognition". *Kluwer Publishers.* 1993.

[2] Acero, A. and Huang, X. "Robust Mean Normalization for Speech Recognition". *US Patent pending,* 1994.

[3] Alleva F., Huang X., and Hwang M. "An Improved Search Algorithm for Continuous Speech Recognition". *IEEE International Conference on Acoustics, Speech, and Signal Processing,* 1993.

[4] Asadi O., Schwartz R., and Makhoul, J. "Automatic Modeling of Adding New Words to a Large-Vocabulary Continuous Speech Recognition System". *IEEE International Conference on Acoustics, Speech, and Signal Processing,* 1991.

[5] Beeferman, D. and Huang, X. "Confidence Measure and Its Applications to Speech Recognition". *US Patent pending.* 1994.

[6] Huang X., Alleva F., Hwang M., and Rosenfeld R. "An Overview of Sphinx-II Speech Recognition System". *Proceedings of ARPA Human Language Technology Workshop,* March 1993.

[7] Huang, X and Zhang, S. "Data Compression for Speech Recognition". *US Patent pending,* 1993.

[8] Huang X., Acero A., Alleva F., Beeferman D., Hwang M., and Mahajan M. "From CMU Sphinx-II to Microsoft Whisper - Making Speech Recognition Usable", in *Automatic Speech and Speaker Recognition - Advanced Topics.* Lee, Paliwal, and Soong editors, *Kluwer Publishers,* 1994.

[9] Hwang M. and Huang X. "Subphonetic Modeling with Markov States -- Senone". *IEEE International Conference on Acoustics, Speech, and Signal Processing,* 1992.

[10] Lee, K.F.,"Automatic Speech Recognition: The Development of the SPHINX System". *Kluwer Publishers,* Boston 1989.

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

[12] Jelinek, F. "Up From Trigrams". *Proceedings of the EuroSpeech Conf,* Geneva, Italy 1991.

[13] Ward W. "Modeling Non-Verbal Sounds for Speech Recognition". *Proceedings of DARPA Speech and Language Workshop,* October 1989.

[14] Wilpon J., Rabiner L., Lee C., and Goldman. "Automatic Recognition of Keywords in Unconstrained Speech using Hidden Markov Models". *IEEE Trans. on Acoustics, Speech, and Signal Processing,* Vol.- ASSP-38, pp. 1870-1878, 1990.