ĐỀ XUẤT NHẬN DẠNG TIẾNG VIỆT NAM CHO ĐIỆN THOẠI DI ĐỘNG

(Nguyễn Văn Khiêm, Lê Quân Hà, Hoàng Tiến Long, Nguyễn Hữu Tình, Nguyễn Ngọc Thắm, Đỗ Hồng Thy)*

▪ TÓM TẮT

Đã 20 năm qua, nhận dạng tiếng nói vẫn là một nỗ lực lớn để tạo ra trí tuệ cho máy tính, nỗ lực không ngừng này đã mang lại ứng dụng trong quản lý điện thoại. Khởi đầu với nhận dạng đọc các chữ số từ 0 đến 9 trong ứng dụng này (digit recognition), sau đó là các bài toán nhận dạng các từ cô lập (isolated word recognition). Từ sau thập niên 90, chúng ta bắt đầu bước vào lĩnh vực nhận dạng tiếng nói với từ vựng lớn, khi đó các yêu cầu về yếu tố bền vững của nhận dạng tiếng nói trở nên cần thiết, nghĩa là: hệ thống không dễ đổ vỡ khi gặp bất kỳ một lỗi nhận dạng, một lỗi phần mềm; khi gặp một tình huống nhận dạng ngoài ý muốn, hệ thống dễ dàng khôi phục để tiếp tục tiến trình nhận dạng liên tục.

Sự xuất hiện của việc nhận dạng tiếng nói trên điện thoại di động và các thiết bị nhúng đã mở ra loại hình nghiên cứu mới về các ứng dụng tương tác giữa con người và máy tính. Nhưng hầu hết các hoạt động trong lĩnh vực này đến nay đã bị giới hạn do các vấn đề về độc quyền phần mềm, hoặc chỉ nhận dạng những câu có cấu trúc ngữ pháp đơn giản và bị hạn chế. Trong phần nghiên cứu này, chúng tôi sẽ trình bày sơ lược về Pocket Sphinx, một hệ thống mã nguồn mở về nhận dạng tiếng nói liên tục từ vựng lớn trên các thiết bị cầm tay. Chúng tôi đã nhúng được tiếng Việt từ vựng lớn là 7660 từ tiếng Việt, đạt độ chính xác là 98,13% tỉ lệ lỗi từ 1,87%.

PROPOSAL FOR IETNAMESE RECOGNITION FOR MOBILE PHONE

▪ *ABSTRACT*

Over the past 20 years, speech recognition has been still a major effort to create intelligence for the computer. The ceaseless effort has brought the application in the phone management. We started with recognition of reading numbers from 0 to 9 in this application (digit recognition), then the problems of isolated word recognition. Since the 1990s onwards, we have started to enter a field of speech recognition with large vocabulary. Thus, requests for the sustainability element of the speech recognition becomes necessary, that is, the system is not easily broken to meet any recognized errors or a software errors. When a situation of unintended recognition is encountered, the system shall easily restore to continue the ongoing process of recognition.

* Khoa Công Nghệ Thông Tin, trường Đại Học Công Nghiệp Tp.HCM

▪ GIỚI THIỆU

Các ứng dụng về tiếng nói trên thiết bị nhúng, điện thoại di động thường đòi hỏi phải liên tục và nhận dạng ở thời gian thực. Rất nhiều ứng dụng về giọng nói hiện tại, chẳng hạn như điều khiển chuyển hướng của hệ thống định vị toàn cầu, chọn nhạc cho máy hát nhạc, hoặc các ứng dụng về ngôn ngữ tự nhiên như thiết bị chuyển đổi ngôn ngữ từ giọng nói (speech-to-speech translation) [tham khảo thêm A.Waibel, A. Badran, A. W Black, R. Frederking, D. Gates, A. Lavie, L. Levin, K. Lenzo, L. Mayfield Tomokiyo, J.Reichert, T. Schultz, D. Wallace, M. Woszczyna, và J. Zhang 2003],... đều đòi hỏi phải nhanh, chính xác và linh động.

Việc triển khai và cài đặt các ứng dụng trên các thiết bị nhúng gặp rất nhiểu khó khăn, trong đó khó khăn lớn nhất là yêu cầu nhận dạng giọng nói liên tục cho một ngữ cảnh từ vựng từ vừa đến lớn. Ngoài ra còn có các trở ngại về phần cứng: CPU của thiết bị nhúng không hỗ trợ kiểu dấu phẩy động, bộ nhớ RAM thiếu, khả năng lưu trữ và băng thông trên thiết bị nhúng cũng rất hạn chế. Vì những lý do này, mà các công việc về nhận dạng tiếng nói trước đây [xem H. Franco, J. Zheng, J. Butzberger, F. Cesari, M. Frandsen, J. Arnold, V. R. R. Gadde, A. Stolcke, và V. Abrash 2002], [T. W. Ḳohler, C. Ṷugen, S. Ṩtuker, và A. Waibel 2005] chỉ giới hạn vào nhận dạng những câu có cấu trúc ngữ pháp đơn giản.

Ngoài những hạn chế về phần cứng, chúng ta còn phải đối mặt với trở ngại trong việc xây dựng hệ thống nhận dạng. Để xây dựng hệ thống này đòi hỏi phải sử dụng các bộ công cụ, nhưng những bộ công cụ này thường có bản quyền với giá rất đắt và không có mã nguồn kèm theo. Đồng thời, các hệ điều hành trên thiết bị nhúng thường bị thiếu các tính năng cho các nhà phát triển không giống như trên các hệ thống máy tính để bàn.

▪ HỆ THỐNG OCKETSPHINX

Bộ nhận dạng SPHINX là nền tảng rất tốt cho sự phát triển nhận dạng giọng nói, và chúng đang được sử dụng bởi các nhà nghiên cứu trong các lĩnh vực ví dụ như: hệ thống đối thoại và hệ thống máy tính hỗ trợ học tập... Trong số các bộ nhận dạng CMU SPHINX, PocketSphinx là công cụ đã được tối ưu cho nhận dạng tiếng nói trên thiết bị nhúng và điện thoại di động.

■ TỐI ƯU HÓA

Do phần cứng của thiết bị nhúng và điện thoại di động so với máy PC có nhiều khác biệt cho nên có các lưu ý sau:

- Tốc độ truy cập bộ nhớ chậm
- Tổ chức dữ liệu sao cho tương thích với phần cứng CPU
- Cần thay đổi các đoạn mã không phù hợp hệ thống.

Vì vậy cần phải thực hiện một số tối ưu sau.

A. Tối ưu hóa bộ nhớ

+ Ánh xạ tập tin I/O vào bộ nhớ: Đối với các thiết bị nhúng có bộ nhớ RAM rất ít, dữ liệu của mô hình cú âm nên đặt ở chế độ read-only để nó có thể được đọc trực tiếp từ ROM. Trên hệ điều hành của các thiết bị nhúng, bộ nhớ ROM được cấu trúc như một tập tin hệ thống, và như vậy nó có thể được truy cập trực tiếp bằng cách sử dụng chức năng ánh xạ tập tin vào bộ nhớ, chẳng hạn như mmap() trên UNIX hoặc MapViewOfFile() trên Windows.

+ Byte ordering: PocketSphinx cần có định dạng của dữ liệu khác với dữ liệu của SPHINX để cho phép chúng được ánh xạ vào bộ nhớ. Vì vậy cần sửa đổi bộ huấn luyện HMM, SPHINXTRAIN, để đầu ra các tập tin phù hợp với hệ thống, vì vậy cho phép ánh tập tin này vào bộ nhớ theo đúng trật tự byte.

+ Định tuyến dữ liệu: Các CPU ngày nay đều có hỗ trợ định tuyến dữ liệu. Ví dụ, một trường dữ liệu 32-bit thì được yêu cầu gán cho các địa chỉ có giá trị 4-byte. Bởi vì các trường dữ liệu trong các file mô hình có độ dài khác nhau, nên chúng ta cần phải thêm dữ liệu vào cuối nó. Kết quả là trong khi phiên bản hiện tại có thể đọc được các file mô hình từ các phiên bản trước, thì các file được tạo ra từ nó không thể tương thích ngược.

+ Ánh xạ Triphone-senone: Tốc độ, nhỏ gọn là mục tiêu của PocketSphinx. Theo thực nghiệm mô hình dữ liệu của PocketSphinx nên lưu ở cấu trúc dạng cây , đây là giải pháp tốt nhất nâng cao năng suất bộ nhớ. Kết quả sử dụng bộ nhớ đã giảm và thời gian khởi động nhanh hơn.

B. Tối ưu hóa cấp thấp

+ Sử dụng dấu phẩy tĩnh: Bộ vi xử lý Strong ARM không hỗ trợ các toán tử cho kiểu dấu phẩy động. Vì thế sự tính toán trên dấu phẩy động sẽ được mô phỏng trong phần mềm bằng cách sử dụng các phép toán được cung cấp bởi trình biên dịch hay của các thư viện runtime. Nó sẽ giả lập các chức năng của một bộ xử lý dấu chấm động, nhưng như vậy sẽ làm cho việc tính toán trên các con số lớn rất chậm, như việc lấy ra đặc tính cú âm và tính toán Gaussian. Vì vậy ta có thể biểu diễn một số thập phân thành phân số và mẫu số thường dùng là một số chia hết cho hai (cho hiệu quả tốt nhất).

Việc sử dụng dấu phẩy tĩnh chắc chắn liên quan đến một số lỗi làm tròn, nó xảy ra sau mỗi lần thực hiện phép tính. Việc chọn thuật toán không những phải đảm bảo làm giảm số lượng tính toán, tăng tốc độ, đồng thời phải duy trì độ chính xác. Ví dụ, dùng FFT để tách số một số thực thành phần nguyên và phần thập phân [H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus vol. 35, no. 6, pp. 849–863, 1987]. Tuy nhiên, khi sử dụng FFT trên dấy phẩy tĩnh đã làm tăng đáng kể tỷ lệ lỗi từ, trong một số trường hợp lên đến 20%.

+ Tối ưu hóa dữ liệu và cấu trúc điều khiển: Kiến trúc ARM đã được tối ưu hoá rất nhiều cho việc tính toán trên kiểu dữ liệu số nguyên và Boolean. Hầu hết các dữ liệu được cung cấp bao gồm một trường "shift count", cho phép dịch chuyển bit theo một giá trị mà không làm thay đổi giá trị ban đầu. ARM là một kiến trúc 32-bit với 16 thanh ghi đa năng. Việc giữ dữ liệu trong các thanh ghi rất quan trọng để thực hiện các phép toán, nó giúp làm nhanh hơn việc truy cập vào bộ nhớ 32 bit tại một thời điểm, vì vậy tránh truy cập trực tiếp vào thanh ghi khi có thể. Nhìn chung, một trình biên dịch tối ưu hóa tốt có thể tạo ra hiệu quả sử dụng tập tin đăng ký.

Trong PocketSphinx, danh sách senones được cài đặt trong mảng byte. Tuy nhiên, khi mảng byte lớn được tải lên bộ đệm của bộ xử lý, và việc truy cập lượng lớn byte sẽ làm chậm tốc độ của CPU. Vì thế, chúng được thay đổi sang một véc tơ bit, dùng vòng lặp để quét vector bit này, và thao tác 32-bit word cùng một thời điểm.

TỐI ƯU HÓA THUẬT TOÁN

Chúng tôi thấy rằng số lượng lớn các tính toán được dùng trong bốn khu vực: tính

toán đặt tính cú âm (MFCC), tính toán Gaussian (codebook), tính toán mô hình Gaussian hỗn hợp, và đánh giá HMM (tìm kiếm Viterbi). Khoảng tỷ lệ các thời gian được dùng trong bốn lĩnh vực được hiển thị trong Bảng 1.

Trong việc tối ưu hóa thuật toán, tính toán mô hình hỗn hợp Gaussian (GMM) được chú ý nhiều nhất, từ công việc trước đó [A. Chan et al. 2004] chúng ta đã có framework rất tốt cho tính toán gần đúng GMM. Trong framework này, GMM ước lượng được chia làm 4 tầng tính toán:

Bảng 1. Tỷ lệ phần trăm thời gian tính toán

Thành phần	Desktop	Nhúng
Codebook	27.43%	24.59%
HMM	24.68%	22.11%
MFCC	14.39%	11.51%
Senone	7.67%	11.71%

- Tầng khung: tính tất cả GMM cho khung dữ liệu đầu vào.
- Tầng GMM: tính toán một GMM đơn.
- Tầng Gaussian: tính toán một Gaussian đơn.
- Tầng thành phần: tính toán các thành phần liên quan đến vector đặc tính.

Lược đồ này cho phép phân loại chính xác các kỹ thuật nâng tốc độ khác nhau bởi các tầng mà chúng hoạt động trên đó, và cho phép chúng ta xác định các kỹ thuật khác nhau được áp dụng kết hợp với nhau như thế nào. Tuy nhiên, framework này được áp dụng chủ yếu cho hệ thống sử dụng HMM phân bố liên tục (CDHMM). Trong việc áp dụng các ý tưởng của nó cho mô hình HMM bán liên tục (SCHMM), có một vài sự khác biệt cần lưu ý:

- Trong việc tính toán của mô hình cú âm bán liên tục, một "codebook" của mật độ Gaussian được chia sẻ giữa tất cả các mô hình hỗn hợp.
- Số lượng Gaussians hỗn hợp thường là 128 đến 2048, lớn hơn nhiều so với 16-32 được sử dụng cho CDHMM.
- Hệ thống SCHMM cơ sở thường là miêu tả cho các vector đặc tính với nhiều luồng độc lập.

Việc áp dụng mô hình bốn tầng, với mỗi tầng có cấu trúc khác nhau. Nhưng trong khi đó codebook được chia sẻ giữa tất cả các tầng, toàn bộ codebook phải được tính ở mỗi tầng. Giới hạn này cho phép các phép tính ở tầng GMM có thể giảm bớt sự tính toán. Chúng ta áp dụng kỹ thuật sau đây cho mỗi tầng:

- **Tầng khung:** Chúng ta áp dụng chuẩn hóa khung (downsampling) [M. Woszczyna 1998]. Mặc dù các kết quả này làm mất tính chính xác, nhưng nó là cách duy nhất để nâng tốc độ trên tầng Gaussian.
- **Tầng GMM:** Áp dụng GMM cơ sở độc lập ngữ cảnh [A. Lee, T. Kawahara, and K. Shikano 2001, vol. 1, pp. 69–72], [A. Chan, M. Ravishankar, and A. Rudnicky 2005].
- **Tầng Gaussian:** Chúng tôi xem xét nhiều khả năng như: Sub-VQ-based Gaussian Selection [M. Ravishankar, R. Bisiani, and E. Thayer 1997, pp. 151–154] nhưng tất cả chúng đều không cải thiện được tốc độ nhiều. Vì thế, chúng tôi quyết định sử dụng phương pháp cây cơ sở Gaussian Selection.
- **Tầng thành phần:** PocketSphinx có

sẵn thành phần tính Gaussian [xem B. Pellom, R. Sarikaya, and J. H. L. Hansen vol. 8, no. 8, pp. 221–224, July 2001]. Sau đó sử dụng thông tin từ cây cơ sở để cải thiện hiệu quả tính toán của thành phần này.

Trong tầng khung, chúng tôi bước đầu áp dụng chuẩn hóa khung (downsampling) một cách đơn giản, bởi chỉ cần bỏ qua tất cả các codebook và sự tính toán GMM ở mọi khung khác. Tuy nhiên, chúng ta chỉnh sửa điều này sau để tính lại đỉnh *N* Gaussians từ khung trước và sử dụng cái đó để tính các senones từ khung hiện tại. Kết quả: thực hiện nhanh hơn khoảng nhỏ (0.6%) và kết quả đạt được tỉ lệ lỗi từ giảm khoảng 10%

Trong tầng Gaussian, chúng tôi áp dụng phiên bản chỉnh sửa của thuật toán BBI, như được mô tả trong [B. Pellom, R. Sarikaya, and J. H. L. Hansen vol. 8, no. 8, pp. 221–224, July 2001]. Thuật toán này đưa bộ Gaussians vào trong cấu trúc cây *kd* điều này cho phép bộ con Gaussian tìm kiếm nhanh trong không gian đặc tính để đưa ra vector đặc tính. Đối với mỗi dòng đặc tính cú âm trong codebook, chúng tôi xây dựng một cây có độ sâu riêng biệt (thường độ sâu 8 hoặc 10) với hộp ngưỡng Gaussian đã định sẵn.

Mặc dù các loại cây được xây dựng ngoại tuyến, chiều sâu của tìm kiếm trong cây có thể được điều khiển như một tham số để giải mã tại thời gian chạy. Cái này cho phép các yêu cầu bộ nhớ cho cây không nhiều lắm. Chúng tôi cũng khám phá ý tưởng hạn chế số lượng tối đa Gaussians để tìm kiếm trong mỗi nút lá. Để thực hiện khả thi, chúng tôi đã sắp xếp danh sách các Gaussians trong các nút lá.

■ DỮ LIỆU VÀ HUẤN LUYỆN TIẾNG VIỆT

A. Lexicon

Lexicon là bộ từ điển đng để thể hiện ếc từ thnh ếc đơn vị phí m (phonemes). Nĩ l một thnh phần quan trọng trong hệ thống nhận dạng tiếng nĩi. Chng tơi đ xy dựng được lexicon tiếng Việt theo phiên âm chuẩn quốc tế. Lexicon tiếng Việt hơn 12 nghìn từ sử dụng 41 phonemes cho cả hai miền Nam và Bắc.

B. Dữ Liệu

Dữ liệu học là một phần không thể thiếu trong nhận dạng tiếng nói. Dữ liệu học quyết định trực tiếp đến kết quả nhận dạng. Dữ liệu học gồm hai phần là dữ liệu văn bản và dữ liệu âm thanh. Dữ liệu âm thành là những tập tin âm thanh thu âm những câu trong dữ liệu văn bản.

C. Dữ liệu văn bản

Tùy vào mục đích của việc nghiên cứu và chương trình ứng dụng nhận dạng tiếng nói khác nhau thì có bộ dữ liệu văn bản khác nhau. Thường thì bộ dữ liệu văn bản được chọn theo chủ đề của ứng dụng.

D. Dữ liệu âm thanh

Dữ liệu âm thanh phụ thuộc vào bộ dữ liệu văn bản. Nó bao gồm tất cả các tập tin âm thanh thu âm các câu trong bộ dữ liệu văn bản. Bộ dữ liệu văn bản cho nhận dạng số gồm 200 câu thì bộ dữ liệu âm thanh là 200 tập tin âm thanh. Chúng tôi ghi âm dữ liệu thành tập tin có đuôi là .raw. Tập tin âm thanh .raw có độ nén cao, dung lượng nhỏ thích hợp cho việc ghi âm dữ liệu lớn.

Một tập tin âm thanh chuẩn là một tập tin không có tiếng ồn và nhiễu, các từ

phải được đọc rõ ràng.

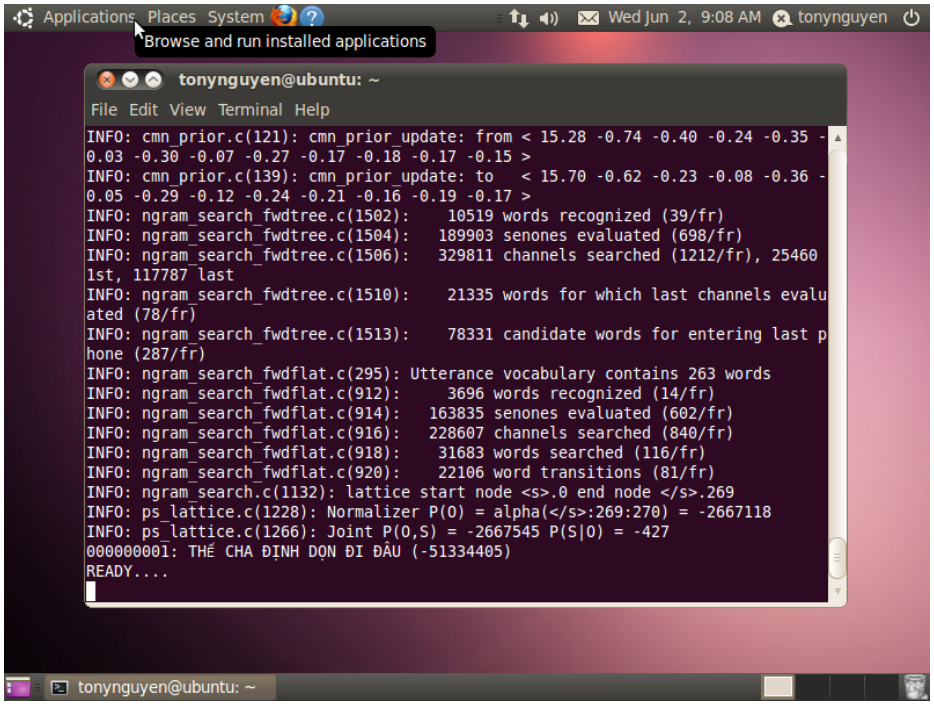
Bộ dữ liệu âm thanh phải được thu âm rõ ràng, dứt khoát từng từ. Người thu âm dữ liệu học cũng đóng vai trò rất quan trọng. Người thu âm nằm trong độ tuổi từ 18 đến 51 chia đều theo độ tuổi, cân bằng cả giọng nam và giọng nữ. Số lượng người thu âm lớn, trải đều theo lứa tuổi, cân bằng số nam và nữ làm cho hệ thống trở lên phong phú hơn, linh hoạt hơn và khả năng thích ứng cao. Ví dụ như huấn luyện 1000 người đọc, khi người thứ 1001 đọc thì hệ thống dễ dàng thích nghi với giọng của người này và cho kết quả nhận dạng chính xác.

E. Tiếng ồn và nhiễu trong dữ liệu âm thanh

Tiếng ồn, nhiễu ảnh hưởng rất lớn đến quá trình học dữ liệu và nhận dạng. Tiếng ồn, nhiễu là do nhiều nguyên nhân như tiếng ồn xe cộ, công trường, người nói chuyện..., và nhiễu chủ yếu là do microphone.

■ HUẤN LUYỆN TIẾNG VIỆT

Bộ công cụ huấn luyện được chúng tôi sử dụng là SphinxTrain: đây là công cụ huấn luyện tiếng Anh phát triển bởi trường đại học Carnegie Mellon. Nay với đầu vào của chúng tôi là dữ liệu tiếng Việt. Với cấu trúc định sẵn, quá trình huấn luyện thực hiện sẽ tạo ra các file mô hình HMM của tiếng Việt, các tập tin HMM này đã được đưa vào PocketSphinx .



Hình 1. Ứng dụng nhận dạng tiếng nói Việt Nam trên điện thoại di động đang nhận dạng câu "THẾ CHA ĐỊNH DỌN ĐI ĐÂU".

■ KẾT LUẬN VÀ CÔNG VIỆC SẮP TỚI

Trong tương lai, chúng tôi sẽ áp dụng hệ thống này cho một công việc với mô hình ngôn ngữ cao hơn và vốn từ vựng lớn hơn. Một ứng cử viên cho sự tối ưu hóa trong tương lai là thuật toán tìm kiếm Viterbi, cái mà chúng tôi đã không thảo

luận sâu vào trong phần này. Chúng tôi cũng đã triển khai POCKETSPHINX để nhận dạng tiếng nói Việt Nam trên hệ máy phổ biến Pocket PC, hệ điều hành Windows®CE và hệ điều hành Linux.

CÁC KẾT QUẢ ĐẠT ĐƯỢC

- Xây dựng xong bộ từ điển lexicon tiếng Việt hơn 12 nghìn từ.

- Xây dựng xong mô hình ngôn ngữ cho tiếng Việt với dữ liệu hơn 20.000 từ.

- Đã huấn luyện được mô hình cú âm cho tiếng Việt.

- Đã nhúng được tiếng Việt cho PocketSphinx.

- Độ chính xác nhận dạng PocketSphinx với kích thước 7660 từ tiếng Việt, đạt độ chính xác là 98,13% tỉ lệ lỗi từ 1,87% dựa trên 150 câu kiểm tra từ vựng lớn

TÀI LIỆU THAM KHẢO

[1]. Waibel, A. Badran, A. W Black, R. Frederking, D. Gates, A. Lavie, L. Levin, K. Lenzo, L. Mayfield Tomokiyo, J. Reichert, T. Schultz, D. Wallace, M. Woszczyna, and J. Zhang. 2003. Speechalator: Two-way speech-to-speech translation in your hand. In Proceedings of NAACL-HLT.

[2]. Kohler, T. W., Fugen, C., Stuker, S. and Waibel, A. 2005. Rapid porting of ASR-systems to mobile devices. In Proceedings of Interspeech.

[3]. Franco, H., Zheng, J., Butzberger, J., Cesari, F., Frandsen, M., Arnold, J., Gadde, V. R. R., Stolcke, A. and Abrash V. 2002. Dynaspeak: SRI's scalable speech recognizer for embedded and mobile systems. In Proceedsings of HLT.

[4]. Chan, A., Sherwani, J., Ravishankar, M. and Rudnicky, A. 2004. Four-layer categorization scheme of fast GMM computation techniques in large vocabulary continuous speech recognition systems. In Proceedings of ICSLP.

[5]. Chan, A., Ravishankar, M. and Rudnicky, A. 2005. On improvements of CI-based GMM selection. In Proceedings of Interspeech.

[6]. Lee, A., Kawahara, T. and Shikano, K. 2001. Gaussian mixture selection using context-independent HMM. In Proceedings of ICASSP, vol. 1, pp. 69–72.

[7]. Waibel, A., Badran, A., Black, A. W., Frederking, R., Gates, D., Lavie, A., Levin, L., Lenzo, K., Tomokiyo, L. M., Reichert, J., Schultz, T., Wallace, D., Woszczyna M. and Zhang J. 2003. Speechalator: Two-way speech-to-speech translation in your hand. In Proceedings of NAACL-HLT.

[8]. Acero, A. and Stern, R. M. 1990. Environmental Robustness in Automatic Speech Recognition. Proc. of ICASSP, pp. 849-852.

[9]. Aubert, X. and Dugast, C. 1995. Improved Acoustic-Phonetic Modelling in Philip's Dictation System by Handling Liaisons and Multiple Pronunciations. Proc. of EuroSpeech'95, vol. 2, pp. 767-770.

[10]. Pellom, B., Sarikaya, R. and Hansen, J. H. L. 2001. Fast likelihood computation techniques in nearest-neighbor based search for continuous speech recognition. IEEE Signal Processing Letters, vol. 8, no. 8, pp. 221–224.

[11]. Bahl, L. R. and Bakis, R. 1989. Large Vocabulary Natural Language Continuous Speech Recognition. Proc of ICASSP'89, pp.465-467.

[12]. Bahl, L. R., Brown, P. F., de Souza, P. V. and Mercer, R. L. 1986. Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. Proc. of ICASSP'86, pp.49-52.

[13]. Bahl, L. R., de Souza, P. C., Gopalakrishnan, P. S., Nahamoo, D., Picheny, M.A. and Watson, T. J. 1994. Robust Methods for Using Context-Dependent Features and Models in a

Continuous Speech Recognizer. Proc. of ICASSP'94, pp.I533-I536.

[14]. Bahl, L. R., Jelinek, F. 1975. Decoding for Channels with Insertions, Deletions and Substiutions, with Applications to Speech Recognition. IEEE Trans. Information Theory, IT-21, pp. 404-411.

[15]. Kawahara, L. T. and Shikano K. 2001. Gaussian mixture selection using context-independent HMM. In Proceedings of ICASSP, vol. 1, pp. 69–72.

[16]. Gold, B. and Morgan, N. 2000. Speech and Audio Signal Processing. John Wiley & Sons, INC, New York.

[17]. Bourland, H. 1995. Towards Increasing Speech Recognition Error Rates. Proc. of EuroSpeech'95, vol. 2, pp. 883-894.

[18]. Lee, K-F., Hon, H-W. and Reedy, R. 1990. An Overview of the SPHINX Speech Recognition System. IEEE Trans. on Acoustic, Speech, Signal Processing, vol. 38, pp.35-45.

[19]. Levinson, S. E. 1986. Continuously Variable Duration Hidden Markov Models *for Automatic Speech Recognition. Computer Speech and Language, 1(1), pp. 29-45.*

[20]. Trask, R. L. 1996. A Dictionary of Phonetics and Phonology. Routledge.

[21]. Young, S. J., Oh, Y. H. and Shin, G. C. 1997. Improved Lexicon Modeling for Continuous Speech Recognition. Proc. of ICASSP'97, pp.1827-1830.