제15회 음성통신 및 신호처리 워크샵(KSCSP '98 15권1호)

C++ 언어와 Standard Library를 이용한 음성인식기 개발

황규웅, 권오욱, 박준, 서영주
한국전자통신연구원 음성언어팀

Development of a Speech Recognition System using C++ Language and Standard library

Kyuwoong Hwang, Oh-Wook Kwon, Jun Park, Young-Ju Suh
Spoken Language Team, ETRI
{hkw,owkwon,junpark}@zenith.etri.re.kr

요약

우리는 C++를 이용하여 음성인식기를 구현하여 기존의 C를 이용한 경우에 비하여 30% 수준의 소스로 표현하였고 인식기의 공동개발, 확장 및 개선, 기술 전수 등이 용이하게 되었으며 이를 음성인식 엔진 및 음성인식 연구를 위한 툴로 사용할 수 있게 되었다. 이 인식기의 특징으로는 연속 음성 및 대화체 음성을 인식할 수 있으며 trigram 언어 모델을 사용하였고 문맥 종속 음소 모델링에서는 기존의 triphone보다 넓은 문맥을 고려하는 n-phone context modeling을 사용하였으며 모델의 선정에는 음성학적 지식을 기반으로 한 질문을 사용한 decision tree를 사용하여 훈련에 나타나지 않은 단어나 문맥인 경우라도 가장 가까운 모델을 선정할 수 있게 하였다. 또, tree lexicon을 사용하여 속도를 개선하였으며 state 단위의 모델 공유를 통해 제한된 데이터를 이용하여 더 많은 모델을 훈련할 수 있어 성능을 개선하였다. 상용화를 염두에 두고 PC에서 구현하였다.

1 서론

최근의 음성인식시스템은 그 크기가 100,000 줄에 이르는 방대한 프로그램이고 여러 사람이 공동 개발하게 되는 경우가 많다. 또 연구용으로 사용되려면 그 내용을 자주 수정하여야 할 필요가 있게 된다. 특히 연구에 새로이 참여하는 사람의 경우 전체 시스템을 파악하는데 만도 많은 시간과 노력이 소요되며 기술 전수를 할 경우에는 더 많은 어려움이 있다. 이러한 문제점들을 해결하기 위해 우리는 기존의 인식기를 C++ 언어와 C++ 언어의 ANSI 표준 library인 standard library를 이용하여 재 작성하였다[1]. 본 논문에서는 이 과정에서 획득한 효율적인 인식기를 작성하기 위한 고려 사항, 메모리 사용과 인식 속도를 개선하기 위한 고려 사항, ETRI 음성인식기의 특징에 대해 기술하였다.

2 C++와 Standard library[6]

다음은 C++ 언어와 standard library를 사용하게 되어 얻은 이익이다.

- 효율적인 표현 : 소스의 크기가 30%로 줄어들었다. 이는 더 짧은 시간에 시스템을 파악할 수 있게 하고 코드 자체의 양이 줄어 오류의 가능성을 줄여준다. 또한 객체를 사용하여 이해하기 쉽고 양도 적은 소스가 되었다.
- standard library: 이 library에서 제공하는 표준 data structure(list, map, vector, string, ...)와 algorithm(find, sort, ...)을 사용하여 작은 프로그램을 만들었고 Standard library를 경험한 사용자를 쉽게 이해할 수 있도록 하였다. 오류를 일으키기 쉬운 malloc, new, delete를 거의 사용하지 않았다.

3 인식엔진과 음성인식연구 도구

본 인식기는 음성인식엔진과 음성인식연구의 도구로 사용될 것을 목표로 하였다. 이를 위해 다음의 사항이 고려 되었다.

- 외부에서 여러 단계의 자세한 정도로 인식기를 제어할 수 있어 인식 엔진으로 사용 가능하다.
- 음성인식기를 개발할 경우에 탐색자체는 소스에서 그렇게 많은 부분을 차지하지 않지만 모델의 선정, distribution과 codebook의 준비 등이 많은 부분을 차지한다. 본 인식기에서는 주변 지원 모듈을 클래스로 만들고 인터페이스를 단순화해 새로운 탐색 알고리즘을 개발할 경우 지원 모듈을 그대로 재활용할 수 있도록 하였다. 이러한 특징은 새로운 알고리즘을 구현하기 위해 필요한 주변 노력을 줄여 주어 아이디어를 쉽게 구현할 수 있도록 하여 주어 이 인식기가 음성인식연구의 library 및 툴로 사용될 수 있도록 하였다.

  일반적으로 연구용 시스템은 Workstation에서 개발하게 되어 상용화하기 위해서는 PC 버전으로 바꾸는 단계가 필요하였다. 이 인식기는 상용화가 쉽게 가능하도록 Workstation이 아닌 PC Windows에서 개발하였다.

- codebook이나 distribution의 경우는 그 정의나 외부와의 작용이 명확하므로 쉽게 객체로 표현되었다.
- 일반적인 HMM 방식의 음성인식기는 Gaussian Mixture로 표현되는 출력 분포 모델을 사용한다. 본 인식기에서는 이 distribution을 직접 class로 정의하지 하지 않고 Model이라는 상위 클래스를 두고 이를 상속 받아 distribution class를 구현하였다. 외부와의 interface는 모두 Model이라는 interface class를 통해 이루어진다. 이와 같이 구현되었으므로 Model class의 interface 정의만 지킨다면 Gaussian mixture가 아닌 다른 형태의 모델도 쉽게 인식기에 추가할 수 있다. 예를 들면 Neural Networks의 출력이나 supra-segmental 정보의 이용도 용이하다[5].

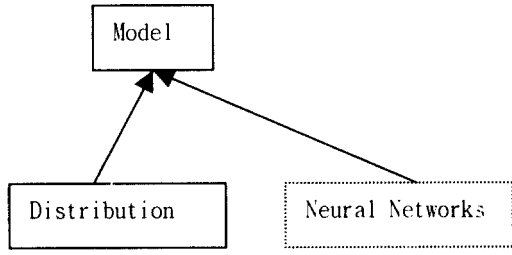

그림 2. 모델의 상속관계

feature의 경우에도 Feature라는 interface class를 먼저 선언하고 구체적인 feature의 class를 정의하였다. 또, 중간 단계의 feature를 각각 별개의 class로 정의하였고 이 중간 단계의 feature들을 지원되는 함수들로 조합하는 것만으로도 새로운 feature를 만들 수 있다. 새로운 feature를 추가할 경우 feature가 처리해야 할 많은 기능들은 상속을 통해 구현되고 단지 한 frame에 대한 feature를 구하는 과정만 구현하면 되므로 새로운 feature를 인식기에 추가하는 과정이 쉽게 처리될 수 있다. 또, 각 feature class는 feature를 요구 받을 경우에만 feature를 구하며 frame 단위로 탐색까지 처리하므로 online 인식이 가능하다.

많은 단어를 인식할 수 있는 인식기의 경우 탐색 트리의 크기는 매우 커진다. 그러나 한 발화를 인식하는 동안 모든 단어가 다 검색되지는 않는다. 따라서 노드 중 일부는 탐색이 종료될 때까지 한 번도 방문 되지 않는 경우도 있다. 이러한 점을 이용하여 특정 노드가 처음으로 방문될 때에는 그 하위 트리에 대해서는 유사도 계산이나 확장을 하지 않는 방식을 택하여 인식속도를 개선하였다.

음성인식기는 많은 계산량과 메모리를 요구한다. 비록 근래의 컴퓨터의 성능이 개선되었다고는 하나 최적화가 요구된다. 특히 탐색 트리의 노드는 매우 많은 개수가 사용되므로 노드 하나가 차지하는 메모리를 줄이는 것은 중요하다. C++에서 virtual function을 사용하게 되면 object별로 virtual function table에 대한 pointer가 하나 필요하고 함수를 호출할 때 이 table을 참조하는 과정이 C에 비해 추가된다. 이는 속도를 저하시킨다. 본 인식기에서는 virtual function을 적절히 이용하여 하나의 노드가 차지하는 메모리를 줄였다. 이는 향후 최적화의 목적에 따라 다른 방향으로 최적화할 수 있을 것이다.

- word end cross word model을 필요에 따라 메모리에 올려 메모리 사용과 인식 속도를 개선하였다.

4 트리 구조의 탐색 단어 사전[4]

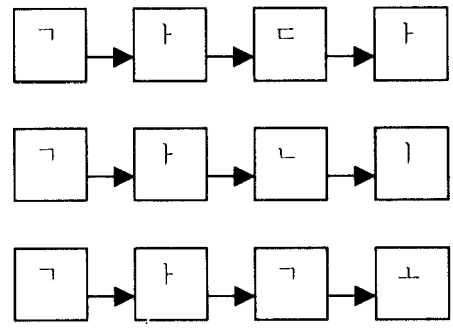

그림 2. Flat lexicon

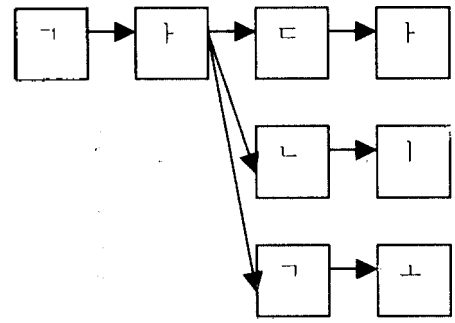

그림 3. Tree lexicon

flat lexicon을 사용하는 기존의 방법에서는 단어를 구성하고 있는 모든 음소에 대해 별도로 처리를 하여야 하지만 tree lexicon을 사용하게 되면 단어 앞 부분의 공통되는 부분에 대해서는 한번의 처리로 해결되어 인식 시간을 단축할 수 있다. 특히, 한국어의 경우 용언의 활용과 조사 때문에 단어의 뒷부분만 다른 경우가 많으므로 더 큰 이익을 볼 수 있다. 탐색에서 단어의 앞부분은 아직 불확실한 상태이므로 많은 단어가 살아있게 되어 계산량이 많은데 이 때 tree lexicon을 적용하게 되므로 그 이익이 더 크다.

flat lexicon 일 경우 단어가 시작할 때 그 단어가 무엇인지를 알 수 있으므로 언어 모델을 적용하여 가능성이 적은 단어는 미리 제거 할 수 있는데 tree lexicon인 경우에는 마지막 노드에 도달해야 그 단어가 무엇인

지 알 수 있게 되므로 미리 언어모델을 적용할 수 없다. 이러한 단점을 보완하기 위해 tree의 각 노드에 들어갈 때 가능성 있는 단어의 unigram 중 최대값을 단계적으로 적용한다. 또 다른 문제점은 단어의 시작에서는 입력되는 path 중 가장 좋은 점수를 가진 것만 살아 남게 되는데 이는 언어모델을 적용하지 않은 것이므로 언어모델을 적용하게 되면 결과가 바뀔 수 있는데 이를 복구해야 한다 이를 위해 tree의 마지막 노드에 도달했을 때 언어모델을 적용하면서 이 단어가 시작할 때 종료되었던 다른 단어들의 목록을 가지고 그 중 언어모델을 고려하였을 경우에 제일 좋은 점수를 가진 단어를 이전 단어로 다시 지정한다. 현재 종료되는 단어를 w 라고 하고 이 단어의 이전 단어가 될 수 있는 후보들 즉, 이 단어가 시작할 때 종료되었던 단어들을 w_i 라고 하자. 그리고 그 때의 점수를 l_i 라고 한다면 복구는 아래의 식에 의해 구해진 단어 w_j 를 이전단어로 지정함으로써 이루어진다.

$$j = \max_i \arg l_i P(w \mid w_i)$$

5 cross word triphone modeling[4]

한 단어의 시작 음소와 끝 음소의 경우에는 실제 탐색에 들어가기 전에는 왼쪽 문맥과 오른쪽 문맥을 알 수 없다. 그래서 좌,우 문맥에 독립적인 모델을 쓰기도 하는데 성능 향상을 위해서는 단어의 좌우 문맥을 고려하여 주어야 한다. 시작 음소에 대해서는 이전의 어느 단어에서 이 단어로 들어오는지를 알 수 있으므로 과거 이력에 따라 현재 음소 모델의 내용을 바꾸어 주는 방법을 사용한다. 즉, 이전 단어가 'ㄱ'으로 종료되는 단어였다면 현재 단어의 시작음소의 여러 변이음 중 왼쪽 문맥이 'ㄱ'인 모델을 선정하여 유사도를 구하게 된다. 끝 음소에 대해서는 오른쪽 단어가 무엇이 될 지 미리 알 수 없으므로 가능한 모든 문맥에 대한 음소 모델을 모두 평가하여 확장하여야 한다. 다음의 그림에 이 과정을 표현하였다. 그림의 왼쪽은 한 단어의 마지막 음소를 표현한 것이다. 마지막 음소는 오른쪽 문맥에 따라 해당되는 모델을 갖는 여러 노드로 펼쳐지게 된다. 오른쪽의 단어 시작에서는 왼쪽 단어의 마지막 음소가 무엇이었는지에 따라 동적으로 모델이 선정되게 된다.

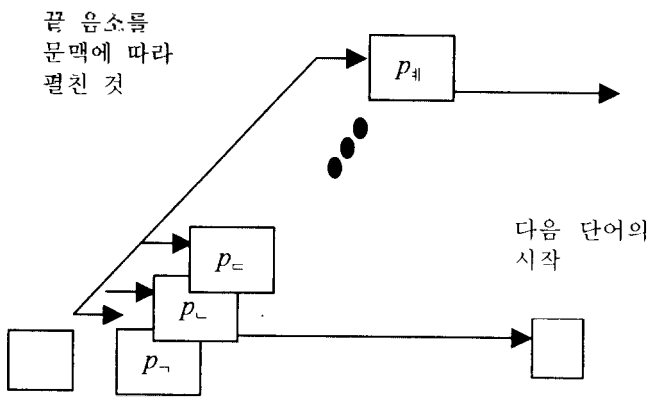

그림 4. Cross word modeling

6 Senone : state 공유[2]

많은 양의 data를 모은다고 해도 훈련데이터의 양에는 언제나 제한이 있게 마련이며 제한된 data를 모델간의 데이터 공유를 통해 더 자세히 모델링하게 되면 성능을 향상시킬 수 있다. ETRI 음성인식기는 하나의 음소를 모델링하기 위해 1 개 또는 3개의 state를 사용하여 음소를 몇 개의 시간적인 세부단위로 나누어 모델링하게 되는데 이 때 다른 음소의 state라고 하더라도 그 모델이 유사하면 공유할 수 있다. 이 공유되는 state를 Senone이라고 하는데 이를 통해 제한된 데이터를 가지고 더 많은 변이음을 모델링할 수 있어 성능을 향상시킬 수 있다.

7 연속 음성 인식

ETRI 음성인식기는 연속음성인식기이다. 이를 위해 trigram 언어 모델을 지원한다. 연속음성에서는 단어가 연결되어 발성되고 이 때 단어 경계에서의 음성은 다음 단어나 이전 단어에 영향을 받아 그 소리값이 변화하므로 단어 경계에 대해서는 다음 단어나 이전단어의 경계에 있는 음소를 고려한 문맥 종속 음소 모델을 사용한다. 언어모델 및 단어경계 모델링을 제거하면 고립단어 인식기로도 사용될 수 있다.

8 대화체 음성 인식

같은 연속음성이라고 하더라도 낭독체의 경우는 간투사(아, 음, 에또, 등의 의미없이 발성 중에 끼어 있는 단어)나 무의미어 등이 적게 나타나지만 자연스러운 대화체에서는 이와 같은 간투사가 많이 발생하고(한 문장에 하나 이상) 말을 더듬거나 잘못 말해서 다시 이야기하는 현상이 빈번하다. 이러한 문제들 중 간투사와 무의미어, 잡음에 대해서는 별도의 모델을 두어 해결하였다.

9 n-phone context modeling[3]

문맥을 고려할 때는 좌우의 1개씩 문맥을 고려하는 triphone modeling이 기본이다. ETRI 음성인식기는 좌우 n 개의 음소 문맥을 고려하여 모델링할 수 있다. 이 때 n 개의 문맥을 모두 고려한다면 모델의 수가 기하급수적으로 증가할 것이므로 적절한 기준이 있어야 한다. 본 인식기에서는 문맥을 세분화할 때 얻는 distribution의 entropy 이득을 보고 이득이 큰 순으로 세분화한다. 이득이 크다는 것은 나뉘어진 모델이 더 순수해졌다는 의미이다.
이 때 음성학적인 질문을 기반으로 구현한 decision tree를 이용하므로 훈련 데이터에 나타나지 않은 문맥이라 하더라도 가장 가까운 문맥을 찾을 수 있으므로 단어 독립 음성인식기라고 할 수 있다.

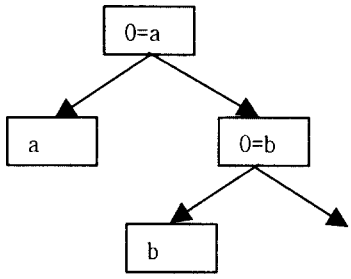

Figure 5. Decision tree for clustering

10 결론

우리는 C++ 언어와 Standard library를 이용하여 효율적이고 확장 및 유지 보수가 편리한 ETRI 음성인식기를 개발하였다. 이 ETRI 음성인식기의 특징은 다음과 같다.

- 연속음성인식기
- 대화체 음성인식기
- n-phone context modeling
- cross word triphone modeling
- trigram language modeling
- tree lexicon
- Senone for state sharing
- PC version
- C++ and Standard library
- 음성인식엔진
- 음성인식 연구용 library

11 향후 계획

현재의 ETRI 음성인식기는 메모리 사용과 인식속도측면에서 최적화가 완료되지 않았다. 이를 계속 추진하여 30Mbyte 정도의 환경에서 1000단어에 대해 실시간으로 수행되는 버전을 만들고자 한다.

12 감사의 글

이 연구는 정보통신부의 지원에 의해 이루어진 결과물입니다.

참고문헌

[1] H. S. Lee, J. Park, and H. R. Kim, "An implementation of Korean spontaneous speech recognition system, " Proc. of Int. Conf. on signal processing applications and technology, pp. 1801-1805, Boston, MA, USA October 1996

[2] M. Y. Hwang, X. D. Huang, and F. Alleva, "Predicting unseen triphones with senones," School of computer science, Carnegie Mellon University, Technical report, CMU-CS-93-139, 1993.

[3] M. Finke, and I. Rogina, "Wide context acoustic modeling in read vs. spontaneous speech," Int. Conf. on Acoustics, Speech, and Signal Processing, Munich, Germany, 1997

[4] M. K. Ravishanker, "Efficient algorithms for speech recognition," School of Computer Science, Carnegie Mellon University, Technical Report CMU-CS-96-143, pp. 42.

[5] Youngjoo Suh, Kyuwoong Hwang, Oh-Wook Kwon, and Jun Park, "Utilizing the Voiced, Unvoiced, and Silence Information to Improve the Performance of Speech Recognizer," Proceedings of the 1998 international technical conference on circuits/systems, computers and communications, vol. 1, pp. 669-672, 1998.

[6] Bjarne Stroustrup, "The C++ programming language third edition," Addison-Wesley, 1997.