한국음향학회지 제21권 제6호 pp. 543~551 (2002)

언어 모델 네트워크에 기반한 대어휘 연속 음성 인식

Large Vocabulary Continuous Speech Recognition Based on Language Model Network

안 동 훈*, 정 민 화*
(Dong-Hoon Ahn*, Min-Hwa Chung*)

*서강대학교 컴퓨터학과 음성언어처리연구실
(접수일자: 2002년 2월 28일; 채택일자: 2002년 7월 31일)

이 논문에서는 20,000 단어급의 대어휘를 대상으로 실시간 연속 음성 인식을 수행할 수 있는 탐색 방법을 제안한다. 기본적인 탐색 방법은 토큰 전파 방식의 비터비 (Viterbi) 디코딩 알고리듬을 이용한 1 패스로 구성된다. 언어 모델 네트워크를 도입하여 다양한 언어 모델들을 일관된 탐색 공간으로 구성하도록 하였으며, 프루닝 (pruning) 단계에서 살아남은 토큰들로부터 동적으로 탐색 공간을 재구성하였다. 용이한 후처리를 위해 워드 그래프 및 N개의 최적 문장을 출력할 수 있도록 비터비 알고리듬을 수정하였다. 이렇게 구성된 디코더는 20,000 단어급 데이터 베이스에 대해 테스트하였으며 인식률 및 RTF 측면에서 평가되었다.
핵심용어: 대어휘 연속음성인식, 언어 모델 네트워크, 백-오프 N-그램, 토큰 전파 알고리듬
투고분야: 음성처리 분야 (2.5, 2.7)

In this paper, we present an efficient decoding method that performs in real time for 20k word continuous speech recognition task. Basic search method is a one-pass Viterbi decoder on the search space constructed from the novel language model network. With the consistent search space representation derived from various language models by the LM network, we incorporate basic pruning strategies, from which tokens alive constitute a dynamic search space. To facilitate post-processing, it produces a word graph and a N-best list subsequently. The decoder is tested on the database of 20k words and evaluated with respect to accuracy and RTF.
Keywords: *Large vocabulary continuous speech recognition, Language model network, Back-off N-gram, Token passing algorithm*
ASK subject classification: *Speech signal processing (2.5, 2.7)*

I. 서 론

연속 음성 인식 시스템은 탐색 공간을 정의하는 언어, 음향, 어휘 모델 등의 지식과 입력된 음성에 대해 탐색 공간으로부터 가장 적절한 단어열을 결정하는 탐색 엔진, 즉 디코더로 구성된다. 디코더는 최적의 단어열을 결정하기 위한 근거로 이들 지식을 사용하므로 시스템의 인식률은 언어 및 음향 모델 등의 모델링 능력에 크게 영향을 받는다. 이들 지식들의 경우에는 정교한 학습 방법을 통해 탐색 공간의 해상도를 높임으로써 디코더로 하여금 보다 정확한 결정을 할 수 있는 가능성을 높이는 데에 있다. 반면, 디코더 입장에서는 다양한 형태의 모델로부터 만들어지는 탐색 공간을 효율적으로 구성하고, 결정 과정에서 가능한 한 탐색 오류 (search error)를 줄일 수 있어야 한다. 이 논문에서는 이 가운데 탐색 공간을

책임저자: 안동훈 (drahn@nlpeng.sogang.ac.kr)
121-742 서울시 마포구 신수동
서강대학교 컴퓨터학과 음성처리연구실
(전화: 02-712-8023; 팩스: 02-704-8273)

효율적으로 구성하는 방법에 대해 논하고자 한다. 실험에 사용한 20,000 단어급 연속 음성 인식 시스템은 수십만 개에 이르는 HMM 상태를 포함한다. 따라서 기본 디코딩 방법인 비터비 (Viterbi) 알고리듬을 수식 그대로 적용하기에는 무리가 있다. 이 논문에서는 효율적인 탐색이 가능하도록 탐색 공간을 유한 상태 네트워크 (finite state network)로 구성한 후, 토큰 전파 방식의 1 패스 디코딩 과정을 통해 최적의 단어열들을 도출하는 방법을 사용하도록 한다. 이러한 탐색 공간은 다음 두 가지 방법을 통해 효과적으로 운용된다.

1. 언어 모델 네트워크 (language model network)를 도입하여 컴파일된 탐색 공간을 구성하도록 하였다. 언어 모델 네트워크를 사용함으로써 디코더의 동작이 언어 모델에 투명하도록 하였으며, 이 위에서 구성된 탐색 공간은 언어 모델, 어휘 모델 등을 컴파일 과정을 통해 동질화함으로써 탐색 함수를 단순화할 수 있다.
2. 토큰 전파 (token propagation) 방식의 비터비 디코더를 사용하였다[12]. 토큰은 프루닝 (pruning) 등의 과정에서 살아남은 활성화된 HMM 상태들을 가리킨다. 비터비 알고리듬은 이러한 토큰들로부터 리스트를 구성하고 이를 동적인 탐색 공간으로 사용하여 전체 인식 네트워크를 탐색해야 하는 부담을 덜 수 있다[5].

이 논문은 다음과 같이 구성되어 있다. 먼저 2절에서는 논문에서 사용한 시스템에 대한 개괄적인 소개를 하며, 3절에서는 컴파일된 탐색 공간인 인식 네트워크를 생성하는 과정에 대해 설명하고, 4절에서는 토큰 전파 방식의 디코더 및 탐색 공간의 동적 구성에 대해 기술한다. 5절에서는 20 k 단어의 데이터베이스를 대상으로 실험한 결과에 대해 설명하며, 마지막으로 6절에서 결론 및 향후 과제에 대하여 다루도록 한다.

II. 기본 인식 시스템

기본 인식 시스템은 화자 독립형이며, 음소 기반의 유사 음소단위 (PLU: phoneme-like-unit)를 인식 단위로 사용한다. 인식 단위는 묵음을 포함하여 48개의 PLU로 구성된 서강대 (SGU) PLU 세트를 이용하였으며, 음향 모델은 상태-공유 (tied-state)의 연속 HMM을 사용하였다. 모든 HMM은 각각 세개의 상태를 가지고, 각 상태의 출력 확률값은 다수의 가우시안 혼합분포로부터 계산된다. 발음 사전의 단어는 의사 형태소 단위로 표현한다[13]. 각 단어마다 하나 또는 그 이상의 발음열을 포함할 수 있으며, 발음 확률을 선택적으로 사용할 수 있다. 언어 모델은 백-오프 (back-off) 기반의 임의의 N-gram을 사용한다. 한편, 시스템의 디코더는 그림 1과 같이 구성된다. 이 디코더는 기본적으로 1 패스 시동기로 트리 구조 네트워크를 이용한다. [10]의 변형된 비터비 디코딩 방법을 도입하여, 결과로 최적의 문장과 워드 그래프 (word graph)를 생성한다. 이 때 디코딩 과정은 토큰 전파 방식으로 구현되었으며[12], 실질적인 탐색 공간은 프루닝에서 살아남은 활성화된 HMM 상태들, 즉 토큰들을 1차원의 리스트로 표현하였다[5,6]. 활성화된 HMM의 수를 줄이기 위해 빔 프루닝 (beam

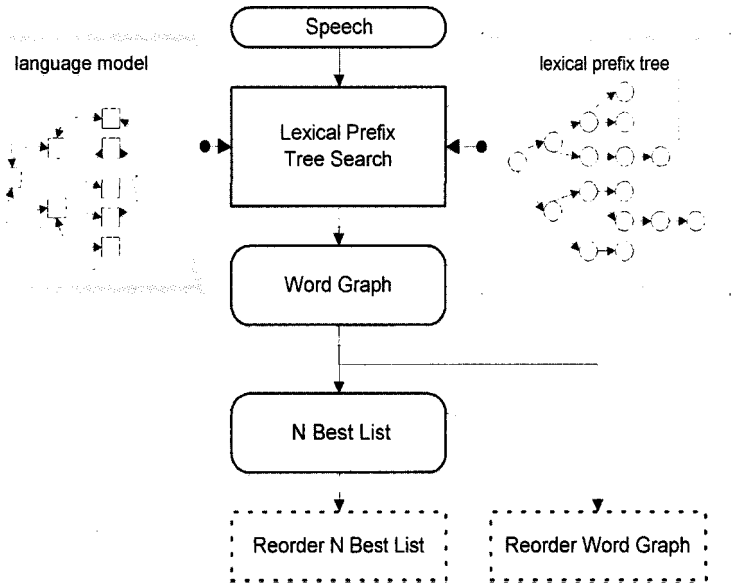


그림 1. 디코더의 구성
Fig. 1. The configuration of decoder.

pruning)과 더불어 언어 모델 미리 참조 (look-ahead) 기법을 사용하였으며[9], 추가 단계로 생성된 워드 그래프로부터 N개의 최적 문장을 추출하도록 하였다[10].

III. 인식 네트워크 생성

3.1. 탐색 공간의 동질화

디코딩 알고리듬이 효과적으로 적용되기 위해서는 각 모델의 변별력도 중요하지만, 서로 이질적인 모델의 하위 탐색 공간들로부터 만들어지는 전체 탐색 공간에 대해서도 모델간의 중복된 정보를 줄이고 동질의 형태로 구성할 필요가 있다. 이를 비터비 알고리듬의 입장에서 설명하면 다음과 같다. 비터비 알고리듬은 원래 동적 프로그래밍 (DP: dynamic programming)의 한 가지로, HMM의 상태와 시간을 각각 축으로 삼는 2차원의 탐색 공간을 사용한다. 그러나 연속 음성 인식의 경우, 탐색에 필요한 언어 모델, 음향 모델, 어휘 모델 등은 서로 성격이 다른 탐색 공간을 형성한다. 따라서 이들을 비터비 알고리듬이 사용할 수 있는 하나의 탐색 공간 안에서 해석할 수 있는 방법이 필요하다. 이 논문에서는 각 모델을 통합하는 컴파일 과정을 통해 탐색 공간을 동질의 형태로 구성한 후, 이 위에서 탐색 함수를 실행하도록 하였다. 이러한 방법은 통합된 탐색 공간을 구성하기 위한 전처리 과정이 불가피하지만, 여러 형태의 지식들을 명시적으로 개별 탐색하는 과정을 은닉할 수 있을 뿐 아니라, 추후 다른 지식을 더 사용해야 하는 경우에도 별다른 탐색 함수의 수정이 필요없게 된다. 실제 시스템에서는 다음과 같은 순서로 탐색 공간을 구성한다.

1. 언어 모델을 언어 모델 네트워크로 표현한다.
2. 언어 모델 네트워크의 각 아크에는 단어에 대한 정보가 포함된다. 따라서 이 아크를 해당 단어의 발음열로 대치하고 다시 발음열을 HMM으로 대치하면, 인식 네트워크를 얻을 수 있다. 이렇게 구성된 인식 네트워크가 전체 탐색 공간을 구성하며, 디코더는 이 가운데 프루닝으로부터 살아남은 상태들만으로 재구성된 동적인 탐색 공간을 사용한다.

3.2. 언어 모델 네트워크

3.2.1. 언어 모델 네트워크의 정의

유한상태 네트워크 (FSN: finite state network)로 표현되는 언어 모델 네트워크는 네트워크의 각 노드는 언어 모델의 상태를 정의하고, 아크는 입력 단어에 따른 새로운 상태로의 전이를 나타낸다. 단어열 $W=w_1w_2\cdots w_n$에 대한 언어 모델 값은 다음과 같은 식으로 계산된다.

$$\begin{aligned} p(w) &= p(w_1w_2\cdots w_n) \\ &= p(w_1)\times p(w_2|w_1)\times\cdots\times p(w_n|w_1\cdots w_{n-1}) \\ &= \prod_i p(w_i|w_1\cdots w_{i-1}) \\ &= \prod_i p(w_i|h_i) \end{aligned}$$

h_i는 언어 모델의 히스토리 (history)로 과거 단어열을 의미하며, $p(w_i|h_i)$는 과거 단어들을 조건으로 현재 단어 w_i가 나타날 확률 (likelihood)을 의미한다. 언어 모델을 만들 때에는 학습상의 데이터 부족 문제를 보완하기 위해 일반적으로 단어열 $h_i=w_1\cdots w_{i-1}$을 일정한 동치 관계 (equivalence relation)를 사용하여 파티션 (partition)한다. 가령, 단어 트라이그램 (trigram)은 히스토리의 마지막 두 단어 w_{i-2}, w_{i-1}이 동일한 경우 동치 관계가 성립한다. 그리고 이에 따라 각 동치 부류 (equivalence class)는 다음과 같은 확률 값을 가진다.

$$p(w_i|w_1\cdots w_{i-1})\approx p(w_i|w_{i-2}w_{i-1})$$

언어 모델이 가질 수 있는 히스토리를 파티션하는 방법에 따라 다양한 언어 모델을 설계할 수 있다. 히스토리를 최근 N개의 단어열을 기준으로 파티션하면 N-gram의 언어 모델을 얻을 수 있으며, 단어 대신 각 단어가 속한 문법적/언어학적인 부류로 나눌 경우 class N-gram을 얻을 수 있다. 이처럼 다양한 언어 모델을 인식 과정에 투명하게 사용하기 위해서는 통일된 표현 방법이 필요하며, 언어 모델 네트워크는 이러한 목적으로 구성된 표현 방법이다. 언어 모델 네트워크는 각 모델에 내재된 언어 모델 컨텍스트를 이용하여 생성된다. 언어 모델 컨텍스트 (LM context)는 히스토리 h와 입력 단어 w를 사용하여 (h, w)의 쌍으로 정의된다. 특히, 단어는 연속된 입력이므로 새로운 단어 입력이 들어올 때마다 특정 컨텍스트에서 새로운 컨텍스트로 전이한다. 이러한 컨텍스트 전이는 모든 언어 모델에 공통된 사실이다. 언어 모델의 컨텍스트는 유한 개이며, 컨텍스트 사이의 전이는 입력 단어에 의해 결정되므로 FSN으로 표현할 수 있다. 이때 FSN의 상태는 언어 모델의 각 컨텍스트에 해당하며, 아크에는 입력 단어와 히스토리로부터 계산된 언어 모델 값이 저장된다. 그림 2는 두 개의 단어 a, b에 대한 모든 전이가 가능한 바이그램 네트워크 (bigram network)를 나타낸다. 각 상태는 히스토리 h와 단어 w의 쌍 (h, w)으로 표현되며, 아크는 입력 단어 w와 특정 컨텍스트의 전이 확률,

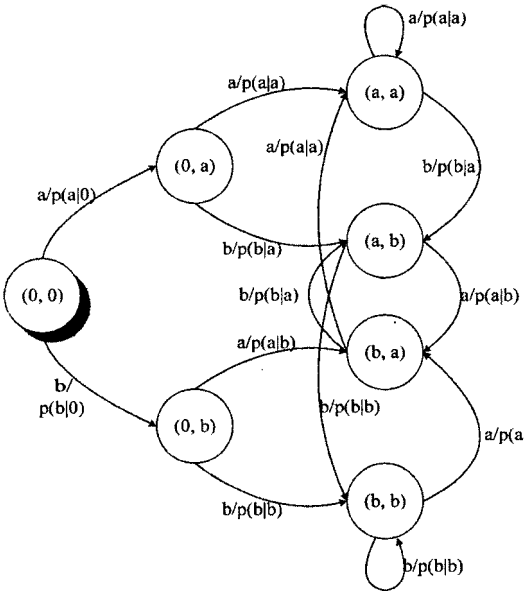


그림 2. 모든 전이가 가능한 바이그램 네트워크
Fig. 2. Full bigram network.

즉, 컨텍스트의 히스토리에 조건을 둔 바이그램 확률 값으로 이루어진다. 초기 상태에는 그림자가 추가되었으며, 네트워크의 어떤 상태도 종료 상태가 될 수 있다.

한편 그림 2의 FSN은 상태 최소화 알고리듬 (state minimization algorithm)을 이용하면 네트워크를 최소화할 수 있다[1]. 최소화 방법은 합동 (congruent)인 상태들을 하나의 집합으로 만들고 이 집합을 새로운 상태로 정의함으로써 상태의 개수를 줄이는 것이다. 따라서 최종적인 언어 모델 네트워크는 그림 3과 같다.

3.2.2. 백-오프 언어 모델의 사용

백-오프 언어 모델은 데이터 부족 문제를 효과적으로 해결할 수 있는 방법이다[3,4]. 백-오프 방법은 먼저 학습 데이터로부터 각 언어 모델 이벤트의 확률 값들을 계산한다. 그리고 각 이벤트 확률 값을 조금씩 "에누리 (discount)"한 다음, 이렇게 에누리된 양을 학습 데이터에 나타나지 않은 언어 모델 이벤트의 확률 값을 추정하는데 사용한다. 다음은 백-오프 방법을 사용한 언어 모델을 설명하는 수식이다[7].

$$p(w|h)=\begin{cases} \alpha(w|h) & \text{if } \mathrm{N}(h,w)>0 \\ \gamma(h)\cdot\beta(w|h') & \text{if } \mathrm{N}(h,w)=0 \end{cases}$$

h'는 h를 백-오프한 히스토리를 나타낸다. 트라이그램의 히스토리를 백-오프하면 바이그램의 히스토리를 얻을 수 있다. 학습 데이터로부터 한번이라도 나타난 이벤트는 에누리된 확률 값 $\alpha(w|h)$를 그대로 사용하도록 하며, 그렇지 못한 경우에는 히스토리를 백-오프하여 추정한다. 즉, 히스토리를 한 단계 줄인 다음 백-오프된 히스토리에 대한 가중치 $\gamma(h)$와 이 히스토리를 조건으로 하는 확률 값 $\beta(w|h')$로부터 언어 모델을 추정하게 된다. 가령, 트라이그램 모델의 $p(a|ab)$을 백-오프로 추정하는 경우 바이그램을 이용하여 다음과 같이 계산된다.

$$p(a|ab)=\gamma(ab)\times\beta(a|b)=\gamma(ab)\times p(a|b)$$

만약 $p(a|b)$가 존재하지 않으면 다시 백-오프하여

$$\begin{aligned} p(a|ab)&=\gamma(ab)\times\beta(a|b)\\ &=\gamma(ab)\times p(a|b)=\gamma(ab)\times\gamma(b)\times p(a) \end{aligned}$$

와 같이 계산한다. 언어 모델 네트워크 입장에서 보면, $p(a|b)$의 $p(a)=p(a|0)$값들은 이미 네트워크에 존재한다. 따라서 $p(a|ab)$에 대한 계산은 백-오프 가중치 γ

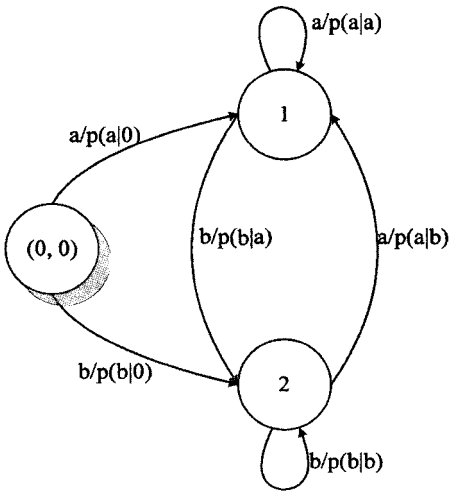


그림 3. 최소화된 바이그램 네트워크
Fig. 3. Minimized bigram network.

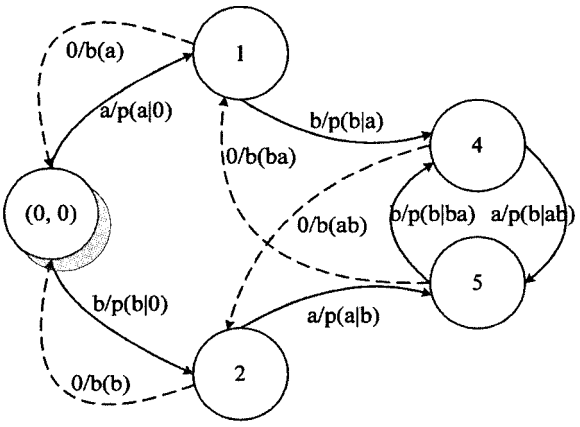


그림 4. 백-오프된 트라이그램 네트워크
Fig. 4. Backed-off trigram network.

를 설명하는 전이만 추가해 주면 된다. 단, 일반적인 확률 값과는 달리 한 단어에 대해 두번 이상의 전이가 필요하므로 백-오프 전이는 "null transition"으로 처리한다. 이 때 백-오프 전이의 출발 상태는 백-오프되기 전의 히스토리를 포함하는 상태이며, 도착 상태는 백-오프 후의 히스토리를 포함하는 상태이다. 그림 4의 상태 (0, 0)은 초기 상태이자 백-오프 상태이다.

3.2.3. 언어 모델 네트워크 생성 알고리듬

언어 모델 네트워크는 초기 상태에서 시작하여, 전이 가능한 단어들로부터 새로운 상태를 BFS (breadth first search) 순서로 생성한다. 이를 위해 FIFO (first-in, first-out)의 큐를 이용하여 만들어진 상태를 이 큐에 추가 한다. 이후 큐에 저장된 각 상태에 대해 다음 상태를 BFS 순서로 생성해 나가면서, 큐가 비워질 때까지 반복한다. 언어 모델 네트워크에는 세 개의 의도된 상태가 존재한다. 네트워크의 초기 상태는 문장의 시작 단어 ⟨s⟩로 하며, 유니그램 (unigram)으로의 백-오프를 위해 히스토리가 없는 널 (null) 상태 Ø를 별도로 추가한다. 또한 문장은 어떠한 단어에서도 종료될 수 있으므로 모든 단어에서 종료 상태 ⟨/s⟩로의 전이를 추가한다.

3.3. 인식 네트워크의 생성

인식 네트워크는 언어 모델 네트워크를 사전의 발음열로 인스턴스화 하여 생성한다. 언어 모델 네트워크의 아크는 특정 상태에서 전이할 수 있는 단어들을 나타내므로, 인스턴스화는 아크의 단어를 HMM열로 바꿔주는 과정에 해당한다. 이 인스턴스화 방법 과정 중에 각 발음열의 어두를 공유하는 단계를 추가하면 트리 구조의 인식 네트워크를 구성할 수 있다. 언어 모델 네트워크로부터 이렇게 생성된 인식 네트워크는 [2]의 후자 트리 (successor tree)를 이용한 표현 방법과 유사하다.

3.3.1. 인식 네트워크의 구조

컴파일된 인식 네트워크는 네가지 유형의 노드를 가진다. 인식 단위로 발음 사전을 구성하는 HMM과 단어의 시작과 끝을 알려주는 단어 시작 (word-begin) 및 단어 끝 (word-end) 노드, 그리고 언어 모델의 "히스토리가 없음"을 의미하는 널 노드가 그것들이다. 단어 시작 노드, HMM 노드, 단어 끝 노드의 순서는 하나의 단어를 표현한다. 단어의 시작과 끝 노드는 발음열을 구성하는 HMM 노드들을 "guard"하여 HMM 레벨의 정보와 단어 레벨의 정보를 구분하는데 사용한다. 구체적으로 말하면, 단어와 단어 사이의 관계인 언어 모델은 한 단어의 단어 끝 노드와 전이되는 노드의 단어 시작 노드 사이에 적용된다. 또한 단어를 여러 개의 발음열로 표현하는 것은 다른 단어와는 상관없는 HMM 레벨의 표현 방법의 문제이다. 따라서 단어가 포함하는 다중 발음열은 단어 시작 노드와 단어 끝 노드 사이에서 인스턴스화 방법에 따라 표현된다. 널 노드는 히스토리를 포함하지 않으므로 백-오프될 수 있는 마지막 노드이다. HMM의 노드는 입력 프레임을 소비하지만 나머지 세가지 유형의 노드는 프레임을 소비하지 않고 곧바로 다음 노드들로 토큰을 전파한다.

3.3.2. 인스턴스화 과정

인스턴스화의 기본 방법은 언어 모델 네트워크의 특정 상태에서 전이 가능한 단어들을 이용하여 트리 구조를 생성해 내는 데에 있다. 전이 가능한 단어들은 언어 모델 네트워크의 아크에 저장되어 있으므로 이들 트리들은 각 아크의 단어에 해당하는 발음열들로부터 만들어질 수 있다. 플랫 구조 인식 네트워크와는 달리, 트리 구조의 경우에는 각 트리에 포함되는 발음열들의 어두를 공유하는 단계가 추가된다. 이렇게 개별적인 트리들을 만든 후 언

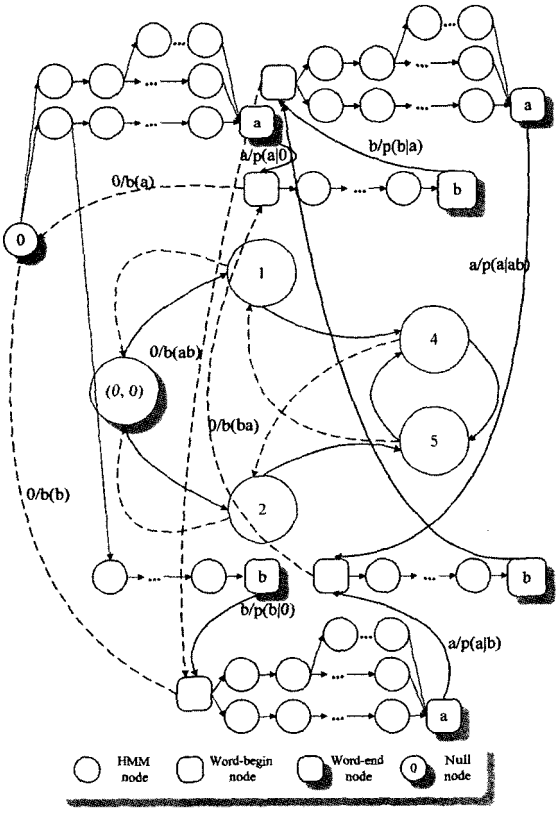


그림 5. 트리 구조 인식 네트워크
Fig. 5. Tree-structured recognition network.

어 모델의 아크에 저장된 확률 값을 트리와 트리 사이에 할당하도록 한다. 그림 5는 트리 구조의 인식 네트워크를 보여준다.

IV. 토큰 전파 방식의 디코딩

4.1. 토큰 전파 알고리듬

비터비 디코딩 알고리듬의 구현은 토큰 전파 알고리듬[12]에 기반한다. 토큰은 활성화된 상태들의 인스턴스에 해당하며 현재 프레임까지의 누적된 확률값과 백트래킹에 관한 정보를 유지한다. 즉, DP의 중간 결과를 저장하는 용도로 사용된다. 토큰은 인식 네트워크의 연결 경로를 따라 다음 상태로 전파되며 해당 상태에서 경쟁이 일어난다. 경쟁에서 이긴 토큰을 해당 상태의 새로운 토큰으로 결정한다. 이를 토큰 재결합 (token recombination) 과정이라고 한다. 모든 입력 프레임을 처리하고 난 후에는 네트워크의 마지막 상태의 토큰으로부터 백트래킹을 시도하면 최적의 문장을 구할 수 있다. 실제 구현에 있어서는 인식 결과는 단어들로 구성된 문장이므로, 단어 레벨의 백트래킹만 수행하면 된다. 따라서 토큰 전파 방식의 비터비 디코딩 과정은 단어 끝 노드에서만 백트래킹 정보를 추가한다. 활성화된 상태 인스턴스를 토큰으로 표현하여 얻을 수 있는 장점은 두가지이다.

첫째, 비터비 디코딩의 탐색 공간은 원칙적으로 2차원 그리드 안에 정의된 모든 상태들이지만, 실제 탐색 과정이는 프루닝에서 살아남은 상태들만이 참여한다. 토큰은 이렇게 해서 살아남은 상태마다 유지되므로 토큰들로 구성된 동적인 탐색 공간을 구성할 수 있다.

둘째, 토큰 재결합시 경쟁에서 이긴 토큰뿐 아니라 상위 M개의 토큰을 유지하면, 워드 그래프를 생성하는데 필요한 각 단어의 시작 시간과 단어의 확률 값을 백트래킹의 정보와 함께 저장할 수 있다. 워드 그래프는 이러한 정보로부터 쉽게 구할 수 있다.

4.2. 탐색 공간의 동적 구성

디코더는 프루닝 등의 휴리스틱을 적용하여 정답과는 거리가 먼 가설 (hypothesis)들을 탈락시키면서 계산량을 조절한다. 한 프레임에서 살아남은 가설들만이 다음 프레임의 디코딩 과정에 참여하기 때문에, 실질적인 탐색 공간의 크기는 상당히 동적으로 달라진다. 반면 원래의 비터비 알고리듬은 2차원의 고정된 그리드 안에서 즉, 시간 축 인덱스와 HMM 축 인덱스에 의해 진행된다. 따라서 인덱스 i를 갖는 HMM의 상태가 프루닝 등으로 이미 탈락되었더라도, DP의 반복 과정 중에 해당 인덱스 i의 차례가 되면 적어도 그것이 탈락되었는지 조사해야 하며, 그렇지 않은 경우에 상태 i를 갱신하게 된다. 이는 전체 탐색 공간의 크기에 비례하는 만큼의 계산 부담을 가져온다. 프루닝을 거쳐 살아남은 상태의 개수가 전체 탐색 공간에 비해 작으면 작을수록 이러한 식의 DP 반복 과정은 확실히 낭비이다. 계산에 참여하지 말았어야 할 부분이 그대로 참여하기 때문이다. 이러한 문제를 해결하기 위해 토큰들을 리스트로 구성하였으며, 그 결과 실질적인 탐색 공간은 살아남은 상태들로부터 동적으로 구성된다. 그리고 메모리와 계산량은 오로지 토큰의 개수에 비례하게 된다[5]. [5]가 이 논문과 다른 점은 [5]의 경우, 언어 모델, 어휘 모델 등의 각 지식마다 별도의 리스트를 유지하므로 단계적인 탐색 과정이 사용된다는 점이다. 토큰은 현재 상태까지의 누적 확률 값과 백트래킹 정보 이외에도 인식 네트워크의 해당 노드에 대한 참조를 포함해야 한다. 토큰의 전파 경로를 인식 네트워크로부터 참조하기 위함이다. 한편 토큰은 시간의 진행 방향을 따라 전파된다. 따라서 올바른 토큰 재결합을 위해서는 토큰을 리스트에 추가할 때 진행 순서가 뒤바뀌지 않도록 유의해야 한다. 이를 토큰 서열 규칙이라고 한다. 프루닝에 의해 탈락한 상태들도 토큰 전파 과정에서 다시 활성화될 수 있기 때문에, 리스트에서 탈락했던 토큰들이 다시 리스트에 추가될 수 있다. 이 경우에는 올바른 전파 순서가 되도록 리스트에 삽입해야 한다. 디코더는 매 입력 프레임마다 리스트의 토큰들을 대상으로 전파 경로를 따라 재결합 과정을 수행한다.

V. 실험 결과

5.1. 실험 환경

5.1.1. 학습 데이터

학습 데이터는 신문, 서적 등으로부터 추출한 낭독체 15,000 문장을 사용하였다. 평균 20개의 형태소, 10개의 어절로 이루어졌으며 전체 약 25시간 분량에 해당한다. 학습 데이터에 대해 48개의 SGU (서강대) PLU로부터 11421개의 트라이폰과 4855개의 상태-공유 HMM을 생성하였다. 발음 사전은 총 22622개의 단어로 이루어졌으며, 다중 발음열을 허용하여 총 28243개의 엔트리를 포함한다. 음성으로부터 추출한 특징 벡터는 13차의 MFCC와 그 델타 및 델타-델타계수를 포함한 39차의 벡터를 이용하였다.

언어 모델은 바이그램 및 트라이그램 모두 절대치 감가(absolute discounting) 방법[7]을 사용하였으며, 이때 사용한 파라미터는 0.5이고, 2회 미만의 출현 빈도를 가진 이벤트는 제외하였다. 마지막으로 테스트 문장으로는 학습에 참여하지 않은 화자의 발화 가운데 미등록 어휘 (OOV: out-of-vocabulary)가 없는 문장 154개를 선택하였다.

5.1.2. 플랫폼 선택

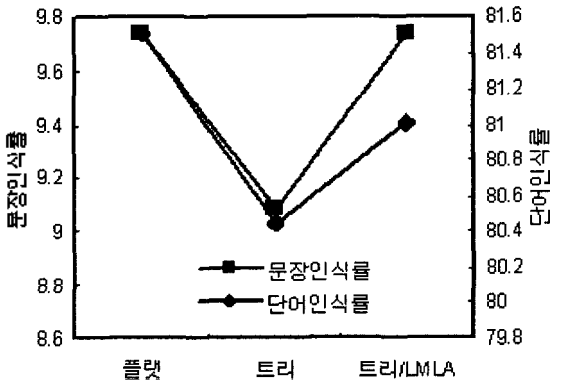
시스템의 디코딩 시간은 사용한 알고리듬뿐 아니라 사용된 플랫폼 환경에 매우 종속적이다[8]. 논문의 디코더는 AMD Athlon 750Mhz 및 Windows 2000이 설치된 PC에서 테스트되었으며, 시스템은 Intel C++ Compiler 5.0으로 컴파일하였다. Microsoft Visual C++ 6.0에 비해 약 10%정도의 디코딩 시간을 줄일 수 있었다.

5.2. 인식 네트워크의 형태에 따른 인식 결과

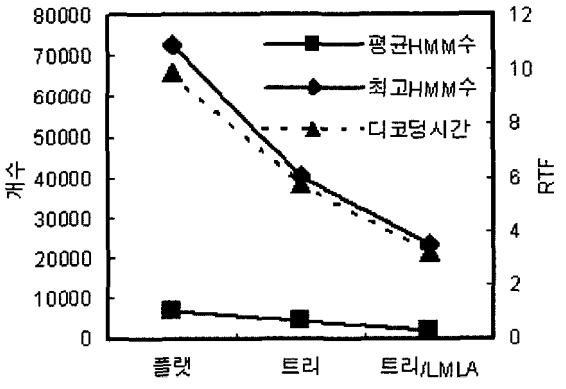
첫번째 실험은 인식 네트워크의 효율성에 대한 실험이다. 빔 크기를 150 (자연로그 도메인)으로 고정시키고 플랫 구조, 트리 구조, 언어 모델 미리 참조 기법을 적용한 트리 구조 (트리/LMLA)에 대해서 테스트한 결과, 그림 6와 같은 결과를 얻었다. 인식률 계산은 모든 삽입, 삭제, 대치 오류를 제외한 정답만을 포함한 정확도를 사용하였으며, 디코딩 시간의 단위인 실시간 인식 배수 (RTF: real time factor)의 경우, 1 RTF는 1초의 음성 분량 (speech segment)을 디코딩하는데 1초가 걸렸음을 의미한다. 인식률의 경우, 플랫 구조의 인식 네트워크를 사용한 경우에 가장 높았지만, 디코딩 시간 및 활성화된 HMM 수에 대해서는 트리/LMLA의 경우가 가장 적었다. 인식률 면에서 트리 구조를 사용한 경우 1.06%, 트리/LMLA의 경우 0.5% 감소인 반면, 디코딩 시간에서는 플랫 구조가 트리 구조에 비해서는 1.7배, 트리/LMLA에 비해서는 3.1배 더 많은 시간을 사용했음을 보여준다. 이러한 결과는 트리 구조를 사용한 경우 디코딩 시간의 대부분을 차지하는 상태 출력 확률 계산 및 토큰 연산에 사용되는 시간을 상당히 줄여줄 수 있음을 보여준다. 특히 언어 모델 미리-참조 기법은 트리 구조의 단점이었던 언어 모델의 지연 적용 문제를 해결하는데에도 도움이 되었음을 알 수 있다. 첫번째 실험 결과를 토대로 이하 모든 실험은 언어 모델 미리 참조 기법을 적용한 트리 구조의 인식 네트워크에서 진행하였다.

5.3. 인식 결과에 대한 빔 프루닝의 영향

그림 7과 그림 8은 빔 사이즈 및 바이그램/트라이그램 사용에 따른 인식률 및 최대 HMM의 개수와 RTF로 대변하는 디코딩 비용의 변화 양상을 보여준다. 언어 모델에 상관없이 빔 크기를 300에서 150까지 줄이더라도 인식률에는 거의 변화가 없는 반면, 인식 시간은 매우 효과적으로 감소하고 있음을 볼 수 있다. 바이그램의 경우 가장 높은 인식률은 빔 크기 150, 3.22RTF에서 81.01%이었으며, 빔 크기 125에서는 1.90RTF에 80.45%이었다. 거의 정확한 실시간 디코딩 (1.02RTF) 결과는 빔 크기 100에서 75.06%의 인식률을 얻을 수 있었다. 이 경우 눈에 띄는 탐색 오류가 발생하였음을 알 수 있다. 트라이그램의 경우에도 동일하다. 트라이그램 사용에 따른 HMM의 개수가 조금씩 증가하였지만 인식 시간에는 큰 차이가 없었으며 최대 인식률은 빔 크기 175에서 5.02RTF, 82.57%, 125에서는 1.88RTF, 81.61%, 100에서는 0.96RTF, 76.78%이었다. 이는 트라이그램이 바이그램보다 변별력이 있음을 보여준다.



(a) 문장/단어 인식률
(a) Sentence/ word recognition rate



(b) 활성화된 평균/최고 HMM 개수 및 디코딩 시간
(b) Average/peak number of active HMMs and decoding time

그림 6. 네트워크 형태에 따른 인식 결과
Fig. 6. Recognition result for network type.

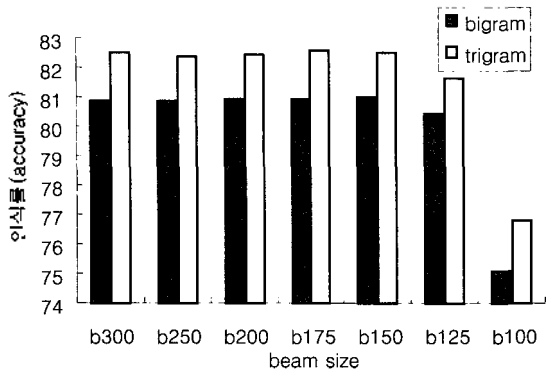


그림 7. 빔 크기에 따른 인식률 변화
Fig. 7. Recognition rates with respect to beam sizes.

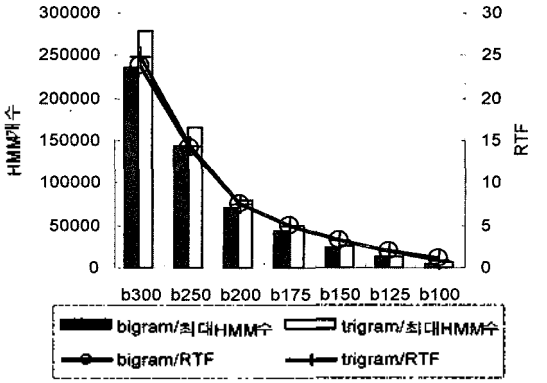


그림 8. 빔 크기에 따른 활성화된 평균/최고 HMM 수 및 디코딩 시간
Fig. 8. Average/peak number of active HMMs and decoding time with respect to beam sizes.

5.4. 워드 그래프 및 N개의 최적 문장 탐색

세번째는 후처리를 위한 워드 그래프와 N개의 최적 문장 탐색에 대한 실험을 수행하였다. 바이그램의 언어 모델과 빔 크기 100, 125, 150에 대해 상태당 토큰의 수를 3, 5, 7, 10개로 설정한 후 워드 그래프를 생성하도록 했으며, 여기서 최대 100개의 최적 문장을 선택하도록 하였다. 그림 9는 인식률 및 인식 시간을 보여주고 있다. 이 그림에서 단어 10은 선택한 100개 중에서 상위 10개의 문장을 선택한 경우의 단어 인식률을 말하며 b100, t3은 빔 크기 100, 상태당 토큰수는 3개를 이용한 경우를 나타낸다. 이 실험의 최고 인식률은 b150, t10에서 4.47 RTF의 시간에 88.98%를 얻은 경우이다. 상위 10개를 선택한 경우 평균 5.8%의 단어 인식률이 향상되었으며 30개의 경우 +1.16% (+는 그만큼 향상했음을 의미), 50개의 경우 +0.28%, 70개의 경우 +0.13%, 100개의 경우 +0.09%의 인식률 향상이 있었다. 반면 인식 시간에 대해서는, 빔 크기 100을 사용한 경우 1.04~1.21 RTF (토큰 한 개 사용시 1.02 RTF), 125를 사용한 경우 1.9~2.5 RTF (1.9 RTF), 150을 사용한 경우 3.58~4.47 RTF (3.22 RTF)에서 보듯이, 약간의 증가만을 가져왔을 뿐이다. 대부분의 인식 결과가 포함된 상위 10개의 문장만을 사용하는 경우, 눈에 띄는 인식 시간의 증가없이도 후처리를 통해 인식률을 높일 수 있음을 보여준다.

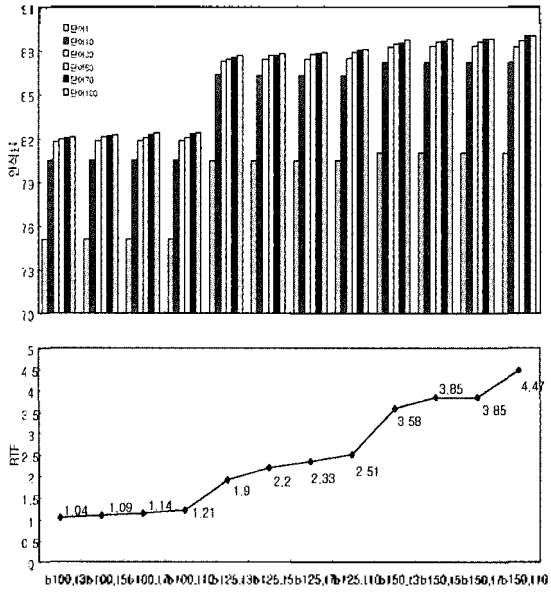


그림 9. 워드 그래프에서 탐색된 100개의 최적 문장
Fig. 9. 100-best sentences from a word graph.

VI. 결론

이 논문에서는 대어휘 연속 음성 인식을 위한 효과적인 탐색 공간의 구성 방법에 대해서 알아보았다. 이 논문에서는 토큰 전파 방식의 1 패스 비터비 디코더를 설계하였으며 효율적인 탐색 공간의 운영 방법에 대해 다루었다. 언어 모델 네트워크와 체계적인 인식 네트워크 생성 방법을 제안하여 여러 레벨의 모델들을 하나의 통합된 탐색 공간을 구성할 수 있었다. 또한 DP 알고리듬과 프루닝의 특성을 고려한 토큰 리스트를 사용함으로써 DP 과정 중 불필요한 반복 과정을 제거하였다. 트리 구조의 인식 네트워크를 사용하여 상태 출력 계산 및 토큰 연산 횟수를 줄였다. 여기에 언어 모델 미리 참조 기법을 적용하여 보다 정밀한 빔을 사용함으로써 많은 수의 토큰들을 탐색 오류의 증가없이 이른 시간에 탈락시킬 수 있었다. 실험

을 통해 언어 모델 미리 참조 기법은 시간 비용을 매우 효과적으로 감소시킬 수 있음을 보여주었다. 또한 1 패스에서 생성된 워드 그래프와 N개의 최적 문장은 인식 결과를 효과적으로 재탐색할 수 있는 수단으로 사용될 수 있다. 결과적으로 논문에서 제안한 디코더는 20 k의 단어를 대상으로 실시간 디코딩이 가능함을 보여주었다.

그러나 실험 결과가 보여주듯이, 언어 및 음향 모델에 내재된 모델 오류는 개선의 여지가 있다. 이들에 대해서는 추가적인 데이터 수집과 더불어 변별력 있는 학습 방법을 통해 개선해야 할 것이다. 또한 디코딩 과정은 실시간으로 수행할 수 있지만, 음성 인식 시스템은 디코더 이외에도 마이크로부터 음성을 입력받는 부분과 인식 결과를 응용 목적에 맞도록 가공하는 과정이 포함된다. 따라서 시스템의 시작 단계에서부터 최종 결과를 사용자에게 보여주는 과정은 다소 지연될 수 밖에 없다. 이러한 지연을 최소화하도록 전반부에서부터 출력 결과에 이르는 단계를 효과적으로 연결화하는 방법도 제안되어야 할 것이다.

참 고 문 헌

1. V. Drobot, *Formal Languages and Automata Theory*, Computer Science Press, 1989.
2. M. Federico, M. Cettolo, F. Brugnara, and G. Antoniol, "Language modelling for efficient beam search," *Computer, Speech and Language*, **9**, 353-379, 1995.
3. S. M. Katz, "Estimation of Probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, **ASSP-35**, 400-401, 1987.
4. R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," *Proc. ICASSP-95*, 181-184, 1995.
5. H. Ney, D. Mergel, A. Noll, and A. Paeseler, "Data driven search organization for continuous speech recognition," *IEEE Transactions on Signal Processing*, **40** (2), 272-281, 1992.
6. H. Ney, R. Haeb-Umbach, B.-H. Tran, and M. Oerder, "Improvements in Beam Search for 10000-Word Continuous Speech Recognition," *Proc. ICASSP-92*, I-9-I-12, 1992.
7. H. Ney, U. Essen, R. Kneser, "On the estimation of small probabilities by leaving-one-out," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17** (12), 1202-1212, 1995.
8. J. J. Odell, "The CUHTK-Entropic 10xRT Broadcast News Transcription System," *Proc DARPA Broadcast News Workshop*, 271-275, 1999.
9. S. Ortmanns, H. Ney, A. Eiden "Language-model look-ahead for large vocabulary speech recognition," *Proc. ICSLP-96*, 2091-2094, 1996.
10. S. Ortmanns, and H. Ney, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer, Speech and Language*, **11** (1), 43-72, 1997.
11. B. H. Tran, F. Seide, and V. Steinbiss, "A word graph based n-best search in continuous speech recognition," *Proc. ICSLP-96*, 2127-2130, 1996.
12. S. J. Young, N. H. Russell, and J. H. S. Thornton, *Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems*, CUED-TR-38, 1989.
13. 이경님, 정민화, "의사형태소 단위의 연속 음성 인식," 제15회 음성통신 및 신호처리 워크샵, 309-314, 1998.

저자 약력

● 안 동 훈 (Dong-Hoon Ahn)

1997년 2월: 서강대학교 컴퓨터학과 졸업 (공학사)
1999년 2월: 서강대학교 대학원 컴퓨터학과 졸업 (공학 석사)
1999년 3월~현재: 서강대학교 대학원 컴퓨터학과 박사 과정
※ 주관심분야 : 연속음성인식, 음향 모델링, 화자 적응

● 정 민 화 (Min-Hwa Chung)

한국음향학회논문지 제20권 제2호 참조