음소단위 TDNN에 기반한 한국어 연속 음성인식을 위한 데이타 자동분할

박 규봉 이 근배 이 종혁
포항공과대학교 전자계산학과

Automatic segmentation for continuous spoken Korean language recognition based on phonemic TDNN

Coophong Baac Geunbae Lee Jong-Hyeok Lee
Dept. of Computer Science and Engineering, POSTECH

요 약

신경망을 이용하는 연속 음성인식에서 학습이라 함은 인위적으로 분할된 음성데이타를 토대로 진행되는 것이 지배적이었다. 그러나 분할된 음성데이타를 마련하기 위해서는 많은 시간과 노력, 숙련 등을 요구할 뿐만아니라 그 자체가 인식도메인의 변화나 확장을 어렵게 하는 하나의 요인이 되기도 한다. 그래서 분할된 음성데이타의 사용을 가급적 피하고 그러면서도 성능을 떨어뜨리지 않는 신경망 학습법들이 나타나고 있다.

본 논문에서는 학습된 인식기를 이용하여 자동으로 한국어 음성데이타를 분할한 후 그 분할된 데이타를 이용하여 다시 인식기를 재학습시켜나가는 반복 과정을 소개하고자 한다. 여기에는 TDNN이 인식기로 사용되며 인식단위는 음소이다. 학습은 cross-validation 기법을 이용하여 제어된다.

1. 서론

한 나라의 언어가 제공할 수 있는 모든 표현을 인식도메인으로 삼기보다는 특정 도메인 하나를 선정하여 인식의 범위를 그 도메인으로 제한하는 방식이 음성인식의 통상적인 형태이다. 이는 현실적인 고려인 동시에 인식기의 성능을 높여보자는 의도로 해석할 수 있다. 그래서 본 연구에서도 호텔예약이라는 축소된 범위의 한국어 음성언어를 인식대상으로 삼고자 한다.

음성인식기로 신경망을 사용한다. 이때 사용되는 신경망은 TDNN(Time-Delay Neural Network) 구조이고 음소를 인식단위로 한다[1]. TDNN은 다른 신경망에 비해 그 규모가 비교적 큼에도 불구하고 우수한 성능을 지닌 것으로 평가받고 있다. TDNN의 학습은 신경망을 위한 가장 일반적인 학습방법인 BP(backpropogation) 알고리듬을 통해 이루어진다.

개략적인 학습 절차는 다음과 같다. 먼저 가볍게 학습된 신경망을 얻기 위해 부트스트래핑(bootstrapping) 단계를 일차적으로 거친다. 다음으로 자동분할과 재학습 과정이 이어지는데, 재학습은 자동분할에서 얻어진 데이타를 근거로 진행된다. 자동분할에는 Viterbi 디코딩 알고리듬이[3] 적용된다. 이와 같은 자동분할 및 재학습 과정을 몇번 반복 수행함으로서 데이타 분할의 정확도를 서서히 높여나가는 한편 분할이 개선된 데이타를 바탕으로 인식기가 인식 에러를 줄이게끔 유도한다. 매번의 재학습에서, 학습의 종결시점은 cross-validation 결과를 근거로 결정한다.

학습을 위해 도메인에서 채택된 문장은 400 여개이고 여기로부터, 음소분포를 감안하여, cross-validation용 데이타와 학습용 데이타를 선별하였다.

이후의 단원들은 TDNN의 기본 구조 및 페러미터 조정방법, 음소 그룹화, 학습 절차 등과 관련된 사항들을 언급하고 있다. 대규모의 실험이 현재 진행중이므로 실험결과는 차후에 소개하기로 하고 본 논문에서는 방법론의 타당성에 역점을 두어 기술하기로 한다.

2. TDNN(Time-Delay Neural Network)

2.1 구조

신경망의 각 노드가 보편적으로 하는 작업은 이전 계층의 활성값에 웨이트를 곱하고 그 합을 비선형함수(일반적으로 시그모이드 함수)에 통과시키는 일이다. TDNN에서는 이러한 노드에 딜레이 개념을 결합시켜 사용한다(그림 1). 예를 들어 $N = 2, J = 16$인 경우, 한 노드에는 시간상 세 지점에서 측정된 16 개짜리 입력 3 쌍이 동시에 들어와 48 개의 웨이트가 존재하게 된다. 이와 같은 구조로 인해 TDNN의 각 노드는 현재와 과거의 입력치들을 연관지어 판단하게 된다. 활성값을 계산하는 비선형함수로는 시그모이드 함수를 사용한다.

음소단위 인식에는 3 개의 계층으로 구성된 TDNN이 사용된다. /ㅂ/, /ㄷ/, /ㄱ/, 3 개의 음소를 인식하는 TDNN의

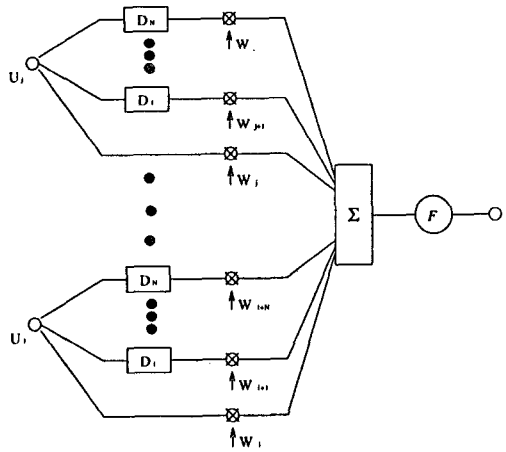

그림 1: TDNN 노드 구조

전체 구조가 그림 2에 제시되어 있다.

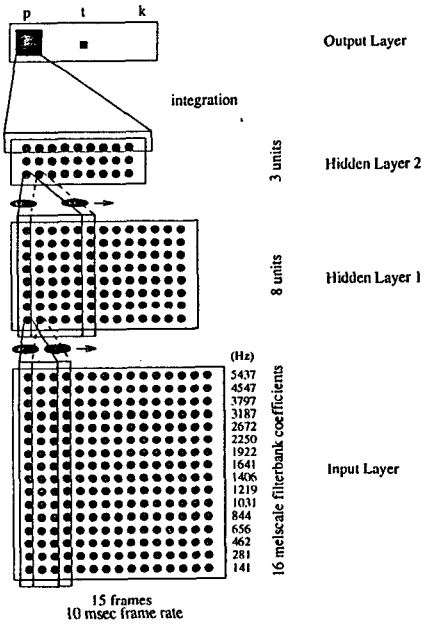

그림 2: TDNN 구조

첫번째 은닉층의 각 노드는 3 프레임 윈도우 내에 들어오는, 두번째 은닉층의 각 노드는 5 프레임 윈도우 내에 들어오는 활성치들을 이전 계층으로부터 받아들인다. 그리고 두번째 은닉층의 각 행은 출력 계층의 해당 노드와 연결된다. 출력 계층의 활성치 계산은 시그모이드 함수를 사용하는 두 은닉층과 달리, 두번째 은닉층의 활성치를 행단위로 모으는 것으로 대신한다.

2.2 패러미터 조정

TDNN 학습, 즉 웨이트 조정에는 BP(backpropogation) 학습 기법이 적용된다. 두 은닉층의 각 노드열은 앞 노드열을 그대로 복제한 형상이다. 그러므로 은닉층에서 같은 행에 위치한 노드들은 동일한 웨이트 집합을 가지게 된다. 즉 한 행에서 서로 대응되는 위치에 놓인 커넥션들은 같은 웨이트를 가지게 된다.

이러한 연결 구조를 구현하기 위해 먼저 신경망 전체를 대상으로 BP 전·후방 패스를 실시한다. 이때 대응되는 위치에 놓인 커넥션들은 서로 다른 새 웨이트들을 가지게 되는데 이 웨이트들의 평균을 구하여 대응되는 커넥션들에 그 평균값을 일률적으로 똑같이 할당해버리면 원하는 모양의 웨이트 구조를 얻을 수 있다. TDNN의 모든 노드들이 입력 패턴에 나타난 유용한 특징들을, 패턴내에서의 그 발생 위치에 상관없이, 감지할 수 있는 것은 바로 이와 같은 연결 구조 때문이다.

2.3 입력 패턴

정규화된 16 x 15 멜스케일 계수들이 하나의 입력 패턴으로서 입력층에 주어진다. 16 kHz로 샘플링된 음성신호에 Hamming window를 씌운 다음 5 msec마다 512-point FFT를 계산해 낸다. 이 FFT 결과에 멜스케일링을 적용하여 16 멜스케일 계수 1 프레임을 구한다. 다시 시간상 이웃하는 두 멜스케일 프레임을 하나로 평균하여, 10 msec마다 16 멜스케일 프레임이 구해지는 것처럼 데이타를 축소시킨다. 이렇게 만들어진 15 개의 멜스케일 프레임이 하나의 입력 패턴으로 주어지게 되는데, 그전에 15 프레임 전체에 대해, 각 멜스케일 계수로부터 계수 평균치를 뺀 다음 각 계수치가 -1에서 +1 사이에 놓이도록 정규화를 시킨다.

3. 음소 그룹화

하나의 TDNN으로 음소 전체를 인식케 하는 것은 학습에 소요되는 시간이나 학습의 질을 따져볼 때 그렇게 바람직하지 않다. 그래서 음소를 여러 그룹으로 묶고 그 그룹 수만큼 TDNN을 둬 각 TDNN이 담당 그룹만을 전문적으로 인식케 하는 다중 인식기 구조가 흔히 사용된다[2]. 자음과 모음은 각각 표 1과 표 2에 나타나 있는 바와 같이 그룹화가 이루어진다. 자음은 조음방법에 따라 6 개의 그룹으로 모음은 발음시 혀끝의 위치에 따라 4 개의 그룹으로 나뉘어진다[6]. 발음상의 어려움 때문에 모음 /애/, /에/; /외/, /왜/, /웨/; /얘/, /예/를 구분하지 않고 하나의 음소로 취급한다. 발음시 차별화가 제대로 이루어지지 않은 음소들을 인식시 구별해 내기란 거의 불가능하다.

표 1: 자음 그룹

	조음방법	음소
자음	폐쇄음	ㅂㄷㄱ
	경음	ㅃㄸㄲ
	격음	ㅍㅌㅋ
	마찰음	ㅅㅆㅎ
	파찰음	ㅈㅉㅊ
	유음(비음)	ㄴㄹㅁㅇ

표 2: 모음 그룹

	혀끝의 위치	음소
모음	F - H	아 야 어 여 와
	F - L	이 위 의 으
	B - H	우 유 오 요 워
	B - L	애(에) 외(왜, 웨) 얘(예)

이상의 10 개 그룹 TDNN 외에 3 개의 TDNN을 더 사용하는데, 하나는 입력 패턴이 자음인지 모음인지를 구별하는 TDNN이고, 나머지 두 개는 입력 패턴이 자음 그룹들 중 또는 모음 그룹들 중 어느 그룹에 속하는 가를 결정하는 TDNN이다. 이들 13 개의 TDNN로부터 나온 인식치들을 통합 활용하는 방안에 대해 아래에서 언급하게 될 것이다.

4. 부트스트래핑

부트스트랩핑은 최상의 결과를 바라고 하는 학습이 아니라 뒤이어 오는 본격적인 학습 과정(자동분할과 재학습)을 다소 안정되게 시작할 수 있게끔 가볍게 인식기를 학습시키는 데 그 목적있다.

부트스트래핑은 본격적인 학습 과정에 들어가기에 앞서 빠른 시간 안에 신경망 패러미터들을 초보적인 수준으로 학습시키는 것을 목표로 하므로, 부트스트래핑에는 적으나마 사람이 직접 분할한 정확한 데이타가 요구된다.

부트스트랩핑의 목적에 비춰 볼 때, 많은 분할 데이타를 부트스트랩핑에 투자할 필요는 없다. 그리고 본 연구의 큰 목적 중의 하나가 자동분할 방식을 개발하여 음성데이타 분할에 드는 많은 노력과 시간을 절약하고자 하는 것이므로, 부트스트랩핑에 요구되는 분할된 데이타는 적정 수준으로 그 양을 제한하는 것이 합당하다. 물론 부트스트랩핑이라 할지라도 분할 데이타가 많을수록 더 나은 학습 결과를 기대할 수 있을 것이다. 본 연구에서는 학습 문장들을 한 번 발음하여 분할한 데이타를 부트스트래핑용으로 활용하였다.

인위적으로 분할된 음성데이타로부터 토큰들을 추출해 내는 방법은 그림 3과 같다[4]. 하나의 분할 구간 안에서 20 msec마다 토큰들을 추출하되 양쪽 구간 경계로부터 20 msec 내에 추출점이 진입하면 추출을 중단한다. 토큰 하나의 크기는 150 msec로서 15 프레임 입력 패턴 하나에 해당한다.

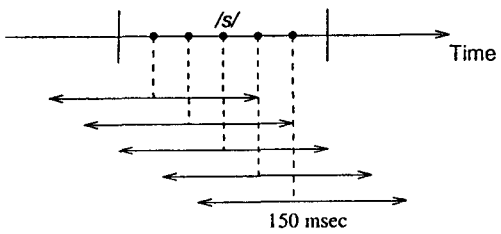

그림 3: 음소 토큰 추출방법

5. 자동분할 및 재학습

부트스트래핑 이후의 학습에 투입되는 음성데이타는 인위적인 음소분할이 전혀 되어 있지 않은 상태이다. 그런데 학습에 필요한 음소 패턴을 구성하기 위해서는 먼저 음성데이타가 음소단위로 분할되어 있어야 하므로, 음소 구간을 결정하는 모종의 자동분할 메카니즘이 이 시점에서 제공되어야 한다.

본 연구에서는 Viterbi 탐색 알고리듬*을 주축으로 하는 자동분할 방식을 채택한다. 분할은 학습용 문장들을 대상으로 이루어진다. 분할 과정은 다음과 같다.

먼저 분할하고자 하는 문장의 음성데이타에 대해, 데이타의 처음에서 시작하여 10 msec 단위로 TDNN 입력 패턴을 생성시킨다. 이렇게 만들어진 각 패턴은 그룹 TDNN들의 입력으로 주어지게 되고 이때 도출되는 신경망 결과들로부터 각 음소에 대한 활성값(주어진 입력 패턴이 해당 음소일 가능성)을 구해낸다. 결과적으로 볼 때 음소 활성값들이 10 msec 단위로 계산되어져 나오는 것이다. 이 음소 활성값 벡터들은 탐색 테이블의 시간축상에 놓여진다.

탐색 테이블의 나머지 축에는 분할하고자 하는 문장의 HMM 모델이 대응된다. 즉 임의의 한 음소에 대해 그림 4와 같은 HMM 모델을 사용하며, 문장 내에 존재하는 음소수만큼 이 음소 HMM 모델들을 결합시켜 문장 전체에 대한 하나의 HMM 모델을 구성하고 이를 탐색 테이블의 다른 한 축에 대응시키는 것이다.

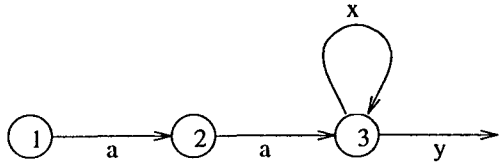

그림 4: 음소 HMM 모델

그림 4에서, 천이 a, x에 대응하는 천이확률(transition prob.)값으로 0.5를 고정 할당시키고 y에는 경험적으로 결정되는 값을 대응시킨다[5]. HMM 모델의 각 스테이트는 생성확률(emission prob.)을 보유하고 있어야 하는데,

---

* Viterbi 탐색 알고리듬의 일반적인 식은 다음과 같다.

1. Initialization

$$\delta_1(i) = \pi_i b_i(o_1), \qquad 1 \le i \le N$$
$$\psi_1(i) = 0.$$

2. Recursion

$$\delta_t(j) = max_{1 \le i \le N}[\delta_{t-1}(i)a_{ij}]b_j(o_t), \qquad 2 \le t \le T,\ 1 \le j \le N$$
$$\psi_t(j) = arg\ max_{1 \le i \le N}[\delta_{t-1}(i)a_{ij}], \qquad 2 \le t \le T,\ 1 \le j \le N.$$

3. Termination

$$P^* = max_{1 \le i \le N}[\delta_T(i)],$$
$$q_T^* = arg\ max_{1 \le i \le N}[\delta_T(i)].$$

4. Backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \cdots, 1.$$

위에서 t는 시간을, i, j는 스테이트를, o_t는 시간 t에서의 관측치(입력 패턴)를, $\delta_t(j)$는 시간 t에서의 관측치가 스테이트 j에 대응되는 패스의 점수들 중 가장 높은 값을, π_j는 스테이트 j에서 시작할 확률을, a_{ij}는 스테이트 i에서 스테이트 j로의 천이확률을, $b_j(o_t)$는 입력패턴 o_t에 대한 음소 활성값들 중 스테이트 j와 관련된 음소의 활성값을 각각 나타낸다. $\psi_t(j)$에는 백트랙킹용 정보가 들어간다.

본 HMM 모델에서는 테이블의 시간축상에 대응되어 있는 활성값들을 정규화시켜 이를 스테이트 생성확률로 삼는다. 각 활성값을 정규화시키는 데는 해당 음소의 기득확률(prior prob.)이 사용된다. 그림 4의 음소 HMM 모델에서 스테이트가 3 개로 고정되어 있는 것은 한 음소의 지속시간이 적어도 30 msec 이상이어야 함을 의미한다. 그런데 하나의 독립된 음소로 취급되는 포즈(pause)에 대해서는 스테이트 한 개짜리 HMM 모델을 사용한다.

탐색 테이블이 구축되었으면 이 테이블에 대해 Viterbi 탐색을 수행하게 되는데 이 탐색으로부터 최적의 음소 분할정보를 얻을 수 있다. 즉 문장 HMM 모델의 각 스테이트가 시간축상의 점(음성데이타 10 msec에 해당) 몇 개와 대응되는가에 대한 최상의 정보를 Viterbi 탐색을 통해 계산해 내는 것이다.

이상의 자동분할 과정을 거쳐 얻은 음소 분할 정보는 새로운 학습 데이타를 만들기 위해 활용된다. 새로 구성된 학습 데이타는 이제 13 개 TDNN을 재학습시키는데 투입되고 각 TDNN은 독립적으로 패러미터를 조정해 나간다. 한 번의 재학습에서, 학습의 종결시점은 cross-validation 결과를 근거로 결정한다.

끝으로 이상과 같은 일련의 자동분할 및 재학습 과정을 여러 번 되풀이 한다. 그와 같은 반복적인 학습을 거듭하면서 분할 정보의 정확성은 점점 높아지게 되고 반대로 신경망의 인식 에러는 차츰 감소해 간다.

6. 전체 음소인식기의 학습

신경망의 학습 효과를 높이기 위한 많은 기술적인 방법들이 제안되어 왔는데 그 쉬운 예로 모멘텀 학습법을 들 수 있을 것이다. 여기서 학습 효과를 높인다 함은, 패러미터 수렴 속도를 촉진시켜 학습에 소요되는 시간을 단축시키고 더불어 학습의 질적인 면도 발전시켜나감을 의미한다.

효과적인 학습을 위해 본 연구에서는 모멘텀뿐만아니라 단계적인 학습 방법을 함께 사용하였다. 단계적인 학습법이란, 처음부터 가용한 학습 패턴 전부를 학습에 투입하는 대신 처음에는 몇 개의 패턴만 학습에 참여시키고 점차 그 양을 늘여가는 점진적인 학습 방법을 말한다[1]. 본 연구에서는, 음소당 패턴수를 처음에는 1 개로 제한하고 다음 단계로 넘어갈 때마다 패턴양을 배가시키는 방법을 사용하였다. 매단계 학습은 마찬가지로 cross-validation에 의해 제어된다.

그리고 지금까지 논의된 모든 학습에 WURM(Weight Update Reduction Method) 기법을 추가로 적용한다. 이 방법 역시 학습 효과를 증진시키기 위한 방안의 하나이다. 그 내용은 비교적 간단한데, 웨이트를 입력패턴이 주어질 때마다 조정하던 것을 몇 회에 한 번씩만 조정되도록 하여 웨이트 조정 횟수를 크게 줄이는 것이다. 단 그 몇 회 동안 발생했던 변화량(gradient)들은 따로 모아두었다가 다음 번 웨이트 조정시 한꺼번에 적용되도록 한다.

한편 앞에서 하나의 독립된 음소로 간주한 포즈는 자음의 폐쇄음 그룹에 포함시켜 학습시킨다.

지금까지 13 개 TDNN 결과들을 종합하여 각 음소에 대한 활성값을 구해내는 구체적인 방법을 논의한 적은 없었다. 사실 그러한 음소 활성값들을 계산할 수 있어야 비로소 자동분할이 가능해진다. 그런데 입력 패턴이 주어졌을 때 음소 활성값들을 구하는 가장 간단한 방법은 각 TDNN 결과에 실험적으로 얻은 가중치를 결합시키는 것인데 이 경우 적게는 3 개의 가중치만 두면 된다.

본 연구에서는 이 가중치 방식의 확장 형태인 LP-TDNN(Large Phonemic TDNN)을 구성하였다[2]. LP-TDNN의 구조는 그림 5와 같다. 출력층은 인식음소수만큼 노드를 가지고, 각 TDNN에서 두번째 은닉층의 매행은 해당 출력노드 하나와 연결된다. 이를 출력노드측에서 보면, 각 출력노드는 3 개의 TDNN과 연결된다. 그림 5는 /ㅂ/ 출력노드가 해당 TDNN 3 개와 연결되어 있는 모습을 보여주고 있다. 은닉층에 부여되는 웨이트들은 재학습 결과를 그대로 복사한 값들이다. 이제 새로 생긴 커넥션(출력층과 은닉층들 사이의 커넥션)의 웨이트들을 초기화 한 다음 LP-TDNN 학습으로 들어간다. 이번 학습은 새로운 웨이트들만을 대상으로 진행되고 은닉층 웨이트들은 고정되어 변하지 않는다. 학습 데이타로는 재학습 과정에서 사용된 것을 그대로 쓰고 여전히 cross-validation에 의해 학습을 제어한다. 이 LP-TDNN 학습을 다 마친 후에야 비로소 자동분할 과정으로 들어갈 수 있으며 분할시 학습된 LP-TDNN을 그대로 사용한다.

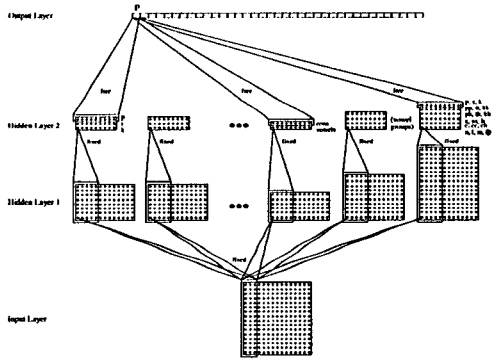

그림 5: LP-TDNN

이상에서 한국어 음소인식을 위한 하나의 방법론을 살펴보았다. 그 개략적인 내용은, 먼저 개별 TDNN들을 독립적으로 부트스트래핑시킨 다음 이들을 하나의 LP-TDNN으로 묶고 이 LP-TDNN을 추가 학습시킨다. 그런 다음 학습된 LP-TDNN을 바탕으로 학습 문장에 대한 자동분할을 실시한다. 그 분할 결과를 토대로 개별 TDNN들을 다시 학습시키고 이와 같은 분할과 재학습 과정을 몇 번 되풀이 한다.

지금까지 본 논문에서 제안된 방법이나 구조 외에 그들을 대체하거나 개선할만한 또다른 형태의 방법이나 구조가 실제로 얼마든지 가능하다. 그러나 본 연구에서는 음소 TDNN에 의해 한국어 음성언어를 인식하는데 있어 자동분할에 의한 신경망 학습의 가능성을 검증하고 그를 위한 신경망 구조를 개발하는데 역점을 두었다. 이는 여태까지 대용량의 손으로 분할된 데이타를 필요로 했던 신경망 음소인식 연구에 커다란 기여를 할 수 있을 것으로 생각한다.

참고 문헌

[1] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. J. Lang, "Phoneme Recognition Using Time-Delay Neural Networks", *IEEE Trans. on ASSP*, vol. 37, no. 3, Mar., 1989.

[2] A. Waibel, H. Sawai, K. Shikano, "Consonant Recognition by Modular Construction of Large Phonemic Time-Delay Neural Networks", *ICASSP-89*, 1989.

[3] L. Rabiner, B. H. Juang, "Fundamentals of speech Recognition", *Prentice Hall*, 1993.

[4] M. Miyatake, H. Sawai, Y. Minami, K. Shikano, "Integrated Training for Spotting Japanese Phonemes Using Large Phonemic Time-Delay Neural Networks", *ICASSP-90*, 1990.

[5] T. Robinson, M. Hochberg, S. Renals, "The use of recurrent neural networks in continuous speech recognition", *Personal communication with T. Robinson*

[6] 정 차균, "TDNN을 이용한 한국어 음성인식에 관한 연구", 포항공대 석사학위논문, 1991.