한글 문서 인식 시스템 SILNOON의 개발 현황

조 창 제, 이 승 호, 고 병기, 김 의경, 이 광호, 하 진영, 김 진 형

한국과학기술원 전산학과

Current Status of SILNOON Project for the Development of Hangul Document Recognition System

Changje Cho, Seungho Lee, Pyongki Ko, Eukyoung Kim, Kwangho Lee, Jinyoung Ha and Jin H Kim

Department of Computer Science, KAIST

요 약

본 논문에서는 한국과학기술원 전산학과 인공지능연구실에서 개발하고 있는 한글 문서 인식 시스템 SILNOON을 소개한다 본 연구는 한글로 작성된 인쇄체 문서를 인식하여 컴퓨터 화일로 구성하고, 인식된 문서를 편집 및 수정하여 레이저 프린터를 통하여 출력할 수 있는 실용적인 시스템의 개발을 그 목적으로 한다. 이 시스템은 크게 전처리, 문자 인식, 후처리 등으로 구성된다. 본 논문에서는 문자 인식, 후처리 과정에 중점을 두어 설명하고 개인용 컴퓨터(IBM PC/AT)에 구현하여 실험한 결과를 발표한다

I. 서 론

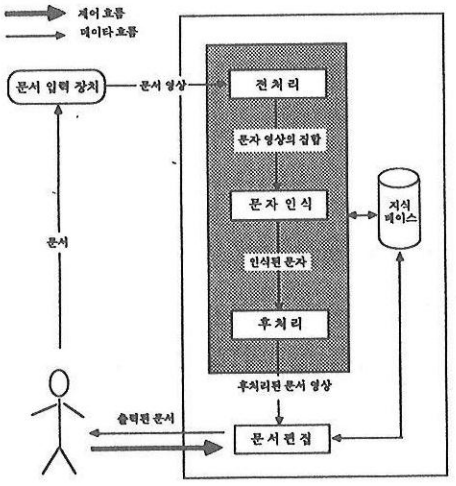
본 논문에서는 한국과학기술원 전산학과 인공지능연구실에서 개발하고 있는 한글 문서 인식 시스템 SILNOON을 소개한다 지금까지 음소의 인식이나 문자의 인식을 위한 연구[1-5]가 오래 동안 계속되어 왔지만 아직 실용적인 한글 문서 인식 시스템을 구성하기 위한 연구는 거의 없었다. 한글 문서 인식 시스템은 우리가 일상적으로 사용하는 한글 문서의 인식을 목표로 한다 한글 문서란 잡지의 한 페이지나 신문, 혹은 연구보고서, 또는 서류 등을 일컫는다 문서는 도형, 사진과 같이 문자가 아닌 영역과 본문 기사와 같이 문자의 영역으로 구분된다 따라서 개발중인 시스템은 문서의 영상을 입력장치로 받아들여 도형 및 사진은 컴퓨터에서 처리할 수 있는 영상 자료로, 문자부분은 각 문자에 해당하는 코드로 출력한다. 사용자의 요구가 있을 때에는 글자 및 도형의 위치 정보도 같이 출력하도록 설계되어 있다 따라서 입력된 문서의 재생이 가능하고 입출력을 Bit-mapped 그래픽 기능을 갖는 편집기와 연결할 수 있도록 설계하였다

II. 한글 문서 인식 시스템의 개요

한글 문서인식 시스템은 그림 1과 같이 크게 전처리, 문자 인식, 후처리의 세 단계로 구분할 수 있다[6] 전처리 과정은 입력영상을 그림부분과 문자부분으로 구분한 다음 그림부분은 영상자료 형태로, 문자부분은 문자영상의 집합으로 보관하여 문자 인식과정에서 처리할 수 있도록 해 주는 것이다 이 과정에서는 문서의 영상이 항상 똑 바로 들어온다고 가정할 수 없기 때문에 입력된 문서영상의 기울어진 각도를 계산하고 그 각도만큼 영상을 돌려서 바로 잡아야 한다 이 과정에서 속도, 기억 용량, 그리고 영상의 변형 등 많은 문제점이 제기된다 그 다음 입력문서의 특성에 따라서 전체적인 구조를 분석한다 즉 입력문서가 논문, 신문 혹은 일반 서적인가에 따라서 그에 대응되는 구조분석 알고리즘을 사용한다 본 연구실에서는 우선 단순한 형태로서 연구보고서의 구성 형식을 이해할 수 있는 알고리즘과 신문의 구조적 특성 분석을 통하여 기사를 추출할 수 있는 알고리즘을 개발하였다[7] 연구보고서 형식은 가로쓰기이며 전단(1-Column) 형식이다 이렇게 단순한 형태의 문서에서는 그림 영역과 문자영역의 구분이 용이하고 문자영역에서는 열(Line)간의 거리, 사용한 활자의 크기 등을 쉽게 추론할 수 있다 활자의 크기를 알아낸 후에 문자영역을 각 열로 구분하고, 각 열은 문자영상의 집합으로 분리해낸다.

문자 인식 단계는 전처리 과정에서 넘겨 받은 문자단위의 영상을 분석하여 이를 부호화하는 단계이다 SILNOON 시스템의 인식 대상은 특정 활자체의 인쇄체 문장이기 때문에 비교적 단순한 알고리즘도 우리의 요구에 만족할 수 있다고 생각한다

문서 인식 시스템의 마지막 단계는 후처리 단계이다 이 단계에서는 문서 인식 시스템이 문자 단위 영상만을 독립적으로 분석하여 도출한 인식 결과를 확대하여 전체를 볼 수 있는 관점에서 오류를 수정하는 과정이다 예를 들어 문자인식 단계에서 "컴퓨터"라는 인식결과가 나왔다면 이는 필히 "컴퓨터"의 오인식일 것이다 이 단계에서는 사용한 인식 알고리즘의 특성, 즉 어느 글자가 어느 글자로 잘못 인식되는 경우의 수가 많았다는 등의 통계적 자료와 음절의 출현빈도, 혹은 단어의 출현 가능성 여부 등의 여러가지 정보를 종합하여야 한다. 가능하다면 문맥적



<그림 1 : 한글 문서 인식 시스템의 구성도>

지식까지 사용하는 것이 더욱 좋은 결과를 낳을 수 있겠지만 현재의 기술 수준으로는 인식한 문장의 "이해"를 통한 문장의 수정은 아직 요원한 단계이다.

III. 통계적 방법에 의한 문자 인식

기존의 한글 문자 인식에 대한 대부분의 연구는 상향식 제어 전략을 사용하여 추출된 기본적인 특징을 특성점 벡터로 구성한 다음에 분류 분석이나 통계적 이론 등을 이용하여 인식하는 통계적 방법을 사용하였거나, 기본 특징 간의 상호관계를 문법 형태로 표현하여 인식하는 구문론적 방법을 사용하였다.

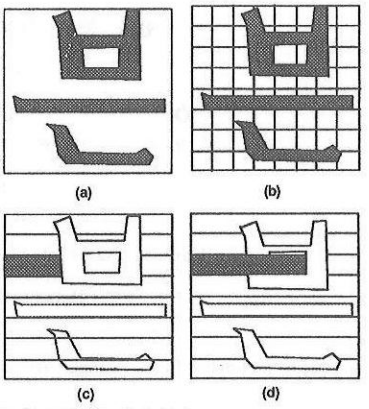
문서를 입력하는 방법으로 카메라나 스캐너 또는 팩시밀리를 사용할 수 있는데, 문서 자체가 2차원이므로 스캐너를 사용하는 것이 실용적이다. 그러나 스캐너의 기계적 제약때문에 문자 영상이 조금씩 변형될 수 있고 변형의 정도는 임의로 되기 때문에 예측할 수 없다. 이렇게 노이즈가 포함된 문자의 자획을 추출하기 위해서, 세선화하면 노이즈부분이 세선화에 반영되어 인식을 어렵게 하는 요인이 된다. 일반적으로 문자 패턴을 인식하는 알고리즘은 기본적인 연산이 간단하여 하드웨어로 쉽게 구현되고 인식율이 높아야 하는데[8] 통계적 방법은 이러한 특성을 잘 반영한다.

1. 특성 추출

통계적 방법으로 한글을 인식하기 위하여 문자의 부분적이고 전체적인 특성을 반영하는 mesh 특성점, peripheral 특성점을 사용하여 각 문자의 특성을 추출한다[9]. 두 특성점은 한글 문자를 서로 다른 견해에서 본 특성을 보여 주며 이런 특성점들은 한글 문자에서 쉽게 구할 수 있다.

가. Mesh 특성점

문자 패턴을 획의 배열로, 즉 검은 부분의 존재로서 특성을 줄 수가 있다. mesh 특성점은 검은 부분의 모양을 반영한다. 이 특성점을 추출하는 방법은 그림 2의 (a)와 같은 문자 패턴을 그림 2의 (b)처럼 8x8의 부분지역(cell)으로 나눈다. 그리고 각 지역의 검은 부분의 면적에 해당하는 값을 계산한다. 따라서 mesh 특성점은 64개의 스칼라 값으로 구성된다.



<그림 2 : mesh 특성점과 peripheral 특성점의 추출>

나. Peripheral 특성점

다른 견해로 문자 패턴은 문자의 윤곽으로 특성을 줄 수가 있다. peripheral 특성점은 문자 패턴의 윤곽을 반영한 특성점을 추출하는 방법은 다음과 같다. 그림 2의 (a)의 문자를 왼쪽, 위쪽, 오른쪽, 아래쪽 각각의 방향에서 다음과 같이 수행한다. 왼쪽 방향을 고려하면, 사각형을 그림 2의 (c)와 같이 8개의 작은 사각형으로 나누어 각 사각형에서 오른쪽으로 따라가면서 처음으로 흰 부분에서 검은 부분으로 변하는 부분까지의 면적을 계산한다(그림 2의 (c)의 검게 칠해진 부분). 똑같은 방향으로 두번째로 흰 부분에서 검은 부분으로 변하는 부분까지의 면적을 계산한다. 이렇게 하여 왼쪽 방향에 대해서 16개의 값을 구하고 나머지 세 방향에 대해서 반복 수행해서 총 64개의 스칼라값으로 구성된 peripheral 특성점을 구한다.

2. 인식 알고리즘

입력 문자를 인식하는 과정은 먼저 앞서의 방법으로 입력문자에 대해 특성점 벡터 X를 구해 시스템에 저장되어 있는 문자 벡터간의 거리를 구해 가장 가까운 문자로 인식하는 방법이다. X는 64-차원의 벡터 $X = (x_1,\ldots,x_{64})^t$ 로 표현한다. 각 문자의 특성점 벡터의 값은 스캐너를 통하여 입력될 때마다 그 값이 조금씩 변하므로 여러 번 같은 글자에 대하여 그 값을 구하여 평균과 분산을 구할 필요가 있다. $M_i=(m_1, m_2,\ldots, m_{64})^t$를 i 번째 문자의 특성점의 평균 벡터라 하고, V_i를 covariance vector라 하자. 문자간의 유사성을 계산하는 식은 다음과 같다.

$$\min_i (X - M_i)^t \, V_i^{-1} (X - M_i)$$

$$V_i(i,j) = \begin{cases} \text{variance of } i^{th} \text{ cell}, & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

그러나 실수 연산은 많은 계산 시간을 요하므로 다음과 같이 정수 연산으로 근사시켜 계산하였다. 여기에서 m_i는 M_i의 각 원소를 정수로 근사시킨 벡터이다.

$$\min_i (X - m_i)^t (X - m_i)$$

이러한 계산 방법의 단점은 시스템에 저장되어 있는 문자 패턴의 수에 비례하여 계산시간이 증가한다는 점이다. 이 점을

개선하기 위하여 문자 패턴을 계층적으로 구성하여 우선적으로 문자의 구조적 특성, 즉 수평, 수직 모음이 존재하는가 등의 간단한 정보를 이용하여 몇 개의 그룹으로 분류한다 다시 말해서 한글 문자는 "으"에서 처럼 긴 수평 모음을 가지는 경우(G1)와 그렇지 않은 경우(G2)로 대별된다 G1 에서는 수평 모음 아래에 자소의 유무에 따라 두 개의 그룹으로 나눌 수 있다 G2에 속하는 문자는 반드시 수직 모음을 가지므로 양성모음인가, 음성모음인가에 따라 두 개의 그룹으로 나눌 수 있다 그룹의 수를 증가함에 따라 위에서 제시한 식을 계산하는 데 시간을 줄일 수 있지만 그룹을 알아내는 데 많은 시간이 걸리고 그룹 분류 오류가 증가하여 인식률을 감소시킨다. 수평모음의 유무는 간단한 계산으로 찾을 수 있고, 양음성 수직 모음의 구분도 쉽게 가능하다.

순수한 통계적 방법은 빠른 계산, 노이즈에 덜 민감한 특징을 가지고 있지만 유사한 문자를 구분하기에는 힘드는 방법이다 이러한 통계적 방법의 특징을 살리면서 구조적 특징을 추가하는 연구와 각 문자의 특징점 벡터를 tree로 구성하여 인식 시간과 인식률을 개선하려는 연구를 계속하고 있다

IV. 후처리

1. 후처리의 필요성

자동 문서인식 시스템의 인식과정에서는 입력문자들의 유사성 및 입력문자에 포함된 노이즈로 인하여 입력문자를 항상 정확하게 인식하지는 못한다. 따라서 오인식된 문자를 수정해줄 수 있는 후처리 과정이 있어야만 신뢰성있는 문서인식 시스템을 구현할 수 있다. 사람은 단어내 문자의 연결관계, 단어간의 연결관계, 문장의 구조, 문서의 주제 등의 다양한 문맥적 지식을 이용하여 비교적 정확하게 문서인식을 수행한다 하지만 컴퓨터가 사람이 사용하는 모든 문맥적 지식으로 의미를 이해하고 수정하는 것은 구현상의 어려움과 많은 계산 비용으로 인하여 그 효율성이 줄어든다. 따라서 일반적으로 자동 문서인식 시스템에서는 단어내 문자의 연결관계에 관한 정보만을 사전으로부터 얻은 다음 그것을 이용하여 단어별로 오인식 수정을 하고 있다

2. 오인식 수정 알고리즘

문자인식 시스템에서의 오인식 수정 알고리즘은 다음과 같이 확률적으로 모델링된다 [12] 문자인식 시스템에 입력된 입력어절을 $Z = Z_1Z_2 \ldots Z_n$이라 하고, 그 입력어절에 대한 인식어절을 $X = X_1X_2 \ldots X_n$이라 하자. 이때 인식된 어절이 X일 때 입력어절이 Z일 확률 P(Z|X)는 Bayes 정리에 의하여 다음과 같이 표현된다

$$P(Z|X) = P(X|Z) * P(Z) / P(X) \quad \text{----- (1)}$$

위의 (1) 식에서 P(X|Z)는 어절 Z가 입력되어 X로 인식될 어절간의 혼동확률을 나타내며 문자인식 시스템의 특성을 반영한다 P(Z)와 P(X)는 각각 Z의 사전확률과 X로 인식되는 확률을 나타낸다 인식어절로 X가 주어졌을 때 모든 입력 가능한 어절중에서 (1) 식에서의 P(Z|X)를 가장 크게하는 어절 Z를 구하는 것이 오인식 수정 알고리즘의 목적이다 (1) 식에서 우변의 분모항인 P(X)는 가능한 모든 Z에 대하여 공통이고 또 Z와 X는 독립이므로, 우변의 분자항인 P(X|Z) * P(Z)를 가장 크게 만드는 Z가 P(Z|X)를 가장 크게 만든다 그러므로 (1) 식에서 양변에 P(X)를 곱해준 다음, Log 함수를 취하면 (1) 식은 다음과 같이 된다

$$\log P(Z|X) + \log P(X) = \log P(X|Z) + \log P(Z)$$
$$G(X,Z) = \log P(X|Z) + \log P(Z) \quad \text{----- (2)}$$

문자인식 시스템의 특성상 한 문자와 그 다음에 인식될 문자 사이에는 서로 독립적이므로 어절간의 혼동확률은 각 문자들의 혼동확률의 곱으로 나타낼 수 있다

$$P(X|Z) \approx P(X_1X_2 \ldots X_n|Z_1Z_2 \ldots Z_n)$$
$$= P(X_1|Z_1) * P(X_2|Z_2) \ldots P(X_n|Z_n) \quad \text{--- (3)}$$

(3) 식에 의하여 (2) 식은 다음과 같이 다시 쓸 수 있다.

$$G(X,Z) = \Sigma_{1 \le i \le n} \log P(X_i|Z_i) + \log P(Z) \quad \text{--- (4)}$$

여기서 P(Z) 즉, 어절들의 사전확률은 일반적으로 구하기가 매우 어려우므로 대신 사전을 사용하여 그 어절이 사전에 있는가에 대한 존재여부로써 사전확률을 대신한다[11]

if Z exists in Dictionary,
then P(Z) = 1
else P(Z) = *-100000* *(작은 음의 값)

그러므로 (4) 식에 의하여 우리는 인식어절 X에 대하여 G(X,Z)를 가장 크게하는 어절 Z를 구할 수 있게 된다

SILNOON 시스템에서 구현된 오인식 수정을 위한 후처리 알고리즘은 하향식 방법인 Dictionary Look-Up 알고리즘과 상향식 방법인 Modified Viterbi 알고리즘을 한글 문장의 띄어쓰기 단위인 어절의 특성에 맞게 결합시켜 개발되었다[10-12] 이 알고리즘을 개략적으로 기술하면 다음과 같다

먼저 인식어절이 입력되면 그 어절을 구성하고 있는 각 문자들에 대한 혼동 문자 리스트를 구한 다음, 그 리스트로부터 생성가능한 모든 문자 스트링들을 만들어 낸다 생성된 각 문자 스트링들에 대하여 먼저 단어 사전에 존재하는가를 확인하고, 사전에 존재할 경우에만 그 문자 스트링을 구성하고 있는 각 문자들의 혼동 확률과 사전 확률의 곱에 대한 값으로서 G(X,Z) 값을 계산한다 본 시스템에서는 어절의 사전확률을 사전의 존재여부에 대한 것과 그 어절을 구성하고 있는 각 문자들의 사전확률의 곱으로써 표현하였다. 계산된 G(X,Z) 값들중에서 가장 큰 값을 갖는 문자 스트링을 입력어절에 대한 수정어절로 간주하여 출력시킨다 만약 이때 생성된 문자 스트링들이 모두 사전에 존재하지 않을 경우에는 생성된 모든 스트링들에 대하여 위와 같은 방법으로 G(X,Z) 값을 계산한 다음, 가장 큰 값을 갖는 문자 스트링을 수정어절로 출력시킨다

3. 사전의 구성

SILNOON 시스템에서 사용된 사전은 어미나 조사 등의 활용이 발달되어 있는 한국어의 특성을 고려하여 각 단어들을 두 부류로 분류하여 구성하였다. 실사 사전에는 체언, 용언의 어간 및 수식언등을 저장하였고 허사 사전에는 조사, 어미 및 접미사 등을 저장하였다 허사 사전은 어절로부터 조사 및 어미의 분리를 용이하게 하기위하여 TRIE 구조로 구성하였고, 실사 사전은 이진 탐색을 위해 가나다 순으로 정렬된 선형구조로 구성하였다

이러한 방식으로 사전을 구성함으로써 사전의 크기와 탐색 시간을 줄일 수 있었으며 효율적으로 사전을 관리할 수 있었다. 문자 스트링에 대한 사전의 탐색은 허사 사전의 TRIE 탐색을 이용하여 먼저 문자 스트링으로부터 허사 부분을 분리해낸 다음, 그 나머지 부분만을 실사 사전에서 이진 탐색을 이용하여 찾는다. 만약 문자 스트링내에 허사 부분이 존재하지 않을 경우에는 문자 스트링 전체를 실사 사전에서 탐색하게 된다.

V. 실험 및 결과 분석

SILNOON 시스템은 개인용 컴퓨터(IBM PC/AT)상에 Microsft C로 구현되었다. 실험한 한글 문자 집합은 한글 기계화 연구소에서 발표한 한글 문자의 찾기 순서에 의해 상위 1500자의 문자들중에서 생소한 문자들은 제외하고 외국어 표기를 위한 문자들을 추가해서 만든 990 자로 하였다. 이 990 자에 대한 누적 사용 빈도율은 99%이다[13]. 한글 문자의 크기는 본 논문에서 사용한 문자의 크기인 QLBP의 H4의 폰트를 사용하였다.

990 자의 문자가 포함되어 있는 7 장의 문서를 가지고서 문자인식 과정에 대하여 실험한 결과가 표 1에 나타나 있다. 그리고 표 2는 일반적인 내용을 포함하고 있는 20 장의 문서들에 대하여 문자인식과 후처리 과정을 실험해서 나온 결과를 보여준다. 각 문서들의 평균 어절 인식률과 평균 어절 수정률은 각각 85.0%와 97.5%로서 비교적 높은 인식률과 수정률을 나타내었다.

후처리 과정에서는 정인식된 어절을 그대로 출력하거나 오인식된 어절을 정수정하는 경우가 대부분이었으나 정인식된 어절을 오수정하는 경우도 약 1.0% 정도 발생하였다. 이것은 후처리 시스템에서 사용된 알고리즘 자체가 다양한 문맥적 지식과 단어의 의미에 의하여 오인식을 수정하는 것이 아니라, 단순히 사전에 있는 구문적 지식만을 가지고서 어절들을 처리하기 때문에 발생한다.

SILNOON 시스템이 1000 자가 포함되어 있는 문서를 처리하는데는 전처리, 문자인식 및 후처리 과정등을 모두 포함하여 약 10분 정도가 소요된다. 이것은 1 초당 1.5 자씩을 처리하는 속도에 해당된다. 처리시간의 대부분은 문자인식 과정을 수행하는데 소비된다.

VI. 결 론

지금까지 한글 문서 인식 시스템을 소개하였다. 이 시스템은 크게 전처리, 문자 인식, 후처리로 구성되는데 본 논문에서는 문자 인식과 후처리 과정에 대해 기술하였다. 문자 인식 알고리즘은 실제로 사용하는 크기의 문자의 인식에 중점을 두었지만, 특정 활자체에만 제한하였다. 앞으로 여러 가지 폰트와 크기로 작성된 문서의 인식에 대한 연구가 수행되어야 하겠다. 입력 문자의 유사성과 포함된 노이즈로 인하여 입력 문자를 항상 정확히 인식하지 못하기 때문에 후처리 과정이 반드시 필요함을 실험으로써 그 유효성을 증명하였다.

	인 식 률	시 간
문서 1	89.4%	약 11분
문서 2	91.7%	약 10분
문서 3	96.4%	약 10분
문서 4	95.4%	약 10분
문서 5	95.4%	약 10분
문서 6	96.2%	약 10분
문서 7	96.3%	약 10분

<표 1 : 문자의 인식률과 시간>

실험 화일	어절수	글자수	어절 인식률 (%)	어절 수정률 (%)	
				정수정률	오수정률
1	118	360	80.5	100.0	0.0
2	103	345	69.9	96.1	1.0
3	92	269	80.4	100.0	0.0
4	71	195	87.3	95.8	1.4
5	82	230	93.9	97.6	0.0
6	102	301	85.3	95.1	2.0
7	80	215	81.3	100.0	0.0
8	123	356	90.2	95.1	1.6
9	87	251	86.2	97.7	2.3
10	109	336	88.1	99.1	0.0
11	121	364	84.3	94.2	0.8
12	134	372	91.0	98.5	0.0
13	71	205	93.0	100.0	0.0
14	165	467	90.9	97.0	1.2
15	128	369	86.7	96.1	3.1
16	97	275	87.6	97.9	2.1
17	141	406	89.4	97.9	0.7
18	90	253	88.9	98.9	1.1
19	101	302	84.2	97.0	1.0
20	116	344	86.2	94.8	1.7

<표 2 : 후처리 실험 결과>

참고 문헌

[1] 이 주근, 이 광우, "한글 문자의 인식에 관한 연구(II)," 전자공학회지, 제 7권 제 3호, pp. 130-136, 1970년 12월.

[2] 김 태균, T. Agui, "Syntactic법에 의한 한글의 패턴 인식에 관한 연구," 전자공학회지, 제 14권 제 5호, pp. 154-160, 1977년 12월.

[3] 최 병욱, T. Ichikawa, H. Fujita, "한글 인식에 있어서의 자소추출," 전자공학회지, 제 18권 제 2호, pp. 36-43, 1981년

[4] 이 주근, 남궁 재찬, 김 영건, "한글 Pattern에서 Subpattern분리와 인식에 관한 연구," 전자공학회지, 제 18권 제 3호, pp. 1-8, 1981년 6월.

[5] 박 종욱, 이 주근, "Shape Pattern에 의한 필기체 한글 인식," 전자공학회지, 제 22권 제 5호, pp. 420-428, 1985년

[6] 이 성환 외 7 인, "문서 인식 및 검색을 위한 전처리 시스템의 설계 및 구현," 한국정보과학회 추계 학술발표회 논문집, pp. 503-509, 1986년 10월.

[7] 김 형훈, 이 성환, 김 진형, "신문의 구조적 분석을 통한 한국 신문 기사의 추출," 인공지능연구회 춘계 학술발표회 논문집, 1988년 3월.

[8] J. Tsukumo and K. Asai, "Machine Printed Chinese and Japanese Character Recognition Method and Experiments for Reading Japanese Pocket Books," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 162-167, 1986.

[9] K. Mori and I. Masuda, "Advaces in Rcognition of Chines CHaracters," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 162-167, 1986.

[10] 박 진규, 김 진형, "한글 문서인식의 오인식 수정에 관한 연구," 한국정보과학회 추계 학술발표회 논문집, pp. 94-97, 1987년 10월.

[11] J. J. Hull and S. N. Shrihari, "Experiments in Text Recognition with Binary n-Gram and Viterbi Algorithm," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, pp. 520-530, 1982.

[12] R. Shinghal and G. T. Toussaint, "Experiments in Recognition with the Modified Viterbi Algorithm," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, pp. 184-193, 1979.

[13] 한국 기계화 연구, 한글 기계화 연구소, 1975.