자동문서판독 후처리를 위한 수정된 n-gram 알고리즘

김일회 * , °유근호 ** , 이철희 *

* 숭실대학교 전자계산학과, ** 육군사관학교 전자공학과,

A Modified Binary n-gram Algroithm for the Postprocessing of the Automatic Document Reading

Il-Hwoe Kim *, Keun-Ho Ryoo **, Cheol-Hee Lee *

*Dept.of Computer Science, Soong Sil Uni, **Dept.of Electronics, Korea Military Academy

ABSTRACT

This Paper proposed the modified binary n-gram algorithm for the contextual post processing system in English sentence.

Backward gram was used to correct the first position error in a word. It is not requires additional storage but more times of comparison It allows interactive correction routine.

Experiments were implemented using PASCAL languge on a micro computer, IBM PC/XT.

This algorithm improves the correction rate around 4 ~ 5% on a limited experimental environments.

1. 서 론

인간이 다루어야 할 정보의 양이 많아짐에 따라 빠른 시간내에 문서를 인식할수 있는 자동 문서 판독기의 개발이 필요하다. 문서의 인식과정은 전처리과정, 문자인식과정 및 인식된 결과를 단어에 대한 정보로서 검색하는 후처리 과정으로 이루어진다.

문자인식 과정에서의 인식률은 잡음이나 왜곡, 모양의 다양성, 전이와 회전등에 의하여 인식률이 좋지 않게 될수 있다. 따라서 문서인식 시스템에서도 인간과 같이 문맥정보 (Contextual information) 를 가지고 문자를 인식한다면 더 좋은 인식률을 성취할 수 있을것이다. 문맥 정보의 요소로는 글자의 순서, 단어의 상관도, 문장 구조, 어법, 문체, 주제의 내용 등이있다. 이들 문맥정보의 요소에 따라 지식 표현 방법이 다르며, 그에 따른수행 시간에도 차이가 있다.

이 논문에서는 비교적 빠른 시간내에 문서를 처리할 수 있는 글자 순서를 문맥정보로 이용하는 방법에 대해서 연구하였다. 이 방법으로는 크게 2 가지 가 있다. Viterbi 방법[1]은 글자들 사이의 확률에 의하여 고정되는 통계적 방법으로 재귀적으로 실현시킬 수있기 때문에 구현이 간단한 반면 최종 결과를 얻기까지 많은 비교가 필요하다. Binary n-gram 방법[4] 은 단어안에서 글자들의 위치에 따른 합법성을 고려하여 틀린위치를 탐지하고 옳은 글자를 고정시키는 구조적 방법이다. 그러므로 효율적인 오류 탐지과정을 이용할 수있다.

그러나 두가지 방법 모두 단어의 첫번째 오류에 대해서는 비교대상이 없기 때문에 viterbi방법에서는 고정이 안되고, binary n-gram 방법에서는 탐지는 되지만 고정은 되지 않는다. 또한 탐지 결과에 따라 거부되는 경우도 발생한다.

이 논문에서는 비교 과정을 뒷글자로 부터 앞글자를 비교할 수있는 backward gram 을 사용하여 첫 문자의 오류에 대해서 고정할 수 있게시도 하였고 거부되는 경우는 heuristic 하게 구한 혼동표를 이용하여 사용자가 선택할 수 있도록 설계하였다.

또한 기존의 방법과 본 논문에서 제안한 방법을 비교 검토하였다. 따라서 제안한 방법에서는 단어의 첫번째문자의 오류의 탐지 뿐만아니라 고정도 가능하며, 거부되는 경우는 사용자가 선택하여 고정이 가능케 하였다.

2. 후처리의 일반적 방법

문맥 지식을 이용한 후처리 기법은 문맥을 나타내는 방법에 의하여 통계적 방법과 구조적 방법으로 나뉠 수 있다. 문맥의 통계적 표현 방법은 문장을 단어 생성처리 모델로 가정하여 인식 하는 방법으로 viterbi 방법과 그것의 수정된 방법이 있다. [2,3] 문맥의 구조적 표현 방법은 인공지능 형태에 접근한 방법으로 문맥의 결정론적 표현에 기초하고 있다. 이 방법은 lexicon 이나 구문론(syntax) 또는 의미론(semantics)의 내부적, 외부적 표현으로 확장시킬 수 있다. binary n-gram 방법은 이진 배열의 형태로 lexicon 의 문맥을 나타낸 것이다.

2.1 VITERBI 방법

Viterbi 방법은 입력되는 글자들의 순서에 의해 전에 입력된 문자와 다음에 입력될 문자 사이의 확률을 조사하여 틀린글자를 수정하는 방법이다.

따라서 이방법은 bottom-up 방법이며 문장을 단어 생성처리 모델로 가정하여 사용하며 일반적으로 영어 문장을 Markovsource로 가정한다.[7,8] 이 방법은 베이지만 결정이론에 근거하여 확률의 최대치를 구하는 방법이므로 사전이 필요하지 않고 재귀적 알고리즘으로 유도되는 동적 프로그래밍 공식에 근거한다.[9] 최대치를 구하는 것은 그래프에서 최단경로를 구하는 문제와 유사하다.[10]

순수한 viterbi 방법의 단점을 보완한 수정된 viterbi 방법으로 MVA(Modified Viterbi Algorithm)가 있다. [2]

2.2 BINARY N-GRAM 방법

Binar n-gram 알고리즘은 단어 안에서 글자들의 위치에 대한 관계성을 고려하여 틀린 글자를 탐지하고 교정하는 방법이다[5]. 글자들의 위치에 대한 정보는 단어의 길이에 따라 lexicon에 저장된다. 이런 형태의 정보는 gram 이라고 불리우며 표현은 각각이 2G (영어의 경우) 의 n 승의 원소로 구성된 이진 배열의 집합이 쓰인다. n 이 2 인 경우를 digram 이라고 하고, 3인 경우를 trigram 이라고 한다.

m 이 단어의 길이 일때 Dm 은 단어의 길이가 m 으로 구성된 sublexicon 이다.[4,6]

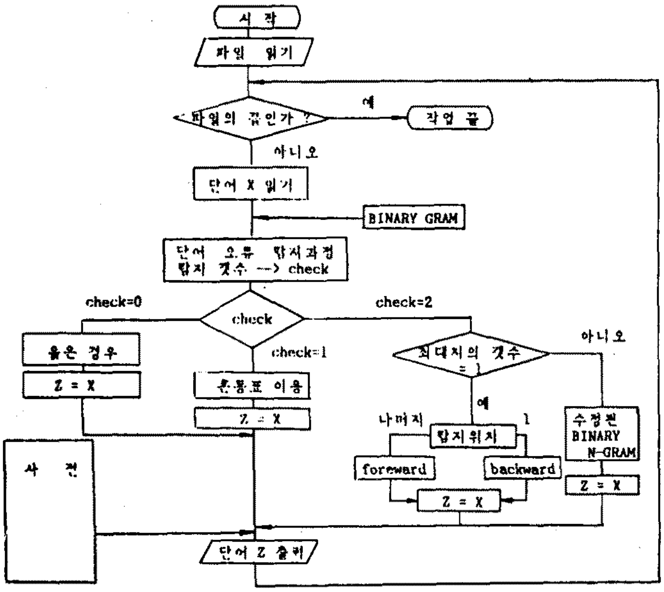
Gram 을 만들때 단어길이 전체의 위치를 고려해서 만드는 방법과 정해진 길이 m 만큼씩 구분하여 만드는 방법이 있는데 전자를 위치적 binary n-gram이라고 하고, 후자를 비 위치적 binary n-gram 이라고 한다.

2.3 혼합 방법

위의 2.1 과 2.2 를 가지고 인식하는 방법으로 첫번째 과정의 지식 정보를 통합하여 검색한 다음 두번째 과정에서 순차적으로 bottom-up과 top-down 과정으로 분기(Cascading) 시키는 방법이다.[10]

3. 후처리 알고리즘

제안된 알고리즘에서 binary digram 을 중심으로 단어의 첫번째 문자가 오류로 탐지되었을 때의 처리와 여러개의 문자가 잘못으로 탐지되었을때의 처리를 선택적으로 gram 과 비교하게 설계 하였으며 사용자와의 상호 작용에 의하여 오류가 고정될 수 있는알고리즘을 제안한다. 제안된 알고리즘의 개요도는 (그림 1)에서 설명되어 진다.



(그림 1) 제안된 알고리즘의 개요도

(그림 1) 에서 X 는 input 된 단어이고, Z 는 후처리된 결과이다. 입력은 file 단위로 하고 "단어 X 읽기" 과정에서 하나 이상의 blank 로 나뉘어져 있는 것은 다른 단어로 가정하여 하나의 단어씩 읽어낸다. 동시에 단어 길이를 조사하여 상응되는 "BINARY GRAM" 과 비교 하고 "오류 탐지 과정" 에서 오류가 발생한 문자의 위치와 갯수를 조사하여 그 갯수를 check 에 기록한다.

check 가 0 인 경우는 오류발생이 없는 경우이므로 output word Z 에 input word X 를 그대로 넣는다.

check 가 1 인 경우는 오류탐지가 되지않아 고정이 불가능한 경우로 이 경우는 혼동표를 이용하여 오인식된 문자를 고정하여 output word Z 에 넣는다.

check가 2인 경우는 최대치의 갯수가 1개인 경우와 여러개인 경우가 있는데 1개인 경우도 탐지위치에 따라 두가지 경우가 발생한다. 탐지위치가 단어의 첫번째가 아니면 foreward gram을 이용하여 고정하고, 첫번째 아니면 backword gram 을 이용하여 고정한다. 그리고 최대치의 갯수가 여러개일 경우 각각에 대해 혼동표를 이용하여 고정한다.[15]

마지막으로 더 정확한 결과를 얻기위하여는 사전을 검색한 다음 결과를 산출할 수 있다.

4. 실험 및 검토

본 연구에서는 binarg digram 방법을 위주로 혼동표와 backward gram 을 이용한 수정된 알고리즘과 본래의 알고리즘과의 인식률 비교, 기억장소와 시간에 대하여 비교 검토 하였다.

사용한 컴퓨터는 주기억 장치가 640KB 인 IBM-PC/XT 호환 기종을 사용하였고, 사용 언어는 Turbo Pascal 을 사용 하였다.

입력되는 파일은 인식과정에서 인식하기 어려운 글자와 다른 논문에서 제시된 혼동빈도에 의하여 오인식된 파일을 가정하여 입력 파일로 사용하였다.[11,12,13] 이 실험에서는 2 개의입력 파일을 가정 하였는데 (그림 2)는 File-1으로 파스칼 언어의 예약어로 구성된 파일이고 (그림 3)은 컴퓨터비젼에 관한 논문중 일부를 발췌하였다.

```
   amd             arroy
  begln              oase              co
nst
                             div
  repedt     do  dovnto
                        set
  elso        them      ena
        flle       fur         fon
ction                   type   to

        goto            if
 im   labol     mod  nll  of   oi
pached                        proc
edvre   progran   untll
         war       whlle       with
```

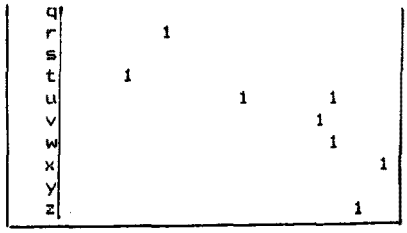
(그림2) 파일 - 1 의 내용

```
    an objectiue computer vision res
 earoh is to provde a nachine with the
 a bllity to anaiyze image in drder to
    their content



            vislin systom
```

(그림 3) 파일 - 2 의 내용

(표 1)은 혼동표로 논문 [11,12,13]에서 제시된 혼동 빈도에 의하여 만들어 졌다.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a				1											1											
b																										
c															1											
d	1																									
e			1																							
f																				1						
g										1																
h											1															
i											1	1						1								
j							1																			
k								1																		
l									1																	
m														1												
n													1		1											
o			1											1												
p																										



(표 1) 혼동표

위에서 1 로 표시된 것은 혼동될 가능성이 있음을 표시한 것이고 그 이외는 혼동될 가능성이 별로 없음을 나타낸다. 혼동표는 오류의 위치가 탐지되지 않거나 1 개이상 최대 가치가 결정될때 사용자가 선택할 수 있게 해 준다.

```
* * * From 1 to 2 relation * * *
  Bf [ 1 : (a,*)] = (00000000000000100010000000000)
  Bf [ 1 : (b,*)] = (00001000000000000000000000000)
  Bf [ 1 : (c,*)] = (10000000000000010000000000000)
  Bf [ 1 : (d,*)] = (00000000100000010000000000000)
  Bf [ 1 : (e,*)] = (00000000000010100000000000000)
  Bf [ 1 : (f,*)] = (00000000100000010000000000000)
  Bf [ 1 : (g,*)] = (00000000000000010000000000000)
  Bf [ 1 : (h,*)] = (00000000000000000000000000000)
  Bf [ 1 : (i,*)] = (00000100000001000000000000000)
  Bf [ 1 : (j,*)] = (00000000000000000000000000000)
```

(표 2) gram 의 일부분

```
* * * From 4 to 1 relation * * *
  Bb [ 4 : (*,a)] = (10000000000000000000000000000)
  Bb [ 4 : (*,b)] = (00000000000000000000000000000)
  Bb [ 4 : (*,c)] = (00000100000000001000000000000)
  Bb [ 4 : (*,d)] = (00000000000000000000000000000)
  Bb [ 4 : (*,e)] = (00101100000010000001010000000)
  Bb [ 4 : (*,f)] = (00000000000000000000000000000)
  Bb [ 4 : (*,g)] = (00000000000000001000000000000)
  Bb [ 4 : (*,h)] = (00000000000000000000000010000)
  Bb [ 4 : (*,i)] = (01000000000000000000001000000)
  Bb [ 4 : (*,j)] = (00000000000000000CC0000000000)
```

(표 3) backword gram 의 일부분

(표 4) 는 program 결과의 일부분이다.

```
* input word =   oase : 4

      X = oase

          ====== D  E  T  B  C  T =====

                INDEX 1 : 1001
                INDEX 2 : 1010

          ===== C O R R E C T ======

                Error Position : 1

                * * Need Backward Gram * *
                abcdefghijklmnopqrstuvwxyz
                (00302100000200010101010000)

                Correct character : C

          ====== Processing T I M E ======

               1.65 secs

     Z = case

* output word =   case
```

(표 4) Program 의 결과 형태

(표 2)는 gram 의 일부분으로 첫번째 문자와, 두 번째 문자의 관계에서 첫 번째에 a 가 나오고 두 번째에 n 이나올수 있다는 것을 나타낸다.

(표 3) 은 단어의 첫 번째 위치의 오류을 고정하는데 사용하는 Backword gram 으로 뒷 문자에서 앞 문자와의 관계를 나타낸다.이것은 gram 에서 첨자만 바꿔 비교하므로 별도의 기억 장소가 필요치 않고 변수만 놀려주면 된다.

실험의 결과는 (1) gram 만 이용했을 때, (2) gram 과 backward gram 을 같이 이용했을 때 (3) 돌다의 사용자의 상호고류에 의한 결과를 비교하였다.

(표 5) 는 File - 1 에 대한 결과표이다.

FILE - 1 O:correct X:fult #:interactive

입력된단어	기존의 Binary bigram			제안된 Binary bigram		
	인식결과		시간	인식결과		시간
arroy	array	0	1.54	array	0	1.54
amd	and	0	1.54	and	0	1.54
begin	begin	0	1.54	begin	0	1.54
oase	aase	X	1.54	case	0	1.54
.	.			.		
.	.			.		
.	.			.		
oi	oi	X	3.13	or	#	14.61
pached	packed	0	6.48	packe d	0	6.48
procedvre	procedure	0	7.64	procedure	0	7.64
progrom	program	0	5.77	program	0	5.77
until	until	0	6.54	until	0	6.54
var	var	0	0.72	var	0	0.72
while	while	0	1.54	while	0	1.54
wlth	whth	X	1.54	with	0	14.84
인 식 률	((33-6) / 33) X 100 = 81.8 %			상호고류 불허 84.8 % 상호고류 허용 100 %		
처리 시간	평 균 3.999 초			평 균 5.243 초		

(표 5) file - 1 의 후처리 결과

(1)번의 인식률은 81.8% 였으나 (2)번은 84.8 %로 향상되었고 (3)번은 오인식된 단어는 출력시키지 않았다. File - 2 에서는 (1)번의 인식률 85% (2) 번은 90% 였고 마찬가지로 (3) 번에서는 거부되는 경우를 사용자와의 상호고류에 의해 재거시킬 수 있었다.

시간은 (1)번의 경우 File - 1 에서 3.999 초 File - 2 dptj 3.848 초 였으나 (3) 번카지 허용할 경우 File - 1 dptj 5.243 초 Flie - 2 에서 5.115 초가 소요됐다. 그렇지만 기억장소는 추가 할 필요가 없었다.

5. 결 론

본 논문은 backwarkd gram 의 개념을 이용하여 단어의 첫번째 문자에서 오류가 발생할 때 내연의 철자를 조절하여 비교 했기 때문에 기억장소의 추가없이 단어의 오류된 글자를 바르게 인식할 수 있었고, 생존백터의 최대치가 여러개 있을 경우는 사용자와의 상호 고류에 의해 최대치 증 알고리즘이 선정한 문자들 중에서 옳은 문자를 사용자가 선택할 수 있게 해 주었다. 또한 오류의 위치가 맏지 되지 않을 경우에는 혼동표를 이용하여 고정하였기 때문에 거부되는 경우를 제거하였다. 인식률은 backward gram을 이용하여 gram 만을 이용했을 때보다 3 ~ 5 % 증가 되었다.

후처리의 이용분야는 컴파일러의 자동구문 오류 탐지 루우틴 우편 번호의 봉투에 기입된 주소가 다를 경우의 고정[14] 및 워드프로세서의 철자 고정 시스템등으로 이용이 예상된다.

한글에서의 후처리도 순수한 dictionary look-up 방법만 쓰는 것보다 binarg n-gram 이나 viterbi 방법을 함께 쓰면 효율적인 후처리를 수행할 수 있을 것이다.

참 고 문 헌

[1] A .J. Viterbi.
"Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", IEEE Trans. on Infromation Theory, Vol. IT-13. No.2. April 1976. pp.260-269.

[2] W. Doster & J. Schurmann.
"An Application on the modified viterbi-algorithm used in text recognition", 5th international conf. of Pattern Recognition peoceding MAIMI BEACH. FLORIDA. Dec. 1980. pp.853-855.

[3] R. Shinghal & G. T. Toussaint.
"Experiments in Text Recognition with the Modified Viterbi Algorithm", IEEE Trans. on PAMI. Vol.PAMI-1. No.2. Apri 1979. pp.184-193.

[4] E.M. Riseman & R.W.Ehrich.
"Contextual Word Recognition Using Binary Digrams", IEEE Trans. on Computer. Vol. C-20. No 4. April 1971. pp.397-403.

[5] J.R. Ullmann.
"A binary n-gram technique for automatic correction and reversal errors in words", The Computer Journal. Vol. 20. No.2. pp. 141-147

[6] E.M. Riseman & A. R. Hanson.
"A contextual Postprocessing System for Error Correction Using Binary n-Gram", IEEE Trans. on Conputer.Vol. C-23. No.5. May 1974. pp.480-493.

[7] D. L. Neuhoff.
"The Viterbi Algorithm As an Aid in Text Recognition", IEEE Trans. on Information Theory. March 1975. pp.222-226.

[8] H. Tanaka. Y. S. Kaneku.
"Recognition of Distorted Pattern Using the Viterbi Algorithm", IEEE Trans. on PAMI. Vol.PAMI-4. No.1. January 1982. pp. 18-25.

[9] R. L. Kashyap & B. J. Oommen.
"An Effective Algorithm for String Correction Using Generalized Edit Distances - I. Description of the Algirithm and Its Optimality", Information Science. Vol.23. pp. 23-142

[10] J. J. Hull. S.N. Srihari & R. Choudhari.
"An Integrated Algorithm for Text Recognition: Comparison with a Cascaded Algorithm", IEEE Trans. on PAMI. Vol.PAMI-5. No.4. Jury 1983. pp.384-395.

[11] J. Schurmann.
"A Multifont Word Recognition System for Postal Address Reading", IEEE Transaction On Computer. Vol. C-27. No.8. August 1978. pp.721-732.

[12] D. J. Burr.
"Designing a Handwriting Reader", IEEE trans. on PAMI. Vol.PAMI-5. NO.5. September 1983. pp. 554-559.

[13] W. S. Rosenbaum & J. J. Hilliard.
"Multifont OCR Postprocessing System", IBM J. Res. Develop. 1975. pp. 398-421.

[14] W. Doster.
"Contextual Postprocessing System for Cooperation with a Multiple - Choice Character - Recognition System", IEEE Trans. on Computer. Vol. C-26. No.11. November 1977. pp. 1090-1101.

[15] 김일희.
"자동문서 판독을 위한 후처리 연구", 숭전대학교 대학원 석사학위 청구논문. 1987.