국어 철자검색 프로그램 키재기

노 용균, 박 동인
시스템공학연구소

A Benchmark Test for Korean Spelling-Checking Programs

Yongkyoon No and Dong-In Park
Systems Engineering Research Institute

요약

국어 철자 검색 프로그램 세 개의 검색 능력을 비교하였다. 오류가 없는 파일, 타자시의 전형적인 오류를 포함하는 파일(자소별 오류율 1%), 그리고 광학적 문자인식 프로그램의 전형적인 오류를 포함하는 파일(자소별 오류율 2.7~2.9%) 등에 대하여 한글과 컴퓨터, 한국 마이크로소프트, 핸디 소프트의 워드프로세서에 도구로 포함된 철자검색 프로그램을 수행하였다. 이 세 프로그램 중에서 한글과 컴퓨터의 제품은 정방향 오판율과 오류율 낮은 파일에 대한 역방향 오판율이 낮았고 핸디 소프트의 제품은 오류율이 높은 파일에 대한 역방향 오판율이 낮았다. 세 프로그램 모두 역방향 오판율이 자소별 오류율의 10배 이상이라는 점에 있어서 심각한 문제를 안고 있는 것으로 판단된다.

1. 서론

자연 언어 처리에 있어서의 초보적인 기술로 철자 검색 및 교정 기능을 들 수 있다. 디지털 방식으로 표시된 글월을 받아서 오자를 포함하는 어절을 자동으로 탐지해내고 그럴싸한 대체후보 어절들을 제시해 주는 기능이다. 영어 철자검색 연구는 1957년경에 시작되었고 첫 응용 프로그램은 1971년에 DEC-10을 위해 Ralph Gorin이 쓴 SPELL이었다. Peterson(1980:677). 국어 철자검색 프로그램의 비조는 한글과 컴퓨터의 1990년 제품이다.

철자 검색은 어절간의 관계까지 고려하는 것과 그렇지 않은 것으로 크게 나눌 수 있는데, 국어 철자 검색용 프로그램으로서 문맥에 민감한 것은 아직 배포되지 않고 있다. 이 연구는 널리 사용되고 있는 국어 철자 검색 프로그램들의 검색 능력을 비교하고 개선 방안을 제시하는 것을 그 목적으로 한다.

2. 시험의 구성

2.1 시험대에 오른 프로그램들

이 키재기에 등장하는 프로그램들은 다음 세 가지이다.

(1) ㄱ. 한글 2.5 (주식회사 한글과 컴퓨터)
ㄴ. 한글워드 5.0 (주식회사 한국 마이크로소프트)
ㄷ. 아리랑 2.0 (주식회사 핸디 컴퓨터)

이 가운데에서 (1ㄱ)은 DOS를 전제하는 프로그램이며 (1ㄴ)과 (1ㄷ)은 MS Windows를 전제하는 프로그램이다.

2.2 시험에 쓰인 글월들

말이 그렇듯이, 글월들도 그 작자의 입지나 작자와 독자와의 관계, 작자의 언어적 취향 등에 따라서 다양함을 드러낸다. 낱말의 선택, 문장 구조의 선택, 그리고 담화의 구성 등이 글에 따라서 많이 다를 수 있다. 예를 들어, 국어의 어미 /으니까/는 평이한 문체에서 쓰일 수 있으나 전형적인 문어체 문장에서는 /으니/가 쓰이고 /으니까/는 안쓰인다. 이유를 나타내는 /으므로/도 마찬가지로 전형적인 구어체 문장에서는 회피되고 동사의 기-골과 명사 때문과 후치사 에에 의해 같은 의미가 표현된다.

이런 이유로 이 시험에서는 판이한 두 장르의 글월을 쓴다. 하나는 호 영송의 단편소설들이고 다른 하나는 황 수영의 단행본 "불국사와 석굴암" 중 제4장에서 가져 온 글들이다. 이 둘은 조합형 한글로 표시되었을 때 각각 약 17킬로바이트 분량으로 12포인트 글자를 써서 인쇄했을 때 A4용지 다섯 장에 조금 못미치는 분량이다. 단편소설은 전형적인 구어체 문장들을, 중수필인 "불국사와 석굴암"은 전문용어와 전형적인 문어체 문장들을 많이 포함하는 것으로 보았다.

2.3 시험대상 글월에 들어있는 오류의 종류와 정도

위의 시험대상 글월을 담은 파일들은 세 종류로 중복 제작되었다. 첫번째로는 미리 치밀하게 교정을 본, 오류가 없는 파일이 만들어졌다. 첫번째 필자가 여러 차례 교정을 보았다. 이 무오류 파일을 근거로 두번째 파일이 만들어졌는데, 이 파일은 타자시의 오류를 반영하도록 인위적으로 만들어졌다.

Damerau(1964)와 Peterson(1980:682)에 따르면 철자오류의 80%가 아래 요인들로 인한 것이다.

(2) ㄱ. 자소 두개의 전위

ㄴ. 자소 하나 추가
ㄷ. 자소 하나 사라짐
ㄹ. 자소 하나 다름

(두 바이트 조합형 파일인 무오류 파일을 로마자로 변환시킨 뒤에) UNIX 도구인 AWK를 써서 글쇠판에서 인접한 글쇠들 사이의 대치, 전위, 첨가, 그리고 삭제 등의 타자오류를 인공적으로 집어넣었는데, 이것은 (2)의 오류패턴들을 다 반영하고 오류 자소의 수를 쉽게 알아내기 위해서였다.

세번째 파일은 무오류 파일을 300dpi급 레이저 프린터에서 인쇄하여 스캐너로 읽어들인 후에 광학적 문자인식 프로그램인 글눈이라는 프로그램(한국인식기술 제작)으로 문자부호가 복원된 파일이다. 즉, 문자인식 프로그램이 잡음 제공의 수단으로 쓰인 것이다.

자음 표시체와 모음 표시체를 각각 한 단위로 취급해서 이 파일들의 오류 포함 정도를 계산하면, 세밀한 육안 교정을 거친 무오류 파일이 0퍼센트, 전형적인 타자시의 오류를 포함하는 파일이 1.0퍼센트, 그리고 문자인식 프로그램의 특징적인 오류를 포함하는 파일이 2.7~2.9퍼센트 정도이다. 오류 포함 정도가 n퍼센트라는 것은 자모 100개 중에서 n개가 다른 자모로 대치, 첨가, 또는 삭제되었다는 것을 의미한다.

이름이 a로 시작하는 파일은 호 영송의 단편 소설집에서 취한 글이고 b로 시작하는 파일은 황 영수의 저서에서 취한 글이다. 이름에 1을 갖는 파일은 오류가 없는 것들이고 2를 갖는 파일은 타자 오류를 포함하는 것들이다. 3을 파일 이름에 갖는 것들은 문자인식 프로그램의 오류를 포함한다.

자소별 오류율을 산출함에 있어서, 타자시의 오류를 포함하는 파일들의 경우에는 공란자 오류까지 산입했지만, a3와 b3, 즉 문자인식 프로그램이 만들어낸 파일들의 경우에는 공란자 오류를 최대한 배려했다. 공란자들의 연속체를 공란자 하나로 대체하여서 오류율을 산출했다.

표 1. 파일별 오류율

파일	자소별 오류율	총 어절 수	오류 어절의 수	무오류 어절의 수	어절별 오류율
a1	0	2,053	0	2,053	0
b1	0	2,066	0	2,066	0
a2	0.0100	2,017	128	1,889	0.0634
b2	0.0103	2,025	132	1,893	0.0651
a3	0.0290	2,199	356	1,843	0.1618
b3	0.0269	2,228	380	1,848	0.1705

자소별 정확도와 어절별 정확도는 어떤 관계에 놓여 있는가? 한 어절이 평균 7개의 자소로 이루어진다고 보고 오류 자소의 분포가 균등하다고 보면, 어절별 정확도 n은 자소별 정확도 m과 아래 등식으로 연관된다.

(3) $n = 6*m - 5$

오류 자소들이 몰려서 분포되어 있을 경우에는 어절별 정확도가 (3)의 그것보다 높을 것이다. 이 등식으로 a3와 b3의 어절별 오류율을 계산하면 각각 0.174와 0.162로서 표1의 끝 열에 제시된 실제 오류율과 아주 가깝다.

3. 시험 결과

아래에 제시된 표2와 표3이 이 시험의 결과를 보여 준다. 각 글월은 어절들(token strings)의 연쇄체인데 어떤 어절은 글월의 여러 위치에 나타나지만 시험대상 프로그램들은 문맥을 고려하지 않기 때문에 동일한 문자연쇄체에 대해서는 (옳든 그르든) 동일한 판단을 반복한다. 따라서 오판 어절의 수를 비교할 때는 개체(token) 뿐만 아니라 유형(type)도 고려되어야 한다. 글월이 특히 반복도가 높을 때에는 유형에 따른 비교가 더욱 중요하다.

각각의 표에서 '정방향 오판'이라고 부른 것들은 옳은 어절들에 대해서 오류가 있다고 잘못 지적하는 경우들을 뜻하고, '역방향 오판'이라고 부른 것들은 오류가 있는데도 멀쩡하다고 판단하는 경우들을 뜻한다. 정방향 오판은 Peterson(1980:677)이 "Type 1" errors라고 부른 것과 일치하고, 역방향 오판은 "Type 2" errors라고 명명한 것과 같다.

표 2. 오판 어절의 수 (개체)

파일	총어절수	정방향 오판				역방향 오판			
		hc	ms	ar	공통	hc	ms	ar	공통
a1	2,053	121	140	133	79	0	0	0	0
b1	2,066	125	139	147	64	0	0	0	0
a2	2,017	117	135	126	77	12	32	39	6
b2	2,025	113	128	127	60	13	30	34	4
a3	2,199	91	109	107	67	182	171	139	112
b3	2,225	92	100	112	42	186	189	172	140

a2와 b2가 오류 없는 원본 a1 및 b1에 비해서 적은 수의 어절을 갖는 이유는 원본에 있는 공란자들 가운데 몇 개가 삭제되어서 이 파일들이 만들어졌기 때문이다. 공란자를 빠뜨리는 것은 타자 오류인 삭제의 한 경우로서 특히 흔한 것으로

보인다. a3와 b3가 원본보다 더 많은 어절을 갖는 이유는 원본에 없는 공란자들이 추가되었기 때문인데, 이것은 주로 문자인식 프로그램이 한 줄의 끝에 있는 글자를 인식한 다음에 그 다음 줄의 첫 자를 인식하기 이전에 (여러 개의) 공란자를 무조건 추가하기 때문이다.

표 3. 오판 어절의 수 (유형)

파일	총어절수	정방향 오판				역방향 오판			
		hc	ms	ar	공통	hc	ms	ar	공통
a1	1,425	89	107	108	56	0	0	0	0
b1	1,366	102	120	123	53	0	0	0	0
a2	1,435	86	103	102	55	12	32	39	6
b2	1,401	95	113	107	46	13	30	34	4
a3	1,556	66	86	84	44	175	164	134	107
b3	1,546	85	93	105	37	172	174	159	128

4. 결론 -- 이 시험이 시사하는 것들

4.1 정방향 오판율의 불변성

옳은 어절을 잘못되었다고 판단하는 사례의 수는 전절의 표1과 표2에서 보듯이 모든 옳은 어절의 수에 비례한다. 한 파일 안에 있는 옳은 어절의 수로 철자검색 프로그램이 오류어절로 오판한 어절의 수를 나누면 0.05~0.07을 얻는다. 시험대상이 된 세 프로그램이 모두 이 좁은 범위 안의 정방향 오판율을 보인다. (한글과 컴퓨터의 제품이 아리랑이나 MS 워드보다 지속적으로 낮은 정방향 오판율을 보인다.)

정방향 오판은 대체로 고유명사와 띄어쓰기를 인색하게 한 어절들에 기인하지만, 표2의 여섯번째 열에서 보듯이 세 프로그램이 공통되게 오판한 어절들의 수가 각 프로그램 특유의 오판어절 수와 거의 비슷하다는 점을 감안하면, 이 부분에서 개선의 여지가 있다고 보아야 한다.

4.2 자소별 오류율 증가에 따른 역방향 오판의 뚜렷한 증가

이 시험을 통해 이루어진 가장 중요한 발견은 역방향 오판에 관한 것으로 아래와 같이 제시할 수 있다.

(3) 역방향 오판 어절의 수는 프로그램 적용 대상 파일의 자소별 정확도가 떨어짐에 따라서 엄청난 속도로 증가한다.

옳지 않은 어절을 옳은 것으로 잘못 판단하는 사례의 수는 해당 파일 안에 있는 오류어절의 총계에 비례한다. 그런데 역방향 오판율은 최저 0.09(파일 a2, 한글과 컴퓨터)부터 최고 0.58(파일 b3, 한국마이크로소프트)까지 큰 폭의 변동을 보인다는 점에 있어서 정방향 오판율의 극히 좁은 변동폭과 뚜렷한 대조를 보인다.

역방향 오판율은 검색 대상 파일의 자소별 오류율이 높을수록 높다. 특히 한글과 컴퓨터의 프로그램의 오판율은 파일의 자소별 오류율이 2.7~3.0배로 높아짐에 따라 5.8배 정도 높아진다.

4.3 역방향 오판을 줄이는 방법 1

표 3에 따르면 아리랑의 철자 검색 프로그램이 역방향 오판을 가장 적게 내놓는다. 그 이유는 띄어서 쓰지 않아야 할 곳을 띄어서 쓴 경우들의 다수를 적발해 내는 능력을 이 프로그램만 갖고 있기 때문이다. 띄어쓰기 과다의 오류는 타자 시에는 범해지지 않고 광학적 문자인식 프로그램이 한 줄의 마지막 글자 다음에 무조건 공란자를 삽입함으로써 발생한다. 이렇게 들어가는 공란자들 가운데에서 얼마만큼이 안들어가야 할 공란자인지는 문자인식 프로그램의 입력 문서의 한 줄이 얼마나 긴지에 달려 있는데, 보통 문서일 경우에 20~50% 정도가 잘못 들어가는 공란자다.

이렇게 잘못 삽입되어 있는 공란자의 문제를 해결하기 위해서는 철자 검색기가 어떤 어절을 오류 있는 어절로 판단할 경우에 그런 판단에 따른 제시 절차로 바로 들어가지 않고, 그 다음 어절을 현재의 피검어절에 접합하여서 새로운 피검어절을 만들고 그것을 검사하는 것을 들 수 있다. (이 새로운 피검어절이 철자 오류가 없는 것으로 판명되면 공란자를 없애라고 사용자에게 추천할 수 있다.) 이 것을 C언어로 표현하면 아래와 같다.

```
(4)     if (acceptable(word1));
        else
        {
                strcpy(potential_word, word1);
                if (acceptable(strcat(potential_word, word2)))
                {
                        suppress_space(word1, word2);
                }
                else
                {
                        process_error(word1);
                }
        }
```

```
strcpy(word1, word2);
strtok(NULL, word2);
```

4.4 역방향 오판을 줄이는 방법 2

기본적으로 어떤 코딩 스킴에 바탕을 두고 철자 검색을 하느냐 하는 것도 중요한 것으로 드러났다. 역방향 오판어절의 수가 오류율이 낮은 파일, a2와 b2에 있어서는 한글과 컴퓨터의 경우에 가장 적은데, 이것은 세 프로그램 가운데에서 한글과 컴퓨터의 제품만이 조합형에 바탕을 두고 검색하기 때문이다. 그 이유는, 완성형에서 지원하지 않는 자소의 결합체들이 완성형으로 변환될 때에는 중화되어서 완성형에서 지원하는 글자로 바뀌어 버리기 때문이다.

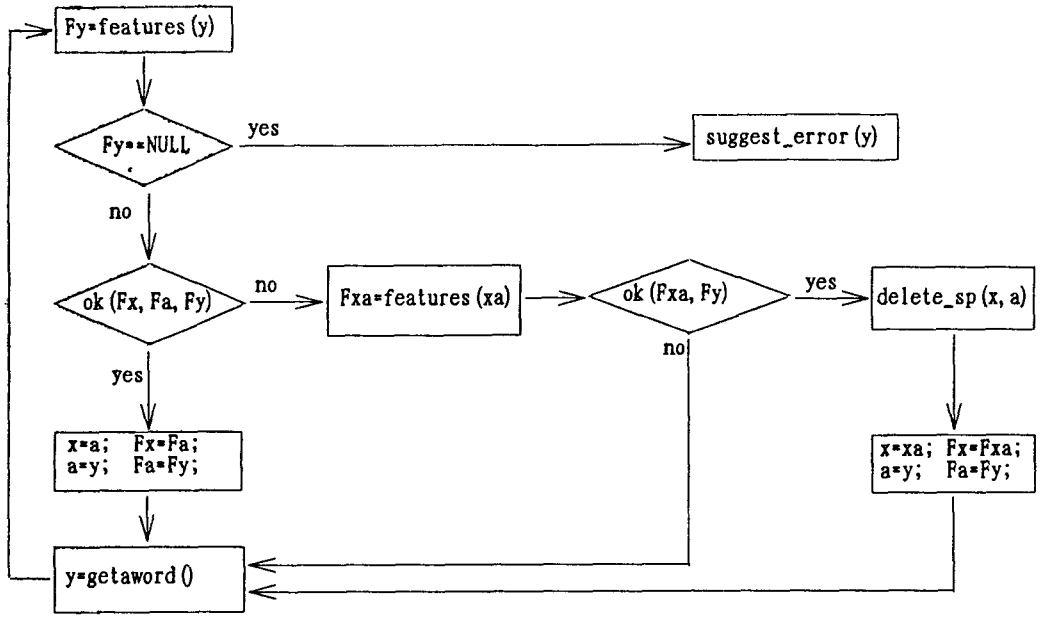
예를 들어, 조합형에서 16진수 94 41은 실제 쓰이지 않는 글자 연쇄체인 'ㄷ 채움 채움'인데, 이것이 완성형에서는 어떤 영문인지 'ㄷ ㅏ 채움'으로 바뀌어 버리는 것이다. 조합형에서 다자는 94 61이라는 값을 가지니까, 결국 94 61뿐 아니라 94 41도 완성형의 다자에 대응한다. 마찬가지로 조합형의 88 61, 88 41, 81 A1등이 모두 완성형의 가자에 대응하고 조합형의 9C 41과 9B B7이 완성형의 띵자에, 조합형의 B8 42와 B7 BC가 완성형의 잎자에 대응한다. 따라서 조합형을 바탕으로 검색하는 한글과 컴퓨터의 제품은 성공을 ㄷ지 못했다의 두번째 어절을 오류어절로 옳게 탐지해 내지만 한국마이크로소프트와 핸디소프트의 제품들은 성공을 다지 못했다로 바뀐 어절들을 검색하므로 이 어절을 오류가 없는 어절로 오판하고 만다. 이런 오판의 사례가 상당히 많은 것으로 보인다. 이것은 조합형 코드를 완성형 코드로 변환하는 부문에서 두 코딩 스킴의 기본 철학이 다름으로써 일대일 대응이 안되는 데에 기인한다.

4.5 역방향 오판을 줄이는 방법 3

앞의 4.2에서 보았듯이 역방향 오판율이 자소별 오류율의 9~30배나 되는데, 범용성 있는 프로그램이라면 역방향 오판율이 자소별 오류율과 크게 다르지 않아야 할 것이다. 역방향 오판은 오류가 있는 어절이 우연하게도 다른 맥락에서는 오류가 없는 어절로 해석될 수 있기 때문에 범해지는데, 이것은 철자검색 프로그램이 독립된 어절을 대상으로 검색하는 한, 즉 그 어절이 들어있는 문장 전체에 대한 고려를 전혀 하지 않는 한 불가피한 것이다.

문장 전체를 범위로 검색하는 데에는 막대한 양의 계산이 필요하고 대상언어인 국어의 문장구조에 대한 정교한 기술이 전제된다. 현재의 국어처리 기술로는 수년 안에 역방향 오판율이 자소별 오류율보다 낮게 할 수가 없는 것으로 보인다. 현실적인 대안으로는 인접한 어절들을 해석해서 형태통사적 특질들을 얻어내고 그 특질들이 그렇게 연속될 수 있는지를 검토하는 것을 들 수 있다. 삽입되지 않아야 할 공란자가 삽입된 것을 탐지해내는 과정을 (5)에 제시한다.

(5) 문법특질을 이용한 공란자 오류 탐지



이것을 평이한 말로 나타내면 아래와 같다.

(6) ㄱ. 새 어절(y)을 해석한다. y가 가능한 어절이고 Fy라는 형태통사적 특질을 갖는다.
ㄴ. 직전 어절 a의 문법특질 Fa, 그 전 어절 x의 문법특질 Fx를 가져와서 Fx--Fa--Fy를 얻는다.
ㄷ. Fx--Fa--Fy가 가능한 연쇄체인지를 검토한다. 가능한 연쇄체가 아니면 ㄹ로 간다. 가능한 연쇄체면 ㅇ으로 건너 뛴다.
ㄹ. x와 a를 접합해서 xa를 얻고 이것을 해석해서 문법특질 Fxa를 얻는다.
ㅁ. Fxa--Fy가 가능한 연쇄체인지를 검토한다. 가능한 연쇄체가 아니면 ㅂ으로 가고 가능한 연쇄체면 ㅅ으로 건너 뛴다.
ㅂ. 어절 연쇄체 x--a--y에 대해서 오류가 있음을 알린다. ㅇ으로 간다.
ㅅ. 어절 x와 어절 a 사이의 공란자를 삭제한다.
ㅇ. 포인터를 어절 y 바로 뒤의 어절로 옮긴다. (5)로 돌아간다.

4.3과 4.4에서 기술한 방법으로 줄일 수 있는 역방향 오판의 수는 전체 사례 수에 비하면 상당히 적은 수다. 핸디소프트의 아리랑이 제공하는 철자 검색 프로그램이 범하는 오판 사례의 수보다 현저히 적은 오판을 범하는 프로그램이라면 (5)에 제시된 알고리듬을 채택해야 할 것이다.

5. 결론

시험 대상이 된 세 프로그램은 검색 능력에 있어서 큰 차이를 보이지 않았다. 굳이 순위를 매긴다면 자소별 오류율이 낮은 파일들을 검색 대상으로 했을 때 정방향 오판율과 역방향 오판율이 낮은 한글과 컴퓨터의 제품과 자소별 오류율이 높은(0.0269~0.0290) 파일들을 검색 대상으로 했을 때 역방향 오판율이 낮은 핸디소프트의 아리랑에 내장된 제품이 승리자라 하겠다.

세 프로그램의 역방향 오판율은 파일의 자소별 오류율의 10~17배에 이른다는 점에 있어서 매우 심각한 문제로 부각되었다. 차세대에 요청되는 철자 검색 프로그램은 역방향 오판율이 파일의 자소별 오류율과 비슷한 정도에 머물게 할 수 있어야 한다. 역방향 오판을 줄이기 위해서는 우선 조합형 코드를 바탕으로 검색작업이 이루어져야 하고, 띄어쓰기 과다의 오류를 감안하여야 한다.

그러나 궁극적인 역방향 오판율 저하는 문장 구조에 관한 정보의 도움을 받음으로써만 달성될 수 있다. 문맥까지 고려하는 철자검색은 어절들을 형태통사적 특질들의 결합으로 해석하는 일에서 시작된다.

[덧붙임 1] 파일 a1에서 세 프로그램 모두 오류가 있는 것으로 잘못 판단한 어절들

초대객들답게	파충이	후각때문에	마동출씨가	실수했읍니다
출장길에	타눔의	쩌릉	마동출씨를	수식구를
애정어린	타눔을	갤갤거리고	마동출씨는	수다장이나
알고있던	타눔은	것이잖겠는가	먹지않는	더우기
어떻구	타눔이	것같다	멋적음을	동출씨의
어쩐지에	타눔과	괴상망칙한	뭐라구	동출씨가
임동걸	털복숭이	말덴씨의	낼름	동출씨로서는
임동걸씨가	재치있는	마동출씨에	냄새같은	동출씨는
임동걸씨도	지장없을	마동출씨에게	넋놓고	동걸박사가
왕개구리같이	주착없는	마동출씨의	부어올랐을	동걸씨가
코때문에	주착바가지같은	마동출씨와	변계량씨의	뒤울렸기
표상해				

[덧붙임 2] 파일 b1에서 세 프로그램 모두 오류가 있는 것으로 잘못 판단한 어절들

층단을	워원사를	주성된	뭉성사는	석단등이
층단형	우견편단의	행병의	무설전에서	신라탑으로서는
치술령의	탱주는	회랑등이	누문등의	당탑
최근년에	장수사	감은사	백률사의	다보
유허가	장수사는	김씨	비로자나불	더우기

유규라	지권인을	괘릉을	보개	동해구였다고
이같은	조양리가	고유섭	볼때	쌍탑
일찌기	좌세와	곰잡은	붉국토와의	쌍탑을
옷주름은	중심삼아	구정리	병안하였다가	쌍탑이
응수사	주존붙을	경루의	승방등	쌍탑배치에서
응수사는	준널을	몽성사		

[덧붙임 3] 파일 a2의 일부분

곧 냄새를 맡아내고 그것을 잘 분별하여, 이를테면 이 음식은 쉬었으니 먹지않는 게 좋겠다든가, 자기의 애인에게는 이 향수가 좋겠다든가 하는 문짱 담당하는 갓이다. 향수 냄새를 잘 분별하는 등의 기능은화장품 회사 닉원 혹은 유복한 여성들에게나 필요한 기능으로 처버릴 수도 있겠지만, 그러나 아저씨의 코때문에 죽을 운명에거 살 운명으로 되넘어올 수 있었던 사람들도 있고 보면 예민한 후강 기능을 가진다는 일도 대단한 미덕이 되는 것이잖겠는가. 그것은 굳이 긴 설명이 필요치 않다. 왜냐하면 요즘도 신문에는 연탄 가스에 중독 사망하는 사람의 기사가 종종 나고있음은 분명한 사실이고, 그런 만큼, 연탄 가스의 위협을 받고 있는 사란믈을 마동츨씨가 구해냈다는 정도의 얘기는 쉬 납득이 갈 것이니까. 오히려 내가 하려는 이야기는 좀 다른 각도에서의 것이다. 연탄 가스에 중독되어 있는 사람들을 구해 준 일은 뭅론 치사를 받을 만한 일이고 내세을 만한 일이겠지만, 우리 코 아저씨 마 동츨씨기 최근에 겪은일은 오히려 그 예민한 후각때문에 고통을 치러야 했던 경우일 것이다. 마 동츱씨는 자기의 거래 회사인 A재벌의 상무이자 어느 사립 대학 경제학 교수인 임동걸씨가 베푼 만찬에 참석하게 되었다. 우리 코 아자씨는 천성이 자유 분방한 성품이니끼 점잖은 자리에도 더러 초대되는 경우가 있는 모양이다. 더우기 상대방이 거래 회사의 중역이며 대학 선배인 데야 굳이뿔참할 필요도 없었을 것이다. 그날 밤의 파티는 임 동걸씨가 모 사립 대학으로부터 명예 박사 학위를 받아 이를 축하하기로 된 모임이었다.. 이날의 주인공은 뭅론 임 등걸 박사 내외였지만 마 동츨씨는 한 단순한 축하객 이상의 역할을 수행하게 되었다. 이것은 뭅론 희한한 코를 가졌다는 까닭에서 비롯된 것이었다. 그의 코는 지금껏 내가 역설해온 것보다도훨씬 더 빛나는 그 어떤 우게0 매력을 풍기고 있어서 마 동츨씨의 코와 한 차례 상면해 본 사람이라면 그 누구라도 이 매력적인 코의 임자가 펼치는 화슬과 애교 넘치는 몸짓에 반해 버리는 것이었다. 그러나 그의 몸짓과 달뱐은 역시 육척 장신의 거구를 배경으로 묘하게 벌름거리고 또는 독자적으로 살아 움직이는 듯한 그 신비스런 코의덕으로, 조금도 경박한 수다장이나 주착바가지같은 느낌은 주지 않았으며 보다는 필생의 노력으로 한 장면 한 장면의 연기를 처리해 나가는 대배우의 그것을 보는 듯한 갑등에 사로잡히게 하는 것이다. 그 코능 때로는 궁지에 몰린 닉슨의 다섯 자나 늡어진 비극넉인 코처럼 연민을 느끼게 하다가 둡연 드골의 영웅적으로 장엄한 코를 연상케 하기도 했다. 그러나 대체로는역시 정다운 아저씨의 코라고 하는 것이 좋을 것이다. 왜냐하면 그 코는 상대방을 느긋한 마음읍 감상에 임할 수 있게 해 주는 인정미 있는 코였기 때문이다. 처음에 파티의 분위기는 다소 서먹서먹하면서 엄숙하였다. 명예 박사학위 수여자의 초대객들답게 손님들은 그들의 자랑스라운 사회적 지위와 신분에 대한 자부심으로 가득 차 있었고, 또 그들은 음 이런 유의만찬에서 점잖게 행세하는 데 익숙해 있었기 때문이었다. 그러나 우리 코 아저씨의 코가 때마침 식탁에 올라앉기 시작한 산해 진미들의 지극히 관능적이고도 도도한 냄새를 맡기 기작하자 본인의 의사와도 관계없이—우리 마 동츨씨는 실상 의식적으ㄹ 자기가 분위기를 조절해야겠다고 생각해서 코를 활용해 본 적은 없다—독자적인 기능을 발휘하기 시작했던 것이다. 이렇게 되니까 마 동츨씨는 저절로 왕성한식욕을 느끼기 시작했다

그의 거구와 그의 거대한 내장이 화려 다양한 식탁 앞에서 침을 끌떵루떡 삼키게 된 일은 조금도

[덧붙임 4] 파일 b3의 일부분

불국사 창건의 여러 가지 배경에 대하여서는 앞에서 그 지리적 여런과 시대적 성격. 그리고 이에 관여한 신라의 임금과 그 재상에 대하여 설명하여 왔다. 이제 장을 바꾸어 불국사와 식굴암을 나누어 따로 설명하고자 한다.

먼서 불국사가 자리잡은 그 위치는 경주에서 동남으로 *40리, 토함산 서남 중복의 경승의 땅을 차지하고 있다. 지금은 교통이 편리하여졌고 옛 도로도 신작로로 바뀌어 버렸으나, 경주에서 이곳까지 이르기 위하여서는 결료 가까운 거리는 아니었을 것이다. 또 호늘의 불국사역에 이르러서도 완만한 경사를 한참 동안 올라야만 하였다. 그러므로 불국사는 상당하 높은 지대에 자리잡고 있기에 먼서 그 주변의 넓은 경관이 불국사가 지니는 뛰어난 점이라고 맣할 수 있을 것이다. 불국사역이 있는 구정리 또는 괘릉을 지나는 올산 가도에서 벌써 멀리 토함신을 바라보고 그 중복에서 한층 푸른 산림 속에 신라의 대사원을 찾을 수가 있다.

그리고 이와 반대로 불국사 또는 석굴암에 오르는 길에서 내려다 본 조양 분지의 광활하고 조용한 경치는 더욱 깊은 인상을 준다. 멀리 신라의 충신 박 제상에 얽한 치술령의 준녈을 바라보며, 또 남으로 길게 뻗은 연봉을 가리킬 수가 있고, 신밀을 돌아서는 영지, 괘릉, 입실리. 조양리가 송림과 연못 사이사이에 널려 있어 아름답고 시원한 조망이라 하겠으며, .평범하고 위대한 경관'이라고 불러 온 까닭이기도 하다. 신라 사람이 이 불국사의 터를 잡는 데 비범한 안목을 발휘하였다고 할 수가 있다. 이 같은 경관의 망을 잡아 동으로는 토함신이 병풍같이 둘러 있으며, 사역 바로 뒤로는 깊은 게곡을 이루어 맑은 냇물이 흘러 경계를 이루고 있다.

이러한 토함산 중복을 깎아서 수단의 광활한 대지를 얻을 수 있었던 것은 불국사를 건립하는 데 필수의 여건이었으나, 이를 자세하게 살핀다면 가람의 중심구역이나 그 서쪽의 1단 낮은 대지, 그리고 이 같은 사역을 남단에서 2단 큰 낙차를 보이며 그곳에 식단을 쌓고 식교를 가설한 사실 등은 모두 이곳 자연의 지세를 그대로 교묘하게 이용한 것이라고 할 수가 있다. 아름다은 대자연에 올기로운 인공을 더하여 불국의 정토를 구현하려 하였던 것이다.

이 같은 불국사의 아름다운 점 중에는 또 하나의 중요한 의의가 숨어 있다.

그것은 불국사가 지니는 오국적 성격으로서. 신라의 사원이면 다소간에 모두 지니고 있다고는 하나 식굴암과 더불어 강조되어야 할 것이다. 이러한 오국적 성격은 그가 지나는 자연적 여건도 넣어서 일찌기 고유섭 선생의 말씀과 같이 ㅣ나라의 병략적 포는 행정상의 궁요한 의미 '를 당초부터 지니고 있는 것이다.

그리하여 동남으로 올산 방면에서 신라의 국도를 위협하는 외적에 대하여 퓟느*중요한 거점으로서의 뜻을 지나고 있디

그 까닭은 신라의 역사에서 미루어 국도 경주를 위협하던 두 지점이라는 것은, 더우기 삼국 통일이 이룩된 이우에 있어서는 첫째 올산이었고, 다음에는 동해구였다고 생각되기 때문이다. 이를 위하여 군사적 요ㅅR로서는 앞에서도 언급한 바 있는 관문성을 쌓았던 것이며, 그 안쯕으로 가까이는 워원사를 모화에 세웠으며, 다시 그에 이어서는 불국시와 식굴암을 이와 같은 외킥 침입의 2대 통로 가까이 또는 그를 바라보는 산 위에 건립하였던 것이다. 앞에서 설명한 바와 같이 신라 8세기 불교의 성격에서 깁씨 왕가의 원당으로서의 성격을 띠고 건립되었으며, 다시 그 뒤에는 이 같은 국방적, 호국적 성격이 있었으니, 이것은 신라의 모른 사원에서 공통으로 지적할 수가 있기도 하다.

이 같은 불국사의 지리직 조건과 관련하여서 그의 소속 사찰로서 토함산을 무대로 불국사와 나리이 장수사, 웅수사, 몽성사 같은 여러 사원의 기록이나 유혀가 남아있는 것도 아울러 주목할 만하다. 그중 징수사는 삼국유사에도 보이고 있어서 곰잡은 곳에 세웠다 이였고, 다른 두 절 중 웅수사는 신 위에 있어 곰을 처음 발견한 땅에, 그리고 몽성사는 꿈을 꾼 곳에 세운 것이라고 전설에는 전하고 있다.

이 같은 김대성에 얽한 인과응보와 불교에의 구,의들 말하는 설화에서 유래한 토함산의 대소가람들은 불국사, 식굴암을 넣어서 불교 교리 그 자체에 입각하는 설교 목적을 위히여 한 큰 역할을 하려던 것으로도 해석된다. 그리하여 불국, 식굳 두 사암의 런립은 다만 깁씨 왕구의 원당으로서의 성격 이외에 위에서 른 불교교리의 설교와 국방적인 의의를 처음부터 지녔다고 넓게 해석할 수가 있을 것이다.

이와 같은 불국사의 위치와 그 성격. 그리고 석굴암과의 관련에서 .불국사의 정로는 토함 신을 등에서 넘어들 제 있고. 국도에서 나와 듬에 있지 않다'고 일■ ㅣ기 고유섭 선생이 지적힌 것은 주목할 만하다 1고유섭= 김대성 한국미술문화사노충, 1948 년 긴핀*

[덧붙임 5] 타자시의 오류를 생성한 awk 프로그램

```
BEGIN {trlit='skcnea'; charpos=0; transposed=0; added=0;
       replaced=0; spzapped=0; wordchange=0;
       freq['s']=0; freq['c']=0; freq['e']=3; freq[' ']=0}
{
   for (i=1;i<=NF;i++) {
      j=length($i)
      for (k=1;k<=j;k++) {
         charpos++
         if (charpos%300==0) {
            if (k<j) {
               printf'%s', substr($i, k+1, 1); transposed+=2
            }
            else {
               printf'%s', substr($i, k, 1); added++
            }
            printf'%s', substr($i, k, 1); k++; affected++
         }
         else {
            if ((ch=substr($i, k, 1))=='s' || ch=='c' || ch=='e') {
               freq[ch] += 1
               if (freq[ch]%50==0) {
                  pos=index(trlit, ch); printf'%s', substr(trlit, pos+1, 1)
                  replaced++; affected++
               }
               else
                  printf'%s', ch
            }
            else
               printf'%s', ch
         }
      }
      freq[' ']++
      if (freq[' ']%50==0) {
                        spzapped++; affected++
      }
      else {
```

```
        printf" "
                        if (affected>0) {
                                wordchange++; affected=0
                        }
      }
      charpos++
   }
   printf"₩n"
}
END { tot=transposed+replaced+added+spzapped
        printf"The number of characters affected is: %d₩n", tot
        printf"₩tTransposed:₩t%d₩n", transposed
        printf"₩tReplaced:₩t%d₩n", replaced
        printf"₩tAdded:₩t₩t%d₩n", added
        printf"₩tSpaces removed:₩t%d₩n", spzapped
        printf"The number of characters in the file is: %d₩n", charpos
        printf"The number of affected words is: %d₩n", wordchange
}
```

참고 문헌

박 인근 1994. '컴퓨터에 우리말 맞추라니: 한글 구현 기술개발 대신 '맞춤법 변경'주장 주객전도'. 한겨레신문 10월 26일자 제10면.

마이크로 소프트웨어 1994. '한글 맞춤법 검사기를 분석한다'. 월간 마이크로 소프트웨어 2월호 (통권 124호). 162~199면.

Damerau, F.J. 1964. "A technique for computer detection and correction of spelling errors." *Communications of the ACM 7, 3.* 171-176.

Peterson, J.L. 1980. "Computer programs for detecting and correcting spelling errors." *Communications of the ACM 23, 12* 676-687.