교정률 최적화를 위한 한국어 철자교정기의 모듈 배열

(A Research on Module Arrangement of Korean Spelling Corrector to Optimize Correction Rate)

윤 근 수 † 권 혁 철 ††
(Keun-Soo Yun) (Hyuk-Chul Kwon)

요 약 본 논문은 한국어 철자교정기의 최적교정률을 보이는 모듈들의 나열순서를 찾는 연구이다. 철자교정기의 모듈 개수가 n개이면 모듈나열 경우의 수는 n!가지가 가능하므로 철자교정기의 최적 교정률을 계산하기가 힘들어 진다. 실험에 사용한 한국어 철자교정기는 현재 19개 모듈들로 구성되어 있다. 입력데이타에 대해서 19!개 모듈을 적용하여 최적교정률을 찾는 것은 현실적으로 불가능하다. 따라서 주어진 입력데이타에 대해 이론적인 최대교정률과 최소교정률을 구하여 교정률 범위를 구하고, 최대교정률에 근접한 최적교정률에 대한 모듈나열순서를 구하는 것이 논문의 목적이다. 최적교정률을 구하기 위해 경험적 지식을 사용하였다.

실험에 사용한 입력데이타는 신문사에서 몇 년간 발생한 오류어절 753,191개의 집합이다. 이 오류집합에 대해 철자교정기의 이론적인 최대교정률은 97.28% (732,764개 / 753,191개)이나 경험적으로 우리가 찾은 최적교정률은 96.62% (727,750개 / 733,191개)이다. 철자교정기의 성능은 99.31% (727,750개 / 732,764개)이다.

키워드 : 철자교정기, 모듈나열, 최대교정률, 최적교정률, N!문제

Abstract We find a module array that takes optimal correction rate of Korean spelling corrector. If there are a lot of module numbers of spelling corrector, it is difficult to calculate optimal correction rate of spelling corrector because permutation of N-modules is N!. This Korean spelling corrector consists of 19 modules. It is impossible to arrange 19 modules actually and the correction rate is various according to input data.

We found the range of correction rate using parallel processing between modules and the optimal correction rate using sequential processing of modules. Input data that are used in an experiment is 753,191 eojeol's sets that happen in newspaper publishing company during several years. About this error set, theoretical maximum correction rate of spelling corrector is 97.28% (732,764 / 753,191). But we got the optimal correction rate 96.62% (727,750 / 733,191). This optimal correction rate is almost near to 99.31% (727,750 / 732,764) of the maximum correction rate.

Key words : Spelling corrector, module arrangement, maximum correction rate, optimal correction rate, N! problem

1. 서 론

철자교정기의 교정률을 높이기 위해서는 아직 교정하지 못하는 부분을 처리하는 새로운 모듈의 개발이 필요하거나 기존 모듈의 손질을 통한 교정률 개선이 필요하다. 한국어 철자교정기의 개발초기에는 시스템의 크기가 작고 모듈의 개수도 몇 개되지 않아 개발자는 전체내용을 고려하면서 시스템을 구축 하였다. 그러나, 철자교정을 위한 모듈이 개발되고 추가되면서 시스템의 크기가 커짐에 따라 나타난 문제는 기존에 교정이 잘 되던 오류어절이 새 모듈의 추가로 인해 교정에 실패하는 것이다. 예를 들면, 오류어절 "5개각으로"에 대하여 모듈 e는 "5개 각으로"을 출력하고, 모듈 m은 "5 개각으로"을 출력한다. 이때 모듈e>모듈m 순서로 실행을 하면 모듈e가 정답을 출력하게 되므로 교정에 성공을 하지만 반대로 모듈m>모듈e로 실행을 하면 모듈 m이 오답을 출

† 정 회 원 : 울산과학대학 컴퓨터정보학부 교수
ksyun@mail.uc.ac.kr
†† 종신회원 : 부산대학교 전기전자정보컴퓨터공학부 교수
hckwon@pusan.ac.kr
논문접수 : 2005년 1월 31일
심사완료 : 2005년 3월 31일

력하게 되므로 교정에 실패한다. 따라서 모듈순서열을 결정하는 것이 교정률에 영향을 미치게 된다. 이들 각 모듈에 대한 설명은 3장에 나와 있다. 이와 같은 문제는 시스템의 크기가 방대해 짐으로 인해 생기는 튜닝문제이며, 오랫동안 시스템이 구축되면서 처음 개발하던 사람이 계속적으로 유지보수를 하게되면 시스템 안정화가 가능하지만 시스템의 설계가 변경되거나 개발자가 바뀌면서 이 문제는 더 심각하게 되었다.

철자교정기의 교정률에 영향을 미치는 것은 모듈자체의 교정률과 모듈의 실행방법을 들 수 있다. 모듈자체의 성능개선은 계속 작업이 이루어지고 있다. 철자교정 모듈들을 실행하는 방법에 있어서 병렬처리와 순차처리 방법을 생각해 볼 수 있다. 병렬처리 방법은 입력 오류어절에 대해 모든 모듈을 실행하여 결과를 취합한 후 적당한 교정결과를 선택하는 과정으로 처리할 수 있다. 모듈들에 대한 병렬처리는 교정률 향상을 위해 가장 이상적이나 너무 많은 시간과 시스템 자원을 요구하여 현실적으로 사용할 수 없다. 다른 방법으로는 모듈들 간 실행순서를 정해 순차처리를 하는 것이다. 가장 교정률이 높고 다른 모듈의 교정에 영향을 적게 미치는 모듈을 먼저 실행하게 하여 모듈의 실행순서를 결정하는 것이다. 이 순차처리시 발생하는 문제는 엄청난 계산량이며, 기존 n-1개 모듈에 새로운 모듈 1개가 추가됨으로 문제의 복잡도는 n배가 증가하게 되고 전체적으로는 n! 문제가 된다. 따라서 이 논문의 실험에서는 모듈들 간 순차처리시 생기는 이 문제를 해결하기 위해 모듈 상호간 교정을 방해하는 정도를 분석하고 시스템의 교정률이 최적화되도록 모듈 간 순서를 결정하는데 초점을 맞추었다.

실험에 사용한 한국어 철자교정기는 계속 개선되고 새로운 모듈이 추가되어 현재 19개 모듈로 구성되어 있다. 이 n=19개 모듈에 대한 순열은 nPn=n!=19!=121,645,100,408,832,000 경우의 수가 된다. 이것은 컴퓨터로 처리하기에는 불가능하다. n번째 모듈이 추가되면 문제의 복잡도는 n배 증가하는 어려운 문제이다. 시스템의 크기가 작을 때는 새 모듈을 적당한 곳에 위치시키면 되었으나 모듈의 개수가 많아짐에 따라 순서를 정하는 것이 쉽지 않게 되었다. 실험에 사용된 데이타에 대해 모듈 간 순서를 다르게 했을 경우 이론적인 최대 교정률은 97.28% (732,764개 /753,191개)이고 최소교정률은 71.04%(535,037개 /753,191개)이었다. 모듈 간 순서에 따라 오류어절에 대한 교정률이 상당한 영향을 받고 있음을 보았다. 철자교정기는 주어진 오류어절 집합(753,191어절)에 대해 최대 교정률은 97.28%인 732,764개를 교정할 수 있으나 이와 같은 교정률을 나타내는 모듈순서를 찾지 못하였다. 대신에 이에 근접한 값인 727,750개를 교정한 모듈순서를 찾았다. 아직 5,014개의 교정이 되지 않는 오류어절이 있으나 0.66%정도이므로 현재 시스템은 거의 최적의 값으로 튜닝이 되어있다고 볼 수 있다.

2. 관련연구

영어에 대한 철자검사교정기는 62년 항공승객 이름에 대한 오류를 교정하는 방법이 처음으로 연구가 시작되어 1970년대에 개발이 완료되었으며, 현재는 문법검사기나 문체검사기가 상용화되어 사용되고 있다. 영어는 단어를 단위로 띄어쓰기를 하며, 단어의 굴절이 심하지 않기 때문에 간단한 방법으로 어휘분석을 할 수 있다. 대부분의 영어 맞춤법 및 교정시스템은 어휘수준에서 처리를 마치지 않고, 의미분석 수준의 처리도 하고있다.

특히, IBM의 퇴고시스템은 철자오류 검사/교정, 문체검사 기능뿐만 아니라 각 단어의 사용빈도, 동의어와 반의어 등이 제공되며, 나이에 따라 이해가 가능한 단어를 분리함으로써 문서를 읽는 대상자의 나이에 따라 문서를 작성할 수 있게 도와주고 있다. 마이크로소프트사는 문장분석에 기반한 영어문법검사기, 틀리기 쉬운 단어에 대한 도움말, 온라인 사전, 용례사전, 동의어와 반의어 사전 등 문서작성에 필요한 다양한 기능을 제공한다.

WWB는 유닉스환경에서 영어문장의 교정을 담당하는 시스템으로 문서의 문법적인 면을 분석하여 사용자에게 도움말을 제공하며, 문서에서 사용한 영어에 대한 정보를 제공한다[1]. IBM의 CRITIQUE 시스템은 파서를 사용하여 문서의 문법, 문체적인 면을 분석하여 도움말을 제공한다[2]. EPISTLE 시스템은 단어, 절, 문장수준의 처리를 하며 도움말을 제공한다[3].

한글 맞춤법 검색 및 교정시스템은 80년대 중반 이후에 컴퓨터를 이용한 문서처리가 증가되면서 시작되었다. 기존의 시스템으로는 한국과학기술원의 HSPELL[4]과 서울대학교 맞춤법 검사교정기[5]가 있었다. 심광섭은 말뭉치에서 추출한 음절 bigram빈도를 음절간 띄어쓰기에 사용하였다[6]. 강승식은 어절 블록 양방향 알고리즘을 개발하여 띄어쓰기 오류를 자동처리하고, 음절bigram을 이용하여 자동교정하였다[7].

한국과학기술원은 좌우접속정보를 이용한 최장일치법과 양방향 접근방법을 이용한 철자오류와 띄어쓰기 오류를 찾기위한 검사와 교정기를 개발하였다. 오류검사를 위해 일반적인 형태소해석방법과 동일한 방법이며, 오류의 종류와 위치를 찾기위해 양쪽방향에서 형태소해석을 하는 방법이다[8].

연세대는 연세대 사전 편찬실에서 구축한 말뭉치 중에서 오류가 발생한 어절에 대한 교정처리를 위해 유닉스 환경에서 시스템을 구현했다. 약 16만 오류 어절에

대해 철자오류, 띄어쓰기 오류, 복합오류, 미등록어 오류로 분류하였고, 이들에 대한 교정률은 77%이다. 또한 처리속도 면에서 한 어절당 5~6초 정도가 소모되었다고 보고하였다[9].

포항공대는 자판특성을 이용한 철자교정과[10] N-gram을 이용하여 교정을 하였다[11], 고려대는 통계적 모델을 이용하여 띄어쓰기를 처리하였다[12].

부산대에서는 단어간 공기정보를 이용한 방법[13], 부분구문분석을 이용한 교정법, 확률과 규칙에 기반한 한국어 문법검사기가 설계되어, 규칙이 적용된 후, 말뭉치를 통해 얻어진 통계적인 가치에 따라 가중치를 부여함으로써 선택하는 방법을 사용하였다[14].

앞 연구들을 오류유형에 따라 분류하고 철자교정기법을 요약하여 표 1에 제시하였다.

표 1 오류유형에 대한 한국어 철자교정기의 교정기법

번호	오류유형	교정기법
1	조사 및 어미 음소변형오류	말뭉치 음소 오류자료
2	띄어쓰기 오류	말뭉치 띄어쓰기 자료 역사전 탐색 n-gram 한글사전 이용
3	음소변형오류	음소대치규칙 테이블 역사전 탐색
4	자판오류(자소변형오류)	자판운지거리 신경망 및 퍼지
5	오인식	n-gram 한글사전 이용

3. 철자교정기구조와 모듈소개

철자 검사기에서 넘어온 오류가능 어절은 19개 모듈로 구성된 철자교정기에 입력이 된다. 입력어절을 전체 모듈이 병렬로 처리한 후 출력을 분석하는 방법도 있겠으나 이 병렬처리 방법은 일반 컴퓨터에서 처리하기에는 너무 많은 자원을 요구하므로 실시간 처리가 불가능하다. 다른 방법으로 순차처리를 하게되는데 이들 모듈을 입력어절에 대해 적용순서를 나열하는 방법이 무려 19! 개나 된다. 이 또한 어려운 방법이지만 본 시스템은 경험적인 방법으로 이들 모듈을 나열하고 있으며, 교정률이 높고 다른 모듈에 영향을 적게 미치는 모듈을 앞쪽으로 두는 방법을 취하여 모듈들을 배치시켜서 사용하고 있다. 그림 1은 교정과정의 전체구성도를 보여준다.

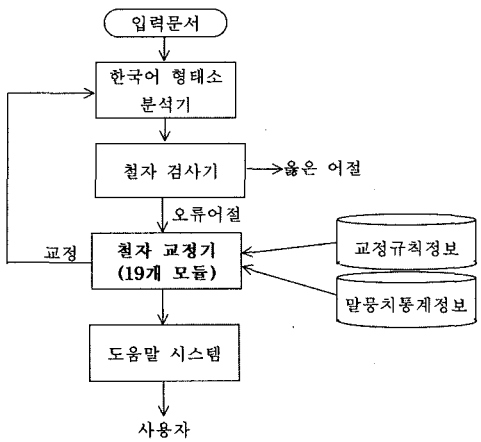


그림 1 한국어 철자교정기의 구성도

다음은 모듈 19개에 대한 개략적인 소개이다. 모듈명과 처리내용 및 처리예제를 보였다. [모듈a]부터 [모듈s]까지 19개 모듈이며 알파벳은 실행순서와 상관이 없다.

[모듈a] GetOyongCandWord()
;수사오용어/오용어 대치
;노가다 >막노동자, 오얏 > 자두

[모듈b] UnconditionalSpacing()
;"bi-gram통계 사용"
;7번만에>7번^만에
가능성에대한>가능성에^대한
감시하고있다>감시하고^있다

[모듈c] UnconditionalReplace()
;실제로 쓰이지 않는 음절에 대한 무조건 패턴대치
;것네>겠네
많는 > 많은
읍니 > 습니
겟 > 겠

[모듈d] ErrorCorrectBySpecialRule()
;특수 교정 규칙을 적용함
;규칙) 동사 + 한데 -> 동사+한 데
규칙) 몯+동사 -> 못 ^ 동사
않+동사 -> 안 ^ 동사
규칙) 공무원수감축이->사람|동물|식물^수^하다-명사

[모듈e] SpacingBySusaNoun()
;수사와 인접한 분류사(수의존명사:'원','명','분'...) 뒤에 띄어써 보기

[모듈f] SpacingBySyllablePair()
-띄어쓸 확률이 높은 음절쌍(bigram)들에 대한 띄어써 보기
-통계 데이타에서 띄어쓸 확률이 아주 높은 경우의 띄어쓰기 적용

[모듈g] ReplaceConsonant()
;ㅋㅋ -> 크크 등으로 바꿔보는 규칙 적용
ㅉㅉ-> 쯧쯧
ㅎㅎ -> 하하

[모듈h] SyllableReplace()
;가중치가 높은 음절 대치 규칙을 적용해 본다

ㅇ로-> 으로, 아->어, 아래->아랫, 아야되->아야하
와 같이 자주 오류를 발생시킬 수 있는 경우에는 대치하여 검사해 본다

[모듈i] SpacingByHeuristicWord()
;두 음절 이상의 특수한 명사 띄어쓰기
가격, 가운데, 간부, 강사 등의 앞에서 띄어 보는 규칙

[모듈j] pacingByHeuristicFirstWord()
;앞에서 등장하는 특수한 단어 뒤에서 띄어쓰기
각종(각종문구류 >각종^문구류), 각층, 각, 갓, 곧 등이 어절의 처음에 나왔을 때 띄어보는 규칙

[모듈k] SpacingByHeuristicOneWord()
;한음절의 특수한 명사 띄어쓰기
간, 값, 것, 겸, 곁 등이 어절의 중간에 나왔을 때 띄어보는 규칙

[모듈l] SpacingByCandidatePos1()
;가중치가 아주 높은 띄어쓰기 규칙을 우선 적용해 본다

[모듈m] SpacingByCandidatePos2()
;가중치가 上中 단계인 띄어쓰기 규칙을 적용

[모듈n] SpacingByCandidatePos3()
;가중치가 中下 단계인 띄어쓰기 규칙을 적용한다

[모듈o] SpacingByCandidatePos3()
;가중치가 아주 낮은 띄어쓰기 규칙

[모듈p] HighPriorityJungsungReplace()
;가중치가 높은 중성 대치
ㅏ->ㅑㅣㅓ, ㅐ->ㅒㅔㅖ, ㅑ->ㅏㅣㅕㅓ, ㅒ->ㅐㅑ
-등과 같이 중성을 대치하여 적용해 보는 규칙

[모듈q] PhonemeReplace ()
;~(ㅆㅆ) -> (ㅆㅇ) 갔썼다 => ~갔었다
;연음법칙 - 종성+'ㅇ' -> 종성을 지우고 앞의 종성을 뒤의 초성으로 옮김
;사잇소리 규칙 - 앞 음절의 종성이 없고 음절의 초성이 경음이면 앞 음절의 종성을 'ㅅ'으로 변경
-등과 같이 초성, 중성, 종성의 조합관계를 보고 대치해서 적용해보는 규칙

[모듈r] DaepyoeumReplace ()
;ㄲ->ㄱ(먹다->먹다), ㄳ->ㄱ, ㄴ->ㄶㅈㅅ, ㄵ->ㄴ
-등과 같이 종성을 대표음으로 대치하여 적용해 보는 규칙

[모듈s] KeyboardDistanceReplace()
;키보드 자판에서 거리에 의해 잘못 사용될 수 있는 것을 대치하여 적용해 보는 규칙
;ㅏ->ㅓ, ㅐ->ㅔ 등은 중성과 관련한 자판 대치의 예로 'ㅏ'는 'ㅓ'와 자판에서 거리가 비슷해서 잘못 사용할 수 있으므로 대치하여 적용해 보는 규칙

4. 철자교정기의 교정률범위와 최적교정률

철자교정기를 구성하고 있는 모듈들 간 실행 순서에 의해 교정률이 영향을 받게 된다. 따라서 어떤 순서에 따라 모듈을 배치하면 교정률을 최적으로 할 수 있는지를 분석하고 조사하여야 한다. 최적 교정률을 찾기위해 먼저 모듈들 간 종속이 없는 독립관계라고 가정을 한 상태에서 모듈들을 독립적으로 각각 실행한 후 최대교정률과 최소교정률을 찾는다. 두 번째, 모듈들 간 관계가 종속이라고 가정한 상태에서 모듈을 순차처리 실행한 후 최적교정률을 찾아 앞에서 찾은 최대교정률 및 최소교정률과 비교하여 어느 정도의 교정률을 보이는지 살펴보고자 한다.

4.1 모듈 간 병렬처리를 통한 교정률범위 결정

철자 교정기의 각 모듈 간에 서로 종속적인 관계를 가지고 있기 때문에 시스템의 최적교정률을 찾기가 쉽지 않다. 모듈 M_i의 출력 중 정답(success) 집합을 O_i, 오답(error) 집합을 X_i, 무응답(pass) 집합을 P_i라고 하자. 최적교정률을 구하기 위해 각 모듈 간에는 $M_i \cap M_j = \phi$, $i \neq j$ 라고 가정할 때 최대교정률 $n\left(\bigcup_{i=1}^{N} O_i\right)/n(U) * 100$ %를 찾는다. 모듈 간 종속상태에서 이 최대교정률을 기준으로 시스템이 가장 근접한 최적교정률을 찾아가는 과정을 살펴본다. 그림2는 입력 오류어절집합에 대하여 각 모듈을 독립적으로 실행하여 얻은 과정을 그린 것이다.

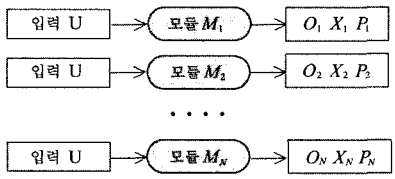


그림 2 모듈 간 병렬처리

여기서 입력집합U는 형태소분석에 실패하여 철자검사기에서 오류어절이라고 판단된 어절로 교정을 필요로 하는 전체오류어절 집합이다. M_i은 i번째 모듈, O_i은 모듈M_i의 출력 중 올바른 교정을 한 어절집합, X_i는 모듈M_i의 출력 중 틀린 교정을 한 어절집합, P_i는 모듈M_i가 출력을 내지 않고 통과해 틀린 어절집합이다. X_i와 P_i는 올바른 어절을 제시하지 못했다는 점에서는 동일하나 전자X_i는 답을 제시하였으나 틀린 경우이고 후자P_i는 아무런 답을 제시하지 않아서 틀린 경우로 서로 구분하였다. 이것은 나중에 모듈들 간에 종속을 고려할 때 X_i는 다음 모듈로 넘어가는 것을 가로채 다음모

듈이 실행될 기회를 주지 않고, P_i는 다음 모듈의 입력으로 들어가게 되어 올바로 교정되는 경우가 있기 때문에 다르게 취급되어야 한다.

표 2는 동일한 오류입력집합U에 대하여 각 모듈을 실행하였을 때 나온 3가지 유형의 출력형태를 보여주고 있다.

각 모듈이 정답을 제시한 O_i에 대한 각각의 합은 1,292,803개이고, 각 $O_i \cap O_j$, ($i \neq j$)에 대한 교집합의 원소개수는 560,039를 구하였다. O_i를 합집합 하였을 경우 원소의 개수는 732,764 개이다. 이 숫자의 의미는 입력집합 U(753,191개)에 대한 철자교정기의 최대한 교정할 수 있는 개수는 732,764개를 넘지 않는다는 것이다. 마찬가지로, $n(X)=n\left(\bigcup_{i=1}^{N} X_i\right)$는 시스템의 모든 모듈에 한번이상 틀린 것으로 선택될 개수를 의미한다.

$$n(O) = n\left(\bigcup_{i=1}^{N} O_i\right) = \sum_{i=1}^{N} n(O_i) - n\left(\bigcup_{i=j=1}^{N,\, i \neq j} (O_i \cap O_j)\right)$$

$$= 1{,}292{,}803 - 560{,}039 = 732{,}764$$

위의 계산으로부터 $n(O) = 732{,}764$은 시스템의 "최대교정 어절 수"를 나타내고,

$$n(X) = n\left(\bigcup_{i=1}^{N} X_i\right) = \sum_{i=1}^{N} n(X_i) - n\left(\bigcup_{i=j=1}^{N,\, i \neq j} (X_i \cap X_j)\right)$$

$$= 233{,}822 - 36{,}095 = 197{,}727$$

$$n(O - X) = n\left(\bigcup_{i=1}^{N} O_i - \bigcup_{i=1}^{N} X_i\right) = n\left(\bigcup_{i=1}^{N} O_i\right) - n\left(\bigcup_{i=1}^{N} O_i \cap \bigcup_{i=1}^{N} X_i\right)$$

$$= 732{,}764 - 130{,}838 = 601{,}926$$

은 시스템의 "최소교정 어절 수"를 나타낸다.

따라서, **최대교정률(CR_{max})**과 **최소교정률(CR_{min})**은 다음과 같다:

$CR_{max} = n(O) / n(U) * 100\ \%$,

$CR_{min} = n(O - X) / n(U) * 100\ \%$.

지금까지 최대교정률과 최소교정률은 시스템의 모듈들을 가장 이상적으로 배치하였을 때의 상한과 최악의 모듈배치를 하였을 때의 값인 하한에 해당하는 것으로 교정률의 범위이다. 주어진 오류입력집합U에 대한 **"최대 교정률"**은 $n(O) / n(U) = 732{,}764 / 753{,}191$ (97.29%)이고, **"최소 교정률"**은 $n(O - X)/n(U) = 601{,}926 / 753{,}191$ (79.92%)이었다.

4.2 모듈 간 순차처리를 통한 최적교정률 결정

앞 절에서처럼 모듈을 병렬 처리한 다음 각 모듈에서 나온 결과를 비교 검토하여 옳은 답을 선택하는 방법은 너무 많은 시스템 자원과 시간을 요구하므로 현실적으로 사용할 수 없기 때문에 실용적으로 활용하기 위해서는 철자교정기의 모듈들을 순차처리하게 된다. 그림 3은 철자교정기의 모듈처리 순서와 출력결과를 보여주고 있다.

처음 입력집합은 P_0이고, 교정이 될 오류어절의 전체집합이다. 각 모듈 M_i의 입력오류집합은 P_{i-1}이고, 출력

표 2 입력 오류어절집합에 대한 각 모듈의 독립 실행결과

번호i	모듈 M_i	입력오류 집합크기	O_i (정답제시)	X_i (오답제시)	P_i (무응답)
1	a	753,191	317,286 (98.53%)	4,722 (1.47%)	431,183 (57.25%)
2	b	753,191	65,702 (97.17%)	1,914 (2.83%)	685,575 (91.02%)
3	c	753,191	222,584 (88.91%)	27,758 (11.09%)	502,849 (66.76%)
4	d	753,191	2,347 (82.55%)	496 (17.45%)	750,348 (99.62%)
5	e	753,191	27,839 (92.40%)	2,289 (7.60%)	723,063 (96.00%)
6	f	753,191	183,115 (96.46%)	6,713 (3.54%)	563,363 (74.80%)
7	g	753,191	0 (0.00%)	0 (0.00%)	753,191 (100.0%)
8	h	753,191	210 (28.04%)	539 (71.96%)	752,442 (99.90%)
9	i	753,191	19,999 (97.64%)	484 (2.36%)	732,708 (97.28%)
10	j	753,191	1,861 (97.13%)	55 (2.87%)	751,275 (99.75%)
11	k	753,191	125,661 (96.07%)	5,138 (3.93%)	622,392 (82.63%)
12	l	753,191	16,436 (89.40%)	1,949 (10.60%)	734,806 (97.56%)
13	m	753,191	117,944 (91.92%)	10,367 (8.08%)	624,880 (82.96%)
14	n	753,191	129,500 (89.39%)	15,367 (10.61%)	608,324 (80.77%)
15	o	753,191	61,661 (90.55%)	6,436 (9.45%)	685,094 (90.96%)
16	p	753,191	68 (0.53%)	12,745 (99.47%)	740,378 (98.30%)
17	q	753,191	82 (2.18%)	3,677 (97.82%)	749,432 (99.50%)
18	r	753,191	10 (0.03%)	35,038 (99.97%)	718,143 (95.35%)
19	s	753,191	498 (0.50%)	98,135 (99.50%)	654,558 (86.90%)
		동일함	$\sum_{i=1}^{N} n(O_i)$=1,292,803	$\sum_{i=1}^{N} n(X_i)$=233,822	

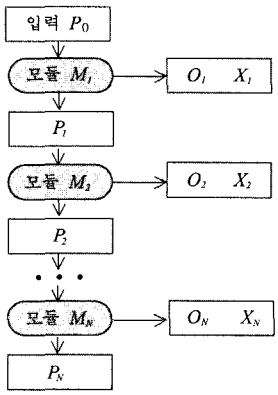


그림 3 모듈 간 순차처리

에는 정답집합이 O_i이고, 오답집합은 X_i이고, 무응답집합은 P_i 으로 3가지 유형으로 나온다. 즉, $n(P_{i-1}) = n(O_i \cup X_i \cup P_i)$ 이며, 다음 식을 만족한다.

$$n(P_0) = \sum_{i=1}^{N} n(P_i) = \sum_{i=1}^{N} n(O_i) + \sum_{i=1}^{N} n(X_i) + n(P_N),$$

$$n\left(\bigcup_{i=1}^{N} O_i\right) = \sum_{i=1}^{N} n(O_i), \quad \because \; n\left(\bigcap_{i=1}^{N} O_i\right) = 0$$

철자교정기의 **최적 교정률(CR_{opt})**은 다음 식으로 표현된다:

$$CR_{opt} = \sum_{i=1}^{N} n(O_i) \; / \; n(U) * 100\ \%.$$

최적교정률은 항상 최대교정률과 최소교정률의 범위 안에 분포하게 된다. 즉,

$$CRmin \leq CRopt \leq CRmax.$$

철자교정기를 구성하고 있는 모듈들 간의 실행 순서에 의해 교정률이 다르게 나올 수 있으므로, 이들 모듈들 간에 서로 교정률에 영향을 미치는 부분을 분석하여 모듈 간 순서를 조정해 줄 필요가 있다. 표 3은 철자교정기의 각 모듈이 출력한 결과 중 오답집합과 정답집합의 교집합을 나타낸다. 이 표에서 의미하는 것은 모듈들 간의 실행순서에 의해 교정률이 달라질 수 있음을 보여준다. 한 가지 예로, 모듈 Ms가 가장 먼저 실행되면 269,374개의 어절이 틀린 것으로 오답을 제시하게 된다. 즉, 모듈 Ms가 269,374개 어절을 가로채서 틀린 답을 내보낸다. 반대로 모듈 Ms가 가장 뒤에 실행된다면 269,374개의 어절이 맞는 것으로 정답을 출력되는데, 이 어절들이 모듈 Ms에 도달하기 전에 다른 모듈들에서 정답으로 처리된다는 뜻이다. 이 숫자는 표 3의 우하위치에 나타나 있다.

표 3에서 (Oa, Xa)위치에 있는 33은 표 4처럼 같은 입력오류어절에 대해 2가지의 정답이 주어진 경우로 시스템은 입력오류어절에 대해 항상 한 가지 응답을 하기 때문에 생긴 문제이다. 나머지 대각선에 위치하는 숫자도 이와 같은 현상을 보이고 있으며 전체 89개가 두 가지 정답을 제시하고 있다. 전체 어절숫자(753191개)에 비해 89개는 0.0001%정도이므로 자기 자신 모듈 간 출력의 교집합의 합 $\sum_{i=1}^{N} n(O_i \cap X_i) = 89$은 무시할 수 있는 개수이다.

표 5는 두 모듈의 정답집합 간 교집합에 대한 분석결과이다. 두 모듈에 대한 합집합연산을 구할 때 중복되는 엔트리 숫자는 이표로부터 제거되어야 이중카운트가 되지 않는다.

표 3과 표 5로부터 두 모듈(Me, Mm)간 서로 순서를 바꾸었을 때의 교정률을 분석하여 어떤 순서가 더 교정률이 나은가를 살펴보고자 한다. Xe는 입력어절에 대해

표 3 모듈출력의 오답집합과 정답집합간 상관표

∩	Xa	Xb	Xc	Xd	Xe	Xf	Xg	Xh	Xi	Xj	Xk	Xl	Xm	Xn	Xo	Xp	Xq	Xr	Xs
Oa	33	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Ob	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Oc	.	.	11	164	359	1,649	.	96	70	15	281	475	2,104	4,180	1,072	3,306	1,056	28,013	50,259
Od	.	.	499	.	.	209	.	3	33	1	20	452	107	897	76	224	28	37	1,071
Oe	.	.	1,161	.	12	283	.	52	35	.	6	.	526	1	11	149	184	1,130	5,075
Of	.	.	9,975	151	312	11	.	59	27	13	27	299	1,435	2,871	296	3,164	954	28,322	48,049
Og	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Oh	.	.	32	.	8	9	.	.	.	.	.	.	29	83	.	.	4	.	15
Oi	.	.	623	11	21	34	.	20	.	1	3,689	49	1,952	666	2,561	340	459	1,539	9,543
Oj	.	.	99	3	.	54	.	5	3	.	19	8	49	146	38	14	16	8	688
Ok	.	.	3,509	103	.	60	.	45	177	33	.	589	1,713	6,714	697	9,839	1,224	32,538	61,246
Ol	.	.	786	80	.	404	.	54	20	2	636	3	830	1,663	458	441	280	172	4,594
Om	.	.	7,629	82	261	1,794	.	194	158	9	289	180	6	3,358	1,492	1,141	587	1,589	13,057
On	.	.	6,619	160	.	854	.	180	71	23	3,016	576	3,221	3	3,537	7,148	2,299	14,796	46,003
Oo	.	.	1,552	30	8	56	.	23	87	13	17	353	1,073	5,620	8	3,466	144	17,913	29,721
Op	.	.	16	.	2	12	.	2	.	.	.	.	8	11	2	.	.	.	19
Oq	.	.	11	.	2	10	.	.	.	.	1	.	9	20	3	2	.	.	32
Or	.	.	1	.	.	1	.	.	.	.	.	.	.	2	.	.	4	.	2
Os	.	.	69	.	9	24	.	.	.	.	2	2	39	100	6	3	10	2	.
계	33	2	32,592	784	994	5,464	0	733	681	110	8,003	2,986	13,101	26,335	10,257	29,237	7,249	126,059	269,374

표 4 입력오류어절에 대해 두 가지 정답을 제시한 예. n(Oa∩Xa)=33

번호	입력오류어절	정답	시스템응답
1	1000여명가량 1000여명가량	1000여 명 가량(o) 1000여 명가량(o)	1000여 명가량(x) 1000여 명가량(o)
...	...	...	...
33	8분20여초를 8분20여초를	8분 20여 초를(o) 8분20여 초를(o)	8분20여 초를(x) 8분20여 초를(o)

표 5 모듈출력의 정답집합간 상관표

∩	Oa	Ob	Oc	Od	Oe	Of	Og	Oh	Oi	Oj	Ok	Ol	Om	On	Oo	Op	Oq	Or	Os
Oa			.	.	.	.		.	.	.	.	.	.	.	.	.	.	.	.
Ob			.	.	.	.		.	.	.	.	.	.	.	.	.	.	.	.
Oc			.	447	14,729	171,777		.	7,777	822	84,096	7,050	82872	74,702	43,401	.	.	.	.
Od			447	.	.	427		.	9	3	410	268	75	474	158	.	2	.	.
Oe			14,729	.	.	14,225		.	236	.	11,739	.	**23953**	.	551	.	.	.	.
Of			171,777	427	14,225	.		.	7,453	622	84,659	3,968	74970	50,880	38,374	.	.	.	.
Og			.	.	.	.		.	.	.	.	.	.	.	.	.	.	.	.
Oh			.	.	.	.		.	.	.	.	.	.	.	.	2	7	2	164
Oi			7,777	9	236	7,453		.	.	54	310	1,181	2997	15,156	160	.	.	.	.
Oj			822	3	.	622		.	54	.	67	229	637	911	.	.	.	.	.
Ok			84,096	410	11,739	84,659		.	310	67	.	1,148	23904	46,186	53,178	.	.	.	.
Ol			7,050	268	.	3,968		.	1,181	229	1,148	.	5	.	.	.	.	.	.
Om			82,872	75	**23,953**	74,970		.	2,997	637	23,904	5	.	.	.	.	.	.	.
On			74,702	474	.	50,880		.	15,156	911	46,186	.	.	.	.	.	.	.	.
Oo			43,401	158	551	38,374		.	160	.	53,178	.	.	.	.	.	.	.	.
Op			.	.	.	.		2	.	.	.	.	.	.	.	.	.	.	.
Oq			.	2	.	.		7	.	.	.	.	.	.	.	.	.	.	9
Or			.	.	.	.		2	.	.	.	.	.	.	.	.	.	.	7
Os			.	.	.	.		164	.	.	.	.	.	.	.	.	9	7	.
계	0	0	487,673	2,273	65,433	447,355	0	175	35,333	3,345	305,697	13,849	209413	188,309	135,822	2	18	9	180

모듈e가 틀린 답을 제시한 경우의 집합이고, Oe는 모듈e가 정답을 제시한 경우의 집합을 나타낸다. 표 3은 모듈출력의 오답집합과 정답집합 간 교집합에 대한 개수를 나타내고, 표 5는 정답집합 간 교집합에 대한 개수를 보여준다.

그림 4는 입력오류집합 n(U)= 753,191에 대하여 모듈Me와 모듈Mm의 출력결과를 각각 나타내고 있으며, 출력 간에 발생 가능한 교집합부분을 나타내고 있다. Oe, Xe, Pe는 서로 독립적이므로 Oe∩Xe =ϕ, Oe∩Pe = ϕ, Pe∩Xe =ϕ 이다 두 모듈 간에 생길 수 있는 교집

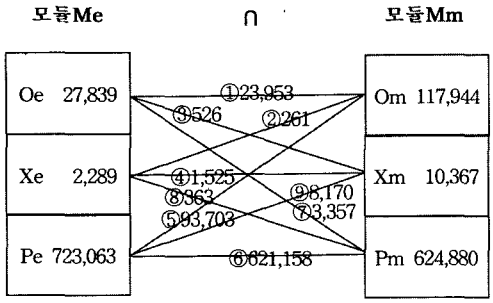


그림 4 모듈Me, Mm의 출력 간 교집합

합의 개수는 9개이지만 중요한 것은 ①=n(Oe∩Om)= 23,953, ②=n(Xe∩Om)= 261, ③=n(Oe∩Xm)= 526 ④=n(Xe∩Xm)=1,525 이다.

성능이 뛰어난 컴퓨터를 사용한다면 두 모듈을 각각 처리하여 출력되는 결과 집합을 분석하고 옳은 결과를 선택하는 과정을 거치면 가장 이상적인 시스템이 된다. 이때 철자 교정이 가능한 오류어절개수는

$$n(Oe \cup Om)=n(Oe)+n(Om)-n(Oe \cap Om)$$
$$=27,839+117,944-23,953 = 121,830$$

이므로 가장 큰 교정 가능한 어절개수이다. 그러나 병렬처리는 계산량이 많아 현실적으로 사용이 어렵고, 각 모듈이 출력한 결과가 서로 다를 때 진위를 결정하기 힘들기 때문에 여기서는 순차처리시 최적교정률에 대한 가이드로만 활용하고자 한다.

그림 5와 그림 6은 두 모듈 간 순서를 바꾸어서 실행해본 결과를 보여준다. Oem은 모듈e>모듈m순으로 실행할 때 정답집합의 합집합을 나타낸다고 하자. 그림 4에서처럼 모듈들 간 종속적이기 때문에 n(Oem)= n(Oe∪Om)=n(Oe) + n(Om) - n(Oe∩Om) = 27,844 + 93,799 - 0 = 121,643이고, 그림 5에서는 n(Ome)=n(Om

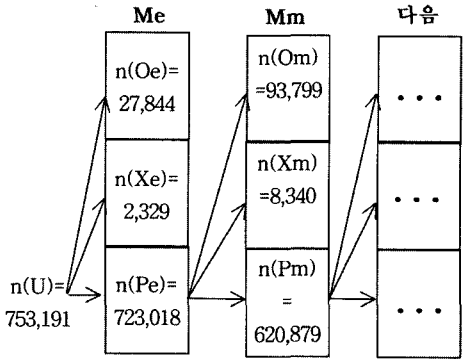


그림 5 모듈순서 Me>Mm 일 때

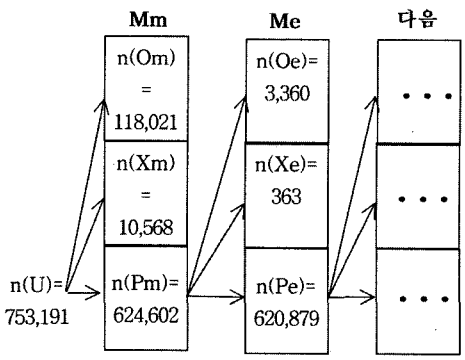


그림 6 모듈순서 Mm>Me 일 때

$\cup$Oe)=n(Om) + n(Oe) - n(Om$\cap$Oe) = 118,021 + 3,360 - 0 = 121,381이다. 따라서 n(Oem) - n(Ome) = 121,643 - 121,381= 262개를 구했다. 이것은 모듈순서를 모듈m>모듈e보다는 모듈e>모듈m순서로 실행하는 것이 교정률이 높아짐을 나타낸다.

예를 들어 입력어절 '1차시험과목인', '21세기경제전망'에 대해 모듈e와 모듈m의 실행순서를 분석해보자. 모듈e는 어절 '1차 시험과목인'에 대해서는 정답을 제시하나 '21세기경 제 전망'에 대해서는 오답을 출력한다. 반면에 모듈m은 정반대의 결과를 출력한다. 이와 같은 경우를 보여주는 어절은 표 3으로부터 n(Om $\cap$ Xe)=261, n(Oe $\cap$ Xm)=526 개이다. 이 숫자의 의미는 모듈e>모듈m 순서로 실행할 경우 261개 어절이 틀린 답을 제시하게 되고, 모듈m>모듈e 순으로 실행할 경우 526개의 틀린 답을 제시한다는 것이다. 따라서 모듈e를 모듈m 보다 먼저 실행하는 것이 필요하다. 표 6은 모듈e와 모듈m간의 예제비교의 결과이다.

표 6 두 입력어절에 대한 모듈e와 모듈m의 실행결과비교

구분	예1	예2
오류어절	21세기경제전망	1차시험과목인
정답어절	21세기 경제전망	1차 시험과목인
모듈e	21세기경 제 전망(x)	1차 시험과목인(o)
모듈m	21세기 경제전망(o)	1차 시험과 목인(x)
동일한 경우	261개	526개

그러나 위의 예처럼 모듈개수가 적을 때는 개발자가 적당히 모듈배치를 하면 최적 성능을 찾을 수 있으나 모듈개수가 많을 때는 모듈 간 간섭현상이 증가하여 오히려 교정률이 떨어진다. 이 문제를 해결하기 위한 시도는 모듈조합이 가능한 모든 경우의 수를 구하여 실험을 하여야 최적교정률을 찾을 수 있다. 그러나 실험에 사용한 철자교정기는 19개 모듈로 되어 있어서 19!이라는 숫자는 도저히 처리할 수 없기 때문에 경험적 방법을 사용하여 교정률을 구하여야 한다.

앞 절에서 우리는 교정률의 범위를 구하였고 이번 절에서는 순차처리시 최적교정률을 구하기 위해 모듈순서를 결정하는 방법을 살펴보았다. 최대교정률을 보이는 모듈순서를 찾을 수 없었으나 경험적 방법을 사용하여 최대교정률에 근접한 **"최적교정률"** 727,750 / 753,191 (96.62%)을 구하였다.

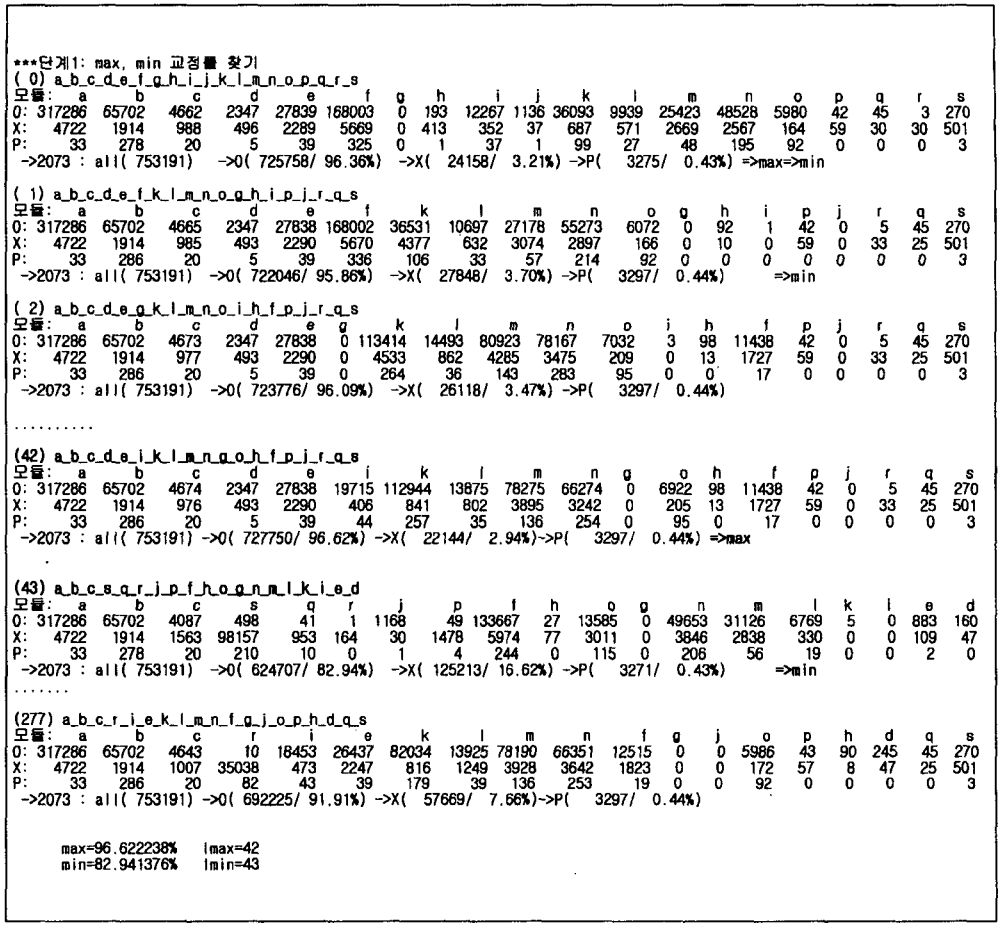
5. 실험결과 및 분석

실험에 사용한 입력 오류어절 집합은 신문사에서 다년간 발생한 오류어절을 수집한 것이며 엔트리는 오류어절과 이를 수작업으로 교정한 정답어절로 구성되어 있다. 엔트리 수는 n(입력오류집합U) = 753,191개 어절이다. 모든 엔트리는 한 어절로 구성되어 있으며 한 어절에 대해 여러 번 띄어쓰기도 처리한다. 또한 한 오류어절에 대해 복수개의 정답어절인 경우도 포함되어 있다. 실험방법은 5.2절에서 보인 경험적 지식을 바탕으로 임의로 278가지의 모듈순서열을 만들어 프로그램을 실행한 후, 그 중에 가장 높은 교정률의 모듈순서열을 선택하였다.

5.1 최적교정률을 구하는 실험

그림 7은 입력오류어절에 대한 순차처리결과를 보여준다. 19!가지 모듈순서열 중 278개의 모듈순서열에 대해 실험한 결과 42번째에서 최적교정률을 보이는 모듈순서열을 찾았다.

그림 7을 보는 방법을 설명하기 위해 표 7을 사용한다. (42) a_ b_ c_ d_ e_ i_ k_ l_ m_ n_ g_ o_ h_ f_ p_ j_ r_ q_ s 라인에서 알파벳은 모듈명이며, 왼쪽부터 오른쪽으로 순차처리한다. 모듈a는 입력어절집합(753,191개)에 대하여 정답(317,286개)과 오답(4,722) 및 무응답(33)을 처리하고 나머지 어절(431,150개=753,191-317,286-4,722-33)을 모듈b에게 넘겨준다. 모듈b는 입력어절집합



```
***단계1: max, min 교정률 찾기
( 0) a_b_c_d_e_f_g_h_i_j_k_l_m_n_o_p_q_r_s
모듈:    a     b     c     d     e     f     g     h     i     j     k     l     m     n     o     p     q     r     s
O: 317286  65702  4662  2347  27839 168003     0   193 12267  1136 36093  9939 25423 48528  5980    42    45     3   270
X:   4722   1914   988   496  2289  5669     0   413   352    37   687   571  2669  2567   164    59    30    30   501
P:     33    278    20     5    39   325     0     1    37     1    99    27    48   195    92     0     0     0     3
 ->2073 : all( 753191)  ->O( 725758/ 96.36%)  ->X(  24158/  3.21%) ->P(   3275/  0.43%) =>max=>min

( 1) a_b_c_d_e_f_k_l_m_n_o_g_h_i_p_j_r_q_s
모듈:    a     b     c     d     e     f     k     l     m     n     o     g     h     i     p     j     r     q     s
O: 317286  65702  4665  2347  27838 168002 36531 10697 27178 55273  6072     0    92     1    42     0     5    45   270
X:   4722   1914   985   493  2290  5670  4377   632  3074  2897   166     0    10     0    59     0    33    25   501
P:     33    286    20     5    39   336   106    33    57   214    92     0     0     0     0     0     0     0     3
 ->2073 : all( 753191)  ->O( 722046/ 95.86%)  ->X(  27848/  3.70%) ->P(   3297/  0.44%)      =>min

( 2) a_b_c_d_e_g_k_l_m_n_o_i_h_f_p_j_r_q_s
모듈:    a     b     c     d     e     g     k     l     m     n     o     i     h     f     p     j     r     q     s
O: 317286  65702  4673  2347  27838     0 113414 14493 80923 78167  7032     3    98 11438    42     0     5    45   270
X:   4722   1914   977   493  2290     0  4533   862  4285  3475   209     0    13  1727    59     0    33    25   501
P:     33    286    20     5    39     0   264    36   143   283    95     0     0    17     0     0     0     0     3
 ->2073 : all( 753191)  ->O( 723776/ 96.09%)  ->X(  26118/  3.47%) ->P(   3297/  0.44%)

.........

(42) a_b_c_d_e_i_k_l_m_n_g_o_h_f_p_j_r_q_s
모듈:    a     b     c     d     e     i     k     l     m     n     g     o     h     f     p     j     r     q     s
O: 317286  65702  4674  2347  27838 19715 112944 13875 78275 66274     0  6922    98 11438    42     0     5    45   270
X:   4722   1914   976   493  2290   406   841   802  3895  3242     0   205    13  1727    59     0    33    25   501
P:     33    286    20     5    39    44   257    35   136   254     0    95     0    17     0     0     0     0     3
 ->2073 : all( 753191) ->O( 727750/ 96.62%) ->X(  22144/  2.94%)->P(   3297/  0.44%) =>max
   .

(43) a_b_c_s_q_r_j_p_f_h_o_g_n_m_l_k_i_e_d
모듈:    a     b     c     s     q     r     j     p     f     h     o     g     n     m     l     k     i     e     d
O: 317286  65702  4087   498    41     1  1168    49 133667    27 13585     0 49653 31126  6769     5     0   883   160
X:   4722   1914  1563 98157   953   164    30  1478  5974    77  3011     0  3846  2838   330     0     0   109    47
P:     33    278    20   210    10     0     1     4   244     0   115     0   206    56    19     0     0     2     0
 ->2073 : all( 753191)  ->O( 624707/ 82.94%)  ->X( 125213/ 16.62%) ->P(   3271/  0.43%)      =>min
.......

(277) a_b_c_r_i_e_k_l_m_n_f_g_j_o_p_h_d_q_s
모듈:    a     b     c     r     i     e     k     l     m     n     f     g     j     o     p     h     d     q     s
O: 317286  65702  4643    10 18453 26437 82034 13925 78190 66351 12515     0     0  5986    43    90   245    45   270
X:   4722   1914  1007 35038   473  2247   816  1249  3928  3642  1823     0     0   172    57     8    47    25   501
P:     33    286    20    82    43    39   179    39   136   253    19     0     0    92     0     0     0     0     3
 ->2073 : all( 753191) ->O( 692225/ 91.91%) ->X(  57669/  7.66%)->P(   3297/  0.44%)


      max=96.622238%   imax=42
      min=82.941376%   imin=43
```

그림 7 모듈들의 순차처리시 실행결과의 일부 (최적교정률을 찾는 과정)

표 7 모듈순서(42)에 대한 상세설명

모듈:	a	b	c	d	...	s	0	all(753,191)
O:	317,286	65,702	4,674	2,347	...	270	0	(727,750/ 96.62%) =>**max**
X:	4,722	1,914	976	493	...	501	0	(22,144/ 2.94%)
P:	33	286	20	5	...	3	2,073	(3,297/ 0.44%)

(431,150개)에 대하여 정답(65,702개)과 오답(1,914개) 및 무응답(286개)을 처리하고 나머지 어절(363,248개)을 모듈c에게 넘겨준다. 이와 같은 방법으로 마지막모듈s까지 처리하고 19개 모듈(a,b,...,s)을 통과하여도 어느 모듈에서도 교정을 하지 못한 2,073개 어절을 출력으로 내보낸다.

표 8은 마지막 모듈까지 처리하지 못한 오류어절의 일부를 보여준다. 이들 중에는 미등록어로 인한 교정실패, 오류어절에 대한 정답이 잘못 제공된 경우, "는16대"같은 어절의 형태소분석 실패, "100만뭔"같은 철자교정 실패 등은 아직 처리하지 못하고 있다. 처리하지 못한 오류들은 모듈자체의 개선과 새로운 모듈의 개발을 통해 교정할 수 있다.

순서(42)에서 최적교정률 96.62%를 얻었다. 순서(43)는 순서(42)의 역순(모듈a, 모듈b, 모듈c를 제외한 나머지 모듈은 역순임)으로 교정률이 82.94%임을 알 수 있다. 이것은 최대 교정률에 근접한 모듈순서를 찾으면 이에 대한 역순은 최소 교정률에 접근한다는 것을 말해준다. (42) a_b_c_d_e_i_k_l_m_n_g_o_h_f_p_j_r_q_s 모듈순서로 실행시 727,750개 어절을 교정할 수 있고 727,750/753,191*100 =96.62%의 교정률을 보인다. (43) a_b_c_s_q_r_j_p_f_h_o_g_n_m_l_k_i_e_d 모듈순서로 실

표 8 철자교정기의 모든 모듈에서 처리되지 않은 오류어절 (2,073개)

1백뉴으로||1백 뉴로	는386세대도||는 386세대도	첨단교통관리스시템||첨단교통관리시스템
2ㅈ간의||2자 간의	는3사는||는 3사는	체제이데올로그에||체제이데올로기에
3천6백만워||3천6백만 워	는40만||는 40만	충남천안시광덕면매당1리||충남 천안시 광덕면 매당 1리
3천킬||3천 킬	는4차례||는 4차례	충무공ㄹ김구||충무공 김구
7월말께ㅈ||7월 말 께 ㅈ	는76||는 76	터어키어과를||터키어과를
8월말께||8월 말께	는76년||는 76년	트겅범죄가중철벌법상의||특경범죄가중처벌법상의
90%이상ㄹ||90% 이상 ㄹ	는8편으로||는 8편으로	특별수사팅늘||특별수사팀 늘
글로벌한민족코리아사의||글로벌한 민족코리아사의	는97년||는 97년	팀된공잡기수||팀 된공잡기 수
김환기데생전도||김환기데 생전도	있6어||있어	페르남부쿠쌔||페르남부쿠 州
네이버스인더스트리의||네이버스 인더스 트리의	있8다고||있다고	프로모터마르셀아브라함과||프로모터 마르셀 아브라함과
는10대팬들도||는 10대 팬들도	있었0다||있었다	플레이어스챔피언십3||플레이어스챔피언십 3
는16대||는 16대	자리고비함아||자리고 비함이	핀랜드인이다||핀란드인이다
는1만원||는 1만 원	전서초케이블||전 서초케이블	필요하없었다는||필요가 없었다는
는20개국||는 20개국	조선조말옆인||조선조 말엽인	핵탄투두우라늄||핵탄두우라늄
는22일||는 22일	조전소장외에||조 전 소장 외에	.....
는29일부터||는 29일부터	챔피엄십1||챔피언십 1	

표 9 실험결과 요약

구분	㉠이론적(4.1절, 모듈간독립,병렬처리)	㉡경험적(4.2절, 모듈간종속,순차처리)	차이 (㉡-㉠)	모듈실행순서
max측	732,764 (97.28%) :최대교정률	727,750 (96.62%) :최적교정률	-5,014	(42) a>b>c>d>e>i>k>l>m> n>g>o>h>f>p>j>r>q>s
min측	601,926 (79.92%) :최소교정률	624,707 (82.94%) :최악교정률	22,781	(43) a>b>c>s>q>r>j>p>f> h>o>g>n>m>l>k>i>e>d

행시 624,707개 어절을 교정할 수 있고, 624,707/753,191 *100= 82.94%의 교정률을 보였다.

입력어절에서 가장 짧은 어절은 "12및">"12 및"이고 가장 긴 어절은 "조사를지속적으로실시해나가기로했으며 관련자료가확보되면대주주">"조사를 지속적으로 실시해 나가기로 했으며 관련 자료가 확보되면 대주주" 이었다. 시스템은 정확하게 교정하였다.

표 9는 입력어절집합에 대한 실험결과의 요약표이다. 아직 5,014개 어절에 대해서는 교정을 못하고 있다. 이 어절들은 모듈의 순서를 계속 바꾸어가면 정확하게 교정하는 모듈순서를 찾을 수는 있지만 시간이 너무 많이 걸리는 과정이다. (최대 오류어절 교정개수) - (최적 오류어절 교정개수)= 5,014개는 전체 입력어절의 0.66% (5,014/753,191*100)정도이므로 거의 최대교정 개수에 근접하여 727,750 / 753,191 *100 = 99.31%이므로 만족할 만한 교정률을 보이고 있다. 계속 모듈순서 조정을 통해 교정률을 높일 수 있으나 많은 비용이 요구되기 때문에 어느 정도 선에서 타협하는 것이 필요하다.

5.2 실험에 사용된 경험적 지식에 대한 분석

최적교정률을 나타내는 (42) a>b>c>d>e>i>k>l>m> n>g>o>h>f>p>j>r>q>s 모듈순서열에 사용된 경험적 지식을 살펴본다. 오용어 대치, 띄어쓰기, 철자교정에 대한 경험적 규칙을 철자교정기의 모듈에 적용하였다.

5.2.1 오용어 대치교정

이들은 절대 교정법을 사용하는 것으로 이 논문의 3장에서 모듈설명처럼 오류어절에 대한 정답어절을 사전에 저장하였다가 일치되는 오류어절이 검색되면 곧바로 정답어절을 제시한다. 경험적으로 볼 때 이와 같은 오류는 다른 모듈보다 앞서서 처리하는 것이 교정률이 높다. 이런 부류에 속하는 예제로는 "노가다>막노동자", "오얏>자두", "7번만에>7번 만에", "가능성에대한>가능성에 대한", "감시하고있다>감시하고 있다", "겟>겠", "읍니>습니" 등이 속한다.

일반 오용어, 어미 오용어, 조사 오용어, 수사오용어 등이 있으며, 이들은 모듈a, 모듈b, 모듈c, 모듈d에서 처리된다. 모듈순서열에서 앞쪽에 배치되어 있다.

5.2.2 띄어 쓰기

띄어쓰기는 띄어쓸 위치를 결정하기가 중요하며, 어떤 위치가 결정되면 그 위치 좌우로 분리된 어절이 모두 옳은 어절인지 확인해야 한다. 분리된 위치의 좌우어절이 모두 옳으면 띄어쓰기가 완료된다. 띄어쓸 위치는 형태소 분석시 어절의 위치와 휴리스틱을 사용해 만들어 놓은 패턴을 비교하여 결정이 된다. 만약 띄어쓸 위치가 여러가지가 나오면 가중치가 높은 것을 먼저 처리한다.

한국어 문서에서 띄어쓰기 오류는 전체오류 중에서 38%정도 차지하며 가장 높다. 띄어쓰기를 처리하는 모듈은 모듈e, 모듈i, 모듈k, 모듈l, 모듈m, 모듈n, 모듈o, 모듈f, 모듈j이며 모듈순서열에서 중간부분에 주로 위치한다.

• 휴리스틱을 이용한 띄어쓰기

띄어쓸 가능성이 큰 패턴들을 만들어 놓고 오류어절이 발견되면 최우선으로 적용한다. 예를 들면, '를'의 뒤에는 띄어쓸 확률이 높기 때문에 무조건 띄어쓴 상태로 교정을 시작한다. 이런 부류에는 '젊'의 앞, '에'와 '대한'의 사이 등이 있다. 이런 패턴을 실마리(clue)라고 한다.

한 음절 띄어쓰기는 주의가 필요하다. 철자가 틀려서 오류가 된 어절이나 미등록어의 경우는 띄어쓰지 않은 것이 오히려 유리할 수 있다. 예를 들어 '김때중'을 '김 때 중'으로 띄어쓰기를 할 가능성이 높다. 이와 같은 문제를 방지하기 위해 띄어쓸 수 있는 한 음절을 제한하고 있다. 또한, 띄어쓸 위치의 앞뒤 형태소 분석정보를 보고 특수한 경우에만 띄어쓰도록 허용하고 있다. 예를 들면, '것'의 앞은 관형형 어미로 분석되는 형태소가 앞에 있을 때만 띄어 쓰도록 한다.

• 형태소 분석시 형태소의 위치를 이용한 띄어쓰기

형태소 분석시 옳은 어절이 되는 위치를 모두 저장해 놓고, 가중치가 높은 위치부터 띄어쓰기를 한다. 가중치는 관형형 어미 뒤, 관형사, 부사, 자주 쓰는 조사 뒤, 연결형 어미 뒤, 다른 명사와 결합이 불가능한 명사 뒤, 나머지 경우 순서로 되어있다. 그러나, 가중치가 낮은 종결형 어미 뒤, 'ㄴ'조사 뒤 등에서는 띄어쓰기를 제한하고 있다. 첫 자가 성씨인 경우에는 띄어쓰기를 하지 않는다.

5.2.3 철자 교정

철자 교정은 띄어쓰기가 실패했을 때 적용하는 부분이다. 자주 틀리는 패턴을 이용한 교정과 키보드 자판거리를 이용한 교정방법이 있다. 이에 해당하는 모듈은 모듈g, 모듈h, 모듈p, 모듈r, 모듈q, 모듈s이며 모듈순서의 뒤쪽에 주로 위치하고 있다.

• 자주 틀리는 패턴 교정

사람들이 자주 틀리는 패턴들을 준비하여 교정하는 방법이며, 특히 전혀 쓰이지 않는 음절의 경우는 무조건 교정을 하게된다. 예를 들면, 무조건 교정하는 패턴은 '됬'->'됐', '몾'->'못'와 같은 것이 있고, 자주 틀리는 패턴은 '잖'<->'찮', '률'<->'율' 등이다.

• 키보드 자판 거리를 이용한 교정

키보드 자판 거리를 고려하여 주위의 자판문자로 바꿔보는 방법이다. 적용하는 순서는 중성, 초성, 종성순이다. 예를 들어, 오류어절이 '생활환겅'이면 '겅'문자를 '경'문자로 바꾸기 위해서 'ㅓ'자판의 주위에 있는 'ㅕ', 'ㅗ', 'ㅏ' 등으로 바꾸어 처리한다.

6. 결 론

이 논문에서 철자교정기의 모듈 개수가 많아짐에 따라 생길 수 있는 문제점을 살펴보았다. 기존에 교정이 되던 오류어절이 새로운 모듈이 개발되고 추가됨으로 오류교정에 실패하는 경우가 생겨 전체 오류교정률이 점점 떨어질 수 있다. 이 논문의 목표는 새로운 교정모듈이 추가될 때 시스템의 교정률을 떨어뜨리지 않는 모듈순서를 결정하도록 하는 것이다. 이렇게 모듈순서열을 찾는 문제는 모듈 개수가 n개일 때 n!난이도이므로 정상적인 방법으로는 처리할 수 없는 계산량이다. 이 문제의 대안으로 모듈들을 병렬처리하여 철자교정기의 교정률의 범위를 구하였고, 모듈순서열을 구하여 순차처리한 후 최적교정률을 계산하였다.

병렬처리는 철자교정기의 교정률의 범위를 구하는 데 사용하였고 모듈간의 순서는 영향을 미치지 않는다. 이 교정률 범위는 최대교정률과 최소교정률로 표현되었다. 주어진 입력오류집합에 대해 각 모듈들을 병렬로 처리한 후 집합연산을 통해 최대교정률은 97.29%, 최소교정률은 79.92%를 찾았다. 철자교정기의 교정률은 모듈의 실행순서가 어떠하든지 관계없이 이 범위 안에 있음이 보장된다.

순차처리는 입력데이타에 대해 모듈순서열을 다르게 하여 실행하기 때문에 교정률이 매번 달라진다. 경험적인 방법을 이용하여 임의로 278개 모듈순서열을 만들었고, 이들을 실행한 후 찾아낸 최적교정률은 96.62%이었다. 아직 5,014개 어절이 처리되지 않은 상태로 되어있으나, 전체 교정가능한 입력어절에 대하여 0.66%정도를 차지하므로 만족할 만한 교정률이다. 최대교정율에 대한 한국어 철자교정기의 성능은 99.31%를 보였다.

본 연구에서 어려웠던 점은 모듈 개수가 N개일 때 계산량이 N!되기 때문에 각 모듈나열에 대한 결과를 저장하기 위한 공간과 계산시간이 너무 많이 요구된 것이다. 향후 연구과제는 관련있는 모듈끼리 묶어 전체 그룹개수를 줄여서 계산량을 줄이고자 한다. 이를 위해서 관련성이 있는 모듈들끼리 묶어서 같은 그룹으로 만드는 작업이 필요하다.

참 고 문 헌

[1] N.H.Macdonald, L.T.Frase, P.Gingrich, and S.A. Keenan, "The WRITER'S WORKBENCH : Computer aids for text analysis," IEEE Trans. Commun. COMM-30, No.1, pp.105-110, 1982.

[2] Stephen D. Richardson, "Enhanced Text Critiquing using a Natural Language Parser," research report RC-11332, IBM Thomas J. Watson Research Center, 1983.

[3] G.E. Heidorn, Jensen, L.A. Miller, R.J. Byrd, and M.S. Chodorow, "The EPISTLE Text-Critiquing System," IBM Syst. J. Vol 21, No 3, pp.305-326, 1982.

[4] 강재우, "접속정보를 이용한 한국어 철자 띄어쓰기 검

사기의 설계 및 구현", 한국과학기술원 전산학과 석사 학위 논문, 1990.
[5] 박종만, "효율적인 한국어 형태소분석기 및 철자 검사 교정기의 구현", 서울대학교 석사학위 논문, 1990.
[6] 심광섭, "음절 간 상호정보를 이용한 한국어 자동띄어쓰기", 정보과학회논문지(B), 23-9, 991-1000, 1996.
[7] 강승식, 장병탁, "음절특성을 이용한 범용 한국어 형태소 분석기 및 맞춤법 검사기", 정보과학회 논문지, 제23권 제5호, 1996.
[8] 김덕봉, 최기선, 강재우, "한국어 형태소와 사전-접속정보를 이용한 한글 철자 및 띄어쓰기 검사기", 언어연구, 제26권 제1호, pp.87-113, 1990.
[9] 이병훈, 윤준태, 송만석, "말뭉치를 기반으로 한 한국어 철자교정기의 구현", 한글 및 한국어 정보처리 학술발표논문집, pp.285-293, 1993.
[10] 정한민, 이근배, 이종혁, "자판특성을 이용한 Neuro-Fuzzy 한국어 철자교정기의 구현", 한글 및 한국어 정보처리 학술발표논문집, pp317-328, 1993.
[11] 이원일, 홍남희, 이종혁, 이근배. 1993. Binary n-gram과 형태소 분석기를 이용한 한국어 철자 교정기. '93 KISS 학술발표 논문집 20:1 (Apr 1993), 813-
[12] 이도길, 이상주, 임희석, 임해창, "한글 문장의 자동 띄어쓰기를 위한 두 가지 통계적 모델", 정보과학회 논문지: 소프트웨어 및 응용 제30권 제4호, pp.358-370, 2003.
[13] Chul-Min Sim, Min-Jung Kim, Hyuk-Chul Kwon, "Automatic Revision of Korean Texts by Collocation Words," Proceedings of the '94 International Conference on Computer Processing of Oriental Languages, pp.280-284, 1994.
[14] 채영숙, "연어 규칙에 기반한 한국어 문서교정시스템의 구현", 부산대학교 전자계산학과 박사학위 논문, 1998.

윤 근 수

1989년 2월 부산대학교 전산통계학과 학사. 1991년 2월 부산대학교 계산통계학과 석사. 1997년 8월 부산대학교 전자계산학과 박사수료. 1992년 3월~1999년 2월 부산경상대학 조교수. 1999년 3월~현 울산과학대학 컴퓨터정보학부 조교수 관심분야는 한국어정보처리, 패턴인식

권 혁 철

1982년 서울대학교 공과대학 전산학 학사. 1984년 서울대학교 공과대학 전산학 석사. 1987년 서울대학교 공과대학 전산학 박사. 1988년~현재 부산대학교 정보컴퓨터 공학부 교수. 1988년~현재 한국정보과학회 프로그래밍언어 연구회 운영위원. 1990년~현재 한국정보과학회 한국어정보처리 연구회 운영위원. 1992년~1993년 미국 Stanford 대학 CSLI연구소 연구원. 1992년~1993년 Xerox Palo Alto Research Center 자문위원. 2003년~현재 BK21 산업자동화 및 정보통신분야 인력양성사업단 단장. 2003년~현재 한국정보과학회 한국어정보처리 연구회 위원장. 관심분야는 한국어정보처리, 정보검색, 프로그래밍언어, 인공지능