# 문서 서식 보존이 가능한 맞춤법 검사 시스템 개발

정택원, 최정용, 남민지, 최정열*
성결대학교 컴퓨터공학과

{chtw2001, aj4941, nam0221, passjay*}@sungkyul.ac.kr

# Development of Spell Checking System that Preserving Document Formatting

Chung Taek Won, Choi Jeong Yong, Nam Min Ji, Choi Jung Yul*
Department of Computer Engineering, Sungkyul Univ.

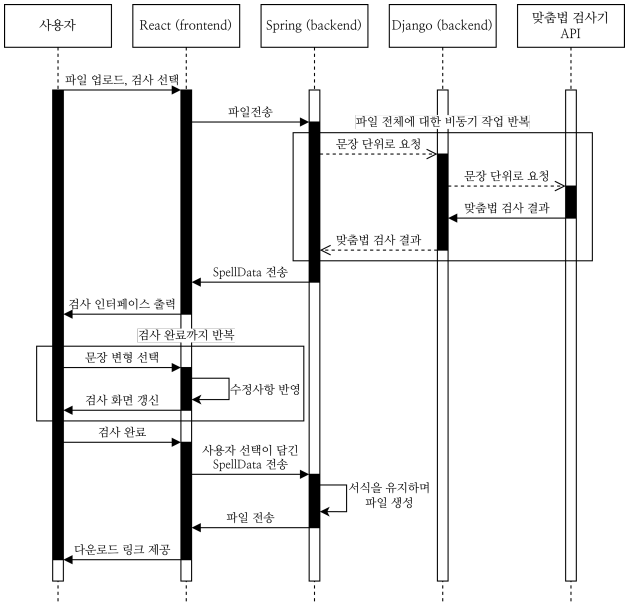
요 약

맞춤법 검사를 수행한 이후에 원본 문서를 수정하면 글꼴, 글씨, 들여쓰기 등의 원본 서식이 망가지는 문제가 발생한다. 본 연구는 원본 문서의 서식을 보존하면서 전체 문서에 대한 맞춤법 검사를 수행할 수 있는 시스템을 제안한다. 제안하는 시스템은 문서 파일의 XML 구조를 활용하여 문법적 오류가 있는 텍스트만 올바른 텍스트로 치환하고, 나머지 서식은 그대로 유지함으로써 서식을 보존하며 맞춤법 검사를 진행할 수 있게 한다. 또한 텍스트, 표, 머리글, 바닥글, 미주, 각주 등의 다양한 구성 요소에 대한 서식을 유지하며, 문서의 재귀적 구조에 알맞은 프로그래밍을 통해 효율적인 처리 방식을 도입한다.

## Ⅰ. 서 론

보고서 작성의 마지막 단계인 검토는 작성된 문서에서 논리적 오류, 문법적 오류 등을 찾아내어 수정하는 중요한 과정이다. 현재 시중에서 주로 사용되는 맞춤법 검사기(부산대 맞춤법, 인크루트 맞춤법 검사기 등)를 사용하면 입력한 문장에 대해 문법적으로 발생한 오류를 손쉽게 찾아낼 수 있다. 그러나 맞춤법 검사 결과를 문서에 적용할 때는 몇 가지 문제가 발생한다. 첫째, 글꼴, 글씨 크기 등의 서식이 갖추어진 문서에 맞춤법 검사를 수행하면 기존의 서식이 깨지게 된다. 이는 맞춤법 검사기의 출력물 형태가 서식을 전혀 고려하지 않은 단순 텍스트이기 때문이다. 둘째, 대부분의 맞춤법 검사기는 제한된 글자 수에 대해서만 맞춤법 검사를 제공한다. 따라서, 긴 문장이나 전체 문서에 대한 맞춤법 검사를 수행하려면 문서의 서식을 재조정하는 비효율적인 과정을 여러 번에 걸쳐 반복적으로 진행해야 한다.

이러한 문제를 해결하기 위해 본 연구에서는 Apache POI 라이브러리[1],[2]와 hwplib 라이브러리[3]를 분석하여 XML 구조로 이루어진 Microsoft Word 파일과 한글(HWP) 파일에 대해 서식을 유지하면서 올바른 텍스트로 치환할 수 있는 XML 문서 통합 자료구조인 SpellData를 개발하였다. 또한, React, Spring, Django 프레임워크를 활용하여 문서의 서식을 유지한 채로 맞춤법 검사를 한 번에 수행할 수 있는 시스템을 개발하였다. 개발한 시스템은 사용자가 선호하는 맞춤법 검사 API[4],[5]를 선택할 수 있으며, 사용자 친화적인 인터페이스를 제공하여 효율적인 문서 검토가 가능하도록 설계하였다.




<그림 1> 제안하는 시스템의 시퀀스 다이어그램

## Ⅱ. 시스템 구성

제안하는 시스템은 맞춤법 검사 기능을 제공하는 웹 기반 애플리케이션으로, React 프레임워크를 사용한 인터페이스와 Spring 및 Django 프레임워크를 사용한 서버로 구성된다. 사용자가 파일을 업로드하고 맞춤법 검사 방식을 선택하면, React 서버가 파일을 Spring 서버로 전달한다. Spring 서버는 파일을 문장 단위로 분할하여 Django 서버로 비동기 방식을 이용해 전송하며, Django 서버 또한 비동기 통신을 사용하여 맞춤법 검사 API 에 검사 요청을 보낸다. 맞춤법 검사 결과는 Spring 서버에서 SpellData 자료구조로 변환되어 React 서버로 다시 전달된다. 사용자는 React 서버가 제공하는 인터페이스를 통해 수정 사항을 확인하고 원하는 부분에

대한 수정을 완료하면, 서식이 유지되며 맞춤법 검사가 완료된 파일을 다운로드 받을 수 있다.

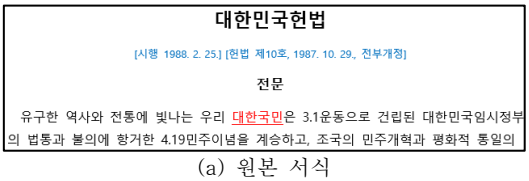
## III. XML 문서 통합 자료구조 개발

XML 구조로 이루어진 문서를 통합하는 자료구조를 개발하기 위해 Apache POI 라이브러리와 hwplib 라이브러리를 활용하여 본문(텍스트, 표 등), 머리글, 바닥글, 미주 및 각주와 같은 다양한 구성 요소로 문서를 분해한다. 각 요소를 분석한 결과, 본문의 텍스트 내부에 표, 미주, 각주가 포함될 수 있었으며 본문뿐만 아니라 표, 머리글, 바닥글, 미주, 각주에는 다시 본문이 포함될 수 있는 재귀적인 구조를 갖고있었다. 이를 기반으로 문서를 분해하고 각 요소를 식별하여 서식에 대한 정보를 효율적으로 저장할 수 있는 SpellData 자료구조를 개발하였다.

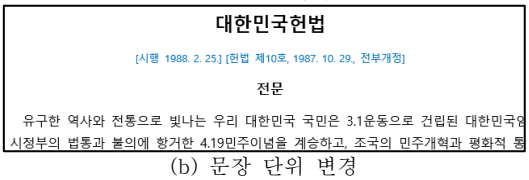
## IV. 문서 분해·결합 동작 과정

모든 구성 요소를 놓치지 않고 분해하기 위해서는 문서 전체에 대한 탐색이 필수적이다. 문서는 본문(텍스트, 표 등), 머리글, 바닥글, 미주, 각주로 분해되며, 각 구성 요소의 형태를 저장한 채로 하나의 문장으로 추출된다. 추출된 문장은 맞춤법 검사기 API 를 통해 검사된 후, 그 결과는 서식 정보를 포함한 SpellData 자료구조에 통합된다. 이 과정에서 문서 내의 요소들은 작성된 순서에 맞게 처리되며, 이후 서식을 유지한 채 텍스트를 치환할 수 있게 한다.

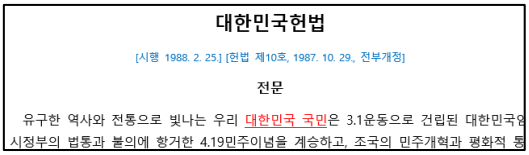
맞춤법 검사 결과를 적용할 때 기존 맞춤법 검사기는 단순 텍스트를 기반으로 하기 때문에 붙여 넣는 위치의 서식으로 통일되는 문제가 있다. 이를 해결하기 위해 제안된 시스템은 XML 구조를 인식하여 문서의 서식과 텍스트를 함께 처리한다. 변경된 부분을 문장 단위로 검사하고 변경 사항을 반영하는 것이 아닌, 문장 내에서 변화하는 부분만 교체하여 서식의 파괴를 최소화할 수 있었다. 그러나 <그림 2>(a)의 원본 서식에 대해 맞춤법 검사를 수행하면 <그림 2>(b)와 같이 여전히 문자열이 통째로 치환되며 해당 문자열의 서식이 획일화되는 문제가 남아있어, 이를 개선하기 위해 편집거리 알고리즘을 도입하였다. 서식을 검사 단위가 아닌 문자 단위로 인식하고, 기존 문자열과 대체 문자열의 변경점을 파악하여 문자에 대한 삽입, 삭제, 대체 작업을 수행하도록 한다. <그림 2>(c)를 보면 알 수 있듯, 이를 통해 서식을 보존하며 맞춤법을 교정할 수 있다. 이러한 순차적인 분해와 처리를 통해 문서 내의 모든 구성 요소에 대해 효율적이고 일관된 맞춤법 검사를 수행할 수 있다.




(a) 원본 서식




(b) 문장 단위 변경

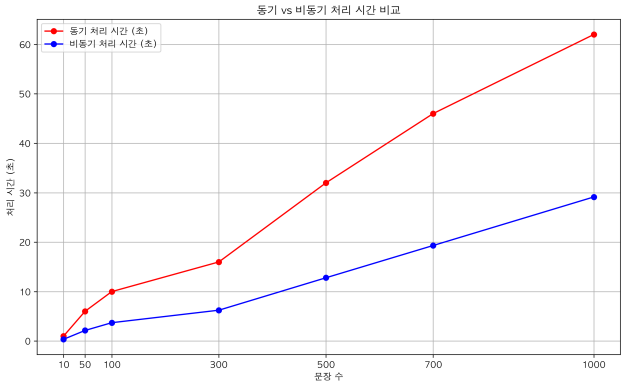



(c) 편집거리 적용

<그림 2> 검사 단위별 문서 분해·결합 동작 과정

## V. 비동기 요청 및 요청 속도 제어를 통한 성능 최적화

Spring 서버에서 Django 서버로의 맞춤법 검사 요청은 기본적으로 동기적으로 처리되며, 한 문장당 약 1초가 소요된다. 이로 인해 문서의 크기가 커질수록 검사 시간이 급격히 늘어나는 문제가 발생하여, Java의 비동기 프로그래밍 인터페이스인 CompletableFuture를 사용한 비동기 요청과 Django 서버의 비동기 함수 구현으로 약 70%의 속도 향상을 달성하였다. 그러나 다수의 비동기 요청이 Django 서버에 과부하를 유발해 타임아웃 에러가 발생하였다. 이를 해결하기 위해 토큰 버킷 알고리즘을 적용하여 제한된 수의 토큰 내에서만 비동기 처리가 가능하게 하였고, 토큰이 소진되면 1초간 대기하도록 구성하였다. 테스트 결과, 토큰 5개 까지는 성능이 향상되었으나 20개를 넘어서면 기존과 동일하게 요청 소실이 발생하였다. 최종적으로, 요청 소실률 0에 수렴하면서 64.44%의 속도 향상을 달성하였다.




<그림 3> 비동기 처리, 토큰 버킷 알고리즘 적용 결과

## VI. 결론

본 논문에서는 기존 맞춤법 검사에서 발생하는 서식 손상 문제를 분석하고, 서식을 보존하며 맞춤법 검사를 수행하는 방안을 제안하였다. 이를 위해 Apache POI와 hwplib 라이브러리를 분석해 서식을 유지하는 방식을 도입하고, 비동기 처리 및 토큰 버킷 알고리즘으로 성능을 최적화하여 약 64.44%의 속도 향상을 달성하였다. 또한 편집거리 알고리즘을 통해 서식 손상을 최소화하였다. 향후에는 자체 맞춤법 검사기 모델 개발을 통한 문법 검사 기능 확장, 다양한 파일 형식 지원을 추가하여 서비스의 범위를 넓히고 개선할 계획이다.

## 참 고 문 헌

[1] Apache POI 라이브러리 코드 https://github.com/apache/poi

[2] Apache POI 라이브러리 공식 문서 https://poi.apache.org/

[3] hwplib 라이브러리 코드 https://github.com/neolord0/hwplib

[4] 인크루트 맞춤법 검사기 https://lab.incruit.com/editor/

[5] 부산대 맞춤법 검사기 https://nara-speller.co.kr/speller/