SNS 텍스트 마이닝을 위한 웹문서 인코딩 자동 인식 기술 방안

모은수*, 이재필*, 이재광*, 이준현*, 이재광**
*한남대학교 컴퓨터공학과
e-mail : {esmo*, jplee*, jglee*, jhlee*}@netwk.hnu.kr
jklee@hnu.kr**

A method of web Document Encoding Automatic Recognition for SNS Text Mining

Eun-Su Mo*, Jae-Pil Lee*, Jae-Gwang Lee*, Jun-hyeon Lee*, Jae-Kwang Lee**
*Dept. of Computer Engineering, Hannam University

요 약

사용자는 자신의 주변상황에 대한 정보를 수집 및 공유하기 위하여 SNS, 포탈사이트 및 커뮤니티를 사용한다. 본 논문에서는 사용자의 특성을 고려한 지역정보 수집 아이디어와 방법론을 제시한다. 또한 각각의 웹 시스템의 데이터를 수집하여, 광범위한 지역정보를 마이닝을 수행하고 가공해내는 시스템을 제안한다. 이를 위해 해결해야하는 이슈는 다음과 같다. 각 웹시스템의 문서들은 운영체제에 따라 인코딩이 달리 사용되는데, 흔히 발생되는 오류 중 하나인 문자깨짐 현상이 그 예이다. 해결방법으로써 문서가 작성된 운영체제의 인코딩정보를 획득해야하며, 이 정보는 서버에서 제공하는 헤더정보에 명시되었거나 문서내에 내장되어 있다. 하지만 일부 웹사이트는 인코딩 정보를 제공하지 않으며, 국가별 인코딩이 다르기 때문에 이를 알기 쉽지않다. 그리하여 본 논문에서 제안하는 방법론은 텍스트 마이닝에 앞서 웹서버에서 제공하는 웹페이지를 읽어들여 인코딩정보를 획득하고, 문자의 깨짐없이 표시할 수 있도록 시스템을 구축하기 위해 Response Header, HTML 의 meta tag 및 읽어드린 문서의 BOM(Byte Order Mark) 정보 및 인코딩 패턴을 통해 인식하도록 하여 글자 깨짐을 완하하도록 시스템을 설계하였다.

1. 서론

포탈사이트 등을 이용하여 독감이나 전염병 등의 정보를 검색 유입량의 증가와 지역으로 판단하는 연구가 진행되고있다[1][2]. 사용자는 자신의 주변상황을 알기 위함은 물론 자신의 생각과 현재 상황을 포탈사이트를 통해 검색하거나 SNS, 블로그, 커뮤니티 사이트등을 통해 공유하며 이러한 정보는 실시간으로 전파되는 특성을 가지고 있다.

급변하는 공간정보를 정확하게 파악하는 방법은 소수의 전문가 보다는 지역에 거주하는 다수이며 특히 SNS 를 사용하는 사용자는 주변상황의 정보와 Hash tag, 위치정보를 포함하며 이 정보는 위치 기반 시스템 최적의 조건을 가지고있다[3]. 트위터, 페이스북, 인스타그램 등의 SNS(소셜네트워크서비스)를 사용하는 사용자수가 증가함에 따라 소셜데이터 마이닝 기술이 주목받고 있다[4]. 하지만 이러한 사이트 기반 정보는 각 국가별 및 시스템 인코딩이 달라 텍스트 데이터 수집에 걸림돌로 작용된다.

즉 인코딩이 달라 문자의 깨짐 현상이 발생하며 키워드 확보가 안되는 문제가 발생하기 때문이다.

텍스트 마이닝은 기본적으로 문자를 인식하여야 하지만 현존하는 인코딩 체계는 국가별, 운용시스템에 때라 다르다. 이는 초기 인코딩이 영어권 국가를 기준으로 만들어졌기 때문이며, 이후 다른 국가를 위해 지속적으로 연구되어 유니코드 체계의 다국어를 지원하는 인코딩이 등장했다[5][6][7].

브라우저에서는 response header 의 인코딩을 읽거나 자동으로 인식하며 렌더링 작업을 수행하여 브라우저에 표시한다. response header 의 인코딩 정보가 없더라도 html meta tag 의 인코딩을 인식하고 그에 맞춰 렌더링을 다시 수행한다. 하지만 헤더 및 meta tag 에 인코딩 정보가 없을경우 자동으로 렌더링을 하지만 글자가 깨지는 일이 일부 발생한다.

최근 들어 이러한 문자 깨짐을 방지하기 위해 UTF-8(Uniformed Transformation Format - 8)을 기본으로 사용하는 사이트들이 많아지고 html 의 meta tag 를 이용 페이지 인코딩을 선언하여 이를 방지하고 있지만 데이터 수집을 위한 크롤링 작업 수행 시 이러한 인코딩을 고려하지 않아 문자가 깨지는 경우가 생기게 된다. 한국의 경우 euc-kr(Extended Unix Korea Code), cp949(Code Page 949)등의 완성형 인코딩과, Unicode 계

열의 확장형 인코딩을 사용하며, 완성형 인코딩의 경우 확장형 인코딩에서 문자가 깨지는 단점을 가진다. 네이버를 예로 들면 메인 등의 페이지들은 UTF-8 이지만 블로그는 MS949 인코딩으로 인해 읽어들인 문자가 깨지는 경우가 발생하며 이러한 글자 깨짐은 국내 사이트 뿐만이 아닌 해외사이트 자료를 수집할때도 동일한 현상을 가지기 때문에 이를 개선하기 위해서는 해당 페이지의 인코딩을 인식하며, 그에 맞춰서 읽어 들여야한다.

본 논문에서는 텍스트 마이닝을 위해 웹 데이터 수집시 발생하는 문자깨짐 방지하기 위해 데이터를 수집전에 인코딩 정보를 추출하여 글자의 깨짐없이 키워드를 읽어드릴 수 있도록 시스템을 설계하였다. 2 장에서는 관련연구를 소개하고 3 장에서는 웹 문서의 인코딩 인식 시스템을 설계하며. 4 장에서는 인코딩 인식 및 키워드 검출 테스트를 진행하였다. 5 장에서는 결론과 향후연구를 제시한다.

2. 관련연구

2.1 검색어를 통한 유행성 독감 감지 및 예측 시스템

사용자가 건강 정보 및 전염병등의 이슈를 인터넷을 통해 검색하여 정보를 얻는데 아이디어를 착안하여 전염병 확산을 예측하기 위해 구글 트랜드의 검색 동향정보와 질병관리본부의 정보 등을 수집한 후 회귀분석, 상관분석 등을 이용하여 전염병을 예측하는 연구이다[1].

2.2 문자깨짐 방지연구

문자 깨진 방지를 위한 서버 상에서의 인코딩 자동인식 적용방법 연구에서는 서버와 클라이언트간의 인코딩이 달라 브라우져에서 지원하는 인코딩 자동인식 기능이 멀티바이트 문자로 이루어진 경우 오작동 하여 문자가 깨질 수 있어 FEAD(Fast Encoding Auto Detector)를 설계하였다. 이는 파일의 BOM 정보와 BOM 이 없는 경우를 체크하여 인코딩을 찾도록한 연구이다[8].

2.3 모질라 인코딩 탐지 프로젝트

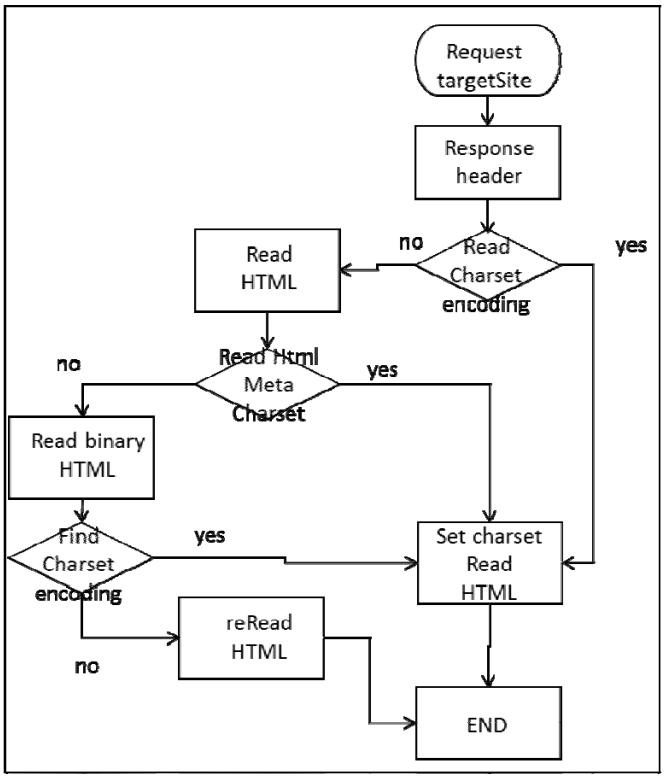
모질라에서 인코딩이 없는 페이지의 인코딩을 찾기위해 시작한 프로젝트로 각 언어별 패턴을 인식하여 인코딩을 찾는 프로젝트이다[9].

3. 웹 문서의 인코딩 인식 시스템 설계

제안하는 인코딩 인식 시나리오는 3 단계로 구성되며 [그림 1]과 같다. 1. response header 의 content-type 정보의 charset 유무, 2. HTML 의 meta tag 로 지정된 charset 유무, 3. HTML 을 byteStream 으로 읽어 BOM 을 이용한 charset 정보를 인식하는 시나리오이다.

response header 를 이용할 경우는 제공하지 않는 경우보다 문서를 다시 읽어드리는 작업이 없어 그 속도가 빠르지만 제공하지 않는 경우가 존재한다. 없는 경우 2 번 시나리오로 돌입하며 html 안의 meta tag 를 조회하며 charset 을 찾게 된다. meta tag 는 일반적으로 html 의 header 안에 정의 하거나 첫줄에 정의하여 header 이후 즉 body 부분은 찾지 않도록 하였다.

하지만 이마저도 지정되지 않을 경우가 존재 하며 이경우 3 번 시나리오로 돌입한다. 3 번 시나리오는 웹 문서를 byteStream 으로 읽어 모질라 인코딩 탐지 프로젝트(Mozilla Charset Detection)를 적용하여 인코딩 정보를 찾도록 하였다. 2, 3 번 시나리오는 페이지를 2 번 이상 호출하여 속도측면에서 다소 떨어지는 면이 있지만 인코딩 인식률을 개선할 수 있었다.



[그림 1] 인코딩 인식 시나리오

4. 재난 키워드 인식 테스트

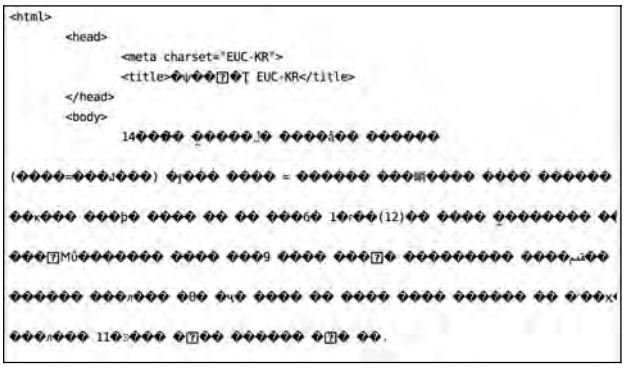
재난재해 키워드 인식을 위해 재난 키워드를 사용하여 한국과 주변국인 일본, 중국의 재난 데이터를 샘플링 하였고, 테스트를 위해 구성한 서버에 국가별 웹 지원 인코딩 별로 샘플페이지를 구성 하였다.

항목	내용
OS	Ubuntu server 14.04.2 LTS
CPU	Intel(R) Core(TM) i5-4570
MEMORY	4G
JDK	Jdk 1.8

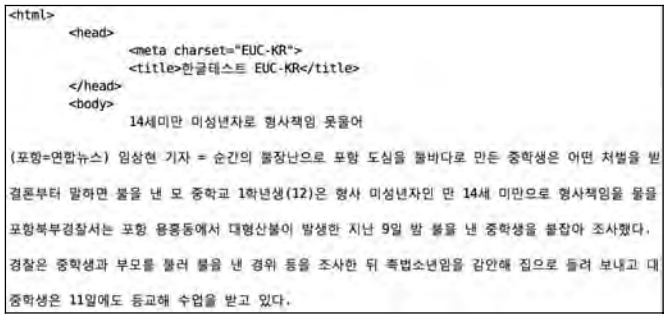
[표 1] 서버사양

서버 및 프로그램 인코딩은 UTF-8 로 기본 설정하였고 시스템 구성은 [표 1]과 같다. 인코딩 인식 시스템을 적용하지 않을 경우 [그림 2]과 같이 EUC-KR

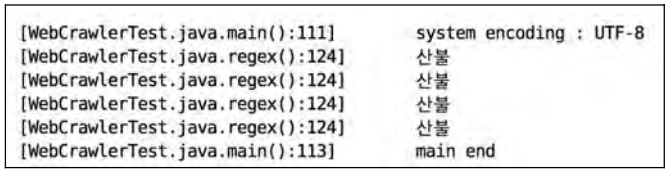
등의 완성형 인코딩은 UTF-8 에서 문자가 깨짐이 발생하여 문자가 깨진다. 하지만 인코딩 자동인식 기술을 적용하면 [그림 3]과 같이 깨짐 없이 읽어드리는걸 확인하였고 이중 재난 키워드를 정규 표현식을 사용하여 검출 하였다 [그림 4]. 테스트를 위해 response header 의 인코딩, html 의 인코딩 을 적용, 미 적용 하여 각각 테스트 하였으며 국가별 인코딩은 3 종류를 사용하였다. 각 인코딩 별로 9 번의 테스트가 진행되어 총 81 회의 테스트가 진행되었다. 1, 2 의 시나리오는 글자의 깨짐 현상이 발생하지 않았지만, 3 번 시나리오의 경우 인코딩은 찾았지만 글자가 깨지는 현상이 14%로 시스템의 인식률 인식률을 86%를 보였다.



[그림 2] 인코딩인식 시스템 적용 전



[그림 3] 인코딩인식 시스템 적용 후



[그림 4] 산불 키워드 인식

5. 결론

사용자는 자신의 주변상황을 알기 위함은 물론 자신의 생각과 현재 상황을 포탈사이트를 통해 검색하거나 커뮤니티, SNS 를 이용하여 전파한다. 사용자의 정보는 실시간으로 전파되는 특성을 가진다. 이러한 특성을 이용하여 데이터 마이닝을 위해 데이터 수집이 필수 이다. 웹 기반 사이트는 국가별, 운영체제 별로 인코딩이 달라 문자가 깨지는 현상이 발생한다. 일반적으로 SNS 마이닝은 특정 사이트 기반으로 수집하며 해당 인코딩 처리만 하였지만 본 논문에서는 재난재해 탐지를 위해 포탈 사이트 등의 웹 사이트기반의 텍스트 마이닝을 위해 문자 깨짐 없이 수집할 수 있도록 header 정보 및 html 인코딩, 문서 인코딩을 인식하는 3 단계 인식 시나리오를 적용하여 시스템을 설계하고 적용하였다. 그 결과 포탈 사이트 및 커뮤니티 사이트, SNS 의 웹 문서를 깨짐없이 읽어드릴수 있었으며 문서내 키워드를 인식함을 확인할 수 있었다. 향후 연구로는 테스트 국가를 좀더 확장하며, 웹 문서의 인코딩 인식률을 개선하여 단일 시나리오로 글자의 깨짐을 방지 할 수 있는 연구가 진행되어야 하겠다.

이 논문은 2014 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (2014R1A1A2055522)

참고문헌

[1] 박정은, “검색어를 통한 유행성 독감 감지 및 예측 시스템”, 한남대학교석사논문, 2010

[2] 이중기, 김성후, 김창수, “인터넷 포털사이트의 서비스를 활용한 재난정보시스템의 구현”, 한국정보통신학회, 한국정보통신학회논문지, 2014.4, 745-751(7 pages)

[3] Eun-Su Mo, Jae-Pil Lee, Jae-Gwang Lee, Jun-Hyeon Lee, Young-Hyuk Kim, Jae-Kwang Lee, "Design of Disaster Collection and Analysis System using Crowd Sensing and Beacon based on Hadoop Framework", Lecture Notes in Computer Science, 2015, 106-116(11 pages)

[4] 박우진, 유기윤, “위치기반 소셜 미디어 데이터의 텍스트 마이닝 기반 공간적 클러스터링 분석연구”, 한국지형공간정보학회, 한국지형공간정보학회지, 2015.6, 89-96(8 pages)

[5] 정의현, “차세대 웹 환경에서의 다국어 식별자 기술 동향”, 한국통신학회, 한국통신학회지, 2006.11, 90-100(11 pages)

[6] NAVER, 한글 인코딩의 이해, http://d2.naver.com/helloworld/76650

[7] NAVER Nuli, 문자집합(Character Set), http://nuli.navercorp.com/sharing/blog/post/1079940

[8] 강희복, 장창수, “서버에서의 인코딩 자동 인식을 통한 한글 깨짐 방지”, 한국정보기술학회, 한국정보기술학회논문지, 2014.10, 201-209(9 pages)

[9] Mozila, A composite approach to language/encoding detection, http://www-archive.mozilla.org/projects/intl/UniversalCharsetDetection.html