Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.3, Jun. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.3.595

의미적 유사성의 효과적 탐지를 위한 데이터 전처리 연구*

강 하 리,† 정 두 원, 이 상 진‡
고려대학교 정보보호대학원

A Study on Preprocessing Method for Effective Semantic-based Similarity Measures using Approximate Matching Algorithm*

Hari Kang,† Doowon Jeong, Sangjin Lee‡
Graduate School of Information Security, Korea University

요 약

디지털 포렌식 분야가 직면한 과제 중 하나는 대량의 데이터를 어떻게 효율적으로 처리할 것인가이다. 디지털 객체 간의 유사성을 빠르게 식별하기 위해 신뢰성 있는 다양한 근사 매칭 알고리즘이 계속하여 제시되어왔다. 하지만 알고리즘만으로 문자열의 의미적 유사성을 식별하면 많은 오탐을 보여 오히려 그 실효성을 끌어내리고 있다. 이와 같은 문제점을 해결하고자 근사 매칭 대상의 전처리 과정을 추가하여, 알고리즘 자체의 신뢰성은 유지하면서 유사도 탐지 정확성을 더 높일 수 있는 방법을 제시한다. 본 논문에서는 의미적 유사성을 식별하고자 eml과 hwp 세트를 가지고 sdhash로 실험하였으며, 실험 결과를 이용하여 그 효과성을 검증한다.

ABSTRACT

One of the challenges of the digital forensics is how to handle certain amounts of data efficiently. Although reliable and various approximate matching algorithms have been presented to quickly identify similarities between digital objects, its practical effectiveness to identify the semantic similarity is low because of frequent false positives. To solve this problem, we suggest adding a pre-processing of the approximate matching target dataset to increase matching accuracy while maintaining the reliability of the approximate matching algorithm. To verify the effectiveness, we experimented with two datasets of eml and hwp using sdhash in order to identify the semantic similarity.

Keywords: Approximate Matching, Semantic similarity, Digital forensics

I. 서 론

디지털 포렌식 분야에서 해시는 무결성 검증 외에도 악성코드 탐지나 알려진 파일을 필터링하는 등 다양하게 사용된다. 해시는 1bit의 데이터만 바뀌어도 해시값이 달라지므로 서로 다른 두 파일의 동일성을 검증할 때 매우 유용하다. 하지만 저작권 분쟁이나 기술유출과 같이 원본과 유사한 파일을 찾고자 할 때에는 사용할 수 없다. 이러한 한계점을 극복하기 위해 최근에는 근사 매칭(approximate matching)을 활용하고 있다. 근사 매칭은 파일, 미디어, 스트림 등, 두 개 이상의 디지털 객체 간의 유사성을 식별하는 기술이다. 이를 이용해 기존에 알려진 악성코드를 변조한 악성 프로그램을 탐지하거나, 문서 파일 및 소스코드 등 디지털 데이터의 무단 도용이나 복제 여부 판단 등 실무적으로 활용할 수 있는 실용 방법들이 연구

접수일(2015년 4월 7일), 게재확정일(2015년 6월 4일)
* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었습니다. (IITP-2015-H8501-15-1003)
† 주저자, imstrong@korea.ac.kr
‡ 교신저자, sangjin@korea.ac.kr(Corresponding author)

되고 있다.

다양한 선행 연구로 근사 매칭 알고리즘의 정확도는 신뢰할만한 수준을 보이고 있으나, 실무 환경에서의 활용도는 높지 않다. 근사 매칭의 궁극적인 목적은 비슷한 콘텐츠를 가진 파일들을 탐색하는 것이나 현재의 알고리즘들은 파일의 바이너리 값 그 자체를 비교하고 있어, 콘텐츠 상 관련이 없는 데이터끼리도 유사한 것으로 판단하는 경우가 많기 때문이다. 이는 알고리즘이 파일의 콘텐츠와 메타데이터를 구분하지 않고, 파일 포맷에 비 종속적이기 때문이다. 이러한 부정확함은 조사자가 유사도 측정 결과를 한 번 더 확인하게 하는 번거로움을 주어 결과적으로 근사 매칭 활용도의 저하를 초래하게 된다.

이에 본 논문에서는 유사성을 측정하기 전 파일 포맷에 따라 콘텐츠와 메타데이터를 분리하는 전처리과정이 유사도 측정 결과에 미치는 영향을 분석하여 효율적인 근사 매칭 활용 방안을 제시한다.

II. 관련 연구

2.1 근사 매칭 알고리즘

미국 국립표준기술연구소 (national institute of standards and technology) 에서는 유사성 식별 방법으로 의미적 (semantic), 구문적 (syntactic), 바이트 단위 (bytewise) 매칭, 세 가지로 크게 분류하여 용어를 정의하고 있다[1].

의미적 유사성 식별은 사람의 눈으로 인지할 수 있는 수준에서 이루어진다. 문서상의 내용, 사진이나 비디오의 유사성을 식별하거나, 얼굴을 인식하는 알고리즘을 예로 들 수 있다.

구문적 유사성 식별은 사람이 아닌 컴퓨터가 인지할 수 있는 수준으로, 디지털 파일의 내부 구조를 비교한다. 겉으로는 같은 그림의 파일 유사성을 식별할 때를 예로 들어, A 파일은 일반적인 그림 파일, B 파일은 스테가노그래피(steganography) 기법을 이용하여 다른 파일이 은닉되어 있다고 가정한다. 사람의 눈으로 A와 B를 비교할 때에는 동일하다고 판단하며, 이를 의미상으로 동일하다고 한다. 이와 달리 컴퓨터는 구문적 유사성 식별을 통해 두 파일이 동일하지 않다고 식별한다.

바이트 단위 매칭(bytewise matching)은 각 데이터를 비트 스트림 수준에서 유사성을 식별한다. 바이트 단위 매칭이 구문적 매칭과 큰 의미에서 비슷하게 보인다. 하지만 바이트 단위 매칭은 데이터 스트림의 구조적 분석은 고려하지 않으며, 데이터를 구성하는 바이트의 순서만을 참조한다. 대표적인 알고리즘으로는 2006년에 발표한 ssdeep(Kornblum[2])과 2010년에 발표한 sdhash(Roussev[3][4])가 있다.

ssdeep은 퍼지 해시(fuzzy hash)라고도 불리는 CTPH(context triggered piecewise hashing) 개념을 이용한다. 데이터를 일정 크기로 나눠서 생긴 각 청크들을 해시하고, 그 해시값들을 연결해서 유사성 다이제스트를 출력한다. 유사성 다이제스트는 그 데이터만의 지문이라고 할 수 있다. 최종적으로는 데이터들의 유사성 다이제스트를 비교하여, 얼마나 공통적인 청크를 가지는지 0점~100점의 유사도 점수로 수치화하여 나타낸다. ssdeep이 데이터 자체를 모두 조각으로 나눈다면, sdhash에서는 통계적으로 그 데이터에서 특별하다고 판단되는 feature들을 골라낸다. 각 feature는 5개의 서브 해시로 나뉘도록 해시하여 연결하고, base 64로 변환하여 sdbf (similarity digest bloom filter)라는 확장자의 유사성 다이제스트로 출력된다. sdhash도 각 데이터의 고유한 유사성 다이제스트인 .sdbf를 비교하고, 각 데이터를 구성하는 feature들이 서로 얼마나 겹치는지 연산하여 0점~100점의 유사도 점수로 나타낸다.

2.2 sdhash 유사도 점수화

sdhash v3.4[4]에서 기본 옵션으로 사용했을 때, 하나의 bloom filter는 256byte이며, 이는 최대 160개의 feature로 구성된다. 하나의 feature는 SHA-1을 통해 160bit의 해시값을 출력한다. 이를 32bit씩의 5개의 서브해시로 나누고, 각 서브해시의 최하위 11bit씩을 bloom filter의 주소 비트로 이용한다. 최대 160개 feature의 서브해시들이 가리키는 주소 비트의 값을 1로 바꾸고, 이 2048bit 길이의 비트스트림이 하나의 bloom filter가 되는 것이다. Table 1.은 이를 예시로 나타낸 것이다.

이와 같이 생성된 bloom filter들을 모아 BASE 64로 인코딩한 스트림을 그 파일의 유사성 다이제스트(*SD*) 또는 fingerprint라 한다. 그리고 두 파일의 유사성 다이제스트를 비교하여 유사도를 수치화한 것이 유사성 점수이다. 유사성 점수는 두 유사성 다이제스트의 bloom filter들을 가지고, 1bit 값을

Table 1. The process of generating a bloom filter on sdhash

① Bloom Filter Initializing

bit add.	0	1	2	3	4	5	6	…	2041	2042	2043	2044	2045	2046	2047
value	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0

② Generating Bloom Filter

feature 1					…	feature 159	feature 160 (max)
sub hash 1	sub hash 2	sub hash 3	sub hash 4	sub hash 5			
$00000000001_{(2)}$	$01000101001_{(2)}$	$11111111110_{(2)}$	$11111111011_{(2)}$	$10001001001_{(2)}$			
$1_{(10)}$	$553_{(10)}$	$2046_{(10)}$	$2043_{(10)}$	$1097_{(10)}$			

⇓

bit add.	0	***1***	2	…	***553***	…	***1097***	…	2041	2042	***2043***	2044	2045	***2046***	2047
value	0	*1*	0	…	*1*	…	*1*	…	0	0	*1*	0	0	*1*	0

갖는 주소 비트가 서로 얼마나 겹치는지를 계산하여 도출한다.

s가 파일 1의 bloom filter 개수($f_1^1, \ldots, f_s^1$)이고 t가 파일 2의 bloom filter개수 ($f_1^2, \ldots, f_t^2$)라고 할 때, sdhash는 s × t 회만큼 bloom filter 비교 작업을 한다. Table 2.와 Table 3.은 이 비교 작업과 점수화 하는 과정을 표로 나타낸 것이고, 이때 두 파일의 최종 유사성 점수는 (1)과 같다.

$$SD_{score}(SD_1, SD_2) = \frac{1}{s}\sum_{i=1}^{s} \max_{1 \le j \le t} BF_{score}(f_i^1, f_j^2) \quad (1)$$

Roussev의 sdhash 점수 해석가이드[4]에 따르면, 일반적으로 21점~100점은 신뢰성이 높은 구간, 11점~20점은 파일 형식에 따라 오탐이 존재하는 구간, 1점~10점은 신뢰성이 매우 낮고 대부분이 flase-positive인 구간, 1점 미만은 파일 1과 파일 2가 공통적으로 가지는 부분이 전체 파일 크기에 비해서 너무 작아 1% 이상으로 표현할 수 없거나, 불일치하는 결과이다.

sdhash에서 0점을 결정하기 위한 임계치 계산과 BF_{score} 함수에 대한 상세 내용은 본 논문에서 생략하도록 한다.

2.3 Approximate Hash Based Matching

근사 매칭은 목적에 따라 의미적, 구문적, 바이트 단위 매칭을 응용할 수 있다. 근사 매칭의 실용 사례를 보면 얼굴이나 지문 등 바이오정보 인식, 악성코드 탐지, 저작권 관련 조사 등 여러 분야에 적용되고 있다. 더불어 최근 Petter Bjelland 외 2인은 근사 매칭을 디지털 포렌식에 활용하기 위한 방안으로 AHBM(Approximate Hash Based Matching)을 제시하였다[6].

AHBM은 sdhash를 이용하며, sdhash는 파일의 1-대-1, 1-대-다, 다-대-다 유사성 비교를 지원한다. AHBM은 이러한 비교 환경에 따라 Search, Streaming, Clustering 세 가지 모드로 구분한다.

Table 2. Comparing between bloom filters of two files on sdhash

① f_1^1 : A bloom filter of the similarity digest of File 1

bit add.	0	1	2	3	4	5	6	…	2041	2042	2043	2044	2045	2046	2047
value	0	1	0	0	0	1	0	…	1	0	1	0	1	1	0

② f_1^2 : A bloom filter of the similarity digest of File 2

bit add.	0	1	2	3	4	5	6	…	2041	2042	2043	2044	2045	2046	2047
value	1	1	0	1	0	1	0	…	0	1	0	0	1	0	1

③ $f_1^1 \cap f_1^2$

bit add.	0	***1***	2	3	4	***5***	6	…	2041	2042	2043	2044	***2045***	2046	2047
value	0	*1*	0	0	0	*1*	0	…	0	0	0	0	*1*	0	0

Table 3. The scoring of the similarity between similarity digests of two files on sdhash

SD_1 \ SD_2	f_1^2	f_2^2	⋯	f_t^2	max BF
f_1^1	$BF_{score}(f_1^1, f_1^2)$	$BF_{score}(f_1^1, f_2^2)$	⋯	$BF_{score}(f_1^1, f_t^2)$	$\max_1$
f_2^1	$BF_{score}(f_2^1, f_1^2)$	$BF_{score}(f_2^1, f_2^2)$	⋯	$BF_{score}(f_2^1, f_t^2)$	$\max_2$
⋯	⋯	⋯	⋯	⋯	⋯
f_s^1	$BF_{score}(f_s^1, f_1^2)$	$BF_{score}(f_s^1, f_2^2)$	⋯	$BF_{score}(f_s^1, f_t^2)$	$\max_s$
$SD_{score}(SD_1, SD_2)$ = average($\max_1$, $\max_2$,⋯, $\max_s$)					

Search는 1-대-다 또는 다-대-다의 비교 환경에서 이루어지며, 큰 데이터세트에서 input 파일과 유사한 것을 찾고자 할 때 이용하는 모드이다. 본 모드에서는 input을 데이터세트의 모든 요소에 대해 비교하여 input과 유사한 파일들을 출력한다. input 개수가 N이고 데이터세트의 크기가 M일 경우 $N \times M$ 만큼의 비교 작업이 이루어진다. 이에 대한 활용 방안으로는 이메일 조사를 제시하였다. 이메일 간의 유사도를 측정하면, 답장, 전달 등의 수·발신 흐름을 쉽게 파악할 수 있다는 것이다.

Streaming 모드에서는 네트워크 트래픽과 같은 스트리밍 패킷을 input으로 하여, 특정 파일이 네트워크를 통해 전송되었는지를 확인하기 위해 유사성을 비교한다. 이 모드도 $N \times M$ 만큼의 비교 작업이 이루어진다.

Clustering은 input으로 큰 데이터세트를 입력하면, 그 데이터세트 안에서 유사한 데이터끼리 그룹핑 하는 모드이다. 이 모드에서는 $N \times N$ 만큼의 비교 작업이 이루어진다.

근사 매칭을 실용화하면서, 조사자가 모든 데이터를 직접 확인하는데 소요할 시간은 줄었다. 하지만 이 역시 오탐은 존재하며, 결국 조사자의 수동적 확인 작업은 여전히 많은 시간이 걸린다.

III. 효율적인 근사 매칭을 위한 방안 제시

효율적인 근사 매칭 결과를 위해서는 매칭 목적에 맞게 데이터의 전처리 과정을 거쳐야 한다.

AHBM Search 모드의 이메일 수사 시나리오는 발신자가 작성한 내용의 유사성을 비교하므로, 크게 의미적 근사 매칭에 속한다. 또한 sdhash 알고리즘을 통해 비트 스트림 수준으로 유사성을 식별하므로 바이트 단위 매칭에 속하기도 한다. 이러한 근사 매칭을 이용할 때, 유사도를 측정하고자 하는 데이터세트에 전처리 작업을 하면 더욱 효과적인 매칭 결과를 확인할 수 있다. 실제로 문서 파일을 확인해보면, 사용자가 작성한 문자열 외에도 파일 시그니처 및 헤더, 서식 설정 등 여러 메타데이터가 추가적으로 존재함을 알 수 있다. 이러한 메타데이터는 파일의 콘텐츠와 관계없이 비슷한 경우가 많아 사용자가 작성한 문자열의 의미와는 상관없이 두 파일의 유사도가 크다고 잘못 판단할 수 있는 요소가 된다.

예를 들어, 어떤 eml 파일의 내용과 의미적으로 유사한 eml을 식별하고자 한다. 이 때, 근사 매칭 알고리즘은 eml 헤더 필드 때문에 관련이 없는 eml을 유사하다고 잘못 판단할 수 있다. 그래서 효율적인 의미적 근사 매칭을 위해서는 비교 대상의 메타데이터를 제외하고, 사용자가 실질적으로 작성한 문자열만을 비교 대상에서 추출하는 전처리 과정이 추가되어야 한다. 본 논문에서는 실험 및 검증 단계에서 이에 대한 효과성을 증명하도록 한다.

IV. 실험 및 검증

4.1 실험 개요

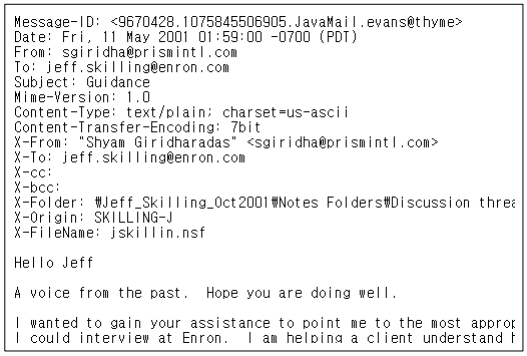
본 장에서는 실험 및 검증을 위해, 저작권 또는 이메일 관련 조사에 쓰일 수 있는 eml과 hwp 데이터세트를 구성하여 실험한다. 유사성 비교 작업은 원본 데이터세트, 전처리 과정을 거친 데이터세트에 대해 각각 진행한다. 근사 매칭 알고리즘은 sdhash 3.4[4]를 이용하였다. sdhash의 근사 매칭 결과, 1에서 100점 사이의 유사성 점수를 갖는 이메일 개수 중 TP(true positive)와 FP(false positive)의 개수를 이용해 정밀도를 도출한다. 이에 필요한 정의는 (2)와 같다.

$$\mathrm{Pr}ecision = \frac{TP}{TP + FP} \tag{2}$$

TP와 FP의 여부는 데이터 간의 의미적 관련성 여부를 직접 확인하여 판단한다.

4.2 eml

eml 파일은 Fig. 1.처럼 Message-ID, 시간, 수·발신자, 제목 등의 헤더 필드와 사용자가 작성한



```
Message-ID: <9670428.1075845506905.JavaMail.evans@thyme>
Date: Fri, 11 May 2001 01:59:00 -0700 (PDT)
From: sgiridha@prismintl.com
To: jeff.skilling@enron.com
Subject: Guidance
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: "Shyam Giridharadas" <sgiridha@prismintl.com>
X-To: jeff.skilling@enron.com
X-cc:
X-bcc:
X-Folder: \Jeff_Skilling_Oct2001\Notes Folders\Discussion threa
X-Origin: SKILLING-J
X-FileName: jskillin.nsf

Hello Jeff

A voice from the past.  Hope you are doing well.

I wanted to gain your assistance to point me to the most approp
I could interview at Enron.  I am helping a client understand t
```

Fig. 1. An example of eml and MIME e-mail

문자열(body text)로 이루어져 있다[7][8]. eml 파일에 대한 전처리 과정은 헤더 필드를 제외하고 body text만 추출하는 작업으로 구성한다. 본 실험에서는 엔론 데이터세트(Enron dataset)[9] 중, 5명의 이메일 17,831개를 테스트세트로 이용하였다. 본 데이터세트는 수신, 발신, 삭제 등 모든 폴더의 이메일을 포함하고 있다.

유사성 측정 방식은 Petter Bjelland 외 2인이 제시한 바와 같다. 하나의 큰 이메일 데이터세트(이메일 17,831개)에 대해 이메일 30개를 input으로 입력하여 $17,831 \times 30 = 534,930$ 번의 유사성 비교 작업을 거친다. input 이메일 30개와 유사하다고 측정된 이메일들이 실제로 의미적 관련성이 있는지 확인하도록 한다. input 데이터로 사용할 이메일 30개는 데이터세트 중 본문의 길이가 512byte 이상인 이메일들을 임의로 선택하였다. sdhash는 최소 512byte 이상의 데이터에 대해 similarity digest를 생성할 수 있기 때문이다.

실험 결과는 Table 4.와 같다. sdhash가 30개의 input 이메일과 유사하다고 탐지한 이메일은 원본 데이터세트(A)에서 508개, 전처리 과정을 거친 데이터세트(B)에서 207개이다.

원본 데이터세트(A)에 대한 유사도 측정 결과는 FP가 TP보다 많아 정밀도가 낮다. 반면 전처리 과정을 거친 데이터세트(B)는 한 건의 오탐만 발견되어 상대적으로 높은 정밀도를 나타낸다. 하지만 FN(false negative)이 증가한 점도 관찰되었다. 데이터세트 전처리 과정 후, 해당 파일들이 sdhash가 권장하는 파일 크기보다 작아진 것이 원인이다.

Fig. 2.와 Fig. 3.은 원본 데이터세트(A)에 대한 결과로, TP와 FP의 유사성 점수 분포도를 나타낸다. 하지만 TP는 1~100점 전체적인 구간에 고루

Table 4. Precision results of approximate matching of the containing header field(A) and preprocessed dataset(B) using 30 chosen e-mails as input with sdhash

	Total	TP	FP	FN	Precision
A	508	208	300	20	0.41
B	207	206	1	22	0.995

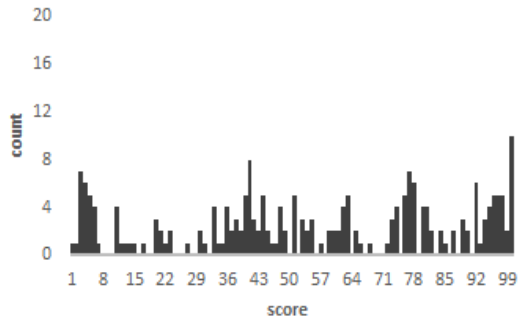


Fig. 2. Distribution of the similarity score in TP(True Positive) about e-mails containing header field

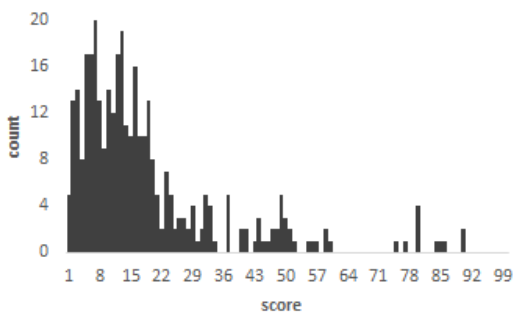


Fig. 3. Distribution of the similarity score in FP(False Positive) about e-mails containing header field

분포하고 있어 Roussev의 점수 해석가이드를 적용할 수 없다. 결국, 조사자는 1~100점 모든 이메일에 대해 유사 여부를 수동적으로 확인할 수밖에 없게 된다.

Fig. 4.는 전처리 과정을 거친 데이터세트(B)에 대한 실험 결과, TP 유사성 점수 분포도이다. 이 또한 전반적인 구간에 분포되어 있으나 90점 이상에 밀집도가 높음을 나타낸다. 한 건의 FP는 유사성 점수가 3점으로 매우 낮은 수치를 나타냈다.

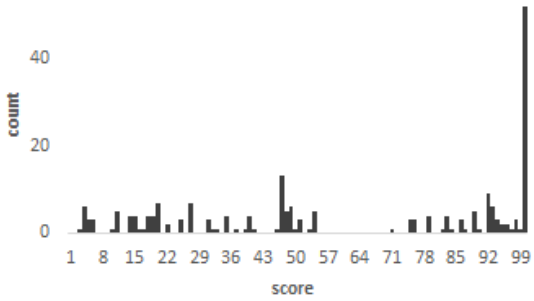


Fig. 4. Distribution of the similarity score in TP(True Positive) about preprocessed e-mail dataset

4.3 hwp

hwp 파일은 사용자가 작성한 내용뿐만 아니라, 문서 정보, 용지, 글자, 문단 속성 등 각종 설정 사항이 포함한다[10]. 사용자는 hwp 파일을 저장할 때 크기를 줄이기 위해 압축 설정을 할 수 있다. 압축에는 zlib를 이용한다. 본 실험에서 hwp 데이터세트에 대한 전처리 과정은 파일 압축 저장 여부를 판단하여 압축 해제한 후, 설정 사항에 대한 각종 메타데이터를 제거하고, 사용자가 작성한 실질적인 문자열만 추출하는 작업으로 구성된다.

데이터세트는 9개의 hwp 파일을 생성하고 시나리오별로 변조하여, 총 25개의 파일로 구성하였다. 생성한 한글 문서 버전은 2014이고, 파일 수정 내용은 다음과 같다.

① 1-1.hwp 단/표 삭제
1-2.hwp 단/표/머리말 삭제, 본문내용 순서 변경
1-3.hwp 단/표/머리말/본문내용 일부 삭제, 문단번호 수정
② 2-1.hwp 머리말 수정, 그림 삭제
2-2.hwp 표 삭제, 본문내용 일부 수정, 문단번호 추가
③ 3-1.hwp 머리말 삭제, 본문내용 일부 수정
3-2.hwp 머리말/그림/문단번호 삭제, 본문내용 일부 수정
④ 4-1.hwp 3단으로 변경, 본문내용 일부 수정
⑤ 5-1.hwp 본문순서/내용 일부 수정
5-2.hwp 본문순서/내용 일부 수정, 글상자로 서식 변경
⑥ 6-1.hwp 3/5 페이지 삭제
⑦ 7-1.hwp 차트 삭제
7-2.hwp 차트/표 이외의 텍스트 모두 삭제
⑧ 8-1.hwp 문단 번호 수정
⑨ 9-1.hwp 메모 삭제
9-2.hwp 메모 삭제, 본문내용 추가

이렇게 생성한 원본과 전처리 과정을 거친 데이터세트 두 개에 대해 각각 실험을 진행한다. Table 5.는 실험 결과이다.

원본 데이터세트(A)에서는 25쌍의 문서가 유사하다고 식별됐고, 그중에 12쌍은 오탐으로 확인했다. Fig. 5.와 Fig. 6.은 전처리 과정을 거치지 않은 hwp 데이터세트에 대한 TP, FP의 유사성 점수 분포를 나타낸다. 이를 통해 TP와 FP를 쉽게 구분할 수 없을 정도로 점수 구간이 매우 겹침을 알 수 있다.

전처리 과정을 거친 데이터세트(B)에서는 21쌍의 문서가 유사하다고 식별됐고, 한 건의 오탐도 발견되지 않아 TP의 정밀도가 매우 높게 측정됐다. 더불어 hwp 파일 포맷 상 메타데이터가 차지하는 비율이 eml 포맷에 비해 매우 많아서 FN 또한 줄었다. Fig. 7. 전처리 과정을 거친 hwp 데이터세트의 TP 유사성 점수 분포도를 나타낸다.

V. 결론 및 향후 연구

근사 매칭의 수요가 증가함에 따라 다양한 알고리즘들이 개발되고 있으나 이를 실무에 적용하기에는 높은 오탐과 미탐을 보여 적극 활용할 수 없는 실정이다. 그러나 본 논문에서는 eml과 hwp 파일 포맷을 기반으로 콘텐츠를 추출하는 전처리 과정을 통해 알고리즘의 정확도를 높일 수 있는 점을 제시하였다. 이는 조사관이 수작업으로 내용을 확인하는 시간을 현저하게 줄일 수 있어 효율성이 높을 것으로 기대된다.

eml과 hwp 외에도 docx, pptx 등의 문서 파일에서도 이러한 전처리 작업은 효과적인 매칭 결과를 출력할 것이다. eml은 발신자가 많은 양의 본문을 작성할수록 파일 크기 대비 메타데이터의 비율이 줄어들지만, hwp, docx, pptx는 사용자가 본문을 작성할수록 페이지 속성, 글씨체, 문단 속성 정보 등의 메타데이터도 함께 증가한다. 이로 인해 문서파일의 전처리 과정은 근사 매칭 시 반드시 필요하다.

향후에는 eml과 hwp 파일 포맷 외에도 다양한 문서 파일에 대한 전처리 과정과, 이를 자동화한 도구 개발에 대한 연구가 필요하다.

Table 5. Precision results of approximate matching of the containing meta data(A) and preprocessed dataset(B) using hwp dataset with sdhash

	Total	TP	FP	FN	Precision
A	25	13	12	11	0.52
B	21	21	0	3	1.00

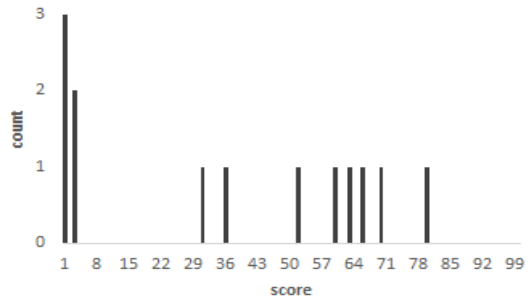


Fig. 5. Distribution of the similarity score in TP(True Positive) about hwp containing meta data

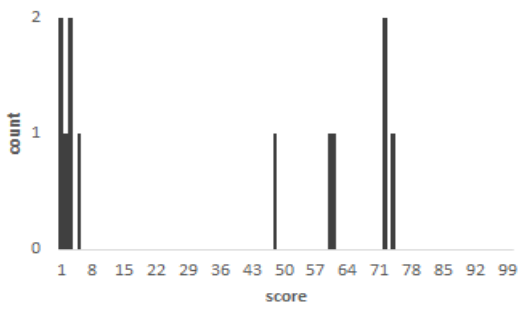


Fig. 6. Distribution of the similarity score in FP(False Positive) about hwp containing meta data

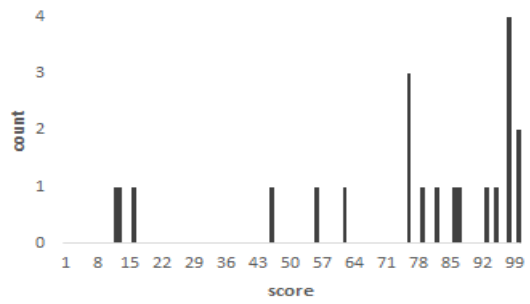


Fig. 7. Distribution of the similarity score in TP(True Positive) about preprocessed hwp dataset

References

[1] NIST SP 800-168, "Approximate Matching : Definition and Terminology," Jul. 2014

[2] Jesse Kornblum, "Identifying almost identical files using context triggered piecewise hashing," Digital Investigation, vol. 3, pp. 91-97, 2006

[3] Vassil Roussev, "Data fingerprinting with similarity digests," Advances in Digital Forensics VI. IFIP AICT, vol. 337, pp. 207-226, 2010

[4] Vassil Roussev, sdhash v3.4, 2013, http://roussev.net/sdhash

[5] Vassil Roussev, "An evaluation of forensic similarity hashes," Digital Investigation, vol. 8, pp. 34-41, Aug. 2011

[6] Petter Christian Bjelland, Katrin Franke and Andre Arnes, "Practical use of Approximate Hash Based Matching in digital investigations," Digital Investigation, vol. 11, pp. 18-26, May. 2014

[7] N. Borenstein and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies," RFC 1521, Sep. 1993

[8] P. Resnick, "Internet Message Format." RFC 2822, Apr. 2001

[9] William W. Cohen, Enron Email Dataset, https://www.cs.cmu.edu/~./enron

[10] Hancom, "Hwp Document File Formats 5.0," 2014, http://www.hancom.com/for MatQna.boardIntro.do

[11] Frank Breitinger and Vassil Roussev, "Automated evaluation of approximate matching algorithms on real data," Digital Investigation, vol. 11, pp. 10-17, May. 2014

[12] Frank Breitinger, Georgios Stivaktakis and Vassil Roussev, "Evaluating detection error trade-offs for bytewise ap-

proximate matching algorithms," Digital Investigation, vol. 11, pp. 81-89, Jun. 2014
[13] Vassil Roussev and Candice Quates, "Content triage with similarity digests : The M57 case study," Digital Investigation, vol. 9, pp. 60-68, Aug. 2012

〈저자소개〉

강 하 리 (Hari Kang) 학생회원
2012년 2월: 순천향대학교 정보보호학과 공학사
2014년 3월~현재: 고려대학교 정보보호대학원 정보보호학과 석사과정
〈관심분야〉 디지털 포렌식, 정보보호 관리체계

정 두 원 (Doo-won Jeong) 학생회원
2011년 8월: 고려대학교 공과대학 산업경영공학과 공학사
2011년 9월~현재: 고려대학교 정보보호대학원 정보보호학과 석·박사통합과정
〈관심분야〉 디지털 포렌식, 데이터 마이닝, 이미지 포렌식

이 상 진 (Sang-jin Lee) 종신회원
1987년 2월: 고려대학교 수학과 학사
1989년 2월: 고려대학교 수학과 석사
1994년 8월: 고려대학교 수학과 박사
1989년 10월~1999년 2월: ETRI 선임 연구원
1999년 3월~2001년 8월: 고려대학교 자연과학대학 조교수
2001년 9월~현재: 고려대학교 정보보호대학원 교수
2008년 3월~현재: 고려대학교 디지털포렌식연구센터 센터장
〈관심분야〉 디지털 포렌식, 심층 암호, 해쉬 함수