Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.6, Dec. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.6.1129

Convolution Filtering을 이용한 캡차 분석*

김 근 영,[1†] 신 동 오,[1] 이 경 희,[2‡] 양 대 헌[1]
[1]인하대학교 컴퓨터정보공학과, [2]수원대학교 전기공학과

CAPTCHA Analysis using Convolution Filtering*

Keun-young Kim,[1†] Dong-oh Shin,[1] Kyung-hee Lee,[2‡] Dae-hun Nyang[1]
[1]Inha University, [2]Suwon University

요 약

캡차는 사람은 쉽게 판단할 수 있으나 기계는 판단할 수 없는 문제를 이용하여 사람과 기계를 구별하는 기술이다. 널리 사용되고 있는 문자열 기반 캡차는 구현이 간단하나 이미지 기반이나 소리 기반 캡차에 비해 상대적으로 보안성이 약하다. 텍스트 기반 캡차의 보안성을 높이기 위해 다양한 기법이 개발되었으며, 그 중 하나는 복잡한 배경이나 노이즈를 사용하여 기계가 문자를 인식하기 어렵게 만드는 것이다. 이 논문에서는 이미지 프로세싱 기법 중 하나인 콘볼루션 필터링(Convolution Filter)을 이용하여 효과적으로 캡차를 공격하는 방법을 제시하고, 이를 네이버 카페의 캡차에 적용하여 분석해보았다.

ABSTRACT

CAPTCHA is a technique which distinguishes human and machine using what human can judge easily but machine can't. Though Text-based-CAPTCHA has been widely used and can be implemented easily, it is less security than other CAPTCHAs such as image-based, or audio-based CAPTCHAs. To enhance the security of text-based CAPTCHA, many techniques have been developed. One of them is making CAPTCHA recognized hard using complex background or noise. In this paper, we introduce how to apply convolution filtering effectively to attack CAPTCHA and actually analyze Naver's CAPTCHA which has been used for joining a cafe with this method.

Keywords: CAPTCHA, Convout, ion filter, Image processing, Crytanalysis

I. 서 론

CAPTCHA(Completely Automated Public Turing test to tell Computers and Humans Apart)는 HIP(Human Interaction Proof) 기술의 일종으로[1], 어떠한 사용자가 실제 사람인지 컴퓨터 프로그램인지 구별하기 위해 사용되는 방법이다. 사람은 인식할 수 있으나 컴퓨터는 인식하기 어렵게 의도적으로 왜곡한 후 그 내용을 물어보는 방식의 테스트로, 이용 통과 여부를 가려 테스트 대상이 사람인지 컴퓨터인지를 판별한다. 캡차는 현재 광고성 게시물 방지, 아이디 자동생성 방지, 이메일 주소 보호, 온라인 선거, 계정 해킹 방지 등에 사용되고 있다[2]. 콘볼루션 필터링 기법은 커널의 값을 조정함으로써 영상을 흐릿하게 하거나 선명하게 하거나 또는 잡음 제거나 경계선 검출 등 여러 방면에 쓰이는데[3] 이와 같은 콘볼루션 필터링의 특징을 이용하여 중요한 의미를 가지는 문자는 강조하고 나머지 부분은 제거함으로써 캡차를 분석하는데 이용할 수 있다.

이 논문에서 기여하는 바는 다음과 같이 두 가지

접수일(2014년 7월 7일), 수정일(2014년 11월 13일), 게재확정일(2014년 11월 13일)
* 이 논문은 인하대학교의 지원에 의하여 연구되었습니다.
† 주저자, kimky@isrl.kr
‡ 교신저자,khlee@suwon.ac.kr(Corresponding author)

로 요약할 수 있다. 하나는 콘볼루션 필터링을 캡차 분석에 적용하였다는 점이고, 나머지 하나는 이 기법을 현재 운영되고 있는 네이버 카페의 캡차를 분석하는데 사용하였다는 점이다.

이 논문의 구성은 다음과 같다. Ⅱ장에서는 문자열 기반 캡차의 특징과 캡차의 분석에 이용한 SVM을 설명하고 Ⅲ장에서는 문자열 기반 캡차의 연구 사례를 소개한다. Ⅳ장에서는 논문의 주제인 콘볼루션 필터링 기법에 대하여 서술하고 Ⅴ장에서는 이를 이용하여 실제 네이버 카페의 세 번째 캡차를 분석하였으며 Ⅵ장에서 이 논문을 마무리한다.

Ⅱ. 기반지식

2.1 문자열 기반 캡차의 특징

캡차는 크게 두 종류의 방어 기술을 사용한다. 하나는 기계가 글자 자체를 인식(recognition)하지 못하도록 하는 것이다. 이를 위해 Fig.1.에서 볼 수 있듯이 캡차에 쓰이는 문자의 종류를 다양하게 하거나 문자의 길이를 랜덤하게 하거나 문자의 크기와 체를 다양하게 하고 문자를 찌그러뜨리거나 기울이고 굴곡지게 배치한다. 다른 하나는 기계가 글자를 제대로 분리(segmentation)하지 못하도록 하는 것이다. 문자 뒤에 복잡한 배경을 사용하거나 문자가 아닌 다른 선을 긋거나 문자들 사이의 간격을 없애는 등의 방법을 사용한다[4]. 이 때 방어기술을 과다하게 사용하면 기계 뿐 아니라 사용자의 캡차 인식 능력 또한 떨어지게 되므로 주의해야 한다.

(a) MSN's CAPTCHA with distorted text

(b) Naver's CAPTCHA with complex background

Fig. 1. Examples of text-based CAPTCHA

2.2 SVM

SVM(Support Vector Machine)은 분류(classification), 회귀(regression) 등 여러 분야에서 사용되는 기계학습 알고리즘의 하나이다[5]. 캡차를 각각의 문자로 분리한 후에 SVM을 이용하여 각 문자를 학습시키고 각 이미지가 어떤 문자인지 질의하면 SVM이 학습된 내용을 바탕으로 응답한다는 것을 이용하여 각 글자가 무엇인지를 알아내기 위하여 이용하였다. 캡차 분석에서 SVM은 자주 이용되는 도구이며 좋은 결과를 보인다[6][7]. 실험에는 오픈소스 중 하나인 LIBSVM을 이용하였다[11].

2.3 콘볼루션 필터링

콘볼루션 필터링은 입력 영상의 각 픽셀의 값에 이웃한 픽셀들의 값에 다른 가중치를 주어 곱한 것을 더함으로써 이루어진다. 예를 들어 Fig.2-a.의 행렬에 Fig.2-b.의 커널을 이용하여 콘볼루션 필터링을 실시한다고 하면, Fig.2-a.의 (2,2)의 201은 필터 적용 후에 { (-164) + (-188) + (-178) + (201*9) + (-174) + (-168) + (-181) }의 값으로 바뀐다. 그림 2-c의 커널을 이용할 경우에는 { (164*0.075) + (178*0.15) + (201*0.15) + (197*0.15) + (174*0.075) + (168*.15) + (181*0.075) }로 바뀌게 된다. 이와 같이 콘볼루션 필터링은 이미지에 필터링 효과를 주기 위해 많이 사

164	188	164	161	195
178	201	197	150	137
174	168	181	190	184
131	179	176	185	198
92	185	179	133	167

(a) Matrix showing image's RGB

-1	-1	-1
-1	9	-1
-1	-1	-1

(b) Kernel for sharpening

0.075	0.15	0.075
0.15	0.15	0.15
0.075	0.15	0.075

(c) Kernel for blurring

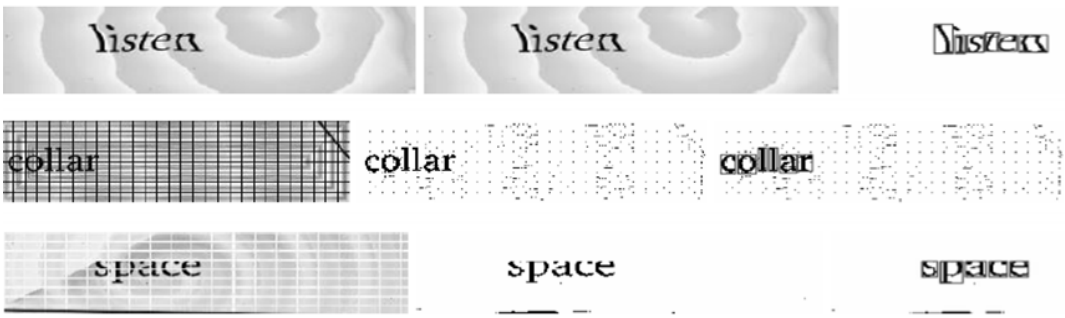
Fig. 2. An image that expressed by RGB and two kernels for convolution filtering

용되는 기법으로, 보통 3x3나 5x5 크기의 커널을 사용한다. 이때, 커널의 값을 조정함으로써 이미지를 흐릿하게 할 수도 선명하게 할 수도 있다. 중간 커널의 값과 이웃한 커널의 값이 차이가 작거나 없으면 블러링(blurring) 효과를, 차이가 크면 샤프닝(sharpening) 효과를 준다[3].

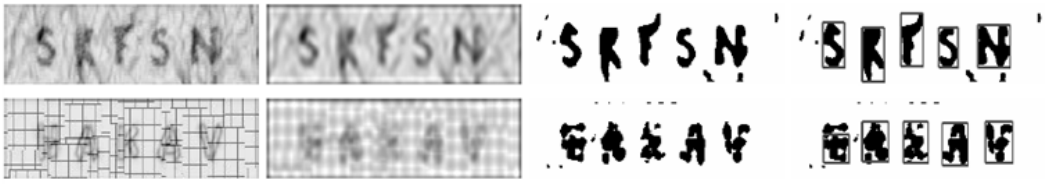
III. 문자열 기반 캡차 연구 사례

2004년 Chellapila등은 기계학습을 이용하여 캡차를 무력화하는 시도를 하였다. Fig.3-a.는 Yahoo/Ez-Gimpy의 3가지 캡차의 글자 분류 및 인식 과정이다. 각각의 캡차는 흑백처리한 후, thresholding을 이용한 이진화 과정을 거친다. 배경에 수평·수직선이 존재하는 두 번째 세 번째 형태의 경우는, 이웃하는 픽셀이 없는 픽셀들을 인식하여 제거한다. 각각의 캡차에서 배경을 지운 후에는 connected component를 인식하여 기계에서 문자를 인식하도록 하였다. Fig.3-b.는 Register 캡차의 글자 분류 및 인식 과정을 보여준다. 어지러운 패턴의 배경을 사용하고 글자를 약간 왜곡한 것이 이 캡차의 특징이다. 이미지를 흐리게하고 이진화한 후에 다섯 개의 커다란 component를 인식하도록 하였다[12].

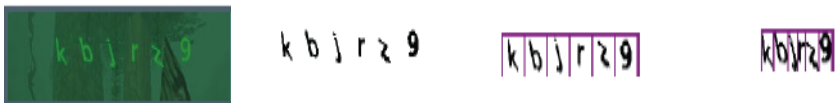
2011년 Bursztein등은 여러 캡차들을 정리하여 실제로 공격을 구현하였다. 그 중 Fig.3-c.는 Blizzard의 캡차로 복잡한 배경을 사용한 것이 특징이다. 저자들은 캡차를 공격하기 위해서 RGB 스펙트럼을 이용함으로써 문자와 배경을 분리해내었다. Fig.3-d.는 배경에 노이즈를 사용한 캡차를 사용한 Captcha.net의 캡차이다. 캡차의 공격에는 주위의 픽셀 값과 threshold값을 이용한 Gibbs 알고리즘이 사용되었다[1].



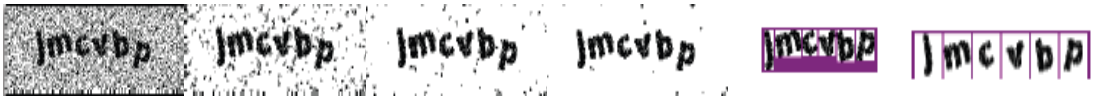
(a) Examples of segmentation and recognition for Yahoo/EZ-Gimpy CAPTCHAs



(b) Examples of segmentation and recognition for Register's CAPTCHA



(c) An example of segmentation and recognition for Blizzard's CAPTCHA



(d) An example of segmentation and recognition for Captcha.net's CAPTCHA

Fig. 3. Examples of segmentation and recognition for various CAPTCHAs with complex background

IV. 캡차 분석에 적용된 콘볼루션 필터링

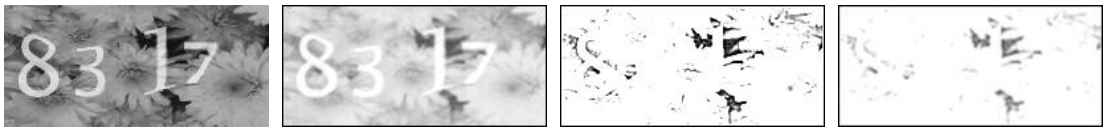
콘볼루션 필터링은 주위의 픽셀 값을 이용한 연산이므로 이웃한 픽셀들의 영향을 받게 된다. 따라서 캡차에서 문자를 나타내는 픽셀은 주변 픽셀과 RGB값이 비슷하여 필터링의 영향을 크게 받지 않는 반면에 노이즈나 배경은 크게 영향을 받으므로 원래의 형태를 잃게 된다. 이 논문에서는 콘볼루션 필터링에 사용되는 커널의 값을 다르게 하여 블러링 효과와 샤프닝 효과를 번갈아 두 번을 반복하여 실시하였다.

적용하는 커널내의 모든 값의 총 합이 1을 넘는다면 픽셀의 값이 원래의 값보다 더 커지게 되는 경우가 많다. 따라서 일정 수준 이상의 값을 가지는 픽셀들은 이와 같은 커널을 이용하여 필터링을 실시했을 경우에 RGB값이 255을 넘어 원래의 형태를 잃고 제거된다.

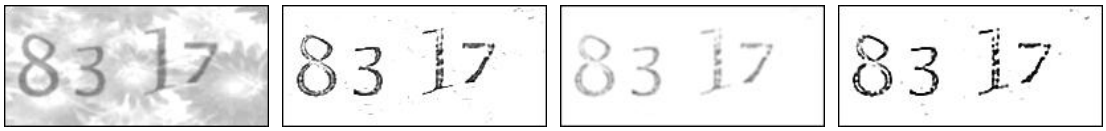
블러링을 실시하면 문자를 제외한 노이즈들은 주변 픽셀들의 영향을 받아서 흐릿해지거나, 그 값이 일정 수준 이상으로 높아져 제거된다. 이 상태의 캡차에 샤프닝을 실시한다. 샤프닝은 해당 픽셀의 값을 주변 픽셀에 비해 크게 강조한다. 또한 샤프닝에 쓰이는 커널은 그 총 합이 1을 월등히 넘는다. 따라서 대부분의 노이즈들과 일부 글자의 가장자리 부분도 배경의 영향을 받아서 사라지고 문자를 나타내는 픽셀들의 중심 부분만 남아있게 된다.

콘볼루션 필터링을 시작하기 전에 이미지를 흑백으로 바꾸어주고 필요하다면 반전 처리 등을 이용하여 문자를 어둡게 해준다. 전처리 과정을 통해 문자의 RGB값이 0에 가까워진다. 콘볼루션 필터링은 산술적 연산을 이용한 것이므로 값이 0에 가까울수록 영향을 덜 받는다. 따라서 문자는 배경과 달리 필터링의 영향을 크게 받지 않게 되어 필터링의 효과를 높일 수 있다. 전처리 과정 유무의 차이는 Fig.4.를 통해 알 수 있다.

Fig.4.의 캡차 뿐만 아니라 문자 외의 노이즈나 배경을 가진 캡차들에 대해서 각 캡차의 특성에 맞게 필터링에 쓰이는 커널의 값이나 필터링 횟수를 바꾸어 같은 배경제거 알고리즘을 적용해 보았다. AOL(Fig.5-a.), Daily Motion(Fig.5-b.)의 캡차는 그림들과 같이 매우 효과적으로 배경과 노이즈가 제거 된 후 문자만 남아있게 되었다.

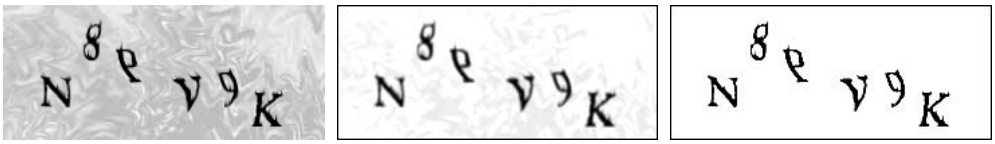


(a) bright text with dark background



(b) dark text with bright background (inverted image of (a))

Fig. 4. Two cases of images that filtered by convolution techniques



(a) An example of AOL CAPTCHA

(b) An example of DailyMotion CAPTCHA

Fig. 5. Examples of images that filtered by convolution techniques: complex background has removed and characters become more shapen

V. 네이버 캡차 분석

Table 1 [8] 과 Table 2 [9] 에 따르면 네이버는 국내 대부분의 사용자가 이용하는 사이트이다. 또한, 커뮤니티 방문 순위에서 상위에 랭크된 커뮤니티들 중, 일방향으로 정보를 제공하는 블로그들을 제외하고 네이버 카페만 유일하게 쌍방으로 정보제공이 가능한 커뮤니티로 네이버의 캡차가 무력화될 경우 많은 국내 이용자들이 광고나 스팸 등에 의한 피해 대상이 될 수 있다.

Table 1. Rank of most visited websites in South Korea (2014.04)

Rank	Domain	Pure visitor (*1000)
1	www.naver.com	30,901
2	www.daum.net	26,517
3	www.tistory.com	18,175
4	www.gmarket.co.kr	13,664
5	www.11st.co.kr	13,182

Table 2. Rank of most visited web communities in South Korea (2014.1.1~2014.05.17)

Rank	Category	Average in the period(%)
1	Other board	55.76
2	Naver blog	16.72
3	Daum blog	7.86
4	Naver cafe	7.08
5	Tistory	5.03

5.1 네이버 캡차의 특징

네이버가 현재 사용하고 있는 캡차의 특징은 다음과 같다.

- 4~8개의 숫자(1~9)로 이루어져 있다.
- 일정 시간 내에 캡차를 다시 요청하면 배경 이미지와 숫자의 왜곡 정도만 달라지고 입력해야 할 숫자 자체는 바뀌지 않는다.
- 가입을 다시 시도할 경우에 숫자가 바뀐다.
- 전체 캡차 이미지의 크기는 200x90으로 고정되어 있으나, 숫자 하나하나의 길이는 고정되어 있지 않다.
- 숫자는 가로 방향으로 떨어져 분포한다.
- 숫자는 크기가 바뀌기도 하지만 약간의 왜곡도 주어진다.
- 숫자는 흰색이지만, 투명하게 처리하여 배경의 영향을 받는다.
- 배경으로 쓰이는 이미지는 자연의 꽃, 풀, 나무, 나뭇잎, 돌 등으로 구성되어 있다.
- 같은 배경이미지가 재사용된다.

5.2 캡차 분석 알고리즘

5.2.1 알고리즘의 흐름

캡차 분석 알고리즘의 의사코드는 Fig.6.과 같다. 일정 시간 내에 캡차를 재요청하면 숫자는 같지만 배경과 왜곡이 변형된 캡차가 주어진다는 특성을 이용하여 한번의 공격에 30개의 캡차를 불러오도록 하였다. 각각의 캡차의 배경을 제거한 후 숫자 별로 나누어 캡차마다 리스트를 생성하여 저장하였다. 이 리스트들의 길이를 비교하여 캡차의 길이를 추정한 후, 리스트의 길이가 추정 길이와 같은 리스트들만 이후의 SVM 판독 과정에 이용하였다. SVM 판독 결과로 생성된 파일을 분석하여, 각 자리별로 가장 많이 나온 숫자를 해당 숫자의 결과로 선택하였다.

```
for i = 0 to 30
    arrayList[i] ← Split(captchaImage[i])
end
    assumedLength ← MostFrequentLength(arrayList)
for i = 0 to 30
  if(assumedLength = arrayList[i].size)
     MakeModelFile(arrayList[i])
end
SVMTest(modelFile)
DeCAPTCHA(resultFile)
```

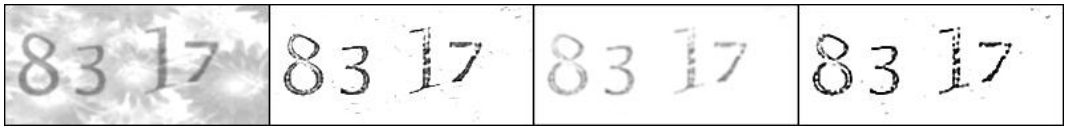
Fig. 6. DeCAPTCHA algorithm

5.2.2 배경 제거

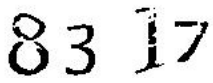
배경제거는 Ⅳ장에서 설명한 콘볼루션 필터링을 이용하였다. Fig.7-a.는 네이버 카페 캡차의 원본이다. 원본 이미지에 색상 반전과 흑백처리를 하면 Fig.7-b.와 같은 이미지가 생성된다. 이 이미지에 콘볼루션 필터링을 이용하여 블러링과 샤프닝 과정을

(a) A source image of Naver's CAPTCHA

(b) Inverting and gray scaling on (a)



(c) Each step of convolution filtering



(d) Binarize

Fig. 7. An example of processes that erasing background and emphasizing text

두 번 반복한다. Fig.7-c.는 커널 값을 조정하여 왼쪽부터 블러링, 샤프닝, 블러링, 샤프닝 필터링을 거친 이미지의 모습이다. 필터링을 마친 이미지에 threshold를 주어, threshold 값을 넘으면 배경으로 추정하여 RGB값을 (255,255,255)로, 넘지 못하면 숫자로 추정하여 RGB값을 (0,0,0)으로 한다. 이진화 과정을 거친 캡차는 Fig.7-d.와 같다.

5.3. 분리

네이버 캡차의 숫자들은 가로 방향으로 모두 떨어져서 분포한다. 따라서 단순히 각 숫자가 시작하는 순간과 끝나는 순간을 인식하여 수직으로 분리할 경우 어렵지 않게 분리가 가능하다.

5.3.1 히스토그램을 이용한 글자 분리

배경이 지워지고 숫자와 배경만 남은 캡차는 이미지의 픽셀을 이용하여 숫자별로 분리가 가능하다. 가로축을 기준으로 세로에 존재하는 검은 픽셀(RGB값이 (0,0,0))의 수를 나타내는 히스토그램을 작성한다. 이 때, 히스토그램의 값이 0을 넘으면, 즉 검은 픽셀이 존재하면 숫자의 일부로 인식한다. 일정 길이 이상 연속적으로 히스토그램의 값이 0을 넘는다면 연속구간의 시작과 끝 픽셀에서 수직으로 잘라 이미지를 리스트에 저장한다. 이 과정에서 자동적으로 숫자의 가로 여백이 제거 된다. 이 과정을 통해 분리된 문자들의 형태는 Fig.8-a.와 같다.

5.3.2 리스트 내의 이미지 조정

5.3.1의 과정을 거치면 각 캡차별로 하나씩 숫자별로 나누어진 이미지들이 저장되어 있는 30개의 리스트가 생성된다. 길이가 4나 5인 캡차들은 숫자간의 거리가 멀어 대부분 단순분리과정에서 모두 분리에 성공하였기 때문에 별도의 처리가 필요 없다. 하지만, 길이가 6 이상인 캡차의 경우 배경을 제거하고 숫자 별로 분리하는 과정에서 배경이 덜 제거되어 Fig.8-b.와 같이 두 숫자가 붙어서 분리된 경우가 존재할 수 있다. 따라서 숫자의 가로 길이가 25를 넘으면 두 숫자가 붙어서 분리 되었다고 판단하여 강제로 분리를 시행한다. 이때 중간의 다섯 픽셀 중에서 수직선 상에 존재하는 검정 픽셀의 수가 가장 작은 점이 경계일 가능성이 높으므로, 그 픽셀을 기준으로 두 숫자를 강제로 분리함으로써 리스트를 수정한다.

(a) Well separated text

(b) Unseparated text

Fig. 8. Two result of text segmentation by histogram

5.4. 크기 조정 및 숫자의 세로 여백, 얼룩 제거

리스트에 저장되어있는 모든 이미지의 세로 여백을 없애고 크기를 20x25로 통한다. 이 때, 세로 여백 제거 알고리즘은 숫자 분리 알고리즘과 원리가 같다. 만약 숫자의 세로 길이가 지나치게 짧으면, 숫자가 아닌 얼룩으로 판단하여 리스트에서 제거한다.

5.5. SVM을 이용한 분류

앞의 과정들을 거치면 30개의 캡차 마다 각각의 리스트들이 생성된다. 이 리스트들의 길이를 비교하여 가장 많이 나온 길이를 캡차의 길이, 즉 숫자의 개수로 간주한다. 이 때 리스트의 길이가 추정 길이와 다른 리스트들은 얼룩이 포함되었거나, 글자가 너무 많이 지워져 얼룩으로 판단되어 리스트에 포함되지 못한 경우이므로 실험에 사용하지 않는다. SVM을 이용하여 각 숫자들을 인식한 후, 각 자리에 대하여 인식 결과 가장 많이 나온 숫자를 해당 숫자로 선택한다.

5.6. 실험

5.6.1 실험 환경

예측 전, SVM 학습에 쓰인 모델 파일은 실험 전에 미리 만들어놓았으며, 모델 파일 생성에 이용한 이미지들은 실험에 사용하지 않았다. 사용한 이미지의 총 개수는 1부터 9까지 150개씩, 총 1350개의 이미지를 사용하였다. 실험을 진행한 환경은 Table 3과 같다.

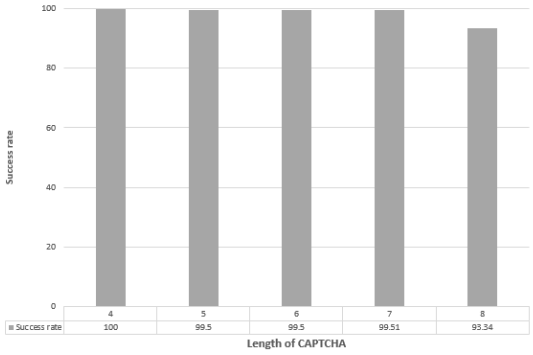


Fig. 9. DeCAPTCHA success rates depending on text length

Table 3. Test environment

Development environment	EClipse
Used library	libSVM
Programming Language	Java
Classification algorithm	SVM 3.17
Computer used in the experiment	Intel® CoreTM i3 M380 @ 2.53GHz. 3.00GB

5.6.2 실험 결과

실험에 사용된 캡차는 총 1,000세트이며, 각 세트는 동일한 숫자이지만 서로 다른 배경을 가진 30개의 캡차 이미지로 구성되어있다. 하나의 캡차를 공격하는데 이용된 이미지는 30개이므로 총 30,000개의 이미지를 사용하였다. 실험에 사용된 이미지들은 순수하게 분석에 소요되는 시간을 측정하기 위하여 실험 전에 미리 컴퓨터에 저장해 놓았다. 캡차 분석 평균 시간은 하나의 세트 당 약 7.76초이다. 이 때 하나의 세트의 캡차는 같은 숫자를 나타내는 30개의 캡차 모두를 의미한다. 따라서 분석 시간은 컴퓨터에 저장되어있는 30개의 이미지를 불러와서 분석한 후에 통계를 낸 시간까지를 의미한다. 여러 개의 숫자 중 하나만 잘못 판독하여도 실패로 간주하였다. 전체 성공률은 98.3%이고, 각 길이별 성공률은 Fig. 9와 같다.

이 때 길이가 8인 경우의 실패율이 다른 경우에 비해 약간 높다. 이는 문자의 개수가 많을수록 문자 사이의 간격과 문자 자체의 크기가 작아 기계에서 정확히 분류하기가 어려워지기 때문이다.

실패의 원인에는 크게 SVM 실험 결과 본래의 숫자와 다른 숫자로 나타내는 분류 실패의 경우와 원래의 글자 길이와 다르게 결과가 나오는 분리에 실패한 경우 두 가지가 있다. 분류에 실패한 경우는 9가지, 분리에 실패한 경우에는 8가지로 실험 결과 비슷한 확률로 등장한다. 분류를 실패한 경우에는 3을 7로(1/9), 5를 2로(4/9), 5를 3으로(5/9) 판단하였다. 이 경우 5를 다른 숫자로 판단한 것이 대부분인데, 이는 필터링 시 5의 윗부분과 아래 둥근 부분 사이가 지워져, 세로 여백을 제거할 때 위쪽 짧은 부분을

(a) Recognized 3 instead of 5 (b) recognized 2 instead of 5 (c) recognized 7 instead of 3

Fig. 10. Cases of text recognition failure

얼룩으로 판별하고 둥근 부분만 문자로 인식하여 2와 3과 같이 둥근 숫자로 결과가 나온 것으로 보인다(Fig.10-a, Fig.10-b). 3을 7로 인식한 경우 Fig.10-c)는 실패 이미지가 사람이 보기에 3과 큰 차이가 없으므로 기계의 특성상 나타난 한계로 파악된다

분리를 실패한 경우는 얼룩이 문자로 판별되어 원래의 길이보다 길게 나온 경우는 없었고 글자가 하나 사라진 경우만 나타났다. 8가지 실패 경우 중에서 7경우가 1이 없어진 경우이고, 하나만 9가 없어진 경우인데, 1과 9 모두 문자의 가로 폭이 좁은 숫자로, 프로그램에서 숫자를 얼룩으로 인식하여 발생한 문자로 추정된다.

VI. 결 론

네이버의 카페 서비스는 자동가입을 방지하기 위하여 캡차 문제를 해결하도록 요구하고 있다. 네이버 캡차는 복잡한 이미지가 배경에 위치하고, 그 위로 반투명의 문자를 보여주어 사람은 인식하기 쉬우나 일반적인 기계는 인식하기 어렵도록 설계되어 있다. 이 논문에서는 네이버 카페의 캡차에 콘볼루션 필터링과 색상 반전을 이용하여 배경을 제거하였다. 그리고 개별적으로 글자를 분리한 뒤 SVM을 이용하면 높은 확률로 캡차의 내용을 인식할 수 있음을 실험을 통해 보였다. 또한, 동일한 방법을 AOL과 DailyMotion에서 사용하는 캡차에 적용했을 때에도 효과적으로 배경 및 노이즈가 제거되고 글자만 남은 이미지를 얻을 수 있음을 보였다. 따라서 복잡한 배경을 가진 캡차는 기계가 인식하기 어렵도록 설계되었으나, 사실상 그 효과가 미미하다고 할 수 있다.

이 논문에서 분석한 캡차들 뿐만 아니라 배경을 가지는 대부분의 캡차들에도 이와 같은 방법을 이용하여 공격을 시도할 경우에 높은 성공률을 보일 수 있을 것으로 기대한다. 또한 이미지 프로세싱 기법에 쉽게 영향을 받지 않으면서도 사용성이 높은 캡차를 설계하는 것은 향후 연구 주제로 남겨둔다.

References

[1] E. Bursztein and M. Martin and Jon C.Mitchell, "Text-based CAPTCHA Strengths and Weaknesses," 18th ACM conference of Computer and Communication security 2011 (CSS'2011), pp. 125-138, 2011

[2] Carnegie Mellon University. "The Official CAPTCHA site," http://www.captcha.net/

[3] V. Podlozhnynk, "Image convolution with CUDA," NVIDIA Corporation white paper, vol. 1.0, June 2007

[4] C. Sutherland, "Usability and Security of Text-based CAPTCHAs," UMM CSci Senior Seminar Conference, Morris, MIN. 2012.

[5] Joachims, T. "Making large-scale SVM learning practical," In Advances in Kernel-Methods - Support Vector Learning, Support Vector Learning, Bl Schollkopf, C.J.C. Burges, and A. J. Smola, Eds., MIT Press, Cambridge, MA, pp. 169-184, 1998

[6] SungHo Kim, DaeHun Nyang, KyungHee Lee, "Breaking character-based CAPTCHA using color information", Journal of The Korea Institute of information Security & Cryptology, 19(6), pp. 105-112, Dec. 2009

[7] DaeHun Nyang, YongHeon Choi, SeokJun Hong, KyungHee Lee, "Analysis of Naver CAPTCHA with Effective Segmentation", Journal of The Korea Institute of information Security &

Cryptology, 23(5), pp. 909-917, Oct. 2013
〔8〕 http://www.nielsen.com/kr/ko/top10s.html?ranking=websites
〔9〕 www.internettrend.co.kr/trendForward.tsp
〔10〕 K. Chellapilla, P. Y. Simard, "Using Machine Learning to Break Visual Human Interaction Proofs (HIPs)," Advances in neural information processing systems 17, pp. 265-272, 2005.
〔11〕 C. C. Chag, C. J. Lin, "LIBSVM: a library for support vector machines," ACM Transactions on intelligent Systems and Technology(TIST) 2(3), article pp. 1-27, 2011

〈저자소개〉

김 근 영 (KeunYoung Kim) 학생회원
2011년 3월~현재: 인하대학교 컴퓨터 정보공학과 학사과정
〈관심분야〉 정보보호, 네트워크 보안, 암호이론

신 동 오 (DongOh Shin) 정회원
2010년 2월: 인하대학교 컴퓨터 정보공학과 졸업
2012년 2월: 인하대학교 컴퓨터 정보공학과 석사
2012년 9월~현재: 인하대학교 컴퓨터 정보공학과 박사과정
〈관심분야〉 인터넷 보안, 네트워크 보안, 금융 보안

이 경 희 (KyungHee Lee) 정회원
1993년 2월: 연세대학교 컴퓨터과학과 학사
1998년 8월: 연세대학교 컴퓨터 과학과 석사
2004년 2월: 연세대학교 컴퓨터 과학과 박사
1993년 1월~1996년 5월: LG소프트(주) 연구원
2000년 12월~2005년 2월: 한국전자통신연구원 선임연구원
2005년 3월~현재: 수원대학교 전기공학과 조교수
〈관심분야〉 바이오인식, 정보보호, 컴퓨터비전, 인공지능, 패턴인식

양 대 헌 (DaeHun Nyang) 종신회원
1994년 2월: 한국과학기술원 과학기술 대학 전기 및 전자 공학과 졸업
1996년 2월: 연세대학교 컴퓨터 과학과 석사
2000년 8월: 연세대학교 컴퓨터 과학과 박사
2000년 9월~2003년 2월: 한국전자통신연구원 정보보호연구본부 선임연구원
2003년 2월~현재 인하대학교 컴퓨터정보공학과 부교수
〈관심분야〉 암호이론, 암호프로토콜, 인증프로토콜, 무선 인터넷 보안