Generator pametnih šifri

Viktor Braut, Matija Osrečki i Dino Sulić

7. siječnja 2014.

Sažetak

Za većinu korisnika sigurnost na računalnim sustavima bazira se na kvalitetnoj šifri. Ovaj rad predstavlja pristup generiranju kvalitetnih šifri na temelju lakoće utipkvanja istih. Za analizu lakoće utipkavanja naučili smo neuronsku mrežu da regresijom ocjeni lakoću utipkavanja nizova znakova fiksne duljine (engl., *n-gram*), u našem slučaju veličine 3. Optimalne parametre neuronske mreže odredili smo unakrsnom validacijom nad skupom od oko 440 podnizova koristeći 4 preklopa. Iako rezultati jako variraju, u konačnici smo dobili dovoljno dobre rezultate za izradu generatora. Generator koristi jednostavnu tehniku gdje kreće s slučajnom odabranom kratkom šifrom dobre kvalitete i svaki sljedeći znak bira stohatički na temelju kvalitete novodobivenog zadnjeg podniza.

1. Uvod

Sigurnost privatnosti na računalnim sustavima danas je bitna više nego ikada. Za većinu korisnika to znači jednu stvar – kvalitetna šifra. I dok sustavi poput GMail-a ugrađuju metode dodatne verifikacije korisnika putem tokena generiranih primjerice mobilnim uređajem te postoje rješenja sigurnosti uporabom kriptografskih metoda javnih i privatnih ključeva (SSH, GPG, itd.), za većinu web servisa, operacijskih sustava i ostalih oblika programske potpore kvalitetna šifra je najzastupljenije rješenje.

Više je svojstava kvalitetne šifre. Najbitnije svojstvo je sigurnost – mora biti dovoljne duljine, sadržavati dovoljno različitih vrsta znakova (mala i velika slova, brojevi i ostali znakovi) koji bi trebali biti slučajno raspoređeni, kako pojedini dijelovi šifre ne bi bile konkretne riječi. Drugo poželjno svojstvo je da je šifru lagano zapamtiti. Nažalost, za većinu je to najbitnije svojstvo zbog čega za šifre koriste imena svojih djevojki, velikih kantautora (jedan je Bob) i slično.

U ovom radu predstavljamo ideju izbora, odnosno generiranja šifre na temelju lakoće utipkavanja iste. Prva pretpostavka je da će takav način zastupati sve znakove na tipkovnici u podjednakoj mjeri te da zbog slučajne prirode generiranja šifri neće doći do podnizova koji se mogu naći u rječnicima, prema tome bi trebalo biti zadovoljeno svojstvo sigurnosti. Druga pretpostavka je da lakoća utipkavanja olakšava mehaničko pamćenje i da će time biti zadovoljeno drugo stvojstvo dobre šifre, iako možda tek nakon kraćeg perioda uvježbavanja šifre.

Uži aspekt ovog zadatka koji je i ujedno najteži jest analiza lakoće utipkavanja proizvoljnih nizova znakova na tipkovnici određenog rasporeda znakova. Pristup koji smo prirodno prihvatili jest uporaba subjektivnih ocjena skupa kraćih nizova u nadi da postoje nekakva statistička ili geometrijska korelacija između transformiranih nizova znakova i naših subjektivnih ocjena. Konkretno, koristili smo neuronske mreže kako bi regresijom odredili lakoće utipkvanja nizova znakova koje nismo vidjeli.

2. Metode

Pošto različiti rasporedi tipki utječu na lakoću utipkavanja, korišten je US raspored tipki. Također ne koristimo znakove za koje je potrebna tipka `shift` i podrazumijeva se da nitko ne koristi `capslock`. Prema tome, sva slova su mala i koristi se podskup znakova. Znakovi `\` i `|` su na US tipkovnici na tipki koja je ponekad iznad, a ponekad lijevo od tipke `enter`, zbog čega također nisu korištene.

2.1. Analiza lakoće tipkanja

2.1.1. Tehnika n-grama

Pretpostavka koju koristimo prilikom analize lakoće utipkavanja proizvoljnog teksta jest da ako uzmemo sve moguće podnizove fiksne duljine (n-grame), lakoća utipkavanja tog niza znakova odgovara prosjeku ocjena lakoća utipkavanja svih njegovih n-grama. Ta se pretpostavka temelji na više intuitivnih ideja. Prva je da ako su n-grami dovoljno dugi i dovoljno kvalitetni, ono što je trebalo utipkati prije n znakova nije toliko bitno, tj. toliko daleka povijest nema utjecaj na ono što slijedi. Druga ideja je da prilikom učenja utipkavanja neke šifre, korisnik će prirodno grupirati slova u manje grupe veličine 3 do 5, koje će moći instantno utipkati ako su kvalitetne. Time ujedno olakšavamo označavanje primjera, smanjujemo broj ulaznih kombinacija i omogućujemo učenje neuronskim mrežom.

2.1.2. Generiranje značajki

Za svaki znak generiramo 8 značajki, koji se mogu podijeliti u tri grupe:

- Koordinate trenutne tipke (2)
- Polarne koordinate vektora od prethodne do trenutne tipke (2)
- Polarne koordinate vektora od lijeve i desne tipke `alt` do trenutne tipke (4)

Pritom se vrijednosti svih značajki normaliziraju na interval $[-1, 1]$.

Koordinate su temelj svih ovih značajki. Koordinatni sustav je postavljen sa $(0, 0)$ na tipki 1. Svaka tipka je kvadrat veličine 1 sa 1 i udaljenost između tipki je 0.2. Prva koordinata je vertikalna (redak), te horizontalna koordinata prve tipke u svakom redu je zadana nizom $[0, 0.6, 1, 1.6]$. Prema tome koordinate znaka a su $(2.4, 1)$, znaka b su $(3.6, 6.4)$ te za $=$ su $(0, 13.2)$.

Polarne koordinate vektora su njegova duljina i kut, koje se trivijalno izračunaju uporabom Pitagorinog poučka i osnovne trigonometrije. Ako su sve udaljenosti između svih slova bliske, vjerojatno se radi o lošem nizu znakova jer ih vjerojatno treba upisati jednom rukom na nezgodan način.

Tipke `alt` služe kao aproksimacija neutralne pozicije ruku tijekom tipkanja. Naime, prsti bi prirodno trebali stajati u srednjem redu slova, nad tipkama `asdf` i `jkl;`, ispod kojih se dva reda niže približno nalaze `alt` tipke. Te dvije koordinate su zapravo izračunate kao tipke koje bi bile direkno ispod tipki x i , (koordinate $(4.8, 2.8)$ i $(4.8, 10)$). Intuicija je da ako su za jednu od ove dvije tipke kutevi za sve znakove u n-gramu podjednaki, radi se o vertikalnom slijedu tipki koji je primjerice nepoželjan, dok su horizontalni slijedovi bolji.

Za n-grame veličine 3, koje mi koristimo, ovo sve skupa predstavlja 24 značajki. Treba napomenuti da postoji 45 znakova koje koristimo, te za n-grame duljine 3, postoji oko 10^5 različitih kombinacija, od kojih je oko 0.5% pokriveno primjerima.

Osim ovog skupa značajki, probali smo i proširenu skup sa kvadratima svake značajke. Više o rezultatima kasnije.

2.2. Stohastički generator šifri

U ovom trenutku pretpostavljamo da imamo spremnu naučenu neuronsku mrežu za ocjenjivanje lakoće utipkavanja nekog n-grama `ocijeni(ngram)`.

Generator radi tako da u prvoj fazi generira n-gram dovoljne kvalitete, a zatim u drugoj fazi gradi šifru znak po znak do željene duljine. U svakom koraku, on nasumice odabere podskup slova s kojom bi šifra završavala n-gramom dovoljno dobre kvalitete, zatim pridaje vjerojatno odabira svakom slovu na temelju te kvalitete te u konačnici nasumice odabire jedno slovo pomoću diskretne slučajne varijable. Ukoliko u nekom trenutku nema dovoljno slova s kojima može kvalitetno nastaviti, generator briše zadnje slovo i nastavlja. U tom smislu ovo nazivamo kvazi-backtracking algoritam.

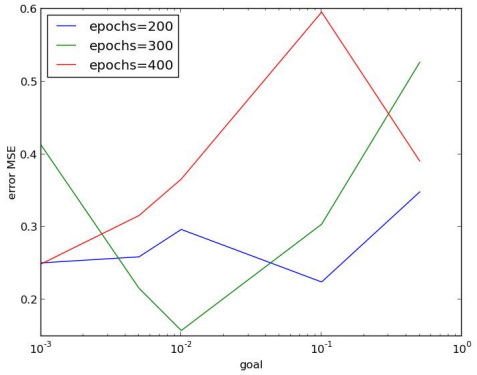
Gore navedeni algoritam ne radi pretjerano brzo i sigurno se može zamijeniti nekim oblikom evolucijskih algoritama ili nekom boljom heurističkom metodom, no za potrebe prototipa i prve implementacije ove ideje radi dovoljno dobro.

Algorithm 1 Stohastički generator šifri

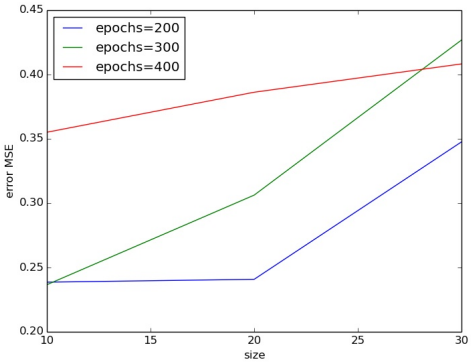
```
prag_pocni ← 0.5
sifra ← ""
loop
    sifra ← slucajni_ngram()
    if ocijeni(sifra) ≥ prag_pocni then
        break
    end if
end loop

prag_dalje ← 0.25
while duljina(sifra) < zeljena_duljina do
    promijesaj(znakovi)
    broj_dobrih ← 0

    for z ∈ znakovi do
        ngram ← sifra[−2 :] + z
        ocjena ← ocijeni(ngram)

        if ocjena ≥ prag_dalje then
            vjerojatnosti[z] ← ocjena + 1
            broj_dobrih ← broj_dobrih + 1
        else
            vjerojatnosti[z] ← 0
        end if
    end for

    normaliziraj(vjerojatnosti)
    if broj_dobrih ≥ 5 then
        sifra ← sifra + slucajni(vjerojatnosti)
    else if duljina(sifra) > 2 then
        sifra ← sifra[: −1]
    else
        kreni ponovo
    end if
end while
```

3. Rezultati

Testiranje mreže je provedeno cross-validacijom. Od 450 primjera 75% je korišteno za učenje neuronske mreže dok je ostalih 25% primjera korištena za testiranje. Primjeri su se rotirali četiri puta i greška je usrednjena, kako se nebi mreža uvijek učila sa istim primjerima. Kroz taj postupak testirano je nekoliko paramatera: broj neurona u prvom sloju, broj epoha učenja i zadovoljavajuća greška nakon koje učenje prestaje. Brojevi neurona koji su testirani su 10, 20 i 30. Broj epoha smo ograničili na 200, 300 i 400 dok smo za zadovoljavajuću grešku uzimali 0.001, 0.005, 0.01, 0.1 i 0.5. Za svaku kombinaciju tih parametara izračunata je kvadratna greška (MSE). Dobiveni podatci predočeni su grafovima na način da se vidi ovisnost greške neuronske mreže za dva parametra te usrednjeni za treći.



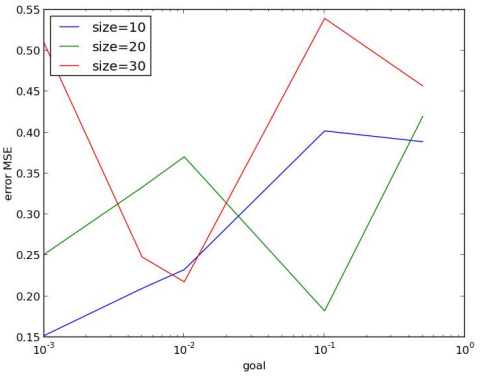
Slika 1.: Odnos ciljane greške i broja epoha učenja na grešku



Slika 2.: Odnos broja neurona i broja epoha učenja na grešku

4. Zaključak

Ovaj rad predstavlja prvi nama poznati pristup ideji da se kvalitetne šifre mogu generirati minimizacijom lakoće tipkanja, time osiguravajući do-



Slika 3.: Odnos ciljane greške i broja neurona na grešku

bar temelj za mehaničko pamćenje i sigurnost šifre. Glavni fokus je prema tome bila analiza lakoće utipkavanja proizvoljnog teksta. U tom aspektu vjerujemo da smo postigli dovoljno dobre rezultate za naš cilj, iako postoji mnogo mjesta za napredak.

Prvi problem je generiranje i ocjenjivanje n-grama. Vjerujemo kako je 3 apsolutno minimalna duljina i trebalo bi bolje isprobati veće duljine, ali najviše do 5-6, jer nakon toga nema puno koristi. Ocjenjivanje je veoma subjektivno, i iako je jedna ideja da je to dobro jer omogućava personalizirane generatore, upitna je statistička korelacija tih ocjena i samih n-grama. Jedna moguća ideja je napraviti program koji testira korisnika i sam daje ocjene na temelju brzine utipkavanja. Takav program bi mogao detektirati kada je dovoljno primjera izgenerirano i time možda skratiti ionako dug postupak.

Drugi problem je kako idalno predstaviti puni skup znakova, odnosno ukorporirati tipku `shift`. Naš prvi skup za učenje sastojao se od 200 n-grama duljine 4 koji su uključivali sve znakove, no sa mapiranjem shifta u posebnu značajku (je li snisnut shift) za svako slovo, nismo dobivali dobre rezultate. Možda je moguće maknuti to iz potprograma za analizu i inducirati u generator, gdje bi generator odlučivao kada staviti `shift` i time prilagodio ocjenu. Naravno, možda najbolje to prepustiti korisniku, pošto u malo vremena može odlučiti gdje bi `shift` odgovarao u već generiranoj šifri.

Osim toga, može se probati još načina generiranja značajki kao i druge metode strojnog učenja.

Literatura

Ovo smo radili isključivo prema vlastitom nahođenju. Na internetu je teško naći materijale na ovu temu. Kao pomoć, koristili smo isključivo dokumentacije programskih biblioteka koje smo koristili (`neurolab`, `numpy`).

Dodatak A: programsko ostvarenje

Za izradu svih elemenata ovog rada korišten je programski jezik `python`. Kako bi sve radilo, potrebno je instalirati pakete `numpy` i `neurolab`.

Glavni program generatora je napisan kao samostalna skripta, ostali moduli se mogu uvesti u recimo `ipython` i koristiti tako. Generator kao prvi argument prima datoteku s postavkama mreže, i onda proizvoljan broj argumenata, svaki od kojih odgovara duljini jedne generirane šifre. Primjer:

```
$ python easypass/main.py data/conf.net 10 12 14
10: p2a'hs]i]v
12: 4f=g=7c12hit
14: bs0.m[baikpef]
```

Ostali moduli su:

datagen: generiranje podataka

util: razne pomoćne metode

features: generiranje značajki

nnet: sve za neuronske mreže

passgen: generator šifri

Za detaljni opis metoda, pogledati dokumentaciju unutar koda.