VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERIŲ MOKSLO KATEDRA

Bakalaurinis darbas

Interneto vartotojų atpažinimas

(Web users identification)

Atliko 4 kurso 2 grupės studentas
Arnas Šleivys

(parašas)

Darbo vadovas:
lekt. Arūnas Janeliūnas

(parašas)

Recenzentas:
dr. Haroldas Giedra

(parašas)

Vilnius

2019

Turinys

Įvadas

Šiandien gyvename išmaniame pasaulyje – kineskopinius televizorius pakeitė išmanieji su protingomis operacinėmis sistemomis, klavišinius telefonus išstūmė lietimu valdomi išmanieji telefonai, kurių funkcionalumas prilygsta kompiuteriui ir prie kurių praleidžiame tiek daug laiko. Jei prieš dešimtmetį kompiuteris šeimoje buvo prabanga, tai šiais laikais neretai kiekvienas turime po savo asmeninį kompiuterį. Viskas po truputį, nejučiomis, persikelia į internetinį pasaulį – socialiniai tinklai, naujienos, paštas ir t.t. – visame tame dalyvaujame kasdien ir praleidžiame gan nemažai laiko.

Šis, smarkiai išaugęs, ir toliau augantis, interneto vartotojų kiekis, iškėlė naujų klausimų kompiuteriniam saugumui. Kaip žinoti, ar už ekrano sėdintis žmogus yra tas, kuriuo dedasi esąs? Būtent ieškant atsakymo į šį klausimą, imta žvalgytis į biometrines apsaugos priemones. Viena tokių – klaviatūros paspaudimų dinamika. Ji remiasi naudotojo atpažinimu pagal tam tikrą, unikalų ir būtent tam žmogui įprastą spausdinimo klaviatūra ritmą bei struktūrą. Spausdinant klavišais, uždelsimas tarp dviejų paspaudimų, pačio paspaudimo trukmė bei pirštų išdėstymas kuria unikalų vartotojo parašą. Šis parašas yra kiekvienam individualus – taip pat, kaip piršto antspaudas, ir galintis padėti atsakyti į iškeltą klausimą.

1. Klaviatūros paspaudimų dinamika laisvam tekstui

Didžioji dalis paspaudimų dinamikos tyrimų [1, 2, 3] yra grįsti stebėjimais iš anksto nustatytame, fiksuotame tekste – pavyzdžiui slaptažodis, el. pašto adresas ir t.t. Visas šis stebėjimas bei atpažinimas vykdavo prisijungimo metu. Po prisijungimo proceso, nėra galimybės toliau stebėti, ar tai tas pats vartotojas, kuris prisijungė – to sprendimas galėtų būti prašymas prisijungti kas tam tikrą laiko tarpą, tačiau tai erzintų vartotojus ir sistema būtų nepatogi. Norint sukurti patikimą sistemą interneto vartotojų atpažinimui, turėtų būti galima tą atpažinimą vykdyti ne tik mažam gabaliukui teksto, tačiau nenutrūkstamai visuomet, kai subjektas spausdina klaviatūra.

Tai, kaip vartotojas spausdina klaviatūra, galima išskirti į paspaudimų savybes laiko prasme, kurios gali būti panaudotos jo identifikacijai. Būtent šis procesas ir vadinasi klaviatūros paspaudimų dinamika. Fiksuoto teksto sistemoms, dažniausiai naudojamos savybės yra „skrydžio trukmė“ (angl. flight time) – laiko tarpas tarp dviejų paspaudimų bei „nuspaudimo trukmė“ (angl. dwell time) – kiek laiko mygtukas buvo įspaustas [4]. Šiame darbe bus naudojamos šiek tiek kitokios paspaudimų savybės. Visas tyrimas išskirtas į dvi pagrindines dalis: pirmoji dalis – duomenų surinkimas realiame internetiniame puslapyje, antroji – tų duomenų apdorojimas vartotojo spausdinimo modelio kūrimui bei atpažinimui.

1.1. Panašūs tyrimai

Panaršius internete, tyrimų, skirtų būtent laisvo teksto autentifikacijai, nebuvo daug. Dauguma jų dirba su mano ankstesniuose darbuose aprašytu CMU benchmark [5] duomenų rinkiniu ar kitais statinio teksto duomenimis. Tai leidžia tirti skirtingus klasifikatorių ar detektorių algoritmus, lyginti jų efektyvumą ir rezultatus, tačiau realaus pasaulio uždavinyje iš to – menka nauda.

Vienas pirmųjų darbų, kurį pavyko rasti šia tema, buvo parašytas 1997 metais straipsnyje „Authentication via keystroke dynamics“ [6]. Šiame tyrime buvo pasinaudota 31 vartotojo paspaudimais, kaip duomenų rinkiniu. Jiems buvo leidžiama rašyti arba tekstą laisva forma, arba rašyti vieną iš pateiktų frazių. Šiuose vartotojų profiliuose buvo skaičiuojamas digrafų trukmės vidurkis bei standartinis nuokrypis. Formuojant profilį, kiekviena trukmė tarp paspaudimų bei pačio paspaudimo ilgis yra lyginamas su atitinkamu vidurkiu ir visos reikšmės didesnės, nei T-standartinis nuokrypis yra išmetamos. Pirmajame bandyme autoriai naudojo Euklidinės erdvės atstumą tarp testavimo ir spėjimo profilių. Bandymų profilis yra klasifikuojamas kaip priklausantis vartotojui, kurio spėjimų profilis pateikia mažiausią Euklidinės erdvės atstumą. Kitame bandyme jie pasinaudojo digrafų svoriais, kad suteikti daugiau svarbos toms raidžių kombinacijos, kurios dažniau pasirodo. Šiame tyrime geriausia ką jiems pavyko pasiekti tai 90% tikslumas klasifikavime, kai naudojamas statinis tekstas. Tačiau kai šis metodas naudojamas laisvame tekste, efektyvumas krenta dramatiškai – iki 23% geriausiu atveju.

2013 metais autoriai Ahmed A. Ahmed bei Issa Traore pristatė labai panašaus i šį darbą tyrimo rezultatus [7]. Jie pabandė panaudoti neuroninį tinklą su monografų ir digrafų analize, kad atspėti trūkstamus digrafus priklausomai nuo santykio su stebėtais paspaudimais. Jie teigia, kad jų pasiūlytas modelis sugeba pasiekti labai pavydėtinus rezultatus su labai mažu mokymosi laiku. Jie panaudojo 53 vartotojų spausdinimų duomenis ir sugebėjo pasiekti EER (angl. Equal error rate) lygų 2.46%. Taip pat teigiama, kad sekančiame eksperimente, EER nukrito iki 2.13% panaudojant tik 17 vartotojų duomenis. Vis dėlto neaišku, kokį konkrečiai neuroninio tinklo modelį jie naudojo ir kiek paspaudimų turėjo kiekvienas vartotojas, nes publikacija yra privati.

1.2. Darbo tikslas

Šio darbo esmė yra sukurti pamatus universaliai, moderniai bei intuityviai aplikacijų programavimo sąsajai (angl. Application Programming Interface, API) interneto vartotojų atpažinimui, kuri remtųsi klaviatūros paspaudimų dinamika. Ji galėtų būti panaudota bet kuriame internetiniame puslapyje, bet kurioje puslapio vietoje, net ir su minimaliomis programavimo žiniomis. Svarbu nustatyti sunkumus, kylančius kuriant tokią sistemą bei rasti metodiką pasiekti rezultatą, kad sistema būtų veiksminga. Taigi, šio darbo tikslai būtų:

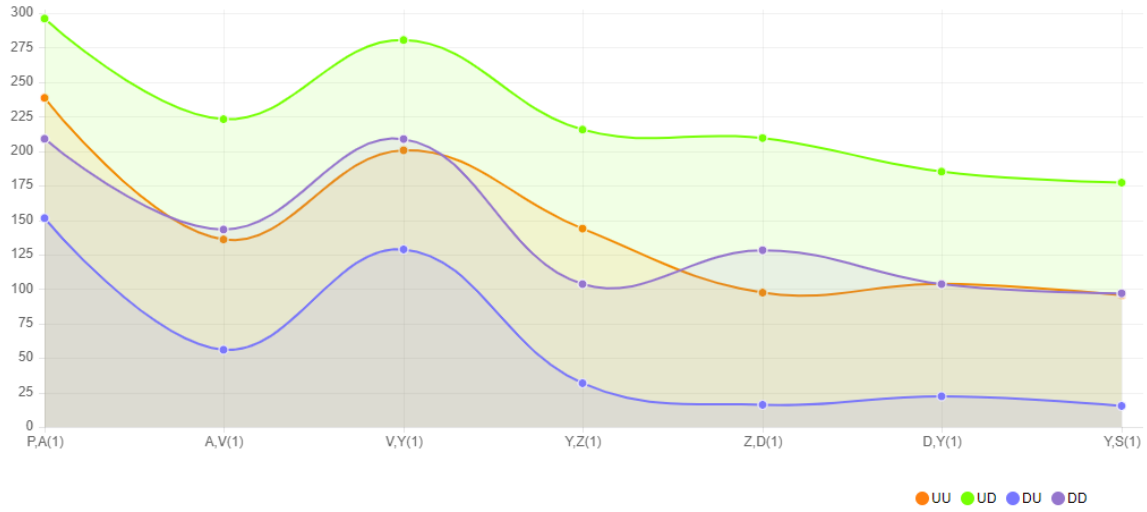
- Apibrėžti, kokiu formatu bus renkami ir saugomi vartotojų duomenys – t.y. kokios paspaudimų savybės bus stebimos
- Rasti būdą, kaip tuos duomenis surinkti vartotojo naršyklėje ir nusiųsti į serverį tolimesnei analizei
- Sukurti serverinę dalį, kuri apdorotų užklausas, ateinančias iš vartotojų naršyklių bei pagal jas atliktų tolimesnius veiksmus
- Rasti veiksmingą būdą, kaip sukurti vartotojų spausdinimo modelius, kurie būtų panaudoti identifikacijai
- Ištirti, kokie gali kilti sunkumai bei į ką reikėtų atkreipti dėmesį implementuojant tokią sistemą pilnu pajėgumu realiame internetiniame puslapyje

2. Duomenų formavimas

Daugumoje anksčiau atliktų tyrimų šia tema [4], paspaudimai buvo matuojami pagal du kriterijus:

- DT (angl. down-time) – nuspaudimo trukmė
- FT (angl. flight-time) – laikas tarp dviejų nuspaudimų

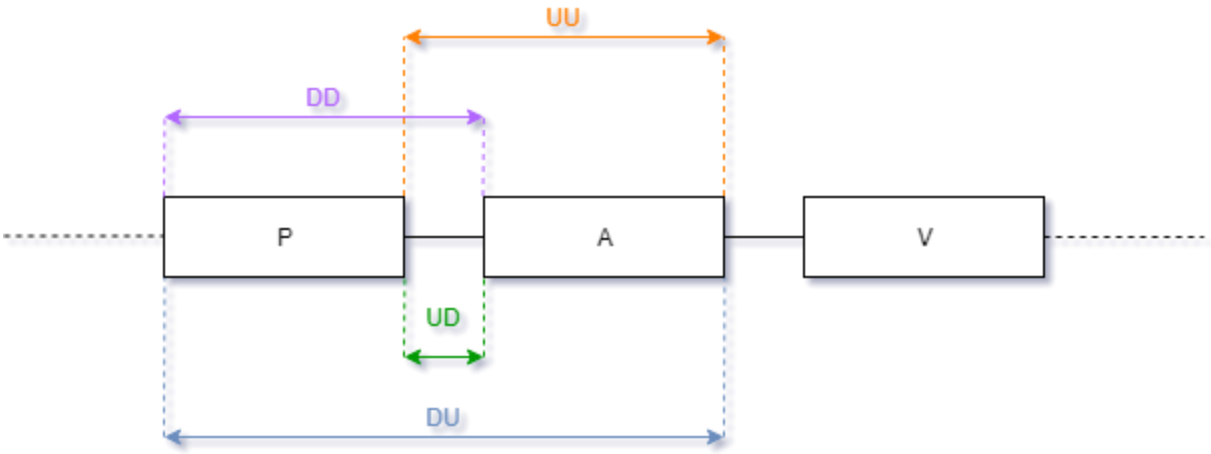
Šiame darbe buvo nuspręsta paspaudimus matuoti kaip digrafus:



pav. 1 Diagrama, vaizduojanti paspaudimus ir jų matavimo vienetus

Pavyzdyje parodyti matavimai, atlikti rašant žodį „pavyzdys“. Kairėje matomas laikas, milisekundėmis. Apačioje pavaizduoti digrafai – klavišų poros. Kiekvienas mygtuko nuspaudimas tampa klasės objektu, su anksčiau minėtais laukais, o tokie du objektai yra formuojami į digrafą, sudarytą iš 6 laukelių:

- Pirmo klavišo kodas
- Antrojo klavišo kodas
- UU (keyup-keyup, grafike pažymėta oranžine spalva) – laiko tarpas tarp pirmojo ir antrojo klavišo atleidimo.
- UD (keyup-keydown, grafike pažymėta žalia spalva) – laiko tarpas tarp pirmojo klavišo atleidimo ir antrojo nuspaudimo.
- DU (keydown-keyup, grafike pažymėta mėlyna spalva) – laiko tarpas tarp primojo klavišo nuspaudimo ir antrojo atleidimo.
- DD (keydown-keydown, grafike pažymėta violetine spalva) – laiko tarpas tarp pirmojo ir antrojo klavišo nuspaudimo.



pav. 2 Grafikas, kuris vaizduoja 4 mygtuko paspaudimo savybes

3. Duomenų rinkimas vartotojo naršyklėje

Pirmoji šio darbo fazė buvo sukurti veikiančią JavaScript programą, kad būtų galima rinkti realių vartotojų paspaudimų duomenis bei juos įrašyti į tam tikrą duomenų bazę.

3.1. Javascript panaudojimo galimybės duomenų rinkimui

JavaScript buvo sukurtas tam, kad padarytų internetinius puslapius „gyvais“. Kartu su HTML bei CSS, ši programavimo kalba sudaro puslapių kūrimo standarto modelį. [8] JavaScript neretai vadinama skriptingo kalba, nes programos čia vadinamos skriptais. Šie skriptai gali būti įkomponuoti tiesiogiai į puslapio HTML kodą ir paleisti automatiškai tada, kai puslapis užsikraus. Jie yra pateikiami bei vykdomi kaip paprastas tekstas – jiems nereikia jokio specialaus paruošimo ar kompiliavimo.

Vartotojo naršyklėje JavaScript gali daryti viską, kas susiję su puslapio manipuliacijomis, sąveika su vartotoju ar serveriu. Pavyzdžiui:

- Reaguoti į vartotojo veiksmus – pelytės paspaudimus, judesius, klaviatūros nuspaudimus
- Siųsti užklausas tinklu į nutolusius serverius (AJAX)
- Atsiminti duomenis kliento naršyklės pusėje (dar vadinama „lokalia atmintimi“)

Visi šie veiksmai vyksta vadinamąjame naršyklės DOM (Document Object Model).

3.2. DOM (Document Object Model)

Kai puslapis užkraunamas, naršyklė sukuria puslapio DOM. Pats DOM nėra programavimo kalba, bet be jo, JavaScript neturėtų jokio modelio ar supratimo apie internetinį puslapį, HTML dokumentus, XML dokumentus ir jų komponentus (elementus). Kiekvienas dokumento elementas – pats dokumentas, *head* elementas, *table* esantys dokumente, *table headers*, tekstas lentelėse – viskas yra dalis DOM, sukurto tam dokumentui. Visi jie gali būti pasiekti ir manipuliuojami naudojantis DOM ir skriptingo kalba, tokia kaip JavaScript. [9]

DOM buvo sukurtas taip, kad būtų nepriklausomas nuo jokios konkrečios programavimo kalbos. Tai leido dokumento struktūrinę reprezentaciją pasiekti iš vieno, pastovaus API. Atsiradus DOM, jis buvo glaudžiai susijęs su JavaScript, tačiau galiausiai jie išaugo į atskiras esybes. Internetinio puslapio turinys yra talpinamas DOM ir gali būti pasiektas bei naudojamas per JavaScript.

Skirtingos naršyklės naudoja skirtingas DOM implementacijas, kurios turi skirtingus atitikimo lygius pagal DOM standartą, tačiau kiekviena interneto naršyklė naudoja tam tikrą *Document Object Model*, kad puslapius būtų galima pasiekti su JavaScript. Pavyzdžiui, šiame darbe buvo naudojami du metodai, darantys tą patį, bet skirtingai aprašomi naršyklėse:

- addEventListener – standartinė JavaScript funkcija, palaikoma IE9+ ir visų kitų naršyklių
- attachEvent – nestandartinė funkcija, naudojama IE8 ir senesnėse naršyklėse

Kai internetiniame puslapyje užkraunamas skriptas, galima iš kart pradėti naudoti DOM API, kad pasiekti *document* ar *window* elementus, kas leidžia pradėti darbą su pačiu dokumentu ar gauti to dokumento vaikus – įvairius internetinio puslapio elementus.

3.3. addEventListener ir attachEvent metodai

Event yra svarbi JavaScript dalis. Internetinis puslapis duoda atsaką, priklausomai nuo to, koks įvykis (angl. event) įvyko. Kai kurie įvykiai yra sugeneruoti vartotojo, o kai kurie API. *Event listener* yra JavaScript procedūra, kuri laukia įvykio atsiradimo. Paprasčiausias to pavyzdys būtų – vartotojas paspaudžia pelytę ar klaviatūros klavišą.

addEventListener yra vidinė JavaScript funkcija, kuri „klauso“, kol įvyks tam tikras įvykis ir, tam atsitikus, iškviečia antruoju argumentu nurodytą funkciją. Prie vieno elemento galima pridėti neribotą skaičių *event listener*, neperrašant senesniųjų.

3.4. keydown ir keyup metodai

JavaScript kalboje yra keturi standartiniai metodai, skirti sekti vartotojo klaviatūros būseną, tačiau šiame darbe naudojome du:

- keydown – iškviečiamas, kai vartotojas nuspaudžia mygtuką. Jei mygtukas laikomas įspaustas ir neatleidžiamas, metodas kartojamas.
- keyup – iškviečiamas, kai vartotojas atleidžia nuspaustą mygtuką.

Įvykus *keydown* arba *keyup* įvykiui, gauname *Event* objektą, kuriame slypi informaciją apie mygtuko paspaudimą:

- event.keyCode – grąžina mygtuko, kuris iššaukė šį *Event*, Unicode simbolio kodą. Šis kodas simbolizuoja ne ASCII kodą, o skaičių, kuris atstoja realų mygtuką klaviatūroje.
- event.timeStamp – trukmė, nusakanti, kiek milisekundžių skiria įvykį nuo 1970 metų sausio 1 dienos vidurnakčio. JavaScript kalboje naudojama nustatyti laiko žymą, kada įvyko *Event*. Vis dėlto, ne visos sistemos suteikia šią informaciją, todėl ši savybė gali būti neprieinama kai kuriose sistemose. Viena tokių – *Android* operacinė sistema. Būtent todėl laiko nustatymui joje buvo naudojama kitas JavaScript metodas – *getTime,* kuris atlieka lygiai tą patį.

3.5. Javascript implementacija duomenų rinkimui

Programa buvo pradėta kurti nuo jau anksčiau aprašytų metodų *keyup, keydown* bei *addEventListener* metodų įgyvendinimo. Idėja tokia, kad kai vartotojas klaviatūroje paspaudžia mygtuką, iškviečiamas metodas, atsakingas už *keydown* įvykio funkcionalumą. Šiame metode sukuriamas naujas objektas, kuris savyje laiko informaciją apie nuspausto mygtuko kodą bei laiką, kada jis buvo nuspaustas. Mygtuką atleidus, į tą patį objektą įrašomas laukelis, skirtas laikui, kada mygtukas buvo atleistas. Taip gauname masyvą, sudarytą iš objektų, kuriuose saugoma informacija apie mygtuko kodą bei nuspaudimo ir atleidimo laiko štampus.

Vėliau buvo pastebėta keletas nestandartinių situacijų:

- „Android“ operacinės sistemos nesuderinamumas. Šioje operacinėje sistemoje nesvarbu koks mygtukas nuspaudžiamas, jo kodas visuomet rodomas kaip 229. Tai yra žinoma problema programuotojų tarpe, tačiau ji neišspręsta iki šiol. Taip pat, čia neveikia standartinis *event.timeStamp* metodas, kuris naudojamas programoje. Būtent todėl reikėjo sukurti atskiras funkcijas sekti „Android“ įrenginio klaviatūros paspaudimus, pasinaudojant kitais JavaScript elementais.
- Ne visi mygtukai turi atitinkamus kodus. Testinėje aplinkoje taip pat buvo pastebėta, jog tam tikri mygtukai kaip kodą grąžina neapibrėžtą reikšmę – taip yra todėl, kad ne visiems mygtukams yra priskirtos reikšmės. Šios problemos sprendimas buvo nurodyti leidžiamų mygtukų kodų masyvą, ir jei nuspaustas mygtukas į jį nepatenka – jam nustatyti veliavėlę (angl. flag), kad jo nereikia sekti.
- Įspausto mygtuko laikymas. JavaScript kalboje yra padaryta, kad jei mygtukas laikomas įspaustas, po tam tikro laiko iššaukiamas naujas *event*, o kartu su juo ir nauja nuspaudimo trukmė ir naujas objektas. To neištaisius, sekimo duomenys būtų iškraipyti. Programoje buvo atlikti pakeitimai, kad jei prieš tai buvęs mygtukas neturi įrašytos atleidimo laiko žymos ir yra toks pat, kaip dabartinis – nustatoma vėliavėlė, jog tai yra įspaustas mygtukas ir naujasis įspaudimas ignoruojamas.

4. Duomenų siuntimas į serverį

HTTP (angl. Hypertext Transfer Protocol) yra sukurtas tam, kad būtų galima komunikacija tarp klientinės pusės ir serverio. HTTP veikia kaip prašymo-atsakymo (angl. request-response) protokolas tarp kliento ir serverio. Klientu gali būti, pavyzdžiui, interneto naršyklė, o serveriu - kompiuterio aplikacija, kuri talpina internetinį puslapį. Du dažniausiai naudojami HTTP metodai yra GET ir POST. Siunčiamos užklausos metodas nurodo funkciją, kurią reikia atlikti ant užklausos URI identifikuoto resurso (URI – angl. Uniform Resource Identifier, kuris identifikuoja resursą, kuriam taikyti užklausą).

4.1. GET užklausa

GET metodas yra naudojamas norint išgauti informaciją iš serverio pasinaudojant duotuoju URI. Užklausos, naudojančios GET turi tik grąžinti duomenis ir neturėti jokio kito efekto kitiems duomenims. Šios užklausos parametrai paduodami kartu su URL adresu, pavyzdžiui:

*arnas.site/api.php?**name1=value1&name2=value2***

GET užklausai būdingos šios savybės:

- Gali būti kešuojama ir išlikti naršyklės istorijoje
- Nederėtų naudoti dirbant su jautriais duomenimis
- Turi ilgio suvaržymus
- Naudojama tik duomenų užklausai (bet ne modifikacijai)

4.2. POST užklausa

POST metodas nusiunčia duomenis, kurie yra jo *body* elemente, į serverį. Jis naudojamas tam, kad serveryje būtų galima sukurti ar atnaujinti resursą. Elemento *body* tipas užklausoje yra nurodamas *header* komponento *Content-Type*. Nurodžius jį kaip *application/x-www-form-urlencoded* duomenų raktai ir reikšmės yra koduojami kaip raktas-reikšmė rinkiniai atskirti simbolio '&'. Raktas su reikšme yra atskiriami ženklu '='.

POST užklausai būdinga:

- Jos nėra kešuojamos bei nelieka naršyklės istorijoje
- Jų ilgiui nėra jokių suvaržymų
- Naudojama duomenų sukūrimui/atnaujinimui
- Nėra jokių suvaržymų duomenų tipui

4.3. Duomenų siuntimas per HTTP užklausą

JavaScript kalboje *HTTP* užklausoms siųsti yra naudojama klasė *XMLHttpRequest*. Ji leidžia keistis duomenimis tarp internetinio puslapio ir serverio. Norint išsiųsti tokią užklausą, reikia sukurti *XMLHttpRequest* objektą, nurodyti *URL*, į kurį bus siunčiama užklausa bei užklausos *body* – t.y. klaviatūros paspaudimų stebėjimo rezultatus. Įvykus transakcijai, gausime

objektą, kuris savyje laikys naudingą informaciją – tokią kaip atsakymo turinį bei rezultato *HTTP* būseną.

Šiame darbe klaviatūros stebėjimas vyksta nuolat – t.y. ne tik tada, kai vartotojas rašo tekstą i tam tikrą, anksčiau nurodytą teksto laukelį. Paspaudimų duomenys vienas po kito „gula“ į klasės objektus, iš kurių vėliau bus konstruojami digrafai. Šis konstravimas ir duomenų siuntimas vyksta tada, kai vartotojas nori pereiti į kitą puslapį. To stebėjimui pasinaudota anksčiau minėtais *addEventListener* bei įvykiu *beforeunload*. Jis iškviečiamas tada, kai dokumentas ir jo resursai yra keitimosi būsenoje (prieš pereinant į kitą puslapį). Pastebėjus tokią būseną, iškviečiama duomenų siuntimo funkcija, kuri sukonstruoja digrafus bei *HTTP POST* užklausos struktūrą. Kaip jau minėta anksčiau, nustačius antraštės tipą kaip *application/x-www-form-urlencoded*, duomenys bus koduojami taip:

id=<*vartotojo ID*>**&data=**<*paspaudimų duomenys*>**&mobile=**<*0 arba 1*>

Su rezultatais siunčiama ir mobilaus įrenginio vėliavėlė (angl. flag) – t.y. ar duomenys surinkti mobiliąjame įrenginyje, ar ne. Taip pat, siunčiant užklausą nurodome, kad ji bus asinchroniška – tai leis interneto naršyklei dirbti be trikdžių, kol užklausa bus vykdoma. Jei to nenurodytumėme – naršyklė lauktų, kol užklausa bus įvykdyta ir visi kiti darbai būtų sustabdyti.

id	user_id	user_timings	mobile
1	1623bf9d01c8c6d5395533218c4cd44a	39,39,1888,1929,2046,1771 39,39,670,635,787,518 39...	0
2	673d18fc134535d5848f0168e79360a1	80,73,192,191,255,128 73,82,320,303,384,239 82,75,...	0
3	d9d20d0591acf7b76865929208048da9	84,69,119,107,196,31 69,75,182,166,270,78 75,73,24...	0
4	01d5e771ad1727a5ad3a904b51953afb	82,69,1046,1059,1065,1040 69,68,835,834,841,828 68...	1
5	132bda2c97dc49847179745f4f064db3	39,39,3312,3351,3543,3119 39,37,695,728,888,535 37...	0
6	4be3e68fe1c25304b51b39ec3eeeaefe	83,78,465,464,471,458 78,73,248,251,255,244 73,80,...	1

pav. 3 Vartotojų duomenų, esančių duomenų bazėje, struktūra

5. Gautų duomenų apdorojimas serveryje

Gautus duomenis iš HTTP užklausos serveris apdoroja pasinaudodamas PHP skriptingo kalba. Ši kalba skirta serverio pusei, sukurta specialiai interneto puslapių kūrimui. PHP yra lengvai įkraunama į HTML failą ir taip pat HTML kodas gali būti rašomas į PHP failą. Skirtingai nei klientinės pusės kalba, tokia kaip HTML, PHP kodas yra vykdomas serveryje, o ne tiesiogiai interneto naršyklėje.

5.1. PHP metodai darbui su duomenų baze

PHP leidžia prisijungti ir dirbti su duomenų bazėmis. Populiariausia iš jų – MySQL duomenų bazė [10]. Šioje duomenų bazėje informacija yra saugoma lentelėse. Lentelėje yra rinkinys susijusių duomenų, kuris susideda iš stulpelių ir eilučių. Norint įrašyti duomenis į MySQL duomenų bazę pasinaudojant PHP, naudojami šie metodai:

- *mysqli_connect* – naudojama prisijungimui prie duomenų bazės. Parametruose nurodoma serverio IP adresas, prisijungimo vardas bei slaptažodis ir lentelės pavadinimas.
- *mysqli_real_escape_string* – ši funkcija prideda specialų „pabėgimo“ simbolį prieš tam tikrus, pavojingus, simbolius *string* tipo kintamąjame, paduodamame į funkciją. Tai padeda išvengti SQL injection atakų, kurios dažniausiai įvykdomos pasinaudojant ' simboliu, kad pridėti kenksmingą SQL kodą į užklausą.
- *mysqli_query* – išsiunčia SQL užklausą į serverį. Grąžina rezultatą ar užklausa pavyko, ar ne.
- *mysqli*_close – uždaro jungtį su duomenų baze.

5.2. PHP *Super*-globalūs kintamieji

PHP kalboje yra keletas kintamųjų, kurie yra vadinami *superglobals*. Tai reiškia, kad jie yra visuomet pasiekiami, nepriklausomai nuo programos *scope* – juos galima pasiekti iš bet kurios funkcijos, klasės ar failo, be jokių papildomų specialių veiksmų. Tokių kintamųjų yra keletas:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_COOKIE
- $_SESSION

$_SERVER yra PHP *super*-globalus kintamasis, kuris savyje turi informaciją apie antraštes (angl. *headers*), kelius (angl. *paths*) bei skriptų lokacijas. Savyje jis turi informaciją ir apie užklausos metodą, kuris buvo panaudotas iškviečiant puslapį (pavyzdžiui *GET* ar *POST*).

$_POST taip pat yra *super*-globalus kintamasis, savyje turintis visą informaciją, kuri buvo siųsta pasinaudojant *POST* metodu. Šis metodas informaciją perduoda per *HTTP* antraštes

(konkrečiai – per *QUERY_STRING* antraštę). Siunčiami duomenys pereina per *HTTP* antraštę, tad jų saugumas priklauso nuo naudojamo saugumo protokolo.

6. Duomenų paruošimas vartotojo spausdinimo modelio kūrimui

Realiame internetiniame puslapyje, kuriame apsilanko didelis srautas vartotojų, natūraliai turėsime didelius kiekius duomenų. Šiame darbe duomenų rinkimo skriptas buvo patalpintas į vieną didžiųjų Lietuvos naujienų portalų ir per savaitę buvo surinkta virš 6 tūkstančių unikalių vartotojų įrašų. Esant tokiems kiekiams duomenų, reikia apgalvoti, kaip ir kada skaityti vartotojų paspaudimų duomenis bei juos apdoroti mokymams, spausdinimo modelių kūrimams.

Duomenų nuskaitymo procesas bus kartojamas du kartus – vieną kartą duomenims, surinktiems kompiuteryje, kitą – mobiliąjame įrenginyje (tam bus pasitelkta *mobile* vėliavėlė). Šio failo struktūra tokia, kaip pavaizduota 4 paveikslėlyje – t.y. viename stulpelyje bus vartotojo ID, o kitame – visa jo paspaudimų informacija išsaugota kaip *String* laukelis. Prieš išsiunčiant duomenis į neuroninį tinklą mokymams, jie yra pertvarkomi į tokį formatą:

secondKey	UU	DD	UD	DU
73	511	511	591	432
69	391	359	471	279
84	416	424	528	311
85	359	384	464	279
86	424	407	503	327
79	464	472	560	375
83	287	280	376	191
32	239	271	335	175
80	544	528	608	464
73	360	368	440	288
76	455	464	528	391
73	287	279	351	215
69	360	336	432	264

pav. 4 Vartotojų duomenys .csv faile

Čia žymima pirmojo bei antrojo mygtukų kodai bei anksčiau aptartos paspaudimų laiko savybės. Toks failas yra sugeneruojamas kiekvienam vartotojui. Pradinis .csv failas pagaminamas pasinaudojant *Python* programavimo kalba – aprašomi duomenų bazės prisijungimo parametrai, užrašoma *SQL* užklausa ir išsiunčiama į serverį. Failo pertvarkymas užrašytas *JAVA* kalboje - *.csv* failas išskaidomas į atskiras eilutes, kurios dar atskiriamos į dalis atsižvelgiant į kablelio poziciją. Tada viskas sujungiama į tokį formatą, kuris pavaizduotas paveikslėlyje. Turint duomenis tokiame formate, galima pradėti mokymus.

Per visą duomenų rinkimo laikotarpį buvo surinkta virš 6000 unikalių vartotojų įrašų. Iš jų – apie 5000 surinkti kompiuteryje, o likę 1000 – mobiliajame įrenginyje. Vidutiniškai, vienas

vartotojas kiekviename įraše turėdavo po 40-50 paspaudimų. Lentelėje pavaizduota, koks skaičius realių vartotojų turėjo atskirų įrašų bei apytiksliai bendrai paspaudimų duomenų bazėje:

Įrašų skaičius	Paspaudimų skaičius	PC lankytojai	Mobile lankytojai
>20	800-3000	19	1
>15	600-1000	10	2
>10	400-750	24	5
>5	200-500	106	21
5	250	41	10
4	200	77	10
3	150	154	32
2	100	405	78
1	40	1776	406

pav. 5 Surinktų duomenų analizė

Iš JavaScript pusės duomenų rinkime buvo sekami ne visi, o tik tam tikri mygtukai. Tokių buvo 90. Taip pat, buvo išfiltruoti tokie paspaudimai, kurių trukmė milisekundėmis yra keturženklis arba didesnis skaičius – sekundės ir daugiau laiko tarpas yra per didelis, kad būtų spausdinama nenutrūkstamai – veikiausiai vartotojas padarė pauzę. Tokie paspaudimai galimai iškraipytų tinklo modelį ir neatspindėtų realaus vartotojo spausdinimo ritmo.

7. Mokymosi algoritmo pasirinkimas

Paspaudimų dinamikos tyrimuose vyrauja daug skirtingų algoritmų [12], kurie parodo gan įvairius rezultatus. Norint, kad mokymai bei atpažinimas būtų efektyvus, svarbu rasti tinkamiausią metodą. Praeitame tyrime buvo išbandyti keletas iš geriausius rezultatus rodančių detektorių:

Detektorius	EER (tik laiko matavimas)	EER (laiko ir ekrano paspaudimų matavimas)
Euklidinės erdvės	18.2%	16.2%
Manheteno	15.7%	13.3%
Mahalanobio	24.1%	17.4%
Manheteno + Mahalanobio	8.9%	8.1%
Deep Belief Nets	4.2%	3.4%

Čia EER - klaidos tikimybė tame ROC kreivės (sprendimus priimančiojo ypatybių kreivė – grafikas, rodantis klasifikatoriaus jautrumo ir specifiškumo sąryšį) taške, kur klaidingo blokavimo bei klaidingo įleidimo tikimybės yra lygios. Trumpiau tariant - tai tarsi „aukso viduriukas“, kur tikro vartotojo neįleidimo į sistemą bei apsimetėlio įleidimo procentai yra lygūs. Taigi, tuomet geriausiai pasirodė naujasis Deep Belief Nets detektorius, su pavydėtinai maža 3-4% klaidos tikimybe. Tačiau išbandžius jį su šio tyrimo duomenimis, pastebėta, kad modelio mokymas užima gan daug laiko, o ir rezultatai nėra tokie geri, kokie buvo gauti praeitame tyrime (~10% EER). Šio detektoriaus implementavimas dabartiniame tyrime buvo atmestas pagrinde dėl apsimokymo laiko trukmės – tai labai svarbus faktorius, turint omenyje, kad tokia sistema gali būti naudojama ir ypač dideliuose internetiniuose portaluose. Todėl teko pasidomėti, kokie šiuo metu populiauriausi detektoriai, kokia jų pagrindinė paskirtis ir pabandyti rasti tinkamiausią tokio tipo sistemai.

8. Rekurentinis neuroninis tinklas

Rekurentinis tinklas yra tokio tipo dirbtinis neuroninis tinklas, kuris sukurtas atpažinti išsidėstymus duomenų eilėse, tokiose kaip tekstas, ištartas žodis ar laiko duomenys [13]. Šie algoritmai atsižvelgia į laiką, išsidėstymą bei turi laikiną atmintį. Skirtingai nuo kitų neuroninių tinklų, šie tinklai turi dvi įvestis – dabartinę bei praeitį, jas sukombinavus nusprendžiama, kaip tinklas reaguos į naujus duomenis. Tai specializuotas neuroninis tinklas naudojamas apdirbti sekvencinius duomenis – paspaudimų duomenys yra taip pat generuojami sekvenciniu būdu, tad tai protingas pasirinkimas klaviatūros paspaudimų dinamikai. Šis metodas parodė puikius rezultatus ir kitose sekvencinių duomenų aplikacijose, tokiose kaip teksto bei balso atpažinimas, laiko duomenų atpažinime. [14,15]

Lyginant su kitais statistiniais ar mašininio mokymosi metodais, šis tinklas pranašus tuom, kad:

- Galima identifikuoti įsilaužėlius tiksliau, nei kituose metoduose. Dažniausiai mokymams reikia gan didelio kiekio paspaudimų duomenų, kad autentifikuoti vartotojus, tačiau šiam metodui pakanka nedidelio paspaudimų duomenų rinkinio. Tas leidžia gan greitai integruoti naujus vartotojus į sistemą.
- Atpažinimo modelis yra stiprinamas naudojant tikrojo vartotojo paspaudimų duomenis, surinktais po treniruočių atnaujinant modelį. Daugelyje kitų mašininio mokymosi algoritmų, atsiradus naujiems duomenims modelis turi būti permokytas nuo nulio. Rekurentiniam neuroniniam tinkle yra galimybė atnaujinti esamą konfigūraciją, pakoregavus tinklo svorius, atsižvelgiant į naujai atsiradusius duomenis. Tai labai pravartu dideliame interetiniame puslapyje, kuriame yra didelis kiekis vartotojų ir duomenų – drąstiškai sumažėja apsimokymo laikai.

8.1. Pranykstantis arba sprogstantis gradientas

Gradientas parodo visų tinklo svorio pokytį, atsižvelgiant į klaidos kitimą. Jei gradientas tampa nežinomas, t.y. pranyksta arba sprogsta, negalima koreguoti tinklo svorių ta kryptimi, kuri mažins klaidos tikimybę ir toks tinklas nebegebės mokytis.

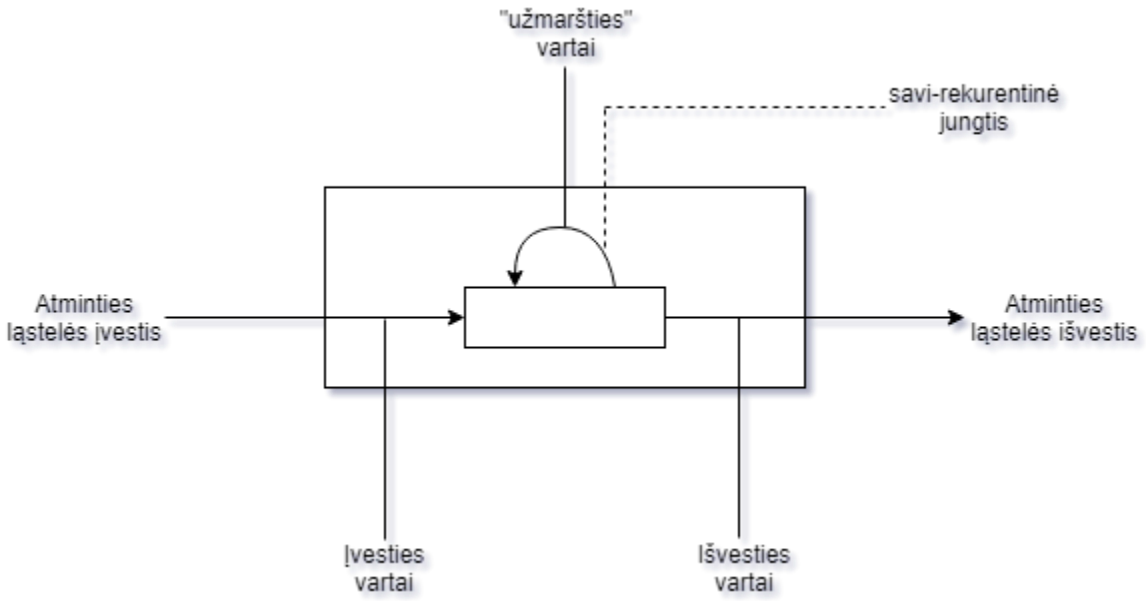
Gilių neuroninių tinklų sluoksniai vienas su kitu koreliuoja per daugybą. Jei dauginsime pakankamai dažnai iš šiek tiek daugiau nei vieno – galima greitai pasiekti labai dideles aukštumas. Taip pat ir su daugyba iš šiek tiek mažiau nei vieno – jei daugyba vyks pakankamai dažnai, greitai bus priartėta prie nulio. Taigi, kadangi neuroninių tinklų sluoksniai vienas su kitu bendrauja per daugybą, jų išvestinėms gresia išnykimas ar sprogimas.

Sprogstantis gradientas kiekvieną tinklo svorį vertina kaip per daug galingą. Tas gali būti išsprendžiama suspaudžiant ar nukertant gradientą. Tačiau jei jis yra išnykstantis – atsiranda problema, kad jis bus per mažas kompiuterio darbui ar tinklo mokymams – tą išspręsti sunkiau.

8.2. Ilgalaikės-trumpalaikės atminties vienetai (angl. LSTM)

Vokiečių mokslininkai Seep Hochreiter ir Juergen Schmidhuber gradiento problemai spręsti pasiūlė naudoti LSTM vienetus. Tai atminties vienetetas „su vartais“, skirtas neuroniniams tinklams. Jis turi 3 vartus, kurie valdo atminties turinį. Šie vartai yra logistinės svorių funkcijos, kur svoriai gali būti išmokti pasinaudojant atgalinio perdavimo grįžimą (angl. back-propogation).

LSTM puikiai tinka neuroninio tinklio mokymosi procesui – jis gali išmokti, ką jam reikia išmokti, atsiminti, ką reikia atsiminti, be jokių specialių treniruočių ar optimizacijų. Įvestie bei „užmaršties“ vartai kontroliuoja atminties celės būseną, kuris yra ilgalaikė atmintis. Išvesties vartai gamina išvesties vektorių arba „paslėptą“ būseną, kuri yra atmintis, skirta naudojimui. Toks veikimo principas leidžia tinklui prisiminti ilgalaikėje persepktyvoje, ko labai trūko paprastame rekurentiniame tinkle. Celės schema:

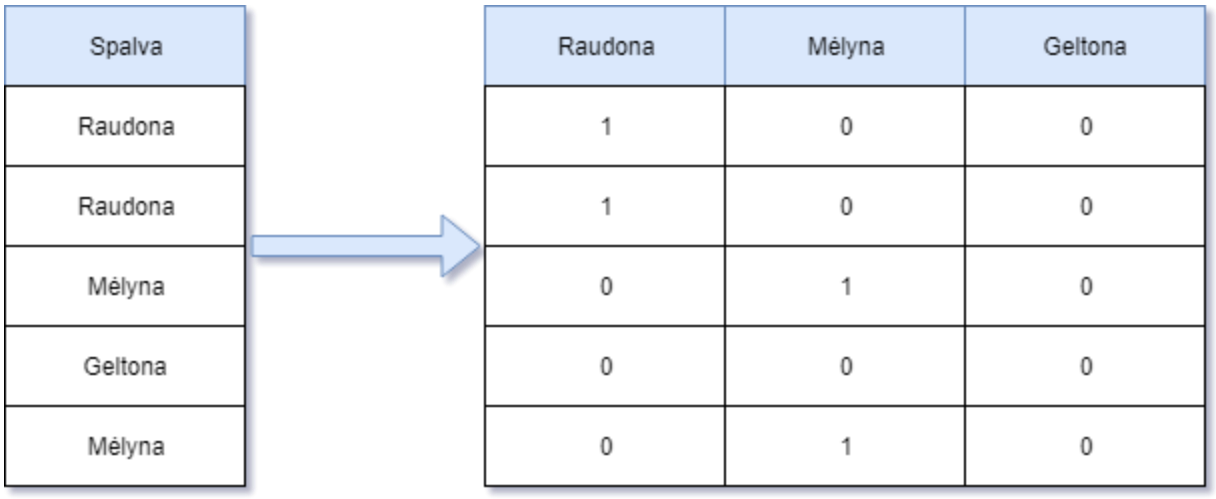


pav. 6 LSTM celės schema

8.3. One-hot vektoriaus kodavimas

Jei turime kategorinių savybių duomenų rinkinyje – pvz. spalvą, kuri gali būti raudona, geltona ar mėlyna – ir norime jas paduoti į mašininio mokymosi algoritmą, reikia jas užkoduoti, kad vietoj *String* tipo kintamųjų turėtumėme skaičius. Taigi, spalvas galima būtų pakeisti į atitinkamus skaičius 1, 2 arba 3. Tačiau toks būdas sukeltų pašalinių efektų, nes modelis galėtų nuspręsti, kad mėlyna > geltona (3 > 2) ar raudona + geltona = mėlyna. Modelis nežino, kad prieš transformaciją šie duomenys buvo kategoriški.

Šios problemos sprendimui ir yra skirtas one-hot vektoriaus kodavimas, kuris sukuria n naujų savybių, kur n – unikalių savybių skaičius (spalvų atveju – 3). Kiekviena šių savybių tampa binarinė ir atstovauja vieną iš unikalių reikšmių. Spalvų pavyzdyje pirmoji savybė būtų 1 arba 0, kuris parodytų, ar tai raudona spalva, ar ne.



pav. 7 One-hot kodavimo veikimo principas

Taigi, šioje sistemoje vartotojas paspaudęs klaviatūros mygtuką sukuria 90 (nes yra 90 sekamų mygtukų) dimensinį koduotą vektorių (angl. one-hot encoding).

9. TensorFlow

Visas šis projektas didžiąja dalimi remiasi į TensorFlow naudojimą. Tai – galingas atviro kodo mašininio mokymosi karkasas, skirtas duomenų tyrimams, neuroniniams tinklams kurti ir giliems mokymams (angl. deep learning). Šiame projekte buvo pasirinktas būtent šis karkasas mokymams, nes jis geba apmokytą neuroninį tinklą bei jo svorius išsaugoti atitinkamai į *.h5* bei *.json* formatus, kuriuos galima patalptinti į duomenų bazę.

9.1. Keras

Keras yra atviro kodo neuroninių tinklų biblioteka, parašyta Python programavimo kalba. Ji veikia kaip TensorFlow papildinys - interfeisas, supaprastinantis darbą su mašininiu mokymusi. Keras turi daugybę implementacijų dažniausiai naudojamų neuroninio tinklo kūrimo įrankių, tokių kaip sluoksniai, objektai, aktyvavimo funkcijos, optimizatoriai ir t.t., kas padaro darbą su paveikslėliais bei tekstiniais duomenimis lengvesnį. Kartu su standartiniais neuroniniais tinklais, jis palaiko konvoliucinius bei rekurentinius neuroninius tinklus, įskaitant ir tokias funkcijas kaip išmetimas (angl. dropout), imties normalizacija bei jungimas (angl. pooling).

9.2. CPU prieš GPU mokymosi procesui

Nors šiame tyrime su mokymosi greičio problema nebuvo susidurta, tačiau atsiradus dideliems kiekiams duomenų bei vartotojų, mokymosi procesas gali pradėti strigti. Tokiu atveju modelių mokymus reiktų perkelti ant GPU (angl. graphical processing unit). Pasirinkimą tarp CPU bei GPU galima būtų apibūdinti taip – jei internetinis puslapis yra nedidelis yra unikalių vartotojų per dieną nėra labai daug – mokymai gali vykti ir ant CPU, tačiau tokiems portalams kaip 15min.lt, vz.lt ir t.t. CPU naudojimas mokymams būtų neįmanomas. Jų skirtumai yra tokie:

- GPU turi paprastesnius branduolius, tačiau jų – šimtai. Taip pat jame lygiagrečiai dirba tūkstančiai gijų.
- CPU turi labai sudėtingus bei galingus branduolius, tačiau jų – vos keli.

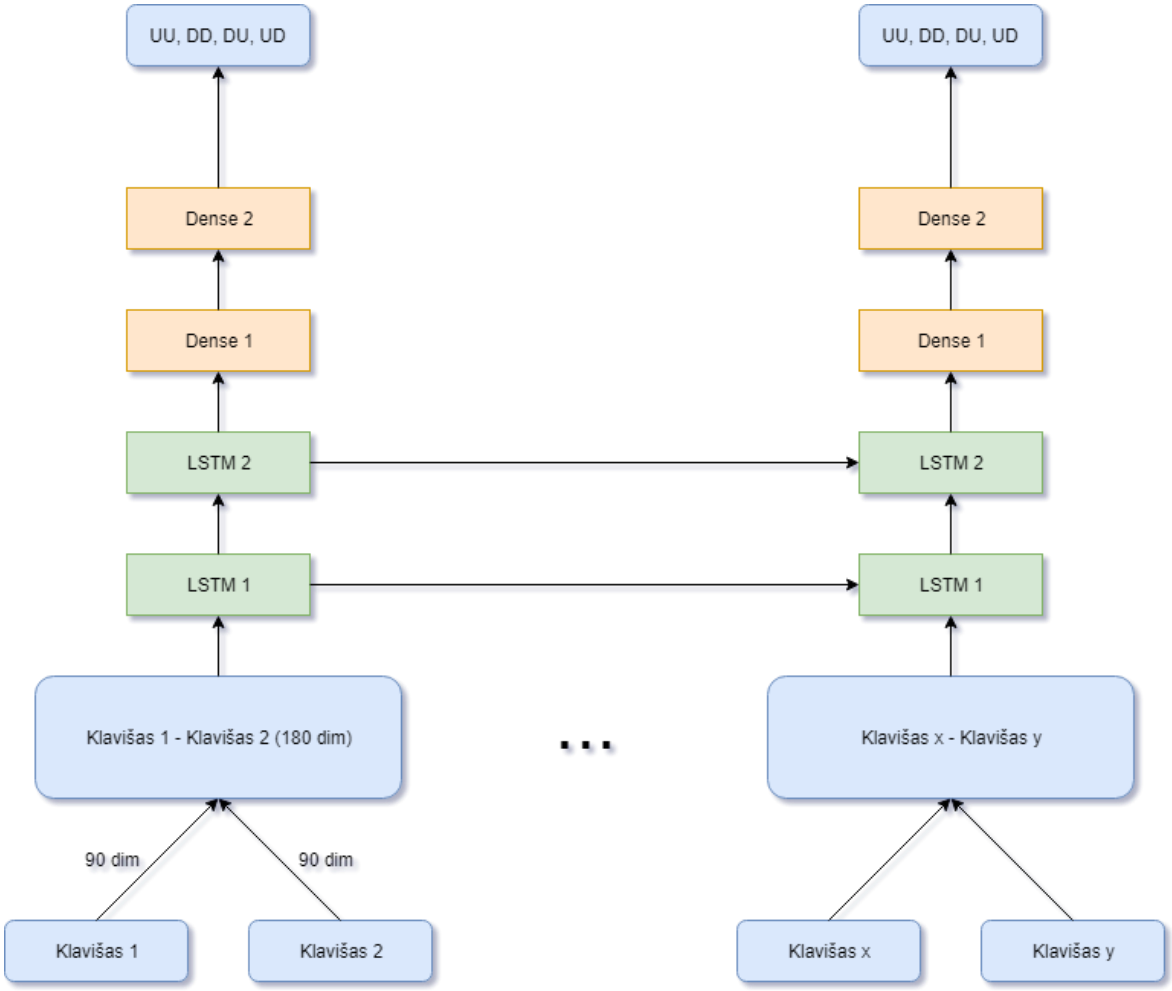
Andriy Lazorenko atliktame tyrime [16] išsiaiškino, jog šiuolaikinis galingas procesorius naudojantis TensorFlow per sekundę gali apdoroti ~400 pavyzdžių, kai tuo tarpu galinga vaizdo plokštė – virš 6 tūkstančių.

10. Rekurentinio neuroninio tinklo struktūra

Kaip jau minėta anksčiau, vartotojui spausdinant renkama informacija apie paspaudimų eiliškumą (kuris po kurio mygtukas yra spaudžiamas, išsaugant jų klaviatūros kodus) bei jų laiko savybes (4 laiko matavimai, pažymėti 1 bei 2 grafuose. Iš to yra kuriamas rekurentinis neuroninis tinklas, kurio tikslas atspėti laiko matavimo savybes tarp dviejų paspaudimų. To produktas – kiekvienam vartotojui bus sukurtas atskiras jam įprasto spausdinimo modelis, kuris vėliau bus talpinamas į duomenų bazę.

Dauguma paspaudimų dinamikos mašininio mokymosi metodų reikalauja įsilaužėlio duomenų apmokant modelį, kad būtų surasti optimalūs parametrai algoritmams [1,2,3,4,5,6,7]. Tai yra nerealistiška realaus pasaulio uždavinyje, kur modelis turi būti kuriamas tik pagal tikrojo vartotojo paspaudimų duomenis. Šio tikslo įgyvendinimui tinklo struktūroje įgyvendintas LSTM (angl. long-short term memory) modelis. Kai vartotojo modelis yra pakankamai apmokytas, jam paduodamas naujas duomenų rinkinys su paspaudimais. Tuomet modelis nuspėja paspaudimų laikus, o skirtumas tarp nuspėtojo bei tikrojo laiko yra naudojamas kaip rezultatas. Taip išmokstamas išskirtinis, būtent tam vartotojui įprastas, spausdinimo elgesys.

Tinklo įvesčiai buvo panaudoti du sujungti užkoduoti vektoriai (dviejų mygtukų paspaudimų), o tikslui - keturios paspaudimų savybės (UU, DD, DU, UD – žr. pav. 1 ir 2). Modelis buvo treniruotas kiekvienam vartotojui individualiai, nenaudojant svetimų paspaudimų duomenų. Šio modelio tikslas yra atspėti paspaudimų laikus, kai jam paduodamas paspaudimų digrafas – jei modelis apmokytas pakankamai gerai, jis išmoks būtent tam vartotojui būdingą spausdinimo ritmą ir gebės atspėti jam įprastus paspaudimų laikus bet kuriai mygtukų porai. Rekurentinio neuroninio tinklo schema:



pav. 8 Tyrime naudoto rekurentinio neuroninio tinklo struktūra

Šiame tinkle naudojami dviejų rūšių sluoksniai:

- Ilgalaikės-trumpalaikės atminties vienetų sluoksnis (žr. 8.2 skyrelį)
- Dense sluoksnis – tai įprastas neuroninio tinklo neuronų sluoksnis. Kiekvienas neuronas gauna įvestis iš visų buvusio sluoksnio neuronų. Sluoksnis turi svorių matricą, *bias* vektorių bei praeito vektoriaus aktivacijas.

Kalbant apie tinklo struktūrą ir žemiau aprašytus parametrus, konkrečių metodų, algoritmų ir dydžių pasirinkimą puikiai iliustruoja ši citata: „Nėra jokių magiškų taisyklių kaip sukonfiguruoti visus šiuos parametrus, kad išgauti geriausią rezultatą. Tu turi bandyti skirtingas reikšmes, kol rasi tai, kas geriausiai veikia sprendžiamai problemai“. [17]

10.1. Batch dydis

Krūvos dydis yra hyper-parametras, kuris nusako elementų skaičių, su kuriais bus atlikti veiksmai prieš atnaujinant vidinius modelio parametrus. Tai tarsi for ciklas, iteruojantis per vieną ar daugiau elementų ir atliekantis spėjimus. Krūvos pabaigoje, spėjimai yra palyginami su tikimaisiais išvesties kintamais ir apskaičiuojamas klaidos dydis. Pasinaudojant šiuo dydžiu, atnaujinimo algoritmas patobulina modelį. Dažniausiai naudojami dydžiai yra 32, 64 bei 128 – šis skaičius privalo būti daugiau arba lygus nei 1, bet mažiau nei duomenų rinkinyje esančių vektorių skaičius. Kadangi paspaudimų kartais gali būti labai mažai, šiam tyrimui batch dydis buvo nustatytas 10.

10.2. Epochos

Epochų skaičius yra dar vienas tinklo parametras, kuris nusako skaičių, kiek kartų mokymosi algoritmas praeis per duomenų rinkinį. Epocha susidaro iš vienos ar daugiau „krūvelių“ (angl. batch). Epocha yra tarsi for ciklas, kurio viduje – dar vienas ciklas, skirtas krūvelių apdorojimams (pvz. turint 100 duomenų vektorių, kai krūvelės dydis nustatytas 10, turėsime 10 ciklų cikle). Epochų skaičius dažnai buna didelis, nuo šimtų iki tūkstančių – tai leidžia mokymosi algoritmui veikti tol, kol klaidos tikimybė modelyje yra minimali. Šiame darbe epochų buvo 100.

10.3. Aktyvavimo funkcijos

Aktyvavimo funkcijos yra matematinės lygybės, kurios nusprendžia neuroninio tinklo išvestį. Jos yra paskirtos kiekvienam tinklo neuronui ir nusprendžia, ar jis bus aktyvuotas ar ne, priklausomai nuo to, ar jo įvestis yra panaši modelio spėjimui. Šios funkcijos taip pat normalizuoja kiekvieno neurono išvestį į skaičių tarp 0 ir 1. Aktyvavimo funkcijos taip pat privalo būti efektyvios, kadangi jos skaičiuojamos tūkstančiams ar net mlijonams neuronų kiekvienam vektoriui.

Neuroniniame tinkle įvestis yra paduodama neuronams įvesties sluoksnyje. Kiekvienas neuronas turi svorį, kuris sudaugintas su įvestimi grąžina išvestį, kuri bus siunčiama kitam sluoksniui. Aktyvavimo funkcija, kitais žodžiais, yra matematiniai „vartai“ tarp neurono įvesties ir išvesties. Ji gali būti paprasta kaip slenkstinė funkcija, kuri arba „įjungia“, arba „išjungia“ neuroną, priklausomai nuo taisyklės ar nustatyto slenksčio.

Yra trijų tipų aktyvavimo funkcijos:

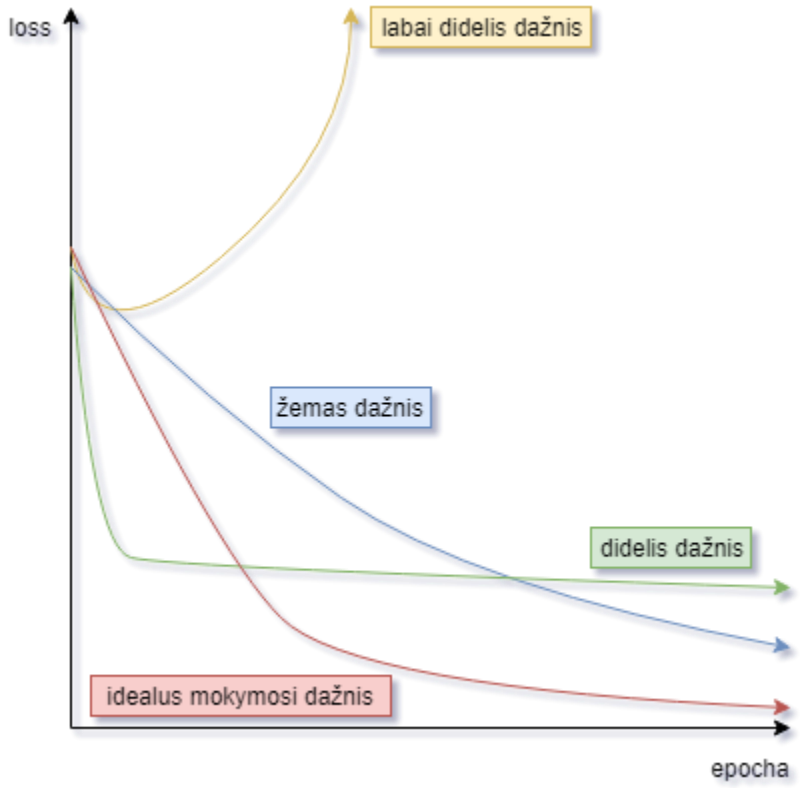
- Binarinė žigsnio funkcija
- Tiesinė aktyvavimo funkcija
- Netiesinė aktyvavimo funkcija

Modernūs neuroniniai tinklų modeliai naudoja netiesines aktyvavimo funkcijas. Jos leidžia modeliui kurti sudėtingas asociacijas tarp tinklo įvesties bei išvesties, kurios yra būtinos sudėtingų duomenų mokymuisi bei modeliavimui. Svarbiausias jų privalumas – jos leidžia „krauti“ skirtingus neuronų sluoksnius vieną ant kito, kas leidžia kurti gilius neuroninius tinklus.

Šiame darbe buvo naudota ReLU – tai dažniausiai naudojama aktyvavimo funkcija neuroniniuose tinkluose. Ji yra tiesinė visoms teigiamoms reikšmėms ir nulinė – neigiamoms. Ji nenaudoja daug resursų, nes nėra komplikuotų skaičiavimų. Modelis, naudojantis tokią aktyvavimo funkciją gali greičiau apsimokyti bei sparčiau veikti.

10.4. Mokymosi dažnis

Mokymosi dažnis yra dar vienas tinklo parametras, kuris kontroliuoja, kiek daug koreguojami tinklo svoriai, atsižvelgiant į *loss* gradientą. Kitais žodžiais, jis daro tiesioginę įtaką kaip greit modelis konverguos į lokalų minimumą – t.y. kaip greit pasieks geriausią tikslumą. Tai daro įtaką modelio mokymosi laikui. Dažniausiai šie dažniai programuotojų yra spėliojami arba nustatomi atsižvelgiant į praeityje buvusius bandymus.



pav. 9 Mokymosi dažnio variantai

Šiame darbe pradinis mokymosi dažnis buvo nustatytas kaip 0.1 ir pastoviai mažinamas per pus, atsižvelgiant į praėjusių bei dabartinių *loss* įrašų medianą.

10.5. Optimizavimo algoritmai

Optimizavimo algoritmai padeda sumažinti (arba padidinti) tikslo funkciją, dar vadinamą klaidos funkcija. Tai matematinė funkcija, priklausanti nuo modelio vidinių mokymosi parametrų – pavyzdžiui svorių ir *bias* reikšmių, kurie naudojami skaičiuojant išeities reikšmes ir yra išmokstami bei atnaujinami atsižvelgiant į optimalų sprendimą – siekiant minimalių praradimų (angl. loss) apsimokymo procese. Šiame tyrime buvo naudotas optimizavimo algoritmas *Adam.*

Tai – populiarus algoritmas gilaus neuroninių tinklų mokymosi arenoje, nes jis sugeba pasiekti gerus rezultatus greitai [11]. Jis sukombinuoja kitų dviejų gerai žinomų optimizavimo algoritmų *Adagrad* ir *RMSProp* geriausias savybes ir yra lengvai konfigūruojamas.

11. Projekto rezultatai

Taigi, kaip minėta auksčiau, modelio kūrimui buvo pasinaudota *Tensorflow* karkasu *Keras*. Modelis aprašytas *Python* programavimo kalba. Iš duomenų bazės gaunami vartotojų paspaudimų duomenys, jie yra pertvarkomi ir siunčiami į rekurentinį neuroninį tinklą. Prieš pateikiant rezultatus pravartu susipažinti su metrika, kuri dažniausiai naudojama įvertinti paspaudimų dinamikos tyrimų efektyvumą ir kuri buvo naudota ir šiame darbe.

11.1. Klaidos tikimybė

Vartotojų autentifikavime galima susidurti su dviejų tipų klaidomis:

- Klaidingo įleidimo (angl. false acceptance rate arba FAR)
- Klaidingo blokavimo (angl. false rejection rate arba FRR)

Klaidingo įleidimo arba FAR klaida žymi tikymbę įsilaužėlį palaikyti kaip tikrą vartotoją, o FRR priešingai – tikrąjį vartotoją palaikyti kaip įsilaužėlį.

FAR nusako sistemos patikimumą – ar tai patikima apsaugos priemonė ar ne. Jei pvz. klaidingo įleidimo (FAR) tikimybė didelė, įsilaužėliai nesunkiai pateks į sistemą. Klaidingo blokavimo klaida arba FRR, savo ruožtu, nusako ar sistemą galima naudoti be didesnių trikdžių. Jei FRR bus didelis, tikrieji vartotojai bus blokuojami nuo patekimo į sistemą ir turės pakartotinai vykdyti autentifikaciją vėl ir vėl. Žinoma, idealaus scenarijaus, kai tiek FAR, tiek FRR bus nuliniai negalime pasiekti – vartotojų rašymo ritmą įtakoja daug išorinių veiksnių (pvz. nuovargis, ligos, svaigalai ir t.t.), kurie šiek tiek iškraipo duomenis. Realiose aplikacijose didesnis demesys dažniau kreipiamas į FAR mažinimą, kadangi vartotojai gali susitaikyti su pakartotinu slatažodžio įvedimu kelis kartus, jei tai suteikia patikimą apsaugą prieš įsilaužėlius.

Bendru atveju, mažinant FAR didiname FRR ir atvirkščiai. Kai jie yra subalansuoti, atsiranda žymuo EER (angl. equal error rate). EER ir yra dažniausiai naudojamas kaip efektyvumo matavimo matas, kai kalbama apie vartotojų identifikavimą naudojant paspaudimų dinamiką.

11.2. Rezultatų lentelė

Rezultatų lentelėje bus pateikti atpažinimo rezultatai tik tų vartotojų, kurių paspaudimų skaičius duomenų rinkinyje buvo 250 arba daugiau. To priežastis – vykdant mokymus ir atpažinimus su mažiau nei 250 paspaudimų rezultatai tampa labai nepastovūs ir iškraipytų tyrimo efektyvumo rezultatus. Vykdant mokymus, visais atvejais testavimams buvo paliekama 50 paspaudimų duomenų – visi kiti budavo panaudojami treniruotėms.

Paspaudimų skaičius	EER (PC)	EER (Mobile)
3000	2.33%	-
2000	2.87%	-
1000	3.40%	5.51%
500	3.62%	6.35%
250	4.55%	8.70%
200	6.79%	10.12%

pav. 10 Rezultatų lentelė

Vykdant mokymus pastebėta, kad kuo mažiau vartotojo paspaudimų užfiksuota – tuo rezultatai nepastovesni. Kylant paspaudimų skaičiui, kyla tikslumas bei pastovumas. Taip pat, lentelėje nepateikti mobiliojo įrenginio duomenys su dviem didžiausiais paspaudimų skaičiais todėl, kad vartotojų, turinčių tokį paspaudimų skaičių tebuvo 1-2, tad iš to daryti išvadas apie efektyvumą būtų netikslu.

Taip pat galima pastebėti, jog nepaisant to, kad paspaudimų skaičius buvo toks pat, mobiliojo įrenginio atpažinimo rezultatų tikslumas nusileidžia kompiuterio klaviatūra surinktiems duomenims. Peržvelgus duomenų bazėje esančius mobiliųjų įrenginių vartotojų įrašus pastebėta, kad šių vartotojų paspaudimai yra ne tokie pastovūs, kaip PC naudotojų – tas ir galėjo įtakoti didesnę klaidos tikimybę.

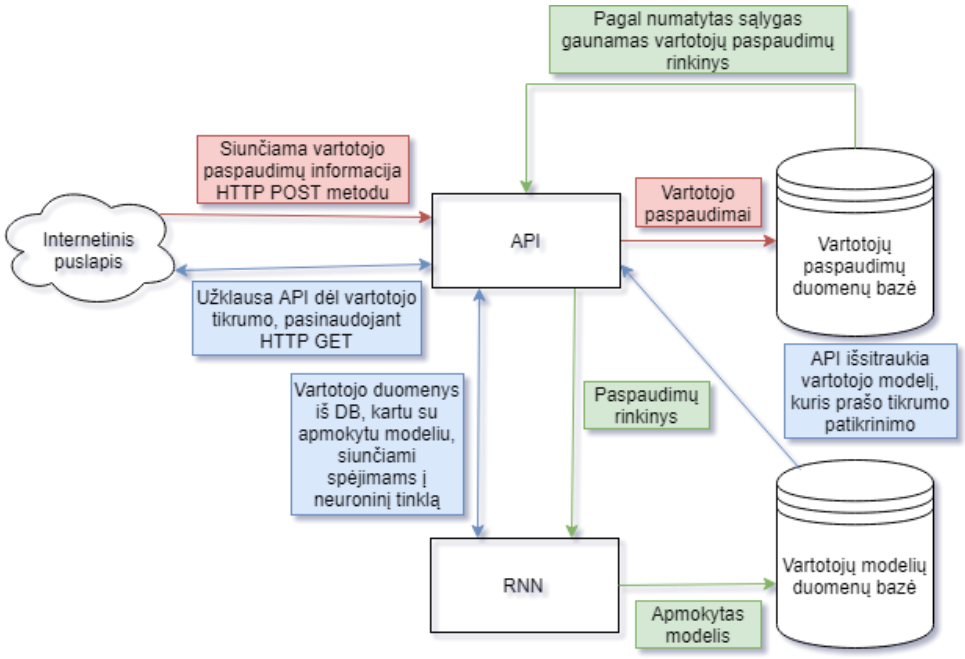
Galiausiai, aukščiausias laiptelis, kurį pavyko išbandyti, buvo 3000 paspaudimų. Tai nėra didelis skaičius, ir, veikiausiai, pradėjus naudoti sistemą ilgalaikėje perspektyvoje, paspaudimų skaičius smarkiai išaugtų. To pasekoje, sprendžiant iš dabartinių rezultatų, sistemos tikslumas būtų dar didesnis – pažvelgus į klaidos tikimybę tarp 2000 ir 3000 paspaudimų, EER sumažėja ant 0.5%, nors paspaudimų skaičius paauga tik ant 1000.

11.3. Mokymosi laikas

Vidutiniškai, šiame darbe naudotam tinklui apsimokyti su vienu vartotojo vektoriumi užtrukdavo apie 375 mikro-sekundžių. Tai yra, jei turime 3000 vartotojo paspaudimų, bus sukurti 1500 digrafai, kurie ir atstoja vektorius. Taigi, tokiu atveju viena mokymosi epocha truks apie 0.5 sekundės. Kadangi buvo naudota 100 epochų, tai vidutiniškai, vieno vartotojo su 3000 paspaudimų apsimokymų trukmė buvo 50-60 sekundžių. Tai gali atrodyti per didelė trukmė realaus pasaulio uždavinyje, kai vartotojų gali būti labai daug, tačiau visas procesas buvo vykdomas ant asmeninio kompiuterio procesoriaus. Kaip jau kalbėta skyrelyje 9.2, mokymosi procesai, vykdomi ant vaizdo plokštės, o ne procesoriaus, vyksta žymiai greičiau. Norint paleisti mokymus ant *GPU*, o ne *CPU*, *Tensorflow* reikalauja, jog vaizdo plokštės kompiliavimo galimybė (angl. computing capability) būtų 3.5 arba aukštesnė. Kokias galimybės turi kiekviena vaizdo plokštė galima pasitikrinti internete, vaizdo plokštės gamintojo puslapyje. Mano turima vaizdo plokštė yra ganėtinai sena, turinti 3.0 *computing capability*, tad procesų pasileisti ant jos – nepavyko. Nauja bei galinga vaizdo plokštė su vieno vartotojo duomenimis veikiausiai susidorotų per kelias sekundes. Taip pat, modelius būtų galima pastoviai atnaujinti, apmokinant naujais duomenimis, vietoj to, kad visi būtų permokomi kas kart iš naujo. Tas taip pat turėtų padidinti spartumą.

11.4. Integracija realiame internetiniame puslapyje

Kaip tokio tipo sistema galėtų būti integruota realiame internetiniame puslapyje, pateikta šioje schemoje:



pav. 11 Sistemos schema

Šioje schemoje esančios trys skirtingos spalvos simbolizuoja tris skirtingus sistemos funkcionalumus:

- Raudona spalva žymi vartotojo paspaudimų informacijos saugojimą į duomenų bazę. Javascript seka paspaudimus, juos talpina į masyvą, o prieš perkraunant puslapį – HTTP POST metodu viską išsiunčia į aplikacijų programavimo sąsają (API). Ji, savo ruožtu, patikrinus užklausos turinį viską išsaugo į duomenų bazę.
- Žalia spalva žymi antrają fazę, kuomet iš duomenų bazės pagal nustatytas sąlygas imami duomenys – t.y. tokios sistemos integruotojas turėtų nuspręsti, kada imti duomenis, kokio senumo, kokio dydžio, ar kaskart mokyti iš naujo, ar imti tik tuos duomenis, kurių nėra modelių duomenų bazėje ir t.t. Tai turėtų priklausyti nuo to, kokio dydžio yra internetinis puslapis ir kiek unikalių vartotojų apsilanko. Tuomet tie duomenys yra siunčiami rekurentiniam neuroniniam tinklui, kuris, po mokymų, kiekvienam vartotojui išsaugo jo modelio strukturą bei tinklo svorius į duomenų bazę.
- Mėlyna spalva žymi identifikavimo procesą. Vartotojo naršyklė išsiunčia HTTP GET užklausą į serverį, kartu su paspaudimų duomenimis ir laukia atsakymo, ar vartotojo duomenys atitinka tokius, kokie numatomi. Jei ne, vartotojui gali būti taikomi apribojimai ar uždrausta prieiga. Tuo tarpu aplikacijų programavimo sąsaja iš vartotojų modelių duomenų bazės išsitraukia reikiamą modelį, kurį, kartu su paspaudimų duomenimis siunčia į rekurentinį neuroninį tinklą. Iš jo bus gautas atsakymas, kiek procentų duomenys atitinka tuos, kurie numatomi.

Išvados

Šio darbo eigoje buvo sukurta veiksminga sistema, atpažįstanti ir identifikuojanti realius internetinio puslapio lankytojus. Įgyvendintas duomenų rinkimas realaus vartotojo interneto naršyklėje, siuntimas į serverį bei talpinimas į duomenų bazę, esančią debesyje, bei su tuo susiję sunkumai bei ypatumai. Taip pat buvo rastas bei įgyvendintas ganėtinai efektyvus metodas kurti vartotojų paspaudimų modelius, naudojamus atpažinime. Galų gale buvo aptarta, kokie sunkumai galėtų iškilti implementuojant tokią sistemą pilnu pajėgumu į realų internetinį portalą bei į ka reikėtų atkreipti dėmesį.

Iš surinktų duomenų ir rezultatų akivaizdu, kad tokia sistema veikia pakankamai efektyviai, kad būtų veiksminga bei panaudojama realybėje. Šiek tiek apmaudu, kad sistemą pavyko išbandyti su maksimaliai ~3000 paspaudimų – tikiu, jog su 10 000 paspaudimų ar net daugiau sistemos efektyvumas išaugtų. Vis dėlto, gautieji rezultatai duoda impulsą vystyti šią sistemą toliau ieškant patobulinimų bei sprendžiant iškilusias problemas, kurios, tikiu, iškiltų paleidus ją pilnu pajėgumu didesniame internetiniame puslapyje.

Mąstant apie ateitį, visą mokymosi procesą būtų pravartų perkelti į debesį – taip jis vyktų sparčiau bei viskas būtų automatizuota, o puslapio savininkui nereiktų papildomų resursų serveryje. Taip pat, sukurta aplikacijų programavimo sąsaja taptų universali – ją būtų galima pritaikyti ne vienam internetiniam puslapiui vienu metu (pavyzdys – typingdna.com).

Literatūra

[1] U. Johanesen: Keystroke Dynamics on a device with touchscreen, Computer Science and Media Technology Department, Gjovik University, 2012

[2] M. Trojahn, F.Arndt, F.Ortmeier: Authentication with Keystroke Dynamics on touchscreen keyboards, 2013, p. 110-120

[3] G. Ho: Tapdynamics: strengthening user authentication on mobile phones with keystroke dynamics. Technical report, Stanford University, 2014.

[4] A.Awan, S.Nizamani, N.Zaman: Password Security Enhancement using Dynamic Keystrokes. A Review, Indian Journal of Science and Technology, 2018

[5] K.Kevin, M.Roy: Keystroke Dynamics – Benchmark Data Set, 2009

[6] F. Monrose, A.Rubin: Authentication via keystroke dynamics, 1997, p. 48–56.

[7] A. A. Ahmed, I. Traore: Biometric Recognition Based on Free-Text Keystroke Dynamics, IEEE Transactions on Cybernetics, 2013, vol. 44, no. 4, p. 458-472

[8] W3C: W3C Technical Reports and Publications, https://www.w3.org/standards/webdesign/, 2009

[9] Mozilla: Introduction to DOM, https://developer.mozilla.org/Document_Object_Model, 2019

[10] PHP MySql Database, https://www.w3schools.com/php/php_mysql_intro.asp, 2019

[11] D.P.Kingma, J.Lei Ba: Adam: A method for stochastic optimization, Published as a conference paper at ICLR, 2015

[12] K. Killourhy, R. Maxion: Comparing Anomaly-Detection Algorithms for Keystroke Dynamics, 2009

[13] M.Miljanovic: Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction, 2012

[14] S.Hasim, A.Beaufays: Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling, 2014

[15] L.Xiangang, W.Xihong: Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recogniction, 2014

[16] A.Lazorenko: TensorFlow performance test: CPU vs GPU, 2017

[17] J.Brownlee: What is the difference between a batch and an epoch in a neural network, 2018

Santrauka

Šio darbo esmė yra sukurti pamatus universaliai, moderniai bei intuityviai aplikacijų programavimo sąsajai vartotojų atpažinimui, kuri remtųsi klaviatūros paspaudimų dinamika. Svarbu nustatyti sunkumus, kylančius kuriant tokią sistemą bei rasti metodiką pasiekti rezultatą, kad sistema būtų veiksminga.

Ataskaita prasideda apibūdinant naudotą duomenų rinkimo būdą tiek iš teorinės, tiek iš praktinės pusės. Taip pat supažindinama su jų formavimu bei paruošimu tolimesniam darbui. Tolimesnė ataskaitos dalis yra koncentruota į metodiką, kuri buvo naudojama vartoto identifikacijai pagal pateiktus duomenis. Galų gale, pateikiami pasiekti rezultatai bei jų aptarimas.

Pagrindinė sukurtos sistemos savybė yra identifikuoti internetinio puslapio vartotoją pagal jam įprastą spausdinimo klaviatūra ritmą. Projekto metu sukurta sistema suteikia tvirtus pagrindus plėtoti ją toliau iki produkcinio lygio, skirto plačiai auditorijai.

Raktažodžiai: Klaviatūros paspaudimų dinamika, rekurentinis neuroninis tinklas, Javascript, Tensorflow, HTTP, MySql

Summary

The purpose of this report is to build base for universal, modern and intuitive application programing interface (API) dedicated to web user identification, based on keystroke dynamics. It is important to pinpoint any difficulties, arising from building system of this kind and to find working method to achieve a result for an effective system.

The report starts by describing data collection method used in the project both teoretically and practically. Also, it is presented how they are formated and prepared for identification algorithm. Futher down the report is concentrated to the methodics, which were used for web users identification based on collected dataset. Finally, results are presented and discussed.

The main purpose of this system is to identify web user based on his usual typing pattern. The system that has been built during this project lays a strong foundation for further researches and a possibility to expand it to a production level for commercial use.

Keywords: Keystroke dynamics, recurrent neural network, Javascript, Tensorflow, HTTP, MySql