Tampereen ammattikorkeakoulu

Dynaamisen verkkosivun verkkoharavointi

Case: Instagram

Miika Halmetoja

OPINNÄYTETYÖ
Kesäkuu 2021
Tietojenkäsittely
Web-palvelut

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Web-palvelut

HALMETOJA, MIIKA:
Dynaamisen verkkosivun verkkoharavointi
Case: Instagram

Opinnäytetyö 22 sivua, joista liitteitä 0 sivua
Kesäkuu 2021

Opinnäytetyön tavoitteena oli selvittää, onko yksityishenkilön mahdollista hyödyntää sosiaalisen median dataa markkinatutkimuksessa dynaamisen verkkosivun verkkoharavoinnin keinoin. Tutkimuksen kohteeksi valikoitui Facebook, Inc. omistama Instagram, jonka vuoksi tutkimus perehtyy käytännön ratkaisun löytämisen ja toteuttamisen lisäksi verkkoharavointia koskevan lainsäädännön ja Facebookin välisen risteyksen tarkasteluun sekä siihen, miten tämän risteyksen liike vaikuttaa kolmansien osapuolten mahdollisuuksiin harjoittaa sosiaalisen median verkkoharavointia.

Instagramin käyttöliittymää ja sen rajoituksia testattiin tutkivan testauksen, HTTP metodien ja koneellisen testauskirjaston yhdistelmällä. Tiedonkeruuseen sovellettiin kahden eri lähestymistavan yhdistelmää: dokumenttioliomallin avulla ohjailu ja rajapintaan tehdyt pyynnöt. Toimintaa pyrittiin inhimillistämään, eli toimimaan eettisesti tavalla, joka ei kohteen näkökulmasta eroa kuormittavuudeltaan tavallisesta käyttäjästä.

Verkkoharavoinnin laillisuus on tapauskohtaista ja tulkinnanvaraista, sillä haravoitava tieto on palvelumuodon vuoksi usein sisäsyntyisesti saatavilla. Kysymykseksi jää, saako kone tarkastella samaa sisältöä, kuin ihminen. Instagramin tapauksessa sen määrittelee viime kädessä Yhdysvaltain korkeimman oikeuden lainsäädännön suhde Instagramin palveluehtoihin.

Tutkimuksessa toteutettiin ja testattiin kahta erilaista tapaa hakea haluttua dataa. Käyttäjän on mahdollista hakea ja tallentaa itselleen näkyvää tietoa koneellisesti, mutta toiminnan riskitekijät ja tiedon määrä suhteessa aikaan määrittelevät sen kannattavuuden. Tutkimuksen tulokset viittaavat tulevaisuuteen, jossa sosiaalisen median yhtiöillä on alustallaan kehittyvään ihmisten käyttäytymistä koskevaan dataan yksinoikeus, jota ei voida murtaa, vaikka sen yrittäminen pysyisi laillisena.

Asiasanat: verkkoharavointi, javascript, sosiaalinen media

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Business Information Systems
Web Services

HALMETOJA, MIIKA:
Scraping a Dynamic Web Page
Case: Instagram

Bachelor's thesis 22 pages, appendices 0 pages
June 2021

The purpose of this study was to investigate whether an independent third party could utilize data from social media for market research purposes by using web scraping methods. Facebook, Inc. owned social media Instagram was chosen for evaluation. The thesis focuses on highlighting the contesting point between Instagram's terms of service and the legislation under which web scraping currently operates.

To understand Instagram's user interface and its restrictions, a combination of exploratory testing, HTTP methods and test automation was used. Data extraction was done through a combination of DOM navigation and direct API calls.

Legality of web scraping is ambiguous and handled on case-by-case basis because the form of a web service often makes the data inherently available. The question remains whether a bot should be allowed to observe the same content that is readily available for a human agent.

The study implemented and tested two different ways to retrieve the desired data. The user can retrieve and store visible information mechanically, but the profitability of operations is affected by operational risk factors and their relation to the amount of information extracted over time. The results of the study point to a future in which social media companies have the exclusive right to human behavioral data generated on their platforms.

Key words: web scraping, javascript, social media

SISÄLLYS

ERITYISSANASTO

Botti	Ohjelma, joka toimii itsenäisesti sille määriteltyjen toimintaohjeiden mukaan varsinkin suurta työmäärää, toistoa, jatkuvaa päivystystä tai rutiinia vaativissa tehtävissä
Dynaaminen verkkosivu	Verkkosivu, joka luodaan vasta, kun selain sitä pyytää
GET-pyyntö	HTTP-protokollan metodi, jota käytetään resurssin hakua varten
GraphQL	Facebookin kehittämä viestintäprotokolla
HTTP	Protokolla, jota selaimet ja verkkopalvelimet käyttävät tiedonsiirtoon
JavaScript	Web-kehityksessä yleisesti käytetty ohjelmointikieli
JSON	Avoimen standardin tiedostomuoto tiedonvälitykseen ensisijaisesti web-sovelluksen ja palvelimen välillä
Node.js	Avoimen lähdekoodin alustariippumaton ajoympäristö JavaScript-koodin suorittamiseen palvelimella
Puppeteer	Node.js -kirjasto, joka tarjoaa ohjelmointirajapinnan Chromen tai Chromiumin ohjailuun kehittäjän työkaluista vastuussa olevan protokollan yli ilman käyttöliittymää
Ohjelmointirajapinta	Muodostaa kahden ohjelman, esimerkiksi käyttöjärjestelmän ja sovelluksen välisen rajapinnan, jonka tarjoamat valmiit palvelut helpottavat ohjelmoijan työtä
XPath	Kieli XML-dokumenttien osien osoittamiseen ja XML-dokumentin rakenteeseen perustuvan tiedon luontiin

1 JOHDANTO

Ihmisten itsestään tuottaman tiedon määrä kasvaa Internetissä ja se lisää erityisesti markkinatutkimuksen mahdollisuuksia. Tiedon väärinkäytökset ovat kuitenkin yleisiä ja sosiaalisen median yhtiöiltä vaaditaan lisääntyvissä määrin tapoja suojella käyttäjiensä yksityisyyttä. Samaan aikaan vaatimukset rajoittavat kolmansien osapuolten, kuten yksityisyrittäjien ja akateemisen tutkimuksen, mahdollisuuksia hyödyntää samaa tietoa.

Ihmisten käyttäytymistä koskevan tiedon osalta avointen standardien poliittinen ongelmallisuus ohjaa yksittäiset toimijat verkkoharavoinnin harmaalle alueelle, jossa se ei ole laitonta, mutta ei myöskään riskitöntä, halpaa tai helppoa. Dynaamisen verkkoharavoinnin toimintaperiaate on kuitenkin yksinkertainen ja vain toiminnan laajuus määrittelee sen ongelmallisuuden. Tutkimuksessa selvitetään, kuinka tiedon keruu Instagramista onnistuu sen riskeistä huolimatta ja samalla kartoitetaan sitä, mikä tekee sen toteutuksesta ongelmallista.

Tutkimus toteutetaan osana yksityisyrittäjän tarvetta kohdistaa sosiaalisen median tiedotustaan seuraajilleen markkinatutkimuksen keinoin.

2 TOTEUTUKSEN EETTISYYS

2.1 Laki

Verkkoharavoinnin kohteeksi valikoitui verkkosivusto, jonka sisältämä tieto on keskeisessä osassa sivuston omistavan yhtiön rahoitusmallissa. Tästä syystä tiedon haravoinnissa täytyy huomioida verkkoharavoinnin lainsäädäntö ja se, miten kohde pyrkii ennakkotapauksia hakemalla muuttamaan sen tulkintaa. Kohteen yritystoiminnan maantieteellisen sijainnin oikeusasteet määrittelevät toiminnan laillisuuden tapauskohtaisesti. Instagramin kohdalla tulkintaa tekee viime kädessä Yhdysvaltain korkein oikeus liittovaltion rikoslain pohjalta (Computer Abuse and Fraud Act, 18 U.S.C. § 1030).

Instagramin omistaja Facebook Inc. on nostanut useita kanteita verkkoharavointia harjoittavia toimijoita kohtaan palveluehtojensa rikkomiseen ja palvelintensa luvattomaan käyttöön vedoten (Facebook, 2020a). Viimeisimpien verkkoharavointia koskevien käsittelyjen perusteella julkisen tiedon haravointia, eikä tiedon kerääjän omilla tunnuksilla tehtyä haravointia, katsota laittomaksi (Yhdysvaltain korkein oikeus, 2020). Mikäli tiedon kerääjä käyttää toisen henkilön tunnuksia kerätäkseen itselleen tälle kuuluvaa tietoa, katsotaan se kyseisen henkilön harhaanjohtamiseksi (Facebook, Inc. v. Massroot8), tai pääsyksi tietoon, johon kerääjällä ei ole Computer Abuse and Fraud Act -säädännön perusteella oikeutta (Facebook, Inc. v. BrandTotal Ltd.).

Toiminnan laillisuus ei estä Facebookia hakemasta vahingonkorvauksia ja estämästä haitalliseksi katsomaansa toimintaa palveluehtojensa nojalla (Facebook, 2010).

2.2 Palveluehdot

Instagramin palveluehtojen rikkominen johtaa rikkeestä riippuen rajoituksiin tai tilin poistamiseen. Verkkoharavoinnin osalta oleellisia ovat kohdat, joissa kielle-

tään tietojen kerääminen tai tilien luonti automatisoidulla tavalla ilman Instagramin erityisesti myöntämää lupaa. Myös uuden tilin luonti on kielletty, mikäli tilisi on aiemmin poistettu käytöstä Instagramin toimesta lain tai palveluehtojen rikkomisen johdosta (Instagram, 2020a).

Instagram käyttää lisäksi yleisesti käytössä olevaa robotin rajausstandardia, jossa annetaan tietoja ja rajoituksia verkkosivuja tutkiville boteille. Instagramin kieltää tunnetuimpien bottien toiminnan lisäksi kaiken toiminnan tuntemattomilta boteilta (Instagram, 2021).

2.3 Johtopäätökset

Toteutus vaatii Instagramin palveluehtojen rikkomista ja rajoitusten kiertämistä. Rajoitusten kiertäminen ei ole laitonta ja sitä on myös julkisesti puollettu (Freelon, 2018; Rogers, 2018; Venturini & Rogers, 2019) erityisesti Cambridge Analytica -dataskandaalin (Isaac & Hanna, 2018) aiheuttamien ohjelmointirajapintarajoitusten jälkimainingeissa, mutta toteuttajan täytyy ottaa huomioon estotoimenpiteiden aiheuttamat kulut ja menetykset. Kuluiksi voidaan lukea välityspalvelinten hinta ja menetyksiksi Instagramin estämät tai poistamat käyttäjätunnukset sekä niihin mahdollisesti liitetyt puhelinnumerot ja henkilötunnukset (Instagram, 2020b), mikäli Instagram havaitsee toiminnan koneelliseksi.

3 TYÖKALUT JA TEKNIIKAT

3.1 Testaus

Halutun tiedon paikantamiseen käytettiin Firefox -selainta ja sen sisältämiä kehittäjän työkaluja. Opinnäytetyön tapauksessa haluttu tieto tuotiin dynaamisesti GET-pyynnöillä Instagramin dokumenttioliomalliin HTML-elementteinä käyttäjän avatessa halutun Instagram -tunnuksen seuraajalistan, tai sitä alaspäin selatessa. Kehittäjän työkalujen verkkoliikenteen tarkastelun välilehteä käytettiin tarvittavien GET-pyyntöjen muodon tunnistamiseen.

Instagram pidättää tiedon tavallisen käyttäjän tekemien pyyntöjen tuntirajoituksista verkkoharavointia hankaloittaakseen (Facebook, 2021). Pyyntöjä koskevien rajoitusten testaamiseen käytettiin kehittäjän työkalujen konsoliin syötettäviä JavaScript -funktioita välittömän äärimmäispyyntömäärän löytämiseksi.

Lisäksi testattiin, onko Puppeteer Node.js -kirjaston avulla toteutetulla navigoinnilla ja toiminnan hidastamisella, tai inhimillistämisellä, merkitystä estotoimenpiteiden välittömyyteen (Rovetta, Suchacka & Masulli, 2020). Puppeteer -toiminnallisuutta sijoitettiin Visual Studio Code -tekstieditorilla toteutetun JavaScript -toiminnallisuuden lomaan.

3.2 Toteutus

Automaattiseen kirjautumiseen ja halutulle viittaussivulle hakeutumiseen käytettiin Puppeteer -kirjastoa ja dokumenttioliomallin avulla ohjailua. Instagram estää XPath:in avulla dokumenttioliomallissa ohjailun, joten toteutuksen tavaksi täytyi valita CSS-luokkien paikantaminen. Myös pikselikoordinaatteihin perustuvaa ohjailua harkittiin sillä perusteella, että haluttujen elementtien koordinaatit ovat mahdollisesti pysyvämpiä, kuin verkkosivun CSS-luokittelu, mikäli kohdesivusto pyrkii vaikeuttamaan dokumenttioliomallin avulla suoritettua verkkoharavointia. Tämä todettiin kuitenkin myöhemmin turhaksi. Huomiona kuitenkin se, että Instagramissa joidenkin ulkoisesti saman näköisenä pysyvien elementtien CSS-

luokittelu muuttuu työpöytä-, tabletti- ja mobiilinäkymien välillä. Varsinainen verkkoharavointi toteutettiin JavaScriptillä ja HTTP-metodeilla.

4 TESTAUS

4.1 Kirjautumistiedot ja henkilöllisyys

Kirjautuminen mahdollistaa suuremman ja mielekkäämmän otannan, mutta silloin IP-osoitteeseen kohdistettujen rajoitusten kiertämisen lisäksi joudutaan luomaan useita käyttäjätunnuksia. Käyttäjätunnuksia varten tarvitaan saman verran sähköpostitunnuksia. Instagram vaatii myös epäilyttävää toimintaa havaitessaan käyttäjältä puhelinvarmennusta ja toiminnan jatkuessa henkilöllisyystodistusta.

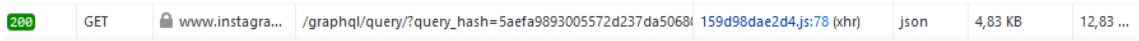
Laajamittainen Instagramin verkkoharavointi on mahdollista niin kauan kuin sivusto ei vaadi henkilöllisyystodistusta käyttäjätunnuksen luonnin yhteydessä. Sen hinta ja ylläpidon vaikeus käyttäjätunnuksia ja IP-osoitteita uudistaessa kasvaa kuitenkin suhteessa toiminnan konemaisuuteen. Työn määrää ja sille kertyvää hintaa voidaan pyrkiä rajoittamaan inhimillistämällä haravointitoimintaa. Tämä on kuitenkin poissa toteutuksen tehokkuudesta, mikäli sen ja inhimillistämisen välistä optimia ei löydetä massatestauksella. Instagram käyttää koneellisen toiminnan havaitsemiseen myös käyttäjiensä toiminnalla koulutettua (Instagram, 2020c), ihmisen käytöskaavan tunnistavaa neuroverkkoa (Facebook, 2020b), jota vastaan koneelliset toimijat käyvät päättymätöntä kilpavarustelua. Tästä syystä käyttäjätunnusten ja niihin yhdistettyjen IP-osoitteiden menettämistä ei voida välttää, vaikka uusia Instagram -käyttäjätilejä voitaisiinkin teoriassa luoda vielä nykypäivänä loputtomasti pelkillä sähköpostiosoitteilla.

Tehokkain lähestymistapa olisi usean käyttäjätunnuksen ja asuinosoitteisiin sidottujen välityspalvelinten kierrätys GET-pyyntöjen välillä. Asuinosoitteisiin sidotut välityspalvelimet ovat kuitenkin tarkoituksen kannattavuuteen suhteutettuna kalliita ja Instagram tunnistaa sekä estää palvelukeskuksissa sijaitsevista halvemmista IP-osoitteista tehdyt pyynnöt haitallisina.

Opinnäytetyön tarkoituksen toteuttamiseksi valitaan kulu- ja riskisyistä mahdollisimman suuri inhimillistäminen, sisäänkirjautuminen ja pyyntöjen teko ilman IP-osoitteiden kierrätystä.

4.2 Halutun tiedon paikantaminen

Tutkiva testaus vaatii hieman loogista päättelykykyä ja yleisymmärrystä verkkosivujen toiminnasta. Opinnäytetyön tapauksessa selaimen kehittäjän työkalujen Network -välisivulta haettiin GET-pyyntö, jolla Instagram tuo haluttua tietoa tietopankistaan GraphQL -ohjelmointirajapintansa kautta front endiin. GET-pyynnön dynaamisia muuttujia vaaditaan tietenkin myös opinnäytetyön tarkoituksessa käytettävien koneellisten pyyntöjen teossa (kuva 1). Opinnäytetyössä haettiin käyttäjän seuraajalistan sisältöä, mutta sama periaate pätee yleisesti dynaamisten verkkosivujen rajapintoihin tehtäviin pyyntöihin.

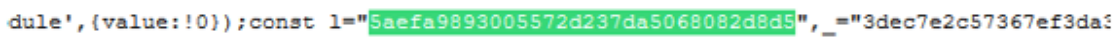


KUVA 1. Kohdekäyttäjän seuraajalistan sisällöstä vastaava pyyntö

Pyynnön neljänneksi viimeinen sarake 159d98dae2d4.js viittaa siihen, että ainakin osa tarvittavista muuttujista löytyy sivustoon linkitetyistä JavaScript -tiedostoista (kuva 2). Näitä voi tarkastella esimerkiksi Firefox -selaimen kehittäjän työkalujen Debugger -välilehdeltä.

KUVA 2. Sivustoon liitetyt tiedostot

Haluttu tieto, eli query_hash -muuttujaan syötettävän kirjain- ja numeroyhdistelmän alkuperäinen julistus, löytyi selaimen tekstihaulla Consumer.js -tiedostosta (kuva 3).

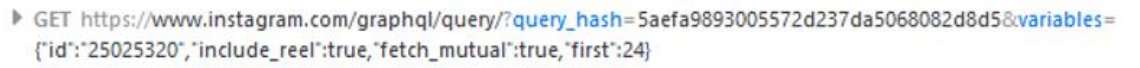


KUVA 3. JavaScript-tiedostosta etsitty muuttuja

Pyyntöä varten tarvitaan myös kohdeprofiilin id. Tämä löytyy kohdeprofiilin etusivua avatessa verkkoliikenteen ensimmäisen, /?__a=1 -päätteisen GET-pyynnön palauttamasta JSON-tiedostosta (kuva 4).

KUVA 4. Kohdekäyttäjätunnuksen juuripyyntö

Palautettua JSON-tiedostoa voidaan tarkastella esimerkiksi GET-pyynnön Response -välilehdeltä. Haluttu tieto löytyy tässä tapauksessa ”user” -olion alta (data.graphql.user.id). Muut seuraajalistan tiedostoja koskevan GET-pyynnön muuttujat ovat valinnaisia, joskin inhimillistämisen osalta esimerkiksi ”first” -muuttuja on olennainen, sillä tavallinen käyttäjä ei voi käyttöliittymässä listaa alaspäin vierittämällä hakea kuin tietyn määrän käyttäjiä kerrallaan. Kaikki muuttujat pyyntöön sisällyttämällä saadaan koostettua opinnäytetyössä haluttu GET-pyyntö (kuva 5).

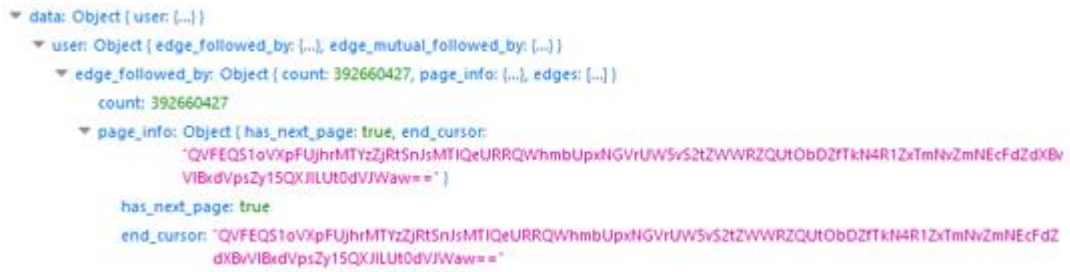


KUVA 5. Kohdekäyttäjän seuraajalistan 24 ensimmäistä käyttäjää

Muiden kuin ensimmäisen seuraajalistaan tehtävän pyynnön kohdalla tarvitaan myös ”after” -muuttujaa, jotta pyyntö kohdistuu järjestyksessä haluttuun JSON-tiedostoon. Muuttujaan syötetään näissä tapauksissa edellisen JSON-tiedostosta löytyvä rekursioon tarkoitettu ”end_cursor” -muuttuja (kuva 6 ja kuva 7).

KUVA 6. Rekursiota käyttävä pyyntö

Mikäli pyyntöjä halutaan tehdä niin kauan, kuin seuraajalistassa on käyttäjätunnuksia, rekursiota jatketaan, kunnes edellisen JSON-tiedoston has_next_page -muuttujan arvo ei ole enää tosi (kuva 7).



KUVA 7. Rekursioon vaadittavat tiedot JSON-tiedostosta

4.3 Pyyntöjen määrä tunnissa

Pyynnöt tehdään Facebookin kehittämään GraphQL -ohjelmointirajapintaan ja Instagram käyttää pyyntöjen rajoituksissa liukuvaa tunnin aikaikkunaa. Yksityisyydensuojan vuoksi ja automatisointia vaikeuttaakseen näitä lukuja ei kuitenkaan kerrota julkisesti yleisen käyttäjän osalta. Vain Facebookin kumppaninohjelmaan hyväksyttyjen sovellusten pyyntörajat ovat julkisia (Facebook, 2021).

Testeissä ilmeni, että täysin rajoittamaton toiminta johtaa noin 240 GET-pyynnön jälkeen estotoimiin, kun toimintaan käytetään täysin uutta tunnusta. Tämä riittää noin 12 000 käyttäjän hakemiseen kohteen seuraajalistasta. Toiminnan venyttäminen usean päivän mittaiseksi saavutti vastaavanlaisia tuloksia äärimmäismäärän osalta. Rajoitukset eivät perustu ainoastaan tunnin aikaikkunaan, vaan ovat myös IP-osoite- ja sessioperusteisia. Lisäksi estojen välittömyyteen vaikutti se, kuinka inhimillistä käytetyn käyttäjätunnuksen toiminta oli. Alkeellinen pyyntöjen aikavälien satunnaistaminen pidensi käytetyn käyttäjätunnuksen elinaikaa, verrattuna tasaisin välein pyyntöjä tekeviin tunnuksiin.

5 TOTEUTUS

5.1 Koneellinen kirjautuminen

Puppeteer Node.js -kirjasto mahdollistaa inhimillisen selainistanssin luonnin ja helpottaa sivustolla navigointia. Valitaan tavallisen työpöytäselaimen ylätunnisteet, luodaan sisäänkirjautumisen koneellistava funktio ja siirrytään kohdekäyttäjätunnuksen juuriosoitteeseen, jotta seuraajalistaa koskevat GET-pyynnöt tulevat oikeasta viiteosoitteesta (kuva 8).

```
const puppeteer = require('puppeteer');

let whoseFollowers = "Instagram";

(async () => {
  const browser = await puppeteer.launch({
    headless: false,
    defaultViewport: null,
    args: ['--window-size=800,600']
  })
  const page = (await browser.pages())[0];
  page.setExtraHTTPHeaders({
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
    "Accept-Encoding": "gzip, deflate, br",
    "Accept-Language": "en-US,en;q=0.9",
    "Referer": "https://www.google.com/",
    "Sec-Fetch-Dest": "document",
    "Sec-Fetch-Mode": "navigate",
    "Sec-Fetch-Site": "cross-site",
    "SEC-FETCH-USER": "?1",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.102 Safari/537.36",
    "X-Amzn-Trace-Id": "Root=1-5f5fe17e-afdab8c01e50caf086574f38"
  })
  page.on('console', consoleObj => console.log(consoleObj.text()));

  //Sign in
  await signInLogic(page);

  //Navigate to a profile
  await page.goto(`https://www.instagram.com/${whoseFollowers}/`)

})();
```

KUVA 8. Puppeteer selainistanssi

Luodaan kirjautumiseen tarvittava logiikka Puppeteer -kirjaston funktioilla sekä yksinkertainen viivästysfunktio toiminnan inhimillistämiseksi. Ohjaillaan dokumenttioliomallin CSS-luokkien avulla (kuva 9).

```
const delay = function delay(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}

const signInLogic = async function signInLogic(page) {

  const user = {
      username: '',
      password: '',
  };

  await page.goto('https://www.instagram.com/accounts/login/');

  await page.waitFor('input[name=username]', { visible: true });
  await delay(345);
  await page.type('input[name=username]', user.email || user.username, { delay: 50 });

  await delay(685);
  await page.type('input[name=password]', user.password, { delay: 50 });

  await delay(685);
  await page.evaluate(() => {
      [...document.querySelectorAll('.sqdOP')].find(element => element.textContent === 'Log In').click();
  });

  await delay(4000);
  await page.evaluate(() => {
      [...document.querySelectorAll('.sqdOP')].find(element => element.textContent === 'Not Now').click();
  });

  await delay(4000);
  await page.evaluate(() => {
      [...document.querySelectorAll('.aOOlW')].find(element => element.textContent === 'Not Now').click();
  });

}
```

KUVA 9. Kirjautumislogiikka

5.2 Halutun tiedon kerääminen ja tallennus

Valitaan listaan sivustosta kaikki ”link” -elementit. Iteroidaan lista läpi ja verrataan jokaisen elementin hyperlinkkiä tiedoston osoitteeseen. Muutetaan HTTP-pyynnön osakäsittelyssä vastaus tekstiksi, verrataan tekstin sisältöä säännölliseen lausekkeeseen ja palautetaan haluttu osa funktiolle (kuva 10).

```
function getHash() {
  let links = document.querySelectorAll('link')
  let consumerJS = "";
  let regex = /const l="[a-z0-9]{32}",_/;
  links.forEach((element)=> {
    if (element.href.includes("/static/bundles/es6/Consumer.js/")) {
          consumerJS = element.href
        }
    })
  return fetch(`${consumerJS}`)
  .then(response => response.text())
  .then(data => data.match(regex)[0].slice(9, 41))
}
```

KUVA 10. Funktio GET-pyynnössä tarvittavan query_hash -muuttujan hakuun

Syötetään funktiolle haluttu käyttäjänimi ja tehdään pyyntö käyttäjän juureen. Muutetaan vastaus JSON-olioksi ja palautetaan käyttäjän id -muuttuja takaisin funktiolle (kuva 11).

```
function getUserId(userName) {
  return fetch(`https://www.instagram.com/${userName}/?__a=1`)
  .then(response => response.json())
  .then(data => data.graphql.user.id)
}
```

KUVA 11. Funktio GET-pyynnössä tarvittavan id -muuttujan hakuun

Syötetään funktiolle seuraajalistan pyyntöön vaadittavat tekstimuotoiset muuttujat ja palautetaan pyynnön vastaus JSON-oliona (kuva 12).

```
function getFollowers(query_hash, variables) {
  return fetch(`https://www.instagram.com/graphql/query/?query_hash=${query_hash}&variables=${JSON.stringify(variables)}`)
  .then(response => response.json())
}
```

KUVA 12. Valmis GET-pyyntö seuraajalistan ensimmäiselle osalle

Luodaan pyyntöjen tuloksista tekstiluokkia, annetaan niille latauslinkki ja tallennetaan ne automaattisesti selaimen latauskansioon (kuva 13).

```
function createTextFile(text, fileName) {
  let data = new Blob([text], {type: 'text/plain'});
  let a = document.createElement('a');
  a.download = fileName + '.txt';
  a.href = window.URL.createObjectURL(data)
  a.click()
}
```

KUVA 13. Funktio haetun tiedon pysyvään tallennukseen

Asynkroninen pääfunktio, jossa julistetaan GET-pyynnölle oleellisten muuttujien lisäksi tallennettavan tiedoston numerointi sekä GET-pyyntöjen rekursio (kuva 14).

```
async function main(userName = "instagram") {
  let follower_amount = 24;
  let file_count = 1;
  let user_id = await getUserId(userName)
  let query_hash = await getHash()
  let variables = {"id": user_id, "include_reel": false, "fetch_mutual": false, "first": follower_amount}

  let followers = await getFollowers(query_hash, variables);
  createTextFile(JSON.stringify(followers), file_count.toString().padStart(3, '0'));
  follower_amount = 13;
  file_count++
  do {
    followers = await getFollowers(query_hash, {...variables, "after": followers.data.user.edge_followed_by.page_info.end_cursor});
    createTextFile(JSON.stringify(followers), file_count.toString().padStart(3, '0'));
    file_count++
  } while (followers.data.user.edge_followed_by.page_info.has_next_page)
}
```

KUVA 14. GET-pyyntöjä koskevan toimintalogiikan kokoava pääfunktio

6 POHDINTA JA JATKOTOIMENPITEET

Tutkimuksen tulokset valottavat erityisesti sosiaalisen median verkkoharavoinnin ongelmallisuutta. Erinäiset henkilötunnistukseen perustuvat rajoitukset, kuten tulevaisuudessa mahdollinen virallisen henkilötodistuksen vaatiminen jokaisen käyttäjätunnuksen luonnin yhteydessä, tekevät ihmisten käyttäytymistä koskevan tiedon keräämisestä mahdotonta ilman palveluntarjoajan erillistä lupaa, mikäli haluttu tieto vaatii kirjautumista. Vaikka kirjautumista ja henkilötunnuksia ei tarvittaisi, sosiaalisen median yhtiöiden kyky käyttäjien toiminnan profilointiin näiden itsensä jakamalla tiedolla johtaa lopulta tilanteeseen, jossa sosiaalisen median yhtiöillä on yksinoikeus heidän alustoillaan kehittyvään ihmistoimintaa koskevaan tietoon. Tämä tieto mahdollistaa jo nykyisellään ihmistoiminnan ja koneellisen toiminnan erotteluun kykenevän neuroverkon. Ajan myötä neuroverkko oppii tunnistamaan, ei pelkästään yksittäisen instanssin koneellista toimintaa sen käyttäytymiskaavasta, vaan myös useita eri instansseja kierrättämällä toteutetut koneelliset tiedonkeruuratkaisut tarkastelemalla suosittuihin tietokokonaisuuksiin keskittyviä instanssirykelmiä. Luvaton sosiaalisen median verkkoharavointi voi olla tulossa tiensä päähän. Tällä on myös hyvät puolensa, sillä se tarkoittaa käyttäjien yksityisyydensuojan paranemista pahansuopia kolmansia osapuolia vastaan.

Tutkimuksen tulosten luotettavuutta yksityiskohtien, kuten pyyntörajoitusten ja estotoimien osalta, ei voida pitää täysin luotettavana, sillä ne ovat jatkuvassa muutoksessa ja niiden täydellinen paljastaminen kokeilevalla testaamisella vaatisi enemmän työtä ja taitoa, kuin tutkimukseen saatiin sijoitettua. Käytettyjä tiedonkeruun periaatteita ja sovelluksia ei kuitenkaan välttämättä tarvitse käyttää juuri sosiaalisen median verkkoharavointiin, ja ne soveltuvat yleisesti minkä tahansa dynaamisen verkkosivun haravointiin. Mikäli halutun tiedon pystyy verkkosivulla havaitsemaan, on se myös aina lähtökohtaisesti koneellisesti kerättävissä. Verkkoharavointitoteutuksen tekijän täytyy vain tehdä päätös sen kannattavuudesta suhteessa sen riskeihin ja hintaan. Massiivinenkin tiedonkeruu on mahdollista hakuinstansseja kierrättämällä, mikäli siitä on valmis maksamaan menetettyjen palvelinosoitteiden ja muiden tarvittavien resurssien muodossa. Suurin osa verkkosivuista ei myöskään ylläpidä tai kouluta Facebookin ja Instagramin tapaan koneellisen toiminnan palvelussaan tunnistavia neuroverkkoja, ainakaan vielä.

Toteutuksessa haluttua tietoa saatiin sen verran niukasti suhteessa luotujen ja menetettyjen käyttäjätunnusten määrään, että jatkoideat tiedotuksen tehostamisesta sen avulla unohdettiin toistaiseksi ainakin siihen saakka, kunnes siihen voitaisiin sijoittaa enemmän aikaa ja alkupääomaa. Ilman massiivista palvelinosoitteiden kierrätystä ja haravoidun tiedon rahoitusmallia, joka kattaisi toiminnan kulut, riskin, työn ja tuoton suhde ei kohtaa.

LÄHTEET

Facebook. (15.4.2010). Automated data collection terms. Haettu osoitteesta https://m.facebook.com/apps/site_scraping_tos_terms.php?hc_location=ufi

Facebook. (11.11.2020 b). Deep Entity Classification (DEC). Haettu osoitteesta https://research.fb.com/publications/deep-entity-classification-abusive-account-detection-for-online-social-networks/

Facebook. (17.6.2021). Rate Limits. Haettu osoitteesta https://developers.facebook.com/docs/graph-api/overview/rate-limiting/

Facebook. (1.10.2020 a). Taking legal action against data scraping. Haettu osoitteesta https://about.fb.com/news/2020/10/taking-legal-action-against-data-scraping/

Facebook, Inc. v. BrandTotal Ltd., Case No. 20-cv-07182-JCS (N.D. Cal. Feb. 19, 2021)

Facebook, Inc. v. Massroot8 https://www.documentcloud.org/documents/6951723-Facebook-lawsuit-against-Massroot8.html

Freelon, D. (2018). Computational research in the post-API age. Political Communication, 35(4), 665–668. https://doi.org/10.1080/10584609.2018.1477506

Instagram. (17.6.2021). Robots.txt. Haettu osoitteesta https://instagram.com/robots.txt

Instagram. (20.12.2020 a). Käyttöehdot. Haettu osoitteesta https://help.instagram.com/478745558852511/?helpref=hc_fnav

Instagram. (13.7.2020 b). Introducing new authenticity measures on Instagram. Haettu osoitteesta https://about.instagram.com/blog/announcements/introducing-new-authenticity-measures-on-instagram

Instagram. (21.8.2020 c). Instagramin tietokäytäntö. Haettu osoitteesta https://help.instagram.com/519522125107875#how-we-use-information

J. Isaak and M. J. Hanna, "User Data Privacy: Facebook, Cambridge Analytica, and Privacy Protection," in Computer, vol. 51, no. 8, pp. 56-59, August 2018. https://doi.org/10.1109/MC.2018.3191268

Rogers, R. (2018). Social media research after the fake news debacle. Partecipazione e Conflitto: The Open Journal of Sociopolitical Studies, 11(2), 557–570. https://doi.org/10.1285/i20356609v11i2p557

Rovetta, Stefano & Suchacka, Grażyna & Masulli, Francesco. (2020). Bot recognition in a Web store: An approach based on unsupervised learning. Journal of Network and Computer Applications. 157. 102577. https://doi.org/10.1016/j.jnca.2020.102577

Venturini, T., & Rogers, R. (2019). “API-Based Research” or How can digital sociology and journalism studies learn from the Facebook and Cambridge Analytica data breach. Digital Journalism, 1–9. https://doi.org/10.1080/21670811.2019.1591927

Yhdysvaltain korkein oikeus. 2020. No. 19-1116 LinkedIn Corporation, v. hiQ Labs, Inc. Julkaistu 25.6.2020. Luettu 1.10.2020. https://www.supremecourt.gov/DocketPDF/19/19-1116/146325/20200625122719793_1%20-%20FINAL%20hiQ%20cert%20opp.pdf

Yhdysvaltain korkein oikeus. 2021. Computer Fraud and Abuse Act, 18 U.S Code § 1030. Haettu osoitteesta https://www.law.cornell.edu/uscode/text/18/1030