VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Twitter srauto duomenų vizualizavimas mobiliuosiuose įrenginiuose

Visualization of Twitter Streamed Data in Mobile Devices

Magistro baigiamasis darbas

Atliko:	Eugenijus Sabaliauskas	(parašas)
Darbo vadovas:	doc. Kristina Lapin	(parašas)
Recenzentas:	doc. Vytautas Čyras	(parašas)

Vilnius – 2016

Santrauka

Šiame darbe nagrinėjamas srauto duomenų analizės ir vizualizavimo priemonės bei konkretus pritaikymas Twitter duomenų vizualizavimui. Literatūros analizės metu išnagrinėta keletą sprendimų, tačiau nebuvo rasta vizualizavimo sprendimo, kuris tenkintų šiuos kriterijus: duomenų srautai, Twitter teksto žinučių srautai, tokių duomenų vizualizavimas būtent mobiliajam įrenginiui ir sprendimo architektūra. Todėl buvo sukurti: naudojimo atvejai, internetinio puslapio sistemos sprendimas ir prototipas su prisitaikančiu dizainu skirtas vizualizuoti Twitter duomenų srauto apdorotą informaciją naudojant skirtingo tipo diagramas. Buvo apžvelgti ir du atskiri alternatyvus sprendimai: pirmas mobili programėlė su vizualizacija ir duomenų apdorojimų, antras – mobili programėlė vizualizavimui kartu su tarpiniu serveriu duomenų apdorojimui. Galiausiai pasirinktas trečias sprendimas – prisitaikančio dizaino internetinio puslapio sistema TweetsViz su tarpiniu duomenų apdorojimo serveriu.

Raktiniai žodžiai: vizualizavimas, duomenų srautas, Twitter, prisitaikantis dizainas, mobilieji įrenginiai.

Summary

This work focuses on streamed data analysis, visualization tools and implementation of Twitter data stream vizualization. During literary analysis a number of solutions were considered, however not all of them could meet all of the following criteria: data streams, textual streamed data of social network Twitter, visualization of Twitter data on a mobile device and architecture of such solution. Therefore, solutions were created: detailed use cases, a website system architecture diagram and a prototype of that website with responsive design. Created website can visualize Twitter streamed data using different diagrams. Two separate alternative solutions were reviewed: first – a mobile app with visualization and data processing, second – a mobile app with visualization and a separate middleware server for data processing. Finally, a third solution was selected– a website system TweetsViz with responsive design an a middleware processing server.

Keywords: visualization, streamed data, Twitter, responsive design, mobile devices.

TURINYS

ĮVADAS

Šiais laikais mobilieji įrenginiai sparčiai plinta. Juos galima suskirstyti į dvi sritis: sumanieji mobilieji telefonai (angl. *smartphone*) ir planšetės (angl. *tablets*). Mobiliųjų įrenginių techninis galingumas, ekranai bei jų raiška didėja [IDC13b], naudojami galingesnės vaizdo kortos, todėl atsiranda galimybė išnaudoti daugiau įrenginio resursų vizualizavimui. Visi šiuolaikiniai mobilieji įrenginiai turi galimybę naudoti EDGE, 3G ir/arba WiFi ryšį, todėl galima nuolat priimti duomenų srauto duomenis. Tačiau ar visos mobiliųjų įrenginių galimybės išnaudojamos? Yra sričių, kuriuose galima kombinuoti kelias technologijas ir įgyvendinti naujus sprendimus. Viena iš tokių sričių tai duomenų srauto vizualizavimas.

Vizualizavimas – tai „skaitinių ir tekstinių duomenų pavertimas grafiniais vaizdais: diagramomis, grafikais, schemomis“ [DGJ14]. „Vizualizavimas leidžia geriau suvokti sudėtingas duomenų aibes, gali padėti nustatyti dominančius jos poaibius. Vizualią informaciją žmogus pajėgus suvokti daug greičiau negu tekstinę, ji palengvina naujų žinių atradimą“ [DKŽ08]. Svarbu išnagrinėti galimus duomenų vizualizavimo metodus mobiliuose įrenginiuose, nes kol kas nėra vieno nusistovėjusio metodo, kuris būtų paplitęs (atmetus filmų srauto vaizdavimą) būtent srauto duomenims. Iki šiol buvo rasti tokie variantai: vaizdų atkūrimas mobiliame įrenginyje, VRML, video kodavimas ir dekodavimas [PCB+11]. Galimi vizualizavimo būdai: histograma, linijinis grafikas, grafas, žemėlapis su statistika, žodžių debesis, radialinės diagramos (angl. *radial chart*) [MKL+13] [RLC+13] [ROR13] [SM08]. Naudojant vizualizavimą galima bus leisti vartotojui filtruoti duomenis pagal pasirinktus parametrus, ir taip paspartinti vartotojo duomenų analizę ir įsisavinimą, išryškinti svarbiausius duomenis ir palengvinti sprendimų priėmimą [DKŽ08].

Norint tirti šią temą taip pat būtina pasirinkti konkretų duomenų šaltinį. Twitter tai internetinis socialinis tinklas (virš 200 mln. vartotojų) paleistas 2006 m., kuriame vartotojai gali parašyti iki 140 simbolių žinutes. Šios žinutės gali būti matomos viešai kitų vartotojų. Vartotojai taip pat gali prenumeruoti kitų vartotojų žinučių srautą. Taigi Twitter žinutes galima traktuoti kaip srauto duomenis, nes yra galimybė užsakyti žinučių srautą, kuris vartotoją pasieks kai tik atsiras naujos žinutės. Taip pat Twitter ypatingas tuo, kad jame galima sukurti ir priskirti metaduomenų žymę (angl. *hashtag*), o vėliau pagal tą žymę naršyti kitų žmonių žinutes. Twitter turi viešai prieinamą taikomųjų programų programavimo sąsają (angl. *Twitter API*) [ZK13]. Tai suteikia galimybę sukurti taikomąja programą, kuri galės parsisiųsti Twitter žinučių duomenų srautą pagal pasirinktus parametrus, o tada tuos duomenis vizualizuoti.

Temos aktualumas ir naujumas

Kol kas nėra viešai prieinamo vizualizavimo modelio skirto vizualiai pateikti tekstinio srauto duomenų analizės rezultatus mobiliuosiuose įrenginiuose. Mobilieji įrenginiai turi įvairių dydžių ekranus, todėl vienas sprendimas nebūtinai veiks visuose įrenginiuose. Efektyviai vizualizuoti daug duomenų mažame ekrane gali būti sudėtinga dėl ekrano užgriozdinimo [GKG+10]. Dar vienas iššūkis yra sparta. Pastebima tendencija, kad mobiliųjų įrenginių vartotojai nori, kad rezultatai būtų pateikti akimirksniu ir jei programėlė veikia lėtai, ją ištrina ir daugiau prie jos nesugrįžta.

Twitter žinučių gausa gali priblokšti, joje sunku greitai surasti ryšius, juos paskaičiuoti ir greitai padaryti išvadas [RLC+13]. Taip pat šiuo metu yra Twitter duomenų vizualizavimo sprendimų, tačiau tie sprendimai yra riboti ir nepritaikyti mobiliesiems įrenginiams. Taigi šis socialinis tinklas yra puikus srauto duomenų šaltinis ir bus tinkamas šiam tyrimui. Apie temos naujumą galime spręsti iš 2012-2013 metais pasirodžiusių mokslinių straipsnių ir iš tokių sistemų kaip TweetXplorer ir ThemeStreams.

TweetXplorer – tai Twitter žinučių analizės ir vizualizavimo programos sistema sukurta 2012 metais. Ši sistema skirta analitikams, kuri palengvina analizės užduotis ir padeda padaryti gilesnes įžvalgas bei išvadas apie įvykusį incidentą arba įvykį naudojant vizualizavimo metodus. Kaip pavyzdys buvo pateiktas žinučių vizualizavimas (JAV žemėlapyje) uragano „Sandy“ metu [MKL+13]. Ši sistema neveikia ant mobiliųjų įrenginių.

Olandijos mokslininkų sukurtas Twitter politinių žinučių analizės ir vizualizavimo sprendimas „ThemeStreams“ buvo pristatytas 2013 metais [ROR13], tačiau šis sprendimas turi apribojimų: vizualizavimas neveikia ant mobilių įrenginių (naršyklėse Opera ir Chrome), vizualizuojami duomenys nėra srauto duomenys (t.y. duomenys iš anksto surinkti), turi ribotą Twitter soc. tinklo žinučių kiekį (2012 metai, Olandijos teritorijoje).

Šiame darbe sukurtas sprendimas gali būti naudojamas tokiuose srityse kaip žurnalistika, marketingas ir sociologija [SM08].

Darbo tikslas

Sukurti Twitter srauto duomenų vizualizavimo algoritmą mobiliam įrenginiui skirtą pavaizduoti duomenų tematikų apibendrinimą pagal raktinius žodžius, efektyvinti informacijos paiešką bei analizę ir padėtų išskirti tendencijas. Sprendimas turi veikti tiek asmeniniame kompiuteryje, tiek mobiliuose įrenginiuose.

Darbo uždaviniai

Tikslui pasiekti bus sprendžiami šie uždaviniai:

1. Ištirti ir palyginti esamus duomenų vizualizavimo būdus.
2. Išnagrinėti kaip galima Twitter žinutes surinkti, išsaugoti ir išanalizuoti.
3. Pasirinkti tinkamus atviro kodo įrankius, paslaugas ir kitą programinę įrangą.
4. Sudaryti vizualizavimo algoritmą ir jį realizuoti pagal pasirinktą metodą.

Uždaviniams išspręsti naudojami šie metodai: dalykinės srities analizė, mokslinės literatūros analizė ir apibendrinimas, modelio projektavimas ir prototipo įgyvendinimas.

Darbo rezultatai

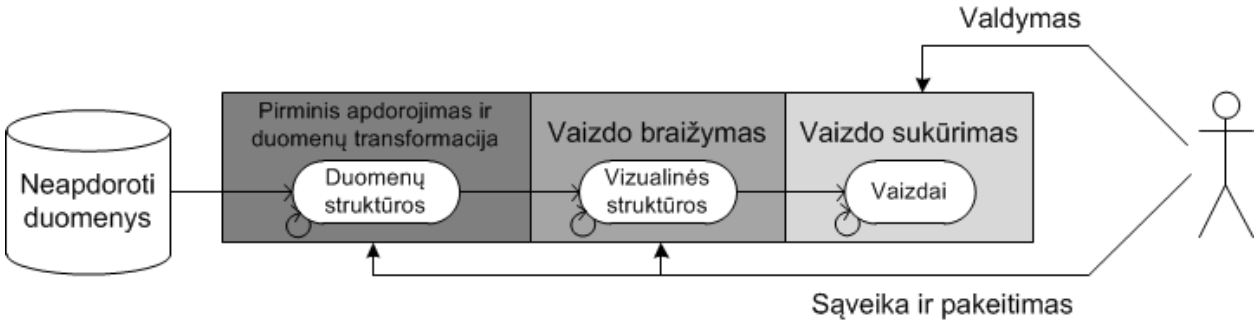
Toliau pateikiami uždavinių rezultatai:

1. Vizualizavimo būdų palyginimas.
2. Twitter prieigos paslaugos panaudojimo aprašymas ir įrankių naudojimo tyrimas.
3. Vizualizavimo algoritmas ir naudojimo atvejai.
4. Vizualizavimo sistemos architektūra
5. Srauto duomenų vizualizavimo prototipas naudojant Twitter prieigą.

1. Esamų vizualizavimo būdų analizė

1.1. Vizualizacijos kūrimo procesas

Vizualizavimas – tai grafinis informacijos pateikimas [DKŽ08]. Pagrindinė vizualizavimo idėja – duomenis pateikti tokia forma, kuri leistų tyrėjui juos suprasti, daryti išvadas ir tiesioginę įtaką tolesniam jų srautui [DKŽ08]. Vizualizacija yra naudinga, nes padeda: geriau suprasti sąryšius, struktūrą ir tendencijas, suvokti kiekybinių duomenų prasmę, suprasti koreliacijas tarp duomenų rinkinių, padaro „nuobodžius“ duomenis patrauklesniais, suteikti didelę vertę per trumpą laiko tarpą [DS13]. Vizualią informaciją žmogus pajėgus suvokti daug greičiau negu tekstinę, ji palengvina naujų žinių atradimą [DKŽ08], ypatingai kai yra daug duomenų ar informacija yra abstrakti. *Abstraktūs duomenys* – tokie duomenys, kurie nebūtinai turi sąryšį su fizine erdve, pvz.: žmonių vardai, produktų kainos, balsavimų rezultatai ir pan. Tokie duomenys turi būti tinkamai apdorojami, prieš pavaizduojant juos grafiškai [Maz09].

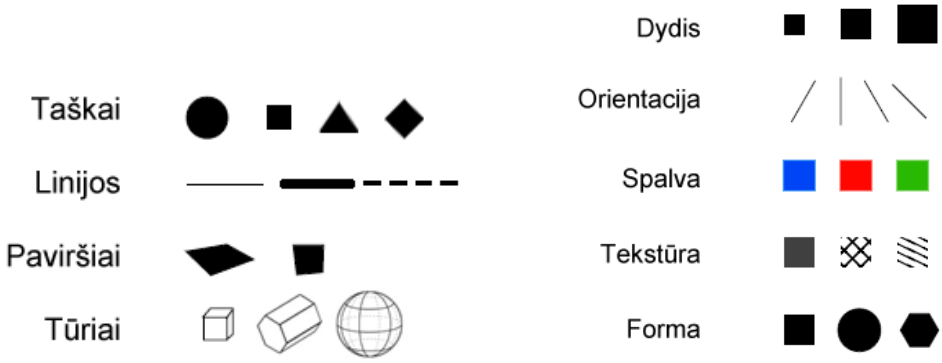


1 pav. Grafinio vaizdo sukūrimo procesas [Maz09]

Grafinio vaizdo sukūrimo procesas susideda iš sekančių žingsnių (1 pav.):

1. Duomenų atrinkimas, apdorojimas, apskaičiavimas, naujų duomenų sukūrimas.
2. Vaizdo braižyme svarbu apibrėžti vaizdavimo struktūras, kurias norime pavaizduoti ir kurios vaizduos duomenų struktūras,. Svarbu atsižvelgti į:
    a. erdvinį pagrindą – kokio dydžio bus vizualizacija, kokios bus naudojamos ašys, ašių matai ir jų tipas (skaitinis, logaritminis, kokybinis, laiko ir pan.).
    b. grafinius elementus – taškai, linijos, plokštumos, tūris ir pan. (2 pav.).
    c. grafines savybes – dydis, orientacija, spalva, tekstūra, forma ir pan. (2 pav.).
3. Vaizdo sukūrimo metu mes paimame tuščią vaizdo braižymo pagrindą (pvz. koordinačių sistemą) ir perkeliame duomenų struktūras į vizualinę struktūrą. Tokiu būdu gauname vizualinę duomenų reprezentaciją. Svarbu taikyti teisingą erdvės ir

duomenų proporciją, kad nebūtų informacijos pertekliaus, bet ir nebūtų informacijos stokos.



2 pav. Kairėje – grafinių elementų pavyzdžiai, dešinėje – grafinių savybių pavyzdžiai [Maz09]

1.1.1. Duomenų apdorojimas

Turint baigtinį duomenų rinkinį, duomenų apdorojimas yra paprastesnis uždavinys nei srauto duomenų analizė. Norint sukurti vizualizaciją pagal Twitter žinutes, mums teks atlikti tekstinę raktažodžių paiešką. Vieną žodį surasti tarp 140 simbolių galima atlikti pasitelkus Morris-Pratt ar panašiais algoritmais. Tačiau sudėtingumas prasideda, kai norime rasti ir tokius žodžius kurie skiriasi galūnę, priesaga, didžiosiomis raidėmis ar yra raktažodžio sinonimas. Dar labiau užduotį apsunkina tai, kad duomenys bus siunčiami pastoviai. Mums pavyko surasti mokslinį darbą [Fon12], kuriame aprašyti algoritmai ir semantinės analizės modelis. Apie tai užsiminta ir moksliniame darbe [ACZ+11] apie „STREAMIT“ sistemą. Šias žinias galima bus pritaikyti, jei bus nuspręsta semantiškai analizuoti Twitter žinutes.

1.1.2. Vizualizacijos išmatavimai

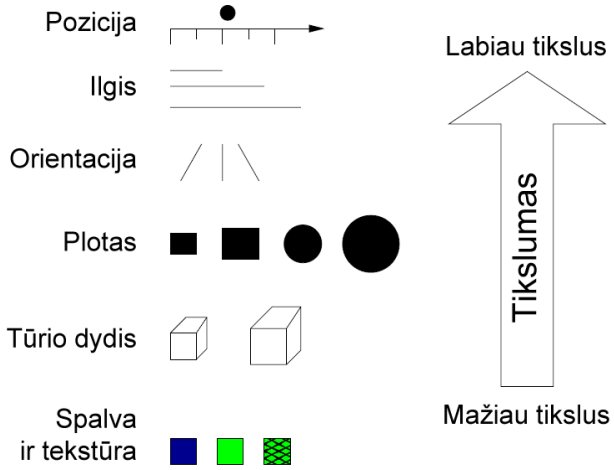
Vaizdo braižymo metu mums svarbu nustatyti ne tik vizualizacijos rėmus, bet ir ašių išmatavimus. Grafiko skalės parinkimas priklauso nuo kintamųjų tipo. Kategoriniams kintamiesiems tai daugiau yra informatyvaus sutvarkymo klausimas, nes priklauso nuo to, ką atvaizduoja kategorijos. Tolydiems kintamiesiems reikia parinkti kraštinius taškus ir sudalinti intervalą [Pla11]. Kadangi srauto duomenys bus siunčiami pastoviai ir jų prognozuoti negalime, todėl turėsime parašyti algoritmą (arba naudoti esamą) kraštutiniams taškams apskaičiuoti kiekvieną kart kai ateina nauji duomenys.

Vizualizacija gali būti vertikali, kvadrato formos arba horizontali. Tai priklauso nuo to kur ji įtraukiama (laikraštyje, moksliniame straipsnyje, internetiniame puslapyje). Mobiliųjų įrenginių ekranai būna vertikalūs, tačiau yra galimybė juos pasukti. Todėl verta grafiką vaizduoti stačiakampio formos, kur ilgesnė briauna yra horizontali, o santykis su vertikalia ašimi būtų

1:1.5 arba su artima auksiniam santykiui 1:1.618. Žmogaus akys geriau įsisavina informaciją, kuri yra pavaizduota horizontaliai [TG83].

1.1.3. Grafinių savybių suvokimas

Ne visos grafinės savybės veikia žmogaus vizualinį suvokimą vienodai. Kai kurios grafinės savybės yra efektyvesnės nei kitos kai yra vaizduojamos kiekybinės reikšmės [Maz09]. 3 pav. pavaizduota tyrimų rezultatai, kurie padeda mums apsispręsti kurias savybes geriausiai naudoti. Nors spalvos yra labai gerai atskiriamos, ši savybė yra apačioje, nes yra žmonių, kurie ne taip atskiria spalvas. Taip pat spalvos gali atrodyti vienaip kompiuteryje, o atspausdintos kitaip. Taigi svarbu naudoti tokias spalvas, kurių skirtumas būtų didžiausias, kad būtų lengva atskirti duomenų grupes (aibes). Ilgiau laiko reikia, kad atskirti tankias tekstūras, tačiau jos padeda, kai turime juodai baltą vaizdavimą. Tiksliausiai atskiriame kiekybines reikšmes, kurios yra pavaizduotos konkrečioje pozicijoje ir mes galime maždaug paskaičiuoti taško reikšmę skalės pagalba.



3 pav. Grafinių ir erdvinių elementų suvokimo tikslumas esant kiekybinėms reikšmėms [Maz09]

1.1.4. Vizualizacijų kūrimo technologijos ir karkasai

Norint sukurti vizualizaciją mobiliajame įrenginyje ir dar ją atnaujinti realiu laiku, turėsime surasti tinkamus karkasus ir atlikti pakeitimus, kad pritaikyti prie mūsų uždavinio. Moksliniuose darbuose [LK12], [MF12], [OK13] pateikta detali informacija apie tai kokias technologijas vertėtų naudoti šiuolaikiniuose mobiliuose įrenginiuose. Tai HTML5 Canvas, SVG, WebGL, tokie JavaScript karkasai kaip: Highcharts, Processing.js, Paper.js, Data Driven Documents (D3), Tree.js, ir pan. [LK12] [MF12] [OK13]. Šiame darbe neatlikinėsime karkasų ir kitų technologijų palyginimo ir pasirinkimo, kadangi dar nesudarėme sistemos modelio.

1.2. Vizualizavimo būdai

Šiame skyriuje panagrinėsime 2D informacijos ir duomenų vizualizacijas (diagramos, grafikai, žemėlapiai), kurias suskirstėme į paprastąsias ir sudėtingesnes.

1.2.1. Paprasti vizualizavimo būdai

Šiame skyriuje trumpai įvardinsim paprasčiausius vizualizavimo būdus tam, kad turėti vienareikšmišką terminologiją, ir kad kituose skyriuose galėtume detaliau aprašyti sudėtingesnes diagramas ir grafikus, kurie dažniausiai yra paremti paprastaisiais. Paprastos diagramos dažniausiai vaizduojamos koordinačių sistemoje su *x* ir *y* ašimis, kurios dažniausiai, bet nebūtinai, tiesiamos nuo nulio į dešinę ir į viršų atitinkamai.

Taškinė diagrama (angl. *scatter plot*) tai tokia diagrama, kurioje duomenys vaizduojami taškais arba ženkleliais, kurių koordinatės atitinka duomenų reikšmes[1].

Linijinė diagrama (angl. *line chart*) tai tokia diagrama, kurioje „vieno arba kelių duomenų rinkinių reikšmės vaizduojamos dvimatėje koordinačių sistemoje ir jungiamos linijomis“[2]. Linija dažniausiai tęsiasi pagal *x* ašį iš kairės į dešinę. Jei *x* ašis reprezentuoja laiką, tuomet paprasta pastebėti *y* ašies reikšmių pokytį laikui bėgant. Linijinė diagrama skirta kiekybiniams duomenims (skaičiams) vaizduoti, todėl gerai tinka tolydiems kintamiesiems [Pla11].

Plokštuminė diagrama (angl. *area chart*) yra linijinė diagrama su papildomai nuspalvintu plotu nuo kiekvienos linijos viršaus iki *x* ašies. Kai vienos yra pastoviai didesnė, tuomet jinai piešiama mažesnės kreivės fone, kad paryškinti skirtumą. Kuomet kreivės persidengia, tuomet naudojamos skaidrios spalvos, kad būtų galima matyti abi kreives.

Stulpelinė diagrama (angl. *bar chart*) ir ***histograma*** (angl. *histogram*) tai tokios diagramos, kurių kiekvienas duomenų elementas vaizduojamas vertikaliu stačiakampiu. Jų aukštis proporcionalus reprezentuojamo kintamojo dydžiui. Stulpelinė diagrama skirta kokybiniams (savybė, rūšis, ypatybė) duomenims vaizduoti, todėl gerai tinka kategoriniams kintamiesiems [Pla11]. Histogramoje vertikalūs stulpeliai skirti kintamojo pasiskirstymui pavaizduoti, kur *x* ašis kažkokios savybės reikšmė, o *y* ašyje – tos savybės dažnumas. Histogramos stulpeliai liečia vienas kitą, kad būtų galima lengviau juos palyginti. ***Juostinė diagrama*** yra panaši į stulpelinę, išskyrus tai, kad stulpeliai vaizduojami horizontaliai ir jų plotis proporcionalus dydžiui.

[1] http://ims.mii.lt/EK%C5%BD/t/ta%C5%A1kin%C4%97%20diagrama.html

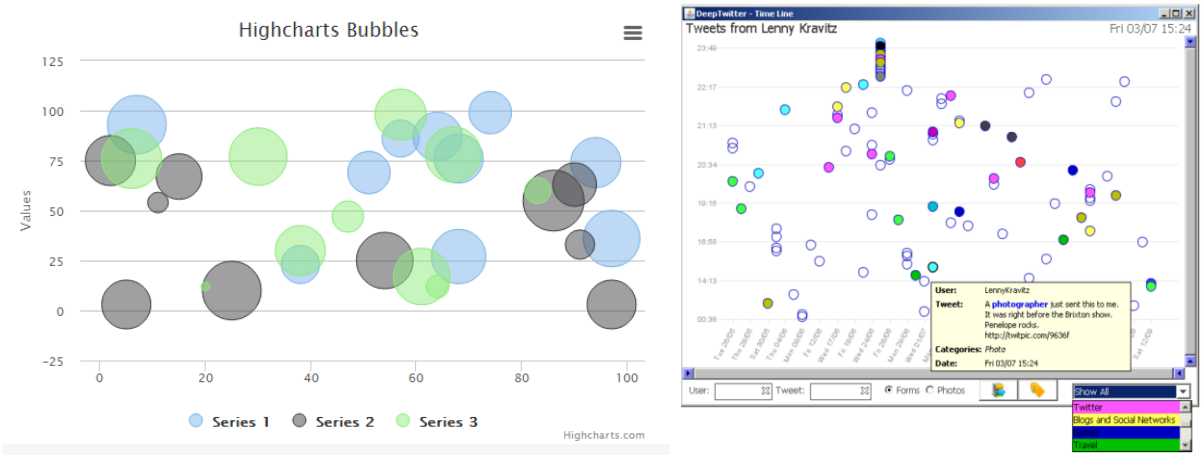
[2] http://ims.mii.lt/ims/%C5%BEodynai/term/l/linijine2_diagrama.html

Skritulinė diagrama (angl. *pie chart*) tai tokia diagrama, kurios reikšmės pavaizduojamos skritulio išpjovomis. Diagramoje sektoriaus lanko ilgis reprezentuoja procentinį sektoriaus dydį palyginus su viso skritulio plotu. Ši diagrama gerai tinka kategoriniams kintamiesiems [Pla11].

Sukrautų stulpelių diagrama (angl. *stacked column diagram*) tai stulpelinė diagrama, kur stulpelis sudarytas iš kelių stulpelių sukrautų vienas ant kito vertikaliai. Tokia diagrama naudinga, kai būtina parodyti kintamųjų sumą konkrečią dieną (ar laikotarpiu), bet kartu ir išryškinti kiekvieno kintamojo reikšmę.

1.2.2. Sudėtingesni vizualizavimo būdai

Rutulinė diagrama (angl. *bubble chart*) tai tokia taškinė diagrama, kurioje galima vaizduoti tris dimensijas duomenų. Pirmi du dydžiai vaizduojami *x* ir *y* ašyse, o trečias dydis priklauso nuo rutulio diametro ar nuo ploto dydžio. Geriausia naudoti ploto dydį, nes žmogus geriausiai atskiria rutulius pagal jų plotą [TG83]. Šioje diagramoje galima naudoti kelias aibes duomenų ir jas žymėti skirtingomis spalvomis, kad lengviau atskirti.

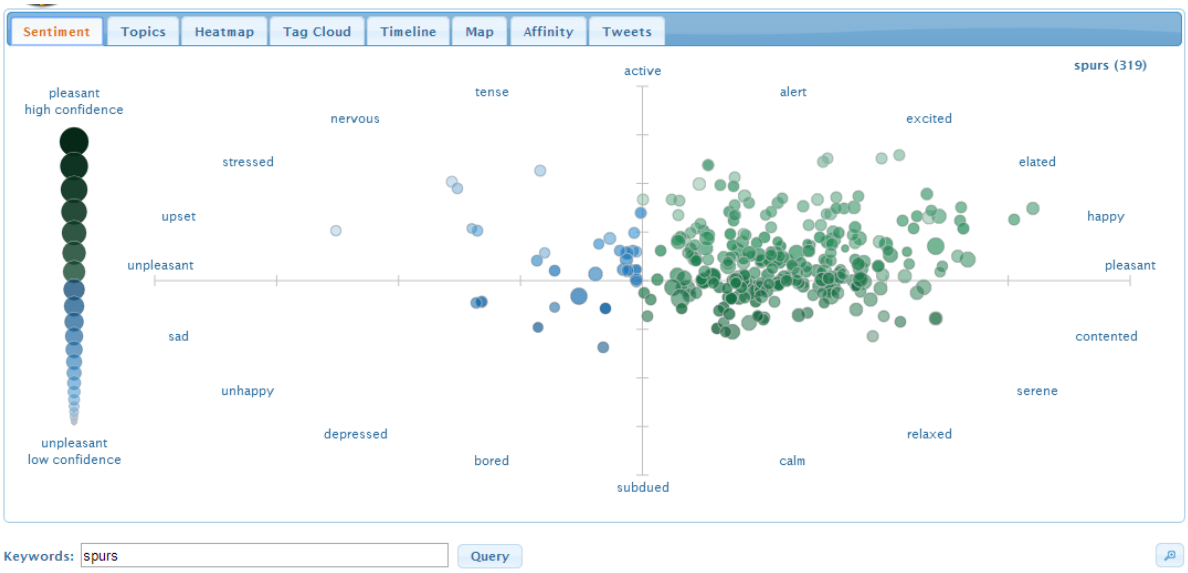


4 pav. Kairėje Highcharts karkaso rutulinė diagrama. Dešinėje DeepTwitter Twitter žinučių pasiskirstymas pagal laiką ir kategoriją [RLC+13]

Pavyzdžiui 4 pav. kairėje matome rutulinę diagramą su trim duomenų aibėm. Tokia vizualizacija tinka duomenų palyginimui, tačiau gali būti sunku nustatyti konkrečius skirtumus. Rutuliai yra skirtingų dydžių ir jeigu nėra greta jų parašytos *x* ir *y* reikšmės, tuomet jas nustatyti darosi sunku. O 4 pav. dešinėje matome, kad DeepTwitter pagrindiniame lange galima vaizduoti vartotojo Twitter žinutes chronologiškai. Duomenys paskirstomi pagal dieną (*x* ašis) ir laiką (*y* ašis) kada žinutės buvo išsiųstos. Tai leidžia tyrėjui stebėti vartotojo elgesio tendencijas, pavyzdžiui, kuriuo paros metu žinutės išsiunčiamos dažniausiai. Žinutės dar skirstomos į kategorijas pagal spalvą: pilka kategorija reiškia žinutės susijusios su Twitter, geltona – žinutės su tinklaraščiais ir socialiniais tinklais, mėlyna – su žaidimais ir t.t.

Išsibarstymo diagrama, taškinis grafikas (angl. *scatter plot*). Taškinis grafikas tai supaprastinta linijinė diagrama tik be linijų, kur kintamieji tiesiog pavaizduojami taškais koordinačių sistemoje. Išsibarstymo diagramoje dažniausiai vizualizuojama dviejų kintamųjų aibių taškų išsibarstymas koordinačių sistemoje. Abu jie gali vaizduoti duomenis dviejuose dimensijose. Diagrama puikiai tinka palyginimui, kai aibių taškai yra nuspalvinami skirtingomis spalvomis, o dar geriau jei aibės taškai koncentruojasi konkrečiuose plotuose.

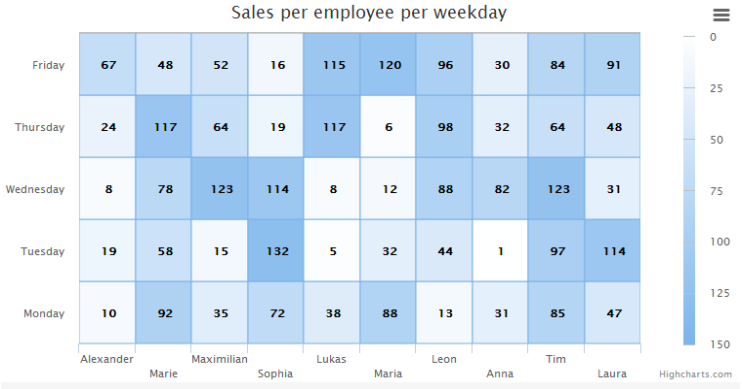
TweetViz tai programa, kurioje vaizduojamas Twitter žinučių emocinis įvertinimas naudojant išbarstymo diagramą, kurią svarbu paminėti dėl originalaus dimensijų pavaizdavimo (5 pav.). Twitter žinutės skirstomos ne tik į teigiamą ir neigiamą, kaip tai daroma kitame darbe [SB13], bet į 20 skirtingų jausmų. Dvi pagrindinės dimensijos yra: nemalonus – malonus (*x* ašis, iš kairės į dešinę) ir pasyvus (prislopintas) – aktyvus arba energingas (*y* ašis). Rutuliai skiriasi savo **spalva** (žalia, tai maloni emocija, o mėlyna – nemaloni), **ryškumu** (kuo rutulys šviesesnis, tuo jausmas aktyvesnis), **dydžiu** ir **skaidrumu** (kuo didesnis ir mažiau skaidrus rutulys, tuo didesnė tikimybė, kad jausmas apskaičiuotas teisingai) [HR13].



5 pav. TweetViz vaizduoja Twitter žinučių emocines reikšmes [HR13]

5 pav. pavaizduotas konkretus naudojimo atvejis su 319 išnagrinėtų Twitter žinučių, kurios buvo rastos pagal paieškos raktinį žodį „spurs“. „Spurs“ tai krepšinio komanda, kuri 2014 metais laimėjo NBA čempionatą. Pagal šią vizualiazaciją galime daryti išvadą, kad dauguma žinučių autorių buvo patenkinti rašydami apie Spurs. TweetViz taip pat leidžia paspausti ant kiekvieno taško ir perskaityti konkrečią žinutę. Galima peržiūrėti kelias tų pačių duomenų vizualizacijas, tokias kaip: spalvinė diagrama, žodžių debesis, sukrautų stulpelių diagrama (1 priedas, 46 pav.), žemėlapis (iš kurių JAV valstijų kilusios žinutės) ir grafą [HR13].

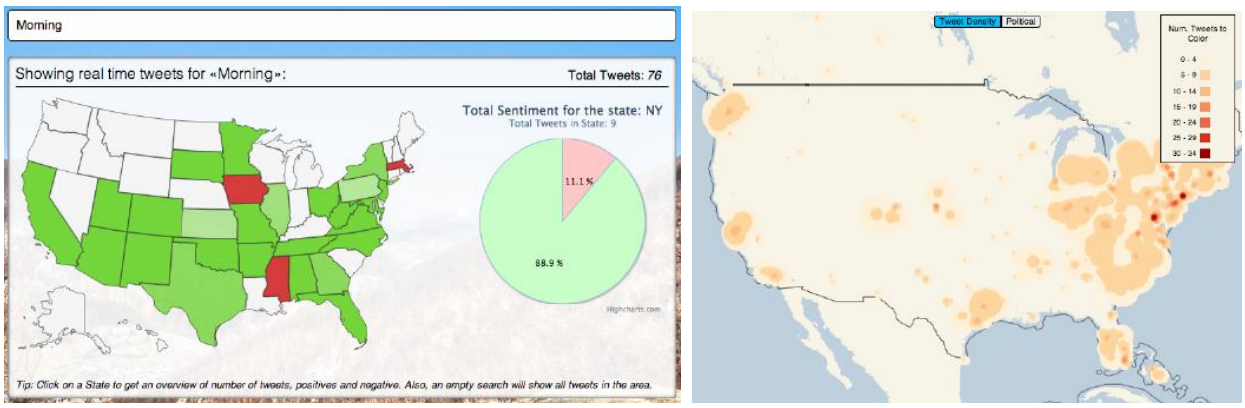
Spalvinė diagrama (angl. *Heatmap*). Išvertus pažodžiui tai šilumos žemėlapis. Paprasčiausia spalvinė diagrama yra stačiakampio formos ir atrodo kaip lentelė su daug langelių. Kiekvieno langelio dydį parodo spalva ir skaičius (ne visada). Kuo daugiau langelių, tuo svarbesnis yra nuspalvinimas. Kuo didesnis skirtumas tarp spalvų, tuo lengviau suprasti dydžių skirtumą. Tarkime kai dėmesio verti langeliai nudažyti raudonai, o mažiau svarbūs – žaliai, tuomet vizualizacija lengviau analizuoti negu nespalvotus langelius su skaičiais. Kaip pavyzdį pateikiame Highcharts karkaso spalvinę diagrama, kurioje parodoma pardavėjų pardavimų statistika savaitės dienomis (6 pav.). Daugiausiai pardavimų padariusių pardavėjų langeliai paryškinti tamsiai mėlyna spalva, o mažiausiai – balta.



6 pav. Pardavėjų pardavimų statistika savaitės dienomis. „Highcharts“[3] karkaso demonstracija

Žemėlapis (angl. *Map*). Žemėlapiu vadinsime tokią vizualizacija, kuri turi panašią teritoriją ir politines ribas, bet skiriasi nuo politinio žemėlapio tuo, kad atskiri jos plotai pateikia kažkokius statistinius duomenis ir tie plotai yra skirtingai nuspalvinti. 7 pav. kairėje matome žemėlapio vizualizaciją iš programos SentiMap, kuri pagal raktažodį ieško Twitter žinutes, analizuoja jų tekstą ir apskaičiuoja vartotojų nuotaiką (pozityvi, negatyvi). Taikomoji programa naudoja algoritmus paremtus statikos ir kompiuterių mokymosi teorijomis: naiviu Bajeso (angl. *Naïve Bayes*) klasifikatoriumi, maksimalia entropija (angl. *maximum entropy*) ir vektorinėmis mašinomis (angl. *support vector machines*). 7 pav. kairėje raudona spalva reiškia neigiamą, žalia – teigiamą, o balta – neutralią nuotaiką [SB13]. Raudonai pažymėtos valstijos yra Ajova, Misisipė ir Masačiusetsas. Šis sprendimas analizuoja tik tas žinutes, kurios turi išsaugotas geografines koordinates, t.y. iš mobiliųjų įrenginių su GPS davikliu.

[3] http://www.highcharts.com/demo/



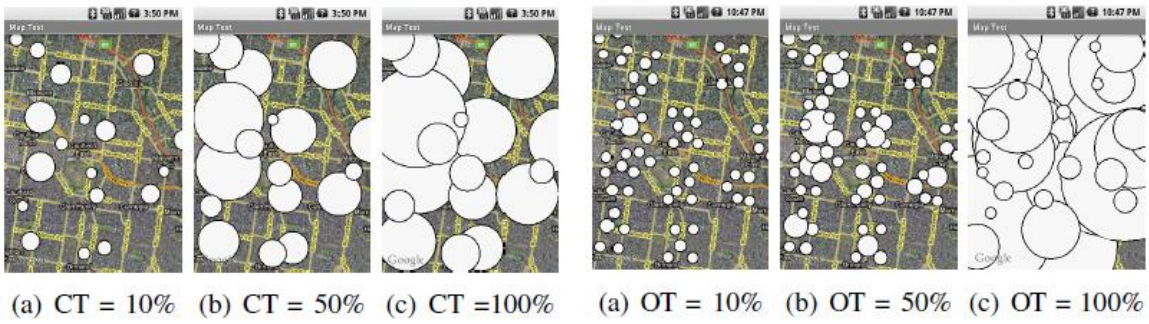
7 pav. Kairėje Twitter vartotojų nuotaiką vaizduojantis SentiMap JAV žemėlapis [SB13], dešinėje TweetXplorer taikomosios programos Twitter žinučių pasiskirstymas žemėlapyje [MKL+13]

Žemėlapio vizualizacija labai gerai tiktų parodyti kuri šalis (ar teritorija) yra labiausiai minima Twitter žinutėse kalbant apie Eurovizijos konkursą. Tokiu būdu galima būtų nuspėti favoritus, dar nepasibaigus balsavimui. Dar vienas būdas panaudojimui – Lietuvos žemėlapis su apskritimis, kuriame būtų pavaizduoti apskritimai virš didžiųjų miestų (Vilnius, Kaunas, Klaipėda, Šiauliai, Panevėžys, Alytus ir pan.), kurių diametras parodytų santykį tarp populiarumo Twitter žinutėse. Apskrities spalva galėtų būti emocinis rodiklis (mėlyna – teigiama, violetinė – neutrali, raudona – neigiama).

Gyvenvietės žemėlapiu vadinsime tokią vizualizaciją, kurioje žemėlapio plotas yra priartintas iki tokio lygio, kad matosi skirtumas tarp gatvių, gyvenamųjų plotų, miško teritorijų ir pan. Ši vizualizacija ypatingai paplitusi mobiliuose įrenginiuose (pvz. „Google Maps“, „Foursquare“ [4], „Maršrutai“ [5]), nes joje patogu vaizduoti gatves, maršrutus ir populiarius lankomus objektus žemėlapyje. Tačiau ką daryti kuomet objektų skaičius per didelis. Ši problema sprendžiama moksliniame darbe apie mobiliųjų įrenginių ekrano užgriozdinimą [GKG+10]. Šis darbas siūlo prisitaikančią netvarkos mažinimo (angl. *Adaptive Clutter Reduction – ACR*) teoriją skirta vizualizacijoms mažuose ekranuose. Pagal šią teoriją sukurtas vizualizacijos metodas angliškų pavaidinimu „Clutter-Aware Clustering Visualizer“ (CACV) [GKG+10]. Darbe aprašomi daugybė parametrų, kurie nusako ekrano užgriozdinimą, kaip pvz. pateikiam du parametrus: CT – priimtino lygio ekrano užpildymas objektais išreikštas procentais. OT – leistinas objektų persidengimo dydis procentais. 8 pav. matome, kad mažinant šituos kintamuosius mažėja ir užgriozdinimas, o tai ypač svarbu, kai didėja objektų skaičius mobiliajame ekrane [GKG+10].

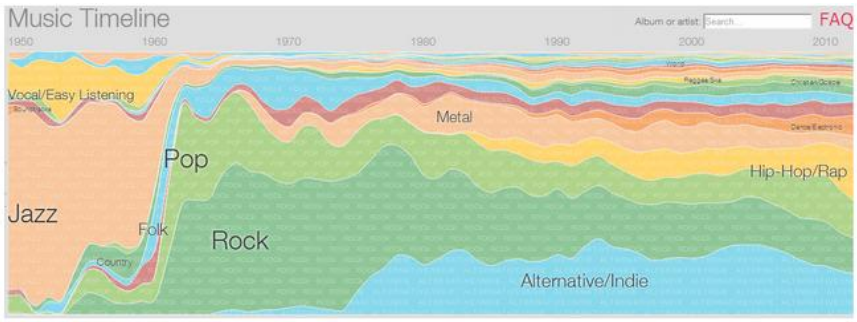
[4] https://play.google.com/store/apps/details?id=com.joelapenna.foursquared&hl=en

[5] https://play.google.com/store/apps/details?id=lt.marsrutai.mobile.android&hl=en



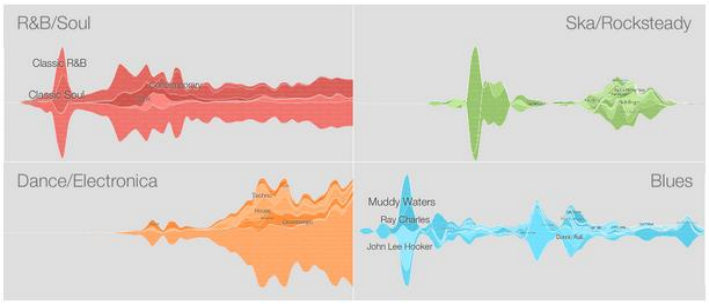
8 pav. Clutter-Aware Clustering Visualizer technologija [GKG+10]

Tėkmės diagrama (angl. *Streamgraph*). Tai tokia daugiasluoksnė plokštuminė diagrama, kurioje sukrautos kelios plokštumos aplink horizontalią centrinę ašį (laikas). Kuo daugiau plokštumų, tuo labiau jos atrodo kaip juostos išsidrėkusios per tam tikrą laikotarpį. Diagramoje galima pamatyti kaip keitėsi kintamasis pagal jų besikeičiantį plotį. Ši diagrama išpopuliarėjo, kai New York Times išspausdino vizualizaciją rodančią filmų pardavimus per 21 metus [BW08].



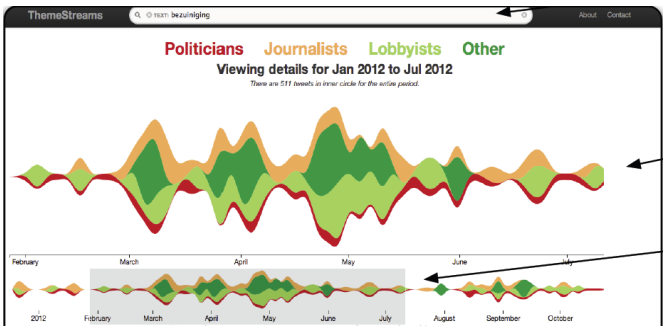
9 pav. Muzikos žanrų evoliucija (1950 – 2010 m.) ir populiarumas [CL14]

Tėkmės diagramą taip pat panaudojo Google darbuotojai, kad interaktyviai vizualizuoti muzikos žanrų populiarumo tendencijas nuo 1950 metų [CL14] (9 pav.). Pagrindinė vizualizacija yra procentinė plokštuminė diagrama, tačiau paspaudus ant konkretaus žanro (tarkim Rock) atsidaro jau detalesnė to žanro tėkmės diagrama. Joje galima matyti subžanrus (pvz.: 60s, 70s, Psychodelic Rock, Classic Rock ir pan.), kuriuos reprezentuoja skirtingais atspalviai nudažytos plokštumos. Paspaudus ant subžanro, pvz. Psychodelic Rock, vėl atsidaro nauja tėkmės diagrama, kurioje jau matyti šio subžanro grupės ir atlikėjai, pvz.: The Doors, Jimi Hendrix ir pan. Paspaudus ant atlikėjo, pvz. Jimi Hendrix, parodomi albumų paveiksliukai bei to atlikėjo ar grupės aprašymas. Naudojant šią interaktyvią vizualizaciją galima greitai palyginti kada koks žanras buvo populiarus, pvz. „R & B / Soul“ žanras turi ilgą istoriją nuo 1960-ųjų ir iki šiol išlieka populiarus, bet „Electronica“ yra palyginus naujas žanras, kuris pradėjo populiarėti maždaug 1990-aisiais (10 pav.).



10 pav. Keturių žanrų palyginimas naudojant Streamgraph[CL14]

Tekmės diagrama taip pat buvo naudojama taikomojoje programoje ThemeStream. Žiūrint į 11 pav. visų pirma matyti bangavimas, kuris reiškia Twitter žinučių padažnėjimą konkrečia tema. Galime daryti išvadą, kad tomis dienomis įvyko kažkas svarbaus, kas susiję su paieškos raktažodžiu. Antra, galime palyginti kuris šaltinis daugiausia rašė susijusių žinučių. Matome, kad pagrindinėmis dienomis rašė daugiausiai lobistai ir paprasti vartotojai. Trečia, galime pasirinkti konkretų laiko tarpą (apatinė pilna vizualizacija) norint įsigilinti tarkim į vieno mėnesio ar net savaitės žinutes. Šio sprendimo trukumai tokie, kad negalime pasirinkti vienos kategorijos (iš keturių) vizualizacijos ir peržiūrėti jos atskirai. Taip pat, grafikas yra statinis, nes duomenys nesikeičia.

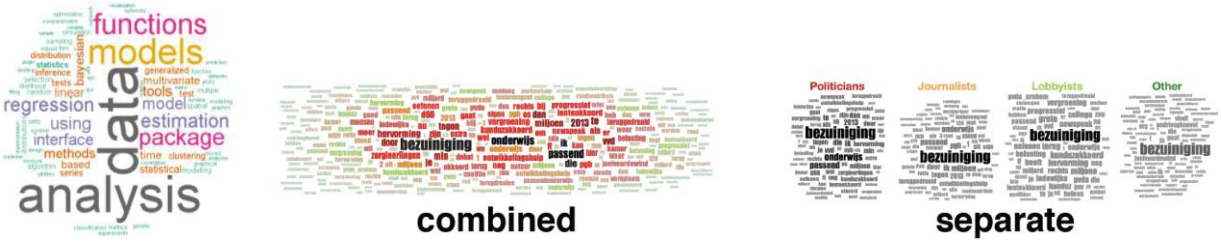


11 pav. ThemeStream Twitter žinučių vizualiazavimas tėkmės diagrama [ROR13]

Žodžių debesis (angl. *Wordcloud*) tai žodžių kratinys stačiakampiame plote. Žodžio dydis parodo dažnumą, kuris matuojamas to žodžio pasikartojimų skaičiumi tekste ar tarkim Twitter žinutėse. Ši vizualizacija leidžia greitai vizualiai analizuoti raktažodžius, kurie yra naudojami ir padeda greitai padaryti išvadą, kurie žodžiai ar kokia tema yra labiausiai paplitusi tekste arba mūsų atveju tarp Twitter vartotojų [Wal12]. 12 pav. rodomas žodžių debesis iš Twitter žinučių, kurias parašė statistikos studentai. Matome, kad yra naudojamos spalvos skirtingiems pasikartojimų dydžiams atskirti.

Sekančios 12 pav. vizualizacijos b) ir c) yra paimtos iš programinės įrangos ThemeStreams. Centre (b) rodomas bendras žodžių debesis (politinė tematika), kurioje spalva nurodo šaltinio tipą, o juoda spalva pažymėti bendri raktažodžiai. Dešiniajame kampe (c) yra

išskirstytas žodžių debesis pagal žinučių šaltinį (politikų – raudonas, žurnalistų – oranžinis, lobistų – šviesiai žalias ir kitų Twitter vartotojų – žalias).



(a) (b) (c)

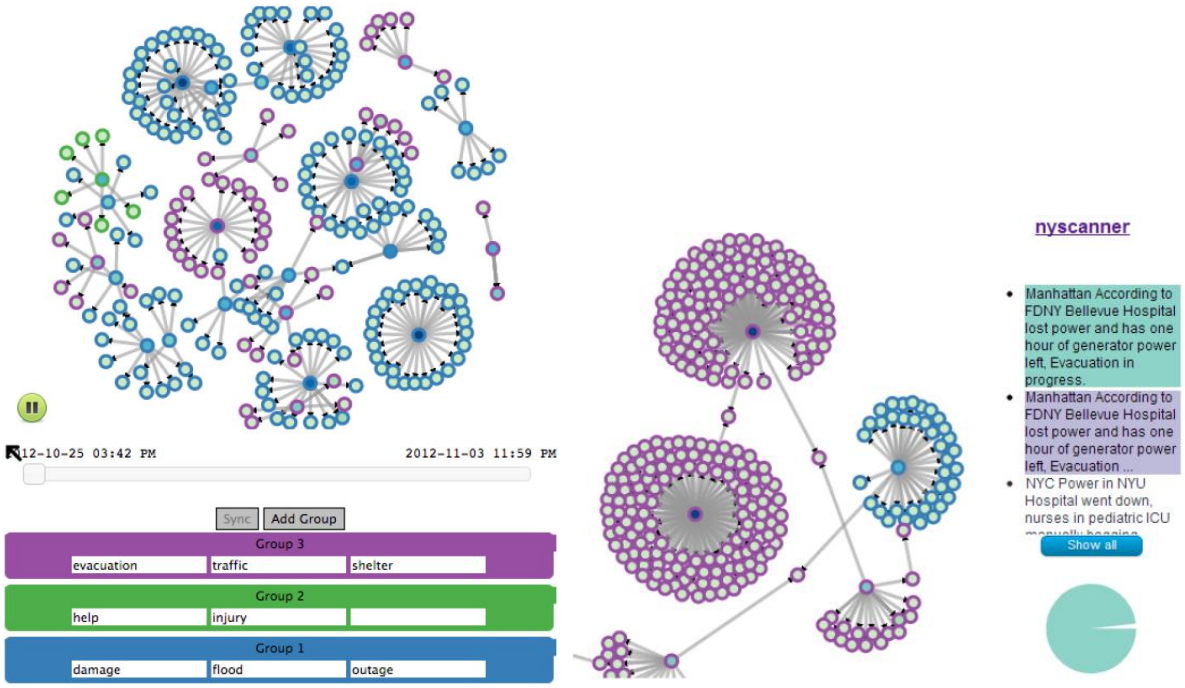
12 pav. a) Žodžių debesis iš žinučių apie statistiką [Wal12], b) bendras, c) išskirstytas žodžių debesis [ROR13]

ThemeStream vartotojai gali perjunginėti vieną ar kitą žodžio debesies tipą [ROR13]. Lengva pastebėti, kad debesys (b) ir (c) turi vieną bendrą pasikartojantį žodį „bezuiniging“, kuris išvertus iš olandų reiškia „santaupos“. Taip yra dėl to, kad šis žodis buvo paieškos raktažodis.

Tinklo diagrama (angl. *Network diagram*) kaip ir ***grafas*** (angl. *Graph*) yra sudarytas iš viršūnių (mazgų) ir briaunų. Plokštumoje viršūnės vaizduojamos taškais, o briaunos – atkarpomis (linijomis), jungiančiomis tuos taškus. Grafai naudojami sudėtingų sistemų modeliavimui (pvz. kompiuterių ir transporto tinklai, molekulės ir t.t.) ir sąryšių vizualizavimui (pvz. socialinių tinklų, duomenų bazių, esybių-ryšių diagramas ir t.t.) [Pla11].

Traukos taškais paremtas grafas (angl. *Force-directed grath*), naudoja algoritmą, kuris nustato kelias viršūnės (kurios turi didžiausią reikšmę, svorį ar dydį) kaip traukos centrus, o aplink šias viršūnes išdėsto susijusias viršūnes taip, kad tarp jų būtų vienodas atstumas ir mažiausias susikirtimų skaičius. Toks grafas turi geras estetines savybes (vienodi briaunų ilgiai, vienodas briaunų paskirstymas, simetrija) [Maz09].

13 pav. matome TweetXplorer [MKL+13] taikomosios programos tinklo grafą, kuriame pavaizduoti originalių ir persiustų (angl. *retweeted*) Twitter žinučių sąryšiai. Kiekviena viršūnė reprezentuoja vartotoją, o briaunos nurodo persiuntimo sąryšį. Jei vartotoją supa daug kitų viršūnių, tuomet jis yra originalios žinutės autorius (traukos taškas), o visi aplink jį pavaizduoti vartotojai persiuntė jo žinutę. Trys skirtingos paieškos temos pavaizduotos skirtingomis spalvomis (1 – mėlyna, 2 – žalia, 3 – violetinė). Pirmą temą (angl. *Group 1*) sudaro žodžiai: žala, potvynis, (el. energijos) nutraukimas. Antrą: pagalba, sužalojimas. Trečią: evakuacija, eismas, pastogė. Viršūnės spalvinamos dviem spalvom, viena viršūnės viduje, kita – išorėje. Vidinė spalva reiškia persiuntimo dydį (šviesus – mažas, tamsus – didelis). Išorinė spalva reiškia kuriai paieškos grupei žinutė priklauso. Matome, kad daugiausia žinučių parašyta pirma tema, o mažiausiai – antra. Dar vienas būdas svarbumui pabrėžti, būtų naudoti ne vidinę spalvą, o skritulio diametrą [KML13].



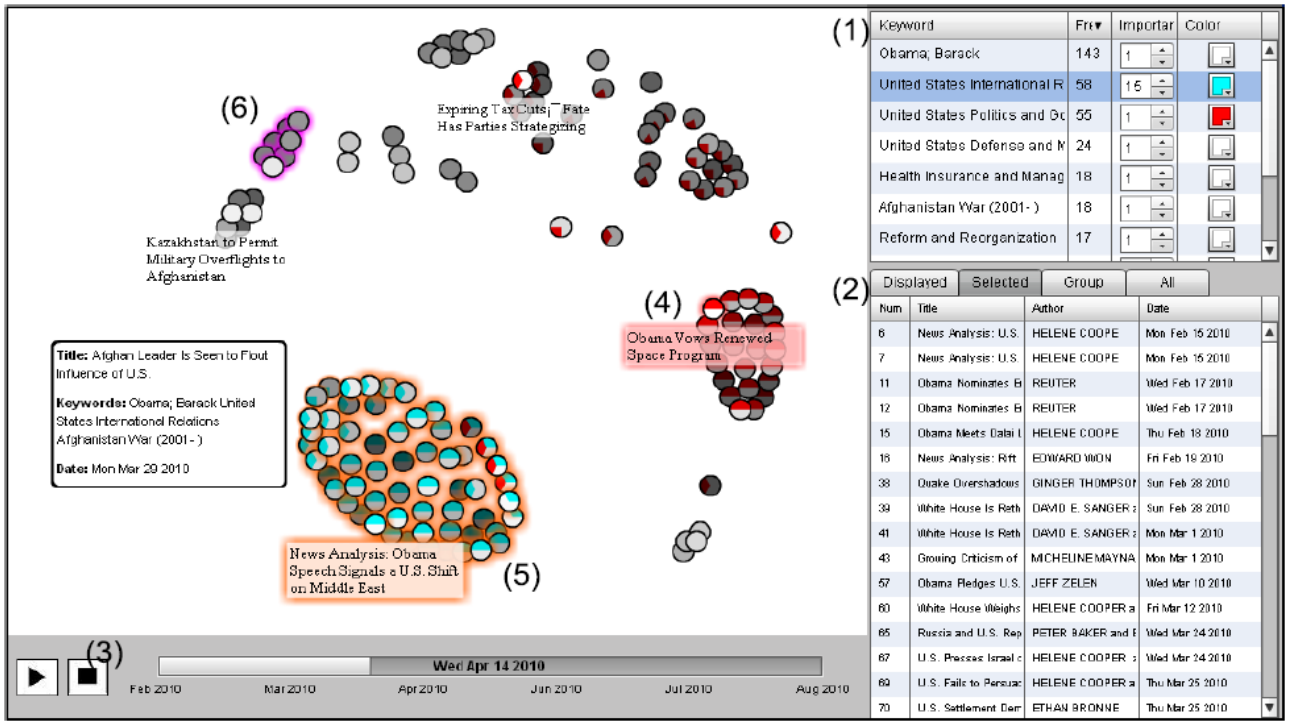
(a) (b)

13 pav. TweetExplorer. Twitter žinučių tinklo diagramos vizualizacija [MKL+13]

TweetXplorer traukos taškais paremtas grafas įgyvendintas naudojant D3 JavaScript karkasą[6]. Norint paspartinti skaičiavimus, grafe buvo vaizduojami tik stipriausią ryšį turinčios viršūnės, tai reiškia ne daugiau kaip 3 persiuntimai nuo originalios žinutės. Žinutės turinį galime pamatyti paspaudę ant atskirų viršūnių [MKL+13].

Traukos taškais paremti grafai buvo panaudojami ir moksliniame darbe [ACZ+11], kad pavaizduoti nuolat didėjantį teksto duomenų srautą. Taikomoji programa STREAMIT tai interaktyvi vizualizacijos sistema (14 pav.), kuri leidžia vartotojams ieškoti srauto tekstinius dokumentus be išankstinių žinių apie tuos duomenis. Sistema naudoja integruotą grafikos kortą kas leidžia akimirksniu animuoti vizualizaciją esant net labai dideliam duomenų kiekiui. Moksliniam darbe [ACZ+11] detaliai paaiškinta kaip veikia taškais paremtas grafas, kokios formulės buvo naudojamos (įskaitant Niutono dėsnio formules) ir koks algoritmas buvo naudojamas dinamiškai apskaičiuojant raktažodžių svarbą/reikšmę. Pagrindinio algoritmo sudėtingumas yra $O(N^2)$, kur N yra taškų (dokumentų skaičius).

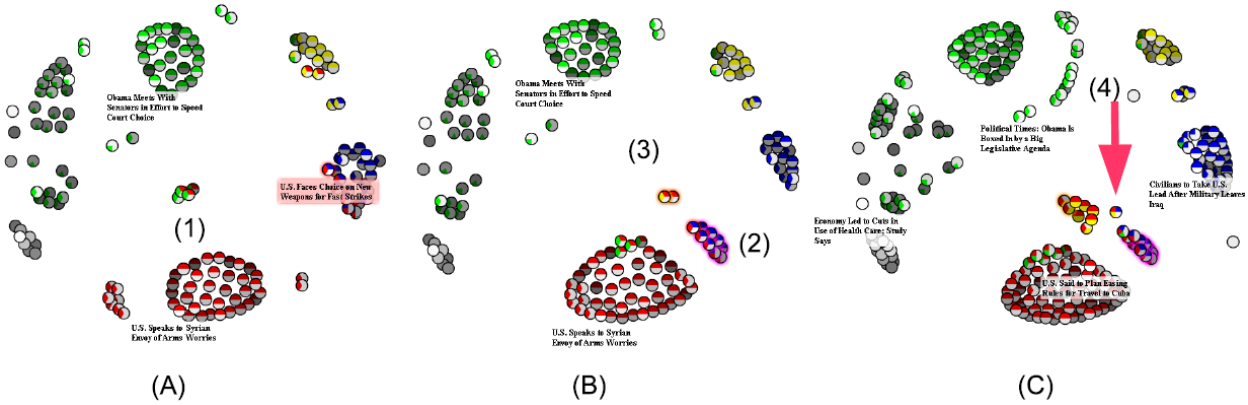
[6] http://d3js.org/



14 pav. STREAMIT vartotojo sąsaja [ACZ+11]

14 pav. matome pagrindinį STREAMIT langą. Lango kairioji dalis yra tekstinio srauto vizualizacija, o dešinė dalis skirta valdymui. 14 pav. dešinėje dalyje 1) yra raktažodžių valdymo lentelė, kurioje raktažodžiai surūšiuoti pagal dažnumą, dalyje 2) yra dokumentų lentelė, kurioje surašyti dokumentų pavadinimai, autoriai ir datos, dalyje 3) animacijos kontroleris, kuris leidžia peržiūrėti kaip vystėsi vizualizacija per tam tikrą laiką, o virš 3) visa centrinė dalis yra pagrindinis grafų vizualizacijos langas.

Vartotojas gali pakeisti raktažodžių svarbos koeficientą, pakeisti su jais susijusių taškų (dokumentų) spalvą. Raktažodžių spalva taške vaizduojama skrituline diagrama. Spalva naudojama ir grupėm pažymėti, pavyzdžiui vartotojas pasirinko raudoną (4), oranžinę (5) ir rožinę (6) spalvas, kad pažymėti jam rupimas grupes. Etiketėmis pažymėti tik naujausi dokumentai iš kiekvienos grupės [ACZ+11]. Jas galima ir išjungti kai laukas persipildo taškais. 14 pav. pavaizduoti raktažodžiai susiję su JAV politika, tokie kaip: Barack Obama, JAV politika, gynyba, karas Afganistane ir pan. Moksliniame darbe aprašoma kokia techninė įranga buvo naudojama norint optimizuoti šitą vizualizacijos sistemą. Bandymai parodė, kad pasiekiama geresnių rezultatų naudojant grafikos kortą (angl. *GPU*), o ne procesorių (angl. *CPU*). Naudojant procesorių praeina daugiau nei sekundė, kol vizualizacija atnaujinama. Su GPU tas įvyksta per mažiau nei sekunde, penkiolika kart greičiau. Sistema gali išanalizuoti apie 15,000 dokumentų su 2,000 skirtingų raktažodžių [ACZ+11].



15 pav. STREAMIT vizualizacijos pokyčiai laikui bėgant ir keičiantis turiniui [ACZ+11]

Moksliniame darbe parodyta, kaip keičiasi grafai priklausomai nuo žinučių turinio. 15 pav. vizualizacija (A) rodo pradinę stadiją esant 136 naujienų straipsniams Rugpjūčio 13d. Vizualizacija (B) – kuomet vartotojas padidino „tarptautinių santykių“ raktažodžio svarbos koeficientą, o paskutinėje (C) – esant 230 naujienų straipsniams Rugsėjo 18d. Raktažodžių spalvos: "Politika" – žalia, "Tarptautiniai ryšiai" – raudona, "terorizmas" – geltonas, ir "gynyba ir kariuomenė" – mėlyna.

1.3. Literatūros apibendrinimas

Šiame skyriuje apibendrinami keli sudėtingesni vizualizavimo būdai atsižvelgiant į šio darbo pagrindinį tikslą: „sukurti srauto duomenų vizualizavimo algoritmą remiantis socialinio tinklo Twitter duomenimis mobiliam įrenginiui“. Taip pat pateikiama 1 lentelė su šaltinių sąrašu ir įvertinimais pagal pasirinktus parametrus.

Rutulinė diagrama netinka mūsų uždaviniui, kadangi esant nuolatiniams atnaujinimams pastoviai keisis rutulių plotas, o tai apsunkina dydžio suvokimą. Rutulio dydis galėtų reprezentuoti Twitter žinučių dažnumą, o *x* ašis žymėtų laiką, bet tuomet kylą klausimas ką vaizduoti *y* ašyje.

Išsibarstymo diagramą galima būtų panaudoti mūsų darbe ir sukurti kažką panašaus į TweetViz [HR13] sprendimą, tik mūsų vizualiazacija turėtų būti nuolat atnaujinama ir taškai turėtų būti matomi mobiliajame įrenginyje esant mažesnėms rezoliucijoms.

Spalvinė diagrama gerai tinka duomenų srautų vizualizacijai, kai turime daugybe įvairių kiekybinių duomenų šaltinių, pavyzdžiui akcijoms tendencijoms vaizduoti. Tačiau jeigu diagramą norėtume panaudoti su raktažodžiais susijusių Twitter žinučių vizualizacijai, tuomet prireiktų ne mažai raktažodžių (mažiausiai 5), kad ta vizualizacija duotų naudą.

Šalies žemėlapis labai gerai tiktų parodyti kuri šalis (ar teritorija) yra labiausiai minima Twitter žinutėse. Galima būtų sukurti vizualizaciją Eurovizijos tendencijoms vaizduoti dar neprasidėjus

balsavimams. Taip pat galima sukurti vizualizaciją su Lietuvos žemėlapiu ir apylinkėm, kad pavaizduoti kur kokie raktažodžiai minimi dažniausiai. Nors **gyvenvietės žemėlapis** dažniausiai naudojamas mobiliuose įrenginiuose, šiame darbe tikriausiai jo nenaudosime. Taip nusprendėme dėl to, kad Twitter žinutės ne visada turi geografines padėties koordinates, todėl vaizdavimui turėtume apribotą duomenų kiekį.

Tėkmės diagrama buvo labai sėkmingai panaudota ThemeStream [ROR13] moksliniam darbe. Mūsų atvejų reikėtų patobulinti vizualizacija taip, kad ji galėtų atvaizduoti srauto duomenis ir tiktų naudojimui mobiliuosiuose įrenginiuose. Šios vizualizacijos kūrimas nėra lengvas, tačiau darbui atlikti pasitelktume formules ir algoritmus iš [BW08] mokslinio darbo.

Žodžių debesis puikiai tiktų dažniausiai pasitaikantiem raktažodžiams pavaizduoti. Manome, kad šią vizualizaciją būtų lengva įgyvendinti, o ir ji nepasižymi interaktyvumu, todėl jos nenaudosime.

Traukos taškais paremtas grafas yra puikus vizualizacijos būdas tinkantis mūsų tikslui išspręsti. Mes galime paimti jau esančių vizualizacijų idėjas ir jas pakeisti / patobulinti taip, kad sprendimas tiktų mūsų užsibrėžtam tikslui išspręsti. TweetXplorer [MKL+13] sistema yra skirta Twitter žinučių analizei, tačiau tik ant asmeninių kompiuterių ir su jau išsaugotais duomenimis. STREAMIT sistema [ACZ+11] veikia su srauto duomenimis, tačiau jinai taip pat nepritaikyta mobiliesiems įrenginiams ir neanalizuoja Twitter žinučių. Šis grafas taip pat yra sudėtingas, tačiau moksliniame darbe [ACZ+11] turime keletą formulių ir algoritmą, kurie padėtų įgyvendinimo metu.

Naujienų portale The Guardian pavyko surasti sprendimą, kuris panašus į traukos taškais paremtą grafą. Sprendime vaizduojama riaušių Londone Twitter žinučių analizė (1 priedo 45 pav.). Tačiau šiame sprendime viršūnės vaizduojamos rutuliais ir jos nėra sujungtos tarpusavyje. Rutulių dydis priklauso nuo to kiek vartotojų tą žinutę persiuntė (angl. *number of retweet*s). Šis sprendimas irgi gali būti panaudotas ateities darbe, tačiau reikėtų surasti (pasitelkus atvirkščiąją inžineriją) šios vizualizacijos algoritmą.

Darbo metu buvo apžvelgta 14 mokslinių straipsnių, 4 knygos, 2 tinklaraščių straipsniai ir viena video prezentacija. Paieškai buvo naudojamos mokslinių publikacijų duomenų bazės EBSCO, IEEE, ACM Digital Library, Science Direct ir kitos pasiekiamos per Google Scholar. Paieškai buvo naudojami raktažodžiai (angliškai): ***visualization, mobile device, stream data, Twitter, sematic analysis, Javascript framework, Highcharts*** ir kiti.

1 lentelė. Literatūros sąrašas ir kriterijai

Nr.	Šaltinis	Šaltinio tipas	Vizualizacija	Srauto duomenys	Mobilieji įrenginiai	Twitter	Javascript karkasai ar kita technologija	Skaitinė analizė	Semantinė analizė	Įvertinimas pagal kriterijus
1	[Maz09]	Knyga	+					+	+	3
2	[Pla11]	Knyga	+					+		2
3	[KML13]	Knyga	+	+		+	+	+	+	6
4	[ACZ+11]	Mokslinis straipsnis	+	+					+	3
5	[LK12]	Mokslinis straipsnis	+		+		+	+	+	5
6	[MF12]	Mokslinis straipsnis	+		+		+			3
7	[MKL+13]	Mokslinis straipsnis	+	+		+		+	+	5
8	[ROR13]	Mokslinis straipsnis	+	+		+			+	4
9	[OK13]	Mokslinis straipsnis	+		+		+	+		4
10	[SB13]	Mokslinis straipsnis	+	+		+	+		+	5
11	[BW08]	Mokslinis straipsnis	+					+		2
12	[Wal12]	Mokslinis straipsnis	+			+			+	3
13	[CL14]	Straipsnis tinklaraštyje	+					+		2
14	[Fon12]	Mokslinis straipsnis	+			+			+	3
15	[RLC+13]	Mokslinis straipsnis	+			+			+	3
16	[HR13]	Straipsnis tinklaraštyje	+			+	+		+	4
17	[DKŽ08]	Knyga	+					+	+	3
18	[Syn13]	Vaizdo įrašas	+				+	+		3
19	[Bjo08]	Mokslinis straipsnis	+		+	+	+	+		5
20	[MLC+13]	Mokslinis straipsnis	+			+			+	3
21	[SM08]	Mokslinis straipsnis	+					+	+	3

Literatūros analizės metu buvo pasirinkti moksliniai straipsniai, kurie visų pirma buvo susiję su duomenų vizualizavimu. Antra, buvo susiję vienu iš sekančių temų: srauto duomenys, mobilieji įrenginiai, Twitter socialinis tinklas, semantinė analizė, Javascript karkasai ir pan. Darbui atlikti buvo naudojami šie metodai: mokslinės literatūros analizė ir apibendrinimas.

1.4. Twitter srauto duomenų tyrimas

Naudojant Twitter oficialaus puslapio paiešką rezultatai pateikiami kaip sąrašas žinučių, kuriuose yra paieškos žodis, vartotojo vardas ar žymė. Naujausios žinutės rodomos viršuje, o senesnės apačioje. Slenkant žemyn puslapis asinchroniškai atsiunčia ir parodo vis daugiau senesnių žinučių. Žinučių sąraše vaizduojami tokia informacija: vartotojo nuotrauka, vardas ir paskyros pavadinimas, žinutės tekstas, prieš kiek maždaug laiko žinutė buvo parašyta (3min., 1val., Sau 17 ir pan.) ir žinutės apačioje prikabintas piešinėlis ar video jei toks egzistuoja. Nuorodos į kitus vartotojus, žymės ir išorinius puslapius yra nuspalvintos mėlynai. Norint palyginti žymių ar raktažodžių populiarumą toks būdas neefektyvus, nes vartotojui pačiam reikėtų skaičiuoti tuos raktažodžius. Yra du būdai gauti žinutes iš Twitter: „REST“ ir „Stream API“. Naudosime „Twitter Stream“ dėl nepertraukiamo duomenų srauto.

1.4.1. Twitter Stream paslauga

„Twitter Stream API“ – Twitter srauto aplikacijų programavimo sąsaja (API), kuri leidžia kūrėjams prisijungti prie Twitter socialinio tinklo su mažo uždelsimo ryšiu ir gauti žinučių srautą, kuri sudaro neapdorotos žinutės su metaduomenimis. Twitter siūlo tris skirtingus duomenų srauto priėjimus, kurie kiekvienas turi savo panaudojimo ypatybes:

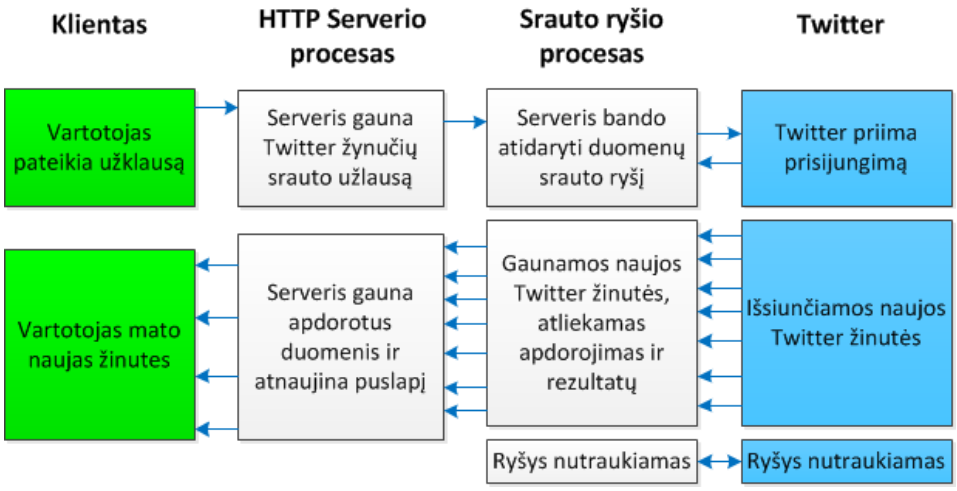
- „Public streams“ – Visas duomenų srautas keliaujantis į Twitter socialinį tinklą. Tinka sekti visus esamus vartotojus ir temas duomenų gavybai.
- „User streams“ – Tik vieno konkretaus vartotojo žinučių srautas ir visi susiję duomenys (ką vartotojas seka, kokią žinutę pažymėjo kaip mėgstamą ir pan.).
- „Site streams“ – Daugelių vartotojų žinučių srautas. Šis sprendimas skirtas serveriams, kurie veikia kaip tarpininkai ir prisijungia kitų vartotojų vardu.

Šiame darbe naudosime „public streams“ Twitter duomenų srautą.

Prisijungimas naudojant OAuth autentikaciją

Ryšys tarp Twitter serverio ir kliento veikia naudojant HTTP protokolą. Bet norint naudoti Twitter Stream API iš pradžių reikės prisijungti su OAuth prieigos raktais. Jie gaunami užregistravus programėlę „Twitter Application Management“ [7] puslapyje. Užregistravome programėlę pavadinimu „TreemapViz“ ir gavome sekančius raktus: *access_token, access_token_secret, consumer_key, consumer_secret*. Visi šie raktai bus naudojami vėliau norint prisijungti. Vienu metu galima turėti tik vieną prisijungimą per Twitter programėlę.

[7] https://apps.twitter.com/



16 pav. Twitter *Stream API* veikimo diagrama, adaptuota iš Twitter puslapio[8]

Ryšis bus nutraukiamas dėl šių priežasčių:

- per daug prisijungimų iš tos pačios paskyros (galimas tik vienas ryšys per paskyrą),
- per lėtas duomenų nuskaitymas (klientas nespėja priimti ir perskaityti visas žinutes, kurios ateina iš Twitter serverio ir tenka nepriimtas žinutes saugoti eilėje),
- jei Twitter serveris yra paleidžiamas iš naujo arba pakeista tinklo konfigūracija.

JSON struktūra

Twitter žinutė pateikiama kaip JSON duomenų objektas (žiūr. 3 priede). Kiekviena žinutė atskiriama nauja eilute. Oficialiame Twitter puslapyje skaitydamas žinutes vartotojas mato šiuos duomenis: vartotojo paveiksliukas, tekstas, žymės, nuorodos ir paveikslėliai. Tačiau didelė dalis duomenų nėra matoma ir šie metaduomenis yra pasiekiami jei turime JSON žinutės reprezentaciją. Toliau pateikiami šiame darbe svarbūs metaduomenis (2 lentelė), o visą metaduomenų sąrašą su paaiškinimais žiūrėti 3 priede.

2 lentelė. Twitter žinutės svarbiausi JSON metaduomenys

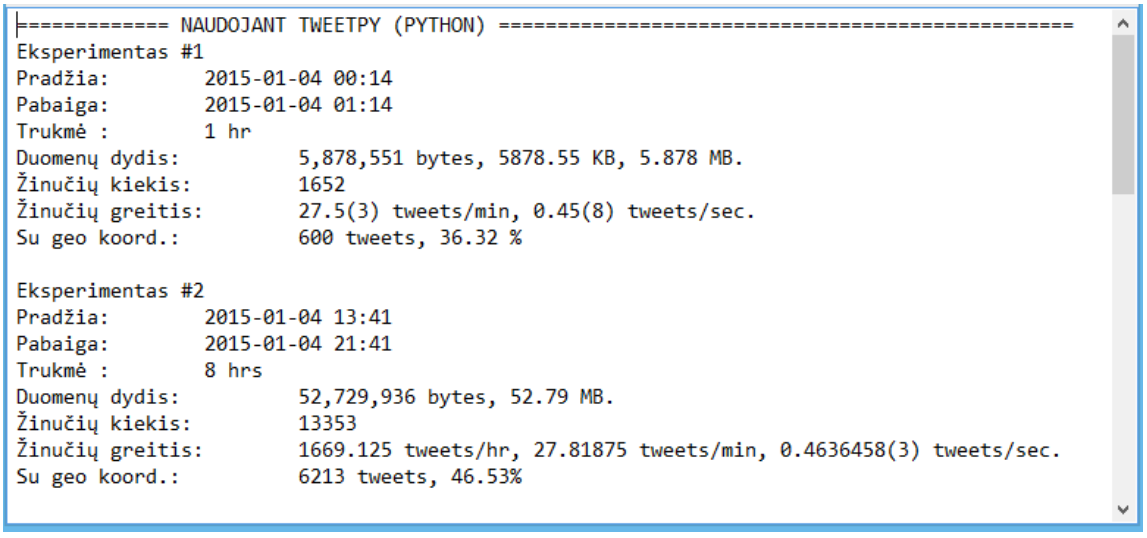
Parametras	**Aprašymas, vertimas**
created_at	Konkreti data ir laikas kada Twitter žinutė buvo išsiųsta, sekundžių tikslumu.
text	Žinutės tekstas (iki 140 simbolių)
user	Atskiras vartotojo duomenų blokas atskirtas riestiniai skliaustais. Toliau user.<elem> reikš vartotojo duomenų elementą

[8] https://dev.twitter.com/streaming/overview

Parametras	Aprašymas, vertimas
user.name	Vartotojo pilnas vardas, pvz.: „Eugenijus Sabaliauskas“
user.screen_name	Paskyros pavadinimas (gali būti vartotojo vardas, bet nebūtinai tikras), pvz.: „eugenijus_s“
coordinates	Tikslios vartotojo koordinatės iš kur buvo išsiųsta žinutė [±*nn.nnnnnn*, ±*nn.nnnnnn*] formatu [ilguma, platuma].
place	Atskiras vietovės duomenų blokas
place.country_code	Šalies kodas, pvz.: JAV yra „US“, Lietuva yra „LT“
entities	Atskiras papildomų duomenų blokas
entities.hashtags	Grotelinė žymė, kreipė (t.y. nukreipiamoji žymė), etiketė

1.4.2. Esamų Twitter API įrankių tyrimas

Twitter turi apibrėžtą REST API sąsają, kurią galima naudoti per naršyklę. Tačiau, kad naudoti Stream API reikia suprogramuotos sąsajos įrankio ar bibliotekos. Yra paviešintas tokių bibliotekų sąrašas[9] su įvairiu programavimo kalbų įgyvendinimais (Python, Java, C++ ir t.t.).



17 pav. Tweetpy įrankio tyrimų rezultatai

Tweetpy. Naudojant Python skriptą **tweetpy**[10] pirmo eksperimento metu buvo paimtas vienos valandos Twitter žinučių srautas ir buvo atsiųstos 1652 žinutės (17 pav.). Vidutinis duomenų srautas **27.5(3)** žinutės/min, **0.45(8)** žin./sek. Antro eksperimento metu greitis

[9] https://dev.twitter.com/overview/api/twitter-libraries

[10] https://github.com/tweepy/tweepy

nepadidėjo. Taigi maždaug kas dvi sekundes gaunama po naują Twitter žinutę. Tai nėra labai greitas srautas, todėl reikėtų palaukti kelias minutes, kad gauti kažkokią naudingą statistiką.

Hosebird Client. Naudojant Java „*Hosebird Client*“ biblioteką (dar vadinama HBC-Core) ir programėlę[11] ketvirto eksperimento metu buvo paimtas ~3 min (172 sek.) Twitter žinučių srautas ir buvo atsiųstos 8684 žinutės (32.2 MB). Vidutinis duomenų srautas **3029.30** žinutės/min, **50.49** žin./sek. Lyginant su Tweetpy – srautas daug greitesnis. Twitter žinutės su koordinatėmis [ilguma, platuma] sudaro 3.35% visų žinučių, vidutinis srautas **1.69** žin./sek. Duomenų srautas gali skirtis, bet jei srautas bus ~1-2 žin./sek. tuomet tokio srauto užtenka efektyviam koordinačių vizualizavimui (2 priedo 47 pav.). Penkto eksperimento metu duomenys buvo panašus į ketvirtojo.

```
============ NAUDOJANT HOSEBIRD-CLIENT (JAVA) ==========================================
Įrankis:        Java api https://github.com/twitter/hbc

Eksperimentas #4
Pradžia:        2015-01-11 00:05:54.389
Pabaiga:        2015-01-11 00:08:46.55
Trukmė :        ~3 min = 172 sec.
Duomenų dydis:          32.2 MB
Žinučių kiekis:         8684
Žinučių greitis:        2894.(6) tweets/min., 8684/172 = 50.49 tweets/sec.
Su geo koord.:          291 tweets, 291/172 = 1.69 tweets/sec.
                        Tai 3.35% esamo srauto (žinutės su metaduomenimis "coordinates":["),

Eksperimentas #5
Pradžia:        2015-01-11 02:00:15.196
Pabaiga:        2015-01-11 02:02:55.174
Trukmė :        ~3 min = 160 sec.
Duomenų dydis:          31.5 MB
Žinučių kiekis:         8471
Žinučių greitis:        2823.(6) tweets/min., 8471/165 = 52.94 tweets/sec.
Su geo koord.:          230 tweets, 230/160 = 1.4375 tweets/sec.
                        Tai 2.72% esamo srauto.
```

18 pav. Hosebird Client įrankio tyrimo rezultatai

Logstash ir **Elasticsearh** yra atviro kodo įrankiai, kurio pagalba galime dar paprasčiau naudoti Tweeter Stream paslaugą. Lostash bendrauja su Twitter serveriu, o saugoti žinutes galima į failą arba į duomenų bazę, pavyzdžiui Elasticsearch. Viename iš pirmųjų testavimo metu buvo pasirinkta rinkti žinutės iš Europos teritorijos į Elasticsearch duomenų bazės lentelę „tweets-eu2“. Žinučių rinkimas trūko 2,5 valandas (2016 Vasario 3-ią, 17:33 iki 20:00) ir buvo surinkta 144323 žinučių. Tai yra 57729 žinutės per valandą arba **16** žinutės per sekundę. Kadangi Elasticsearch indeksuoja tentelės, susidaro pagalbiniai failai, taigi iš viso buvo sukurta 640 MB duomenų, o tas lygu maždaug 4.43 KB per žinutę.

Antrą kartą tiriant Logstash buvo sukurta „tweets-bafta“ DB lentelė ir joje buvo saugomos žinutės iš Twitter serverio susijusios su BAFTA apdovanojimais. Šioje lentelėje buvo išsaugota

[11] https://github.com/twitter/hbc

134690 žinučių per 3 val 15min (Vasario 14 d., 22:45 – Vasario 15d. 2:00). Tai yra 41443 žinutės per valandą, **~11.51** žinutės per sekundę. Sukurta 498 MB duomenų, o tas lygu 3,7 KB per žinutę.

Trečio tyrimo metu buvo sukurta „tweets-oscars2016“ DB lentelė su žinutėmis susijusiomis su Oskarų apdovanojimais. Šioje lentelėje buvo išsaugota 1489334 žinutės per 8.5 val. (Vasario 29-ą, 00:00 – 8:30). Tai yra 175216 žinutės per valandą, **48.67** žinutės per sekundę. Sukurta 5.54 GB duomenų, o tas lygu 3,72 KB per žinutę.

1.4.3. Tyrimo rezultatai ir pastebėjimai

Didžiausia Twitter žinutė, kurią teko surasti buvo 11,081 simbolių (11KB) dydžio, o teksto elementas sudaro tik 1.26%, o tai reiškia, kad daug duomenų reikės atfiltruoti ir ištrinti, kad neužsipildytų laikinoji atmintis. Vidutinis žinutės dydis su metaduomenimis yra apie 3500 simbolių. Mums reikalingų elementų dydis gali būti įvairaus dydžio, bet vidutiniškai jis sudaro 760 simbolių, kas yra 21.7% viso duomenų srauto.

Jei naudojant Hosebird Client žinučių srautas yra **50.49** žin./sek., tai vidutiniškas duomenų srauto greitis yra 178.88 KB/sek. (3 lentelė), o atfiltruotų duomenų srautas: 38 KB/sek. Disko naudojimo greitis būtų 3,82 KB/žin. Jei naudoti Logstash, tuomet žinučių srautas bus apie **48,67** žin./sek., tačiau diske vietos prireiks mažiau (**3.7** KB/sek.), nes tikėtina, kad Elasticsearch suglaudžia duomenis į mažesnį formatą ir saugo papildomą indeksavimo informaciją. Tweetpy įrankis atsiunčia mažiau žinučių, nei Hosebird Client ir Logstash, todėl remiantis tyrimų rezultatais nutarta Tweetpy įrankiu nesinaudoti.

3 lentelė Twitter srauto duomenų įrankių tyrimų rezultatai

Įrankio pavadinimas	Žinutės/sek.	KB/žin.	KB/sek.	Žinučių kiekis
Tweetpy	0,46	3,95	1,83	13 353
Hosebird Client	50,49	3,82	178,88	8 684
Logstash + Elasticsearch	48,67	3,72	181,05	1 489 334

Jeigu sistema turės tarpinį serverį, tuomet geriausia būtų naudoti Logstash žinučių srautui atsiųsti ir Elasticsearch žinutėms saugoti, nes šituos įrankius patogu konfigūruoti kartu. Šiems įrankiams reikalinga Windows, Linux ar Mac operacinė sistema. Jeigu sistema bus įgyvendinta Android mobiliąja programėle, tuomet verta naudoti Java kalbos Hosebird Client biblioteką.

Nagrinėjant eksperimento rezultatus ir išvesties tekstus buvo atkreiptas dėmesys į pagrindinius žinučių metaduomenis: tekstas, žymės, vartotojo koordinatės. Toliau pateikiami pastebėjimai dėl šių metaduomenų reikšmės ir naudos vizualizavimo sistemos kūrime.

Tekstas yra pagrindinė žinutės dalis, joje gali būti ir žymės (prasideda #) ir nuorodos į kitus vartotojus (prasideda @). Norėdami filtruoti duomenų srautą, dažniausiai filtruosime šitą teksto elementą.

Žymės (angl. *hashtag*) skirtos pažymėti raktažodžius ar temas Twitter žinutėje. Jos priskiria žinutę konkrečiai kategorijai. Žymės naudingos, nes pagal jas galima lengviau surasti žinutes su ta pačia žyme. Tai pasiekiama paspaudus ant žymės esant Twitter internetiniame puslapyje. Kas kažkiek laiko atsiranda populiarios žymės, kurias naudoja daug vartotojų. Tokios žymės rodo tendencijas (angl. *trending*). Jei vartotojas žino populiariausias temas, tuomet gali jas atfiltruoti ir gauti tam tikros temos žinutes. Žemiau pateikiamas žymės pavyzdys:

```
{ ... "hashtags":[ {"text":"ComputerScience","indices":[51,67]}], ... }.
```

Vartotojo koordinatės. Geografinės padėties koordinatės yra pateikiamos GeoJSON formatu. GeoJSON – tai atviro standarto formatas skirtas apibrėžti paprastus objektus geografinėje koordinačių sistemoje, tokius kaip taškas (vieta, adresas), atkarpa (gatvė, kelias, autostrada), daugiakampis (apskritis, rajonas, šalis ir kita teritorija) ir šių objektų rinkinys[12]. Taško koordinatės – tai pora [ilguma, platuma], kur ilguma (angl. *Longitude*) yra nuo -180 iki 180 laipsnių, o platuma – nuo -90 iki 90 (angl. *Latitude*). Šie metaduomenys bus reikalinga jei reikėtų kurti žinučių žemėlapį. Pavyzdžiui, VU MIF koordinatės objektas atrodo taip:

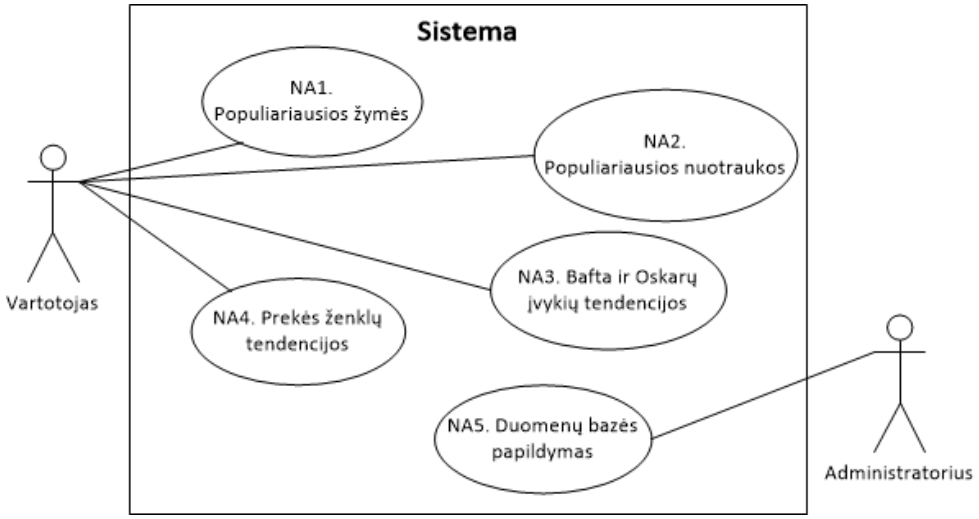
```
{ "type": "Point", "coordinates": [25.273457765579224, 54.67507479641878] }.
```

[12] http://geojson.org/geojson-spec.html

2. Principiniai sprendimai

2.1. Naudojimo atvejai

Šiame poskyryje aprašomi naudojimo atvejai, kurie skirti Twitter žinučių vizualizavimo sistemai. Šie atvejai atspindi sistemos panaudojimo galimybes bei potencialią naudą vartotojui.



19 pav. Naudojimų atvejų diagrama

2.1.1. Bendroji naudojimo atvejų dalis

Apačioje nurodyti bendri atributai (4 lentelė), kurie bus naudojami kiekviename naudojimo atvejyje nebent nurodyta atskirai.

4 lentelė. Bendri naudojimo atvejų atributai

NA atributas	Aprašas
Apimtis	Projektuojama programų sistema (angl. *System Under Design*) toliau – sistema
Pagrindinis veikėjas	Sistemos vartotojas (toliau – vartotojas)
Kiti veikėjai	Sistema (Vizualizacijos sistema, žinučių kaupimo ir filtravimo saityno paslauga), Sistemos administratorius, Twitter serveris
Išankstinės sąlygos	Iš anksto sukauptos Twitter žinutės pagal konkretų filtravimą, pvz.: kelių valandų trukmės Twitter žinutės su Oskarų apdovanojimų raktažodžiais. Šias žinučių grupes vadinsime duomenų bazės lentelėmis (5 lentelė). Kiekvienas naudojimo atvejo scenarijus prasideda tuo, kad vartotojas įsijungia sistemą mobiliame įrenginyje ir sistema parodo langą su mygtukais.

5 lentelė. Duomenų bazės lentelės su išsaugotomis Twitter duomenų srauto žinutėmis

Lentelės pav.	Dokumentų skaičius	Tema ar paskirtis	Laikotarpis
tweets-bafta	134 690	Bafta apdovanojimai	3 valandų trukmė, 2016 m. Vasario 14-15 d.
tweets-eu2	232 971	Žinutės Europos teritorijoje	6 val., 2016 m. Vasario 3 ir 5 dienomis
tweets-oscars2016	1 489 336	Oskarų apdovanojimai	8 val., 2016-02-29.

2.1.2. NA1. Populiariausios žymės

Tikslai:

- Atrasti šiuo metu labiausiai populiariausias Twitter žymes.
- Pastebėti kokiu santykiu skiriasi žymių populiarumas.
- Perskaityti kelias Twitter žinutes, kurios turi bent vieną iš nurodytų žymių.

NA1. Scenarijus nr. 1. Surasti Twitter populiariausias žymes.

1) Vartotojas pasirenka „**Top 10 populiariausių žymių**“ mygtuką.
2) Sistema parodo langą su įvesties parametrais ir pradinę diagramą naudojant duomenų bazės lentelę „tweets-bafta“.
3) Vartotojas įveda žodžius po vieną.
4) Sistema sukuria žodžių sąrašą su mygtukais.
5) Vartotojas mato žodžių sąrašą, kurį jis įvedė, spaudžia „Paleist“.
6) Sistema daro paiešką pagal žodžių sąrašą sukauptame Twitter žinučių archyve.
7) Sistema parodo vizualizaciją (horizontalę stulpelinę diagramą), kur kiekvienas stulpelis atitinka žymės žodį ir stulpeliu ilgis priklauso nuo pasikartojimo dažnumo (populiarumo). Prie kiekvienos žymės parašytas skaičius, kiek kartų ji buvo rasta.
8) Vartotojas apžiūri diagramą ir paspaudžia ant populiariausios žymės.
9) Sistema parodo ribotą sąrašą Twitter žinučių, kuriuose buvo panaudota žymė.
10) Vartotojas perskaito Twitter žinutes.
11) Vartotojas grįžta atgal.
12) Sistema parodo vaizdą, kaip ir 9 žingsnyje.
13) Vartotojas spaudžia „Namai“ ir grįžta į pagrindinį puslapį su mygtukais.

Alternatyvi eiga:

7a1) Vartotojas mato žodžių sąrašą, kurį jis įvedė ir nusprendžia išimti vieną iš žodžių iš sąrašo paspaudamas trinimo mygtuką.

7a2) Sistema ištrina konkretų žodį iš sąrašo.

7a3) Vartotojas spaudžia „paleisti“.

2.1.3. NA2. Populiariausios nuotraukos

Tikslai:

- Peržiūrėti Twitter žinutėse esančias nuotraukas naudojant sistemos sugeneruotą medžio pavidalo žemėlapį (angl. *Treemap*).
- Pagal vizualizaciją nuspręsti kuri nuotrauka populiariausia (labiausiai minima) konkrečiu laikotarpiu.
- Sužinoti, kas pasidalino ta nuotrauka (Twitter vartotojo vardas).
- Perskaityti keletą žinučių, kurios yra įtraukusios tą nuotrauką.

Išankstinė sąlyga: sukaupta kelių valandų trukmės Twitter žinutės, kuriuose yra nuorodos į paveikslėlius ar nuotraukas kituose sistemose (pvz.: Instagram, Twitter picture[13], Flickr ir pan.).

NA2. Scenarijus nr. 1. Surasti populiariausią Twitter nuotrauką.

1) Vartotojas pasirenka „**Populiariausios nuotraukos**“ mygtuką.
2) Sistema parodo langą nuotraukų žemėlapiu, kuriame 10 nuotraukų ir virš jų yra įvesties parametrų langeliai ir mygtukai.
3) Vartotojas pasirenka nuotraukų kiekį ir spaudžia „Paleisti“.
4) Sistema perpiešia Treemap vizualizaciją pagal naujus parametrus. Vizualizacijoje matyti nuotraukos, kurios patalpintos skirtingų dydžių stačiakampiuose.
5) Vartotojas pasirenka filtravimą pagal raktinius žodžius arba žymes, tuomet suveda kelis raktažodžius (pvz.: „saulėta diena“ arba „Vilnius“) ir spaudžia „Paleisti“.
6) Sistema iš naujo sugeneruoja Treemap vizualizaciją pagal parametrus. Šį kartą bus rodomos tik nuotraukos, kurių Twitter žinutės turi vartotojo pasirinktus raktažodžius.
7) Vartotojas pasirenka didžiausią nuotrauką (populiariausią) ir ant jos paspaudžia.
8) Sistema pateikia papildoma informaciją apie vartotoją, kuris įkėlė nuotrauką, iš kur ta nuotrauka (twitter pic, instagram, flickr ar pan.) ir kitas detales.
9) Vartotojas paspaudžia ant mygtuko „Susijusios žinutės“.
10) Sistema parodo ribotą sąrašą Twitter žinučių, kuriuose nuotrauka buvo cituojama (Twitter naudojamas specialus terminas: angl. *retweet*).

[13] https://pbs.twimg.com/media/ABC.jpg

2.1.4. NA3. Bafta ir Oskarų apdovanojimų tendencijos

Šiame naudojimo atvejyje naudojamos dvi duomenų bazės lentelės: „tweets-bafta“ ir „tweets-oscars2016“, kurios surinktos pagal susijusius raktažodžius ir žymes (pvz.: #BAFTA, #oscars, #oscars2016, theAcademy ir pan.). Twitter žinutės buvo renkamos į duomenų bazę būtent tuo metu kai vyko BAFTA ir Oskarų apdovanojimų ceremonijos. Sistema naudoja duomenų bazę jau apdovanojimams pasibaigus, todėl yra žinomi ir nugalėtojai. Taigi chronologiškoje diagramoje galima pastebėti atsirandančias tendencijas, kai apdovanojamas aktorius(-ė) ar filmas. Bafta apdovanojimai vyko anksčiau nei Oskarai, todėl galima palyginti kokias Twitter žinučių tendencijas sukūrė šie du renginiai.

Tikslai:

- Atrasti populiariausius Oskarui nominuotus aktorius, aktores ir filmus.
- Pasirenkant konkrečią nominacijos kategoriją, surasti ryšį tarp nominacijos laimėtojo ir kas iš nominuotų žmonių ar filmų buvo populiariausias Twitter žinutėse.
- Palyginti konkrečios nominacijos tendencijas tarp Bafta ir Osakrų DB lentelės.
- Perskaityti kelias Twitter žinutes susijusiąs su Bafta ar Oskarų apdovanojimais.

NA3. Scenarijus nr. 1. Linijinė diagrama – populiariausi aktoriai

1) Vartotojas pasirenka „**Bafta ir Oskarų apdovanojimų tendencijos**“ mygtuką.
2) Sistema parodo langą su įvesties parametrais.
3) Vartotojas iš sąrašo pasirenka nominaciją „geriausias pagrindinis aktorius“.
4) Sistema parodo sąrašą su nominuotų aktorių vardais ir šalia jų varneles.
5) Sistema sukuria linijinę diagramą suskirstytą pagal valandas, kur kiekviena linija atitinka raktinį žodį, šiuo atveju aktorių(-ę).
6) Vartotojas pasirenka kitus filtravimo parametrus (laiko intervalą, nominaciją) ir spaudžia „Paleisti“.
7) Sistema iš naujo sugeneruoja vizualizaciją pagal parametrus.
8) Vartotojas nori palyginti tik dviejų aktorių tendencijas, todėl pažymi varnelėmis tuos du aktorius.
9) Sistema perpiešia diagramą ir parodo tik tų dviejų aktorių linijas.
10) Vartotojas pasirenka vieną iš konkrečių nominuotų aktorių.
11) Sistema pateikia ribotą sąrašą Twitter žinučių, kuriuose buvo paminėtas aktorius.

NA3. Scenarijus nr. 2. Plokštuminė diagrama – populiariausi filmai

1) Vartotojas pasirenka „**Bafta ir Oskarų įvykių tendencijos**“ mygtuką.
2) Sistema parodo langą su įvesties parametrais.
3) Vartotojas iš sąrašo pasirenka nominaciją „geriausias filmas“.
4) Sistema parodo sąrašą su nominuotų filmų pavadinimais ir varnelėm.
5) Sistema sukuria linijinę diagramą suskirstytą pagal valandas, kur kiekviena linija atitinka raktinį žodį, šiuo atveju filmą.
6) Vartotojas pasirenka diagramos tipą „plokštuminė diagrama“.
7) Sistema perpiešia vizualiazaciją su plokštumine diagrama.
8) Vartotojas nori palyginti tik dviejų filmų tendencijas, todėl pažymi varnelėmis tuos du filmų pavadinimus.
9) Sistema perpiešia diagramą ir parodo tik tų dviejų filmų plokštumas.
10) Vartotojas pasirenka vieną iš konkrečių nominuotų filmų.
11) Sistema pateikia ribotą sąrašą Twitter žinučių, kuriuose buvo paminėtas filmas.

NA3. Scenarijus nr. 3. Skritulinė diagrama – populiariausi pagalbiniai aktoriai

1) Vartotojas pasirenka „**Bafta ir Oskarų įvykių tendencijos**“ mygtuką.
2) Sistema parodo langą su įvesties parametrais.
3) Vartotojas paspaudžia mygtuką „Dviejų nominacijų palyginimas“.
4) Sistema parodo atskirą langą su nustatymais, kuriame du išslenkantys meniu.
5) Vartotojas iš pirmo meniu pasirenka nominaciją „geriausias pagalbinis aktorius“, o iš antro „geriausia pagalbinė aktorė“.
6) Sistema sukuria vizualizaciją, kurioje dvi skritulinės diagramos. Kairėje pusėje aktoriai, dešinėje – aktorė. Išpjovos dydžiai atitinka kaip dažnai buvo minimas konkretus aktorius(-ė).
7) Vartotojas pasirenka kitus filtravimo parametrus (duomenų bazės lentelę, laiko intervalą, nominaciją) ir spaudžia „Paleisti“.
8) Sistema iš naujo sugeneruoja vizualizaciją pagal parametrus.
9) Vartotojas paspaudžia ant kairės skritulio didžiausios išpjovos.
10) Sistema sukuria linijinę diagramą suskirstytą pagal valandas, kur kiekviena linija atitinka raktinį žodį, šiuo atveju aktorių. Tai vizualizacija iš NA3S1.

2.1.5. NA4. Prekės ženklų tendencijos

Twitter žinučių analizė gali būti naudinga skirtinguose srityse (politika, marketingas, sportas ir pan.) ir įvairiems tikslams bei veikloms: rinkodaros analizei, prekės ženklo kūrimui, valdymui ir analizei, klientų poreikio ir nepasitenkinimo analizei ir pan. [Gup15].

Šiam naudojimo atvejui bus reikalinga DB lentelė (tarkim „tweets-brands“) su Twitter žinutėmis, kurios turi bent vieną iš raktažodžių: Nike, Adidas, Reebok, Puma, Converse.

Tikslai:

- Žinučių kiekio palyginimas tarp prekės ženklų.
- Iš kurių pasaulio vietų ar šalių vartotojai rašo apie šiuos prekės ženklus.
- Perskaityti kelias Twitter žinutes, kurios turi bent vieną iš prekės ženklų

NA4. Scenarijus nr. 1. Sukrautų stulpelių diagrama – populiariausi prekės ženklai

1) Vartotojas pasirenka „**Prekės ženklų tendencijos**“ mygtuką.
2) Sistema parodo langą su įvesties parametrais tarp jų sąrašas su prekės ženklų pavadinimais.
3) Vartotojas susirenka iš sąrašo kelis prekės ženklų pavadinimus ir spaudžia „Paleisti“.
4) Sistema sukuria vertikalių sukrautų stulpelių diagramą suskirstytą pagal valandas, kur skirtingos spalvos stulpelis atitinka raktinį žodį, šiuo atveju prekės ženklo pavadinimą.
5) Vartotojas pakeičia vieną iš prekės ženklų pavadinimą.
6) Sistema iš naujo sugeneruoja vizualizaciją pagal parametrus
7) Vartotojas pasirenka vieną iš prekės ženklų mygtukų ir jį paspaudžia.
8) Sistema pateikia sąrašą Twitter žinučių, kuriuose buvo panaudotas prekės ženklo pavadinimas.

NA4. Scenarijus nr. 2. Žemėlapis – prekės ženklo populiarumas pagal šalis

1) Vartotojas pasirenka „**Prekės ženklų tendencijos**“ mygtuką.
2) Sistema parodo langą su įvesties parametrais tarp jų sąrašas su prekės ženklų pavadinimais.
3) Vartotojas susirenka iš sąrašo konkretų prekės ženklą ir pasirenką žemėlapį.
4) Sistema sukuria pasaulio žemėlapį su skaičiais kiek kartų konkrečioje šalyje tas prekės ženklas buvo paminėtas.
5) Vartotojas pakeičia diagramos tipą į stulpelinę diagramą.
6) Sistema sukuria stulpelinę diagrama su top 5-10 šalių su skaičiais kiek kartų konkrečioje šalyje tas prekės ženklas buvo paminėtas.
7) Vartotojas pasirenka vieną iš top 10 šalių mygtukų ir jį paspaudžia.

8) Sistema pateikia ribotą sąrašą Twitter žinučių pagal šalį, iš kurios buvo išsiustos žinutės su prekės ženklo pavadinimu.

Galimi papildomi tikslai [Gup15], kuriems gali sukurti atskirus scenarijus:

- Kokios populiariausios žymės yra naudojamos rašant žinutes apie **X**?
- Kokios labiausiai naudojamos kalbos, kuriomis vartotojai rašo žinutes apie **X**?
- Iš kokios įrenginių vartotojai rašo apie **X**?

2.1.6. NA5. Duomenų bazės papildymas

Šiame naudojimo atvejyje pagrindinis veikėjas yra sistemos administratorius. Administratorius naudos sistemos konfigūracijos failus ir komandinę eilutę.

Tikslai:

- Sukurti atskirą duomenų bazės lentelę.
- Įtraukti naujų žinučių į duomenų bazės lentelę.

NA5. Scenarijus nr. 1. Naujos DB lentelės sukūrimas

1) Administratorius sustabdo žinučių rinkimo posistemę jeigu jinai įjungta.
2) Administratorius atsidaro konfigūracijos failą ir įrašo papildomą duomenų bazės lentelės pavadinimą.
3) Administratorius parašo sąrašą raktažodžių arba koordinačių pagal kuriuos bus renkamos Twitter žinutės.
4) Administratorius paleidžia posistemę.
5) Sistema kreipiasi į Twitter serverius ir pradeda rinktis žinutes.
6) Sistema kaupia žinučių srautą į naują duomenų bazės lentelę.

NA5. Scenarijus nr. 2. Rinkimo parametrų pakeitimas

1) Administratorius sustabdo sistema jeigu jinai įjungta.
2) Administratorius atsidaro konfigūracijos failą ir pakeičia arba papildo DB lentelės raktažodžių masyvą, pvz.: [#BAFTA, #oscars, #oscars2016, theAcademy] ir pan.
3) Administratorius iš naujo paleidžia žinučių rinkimo posistemę.
4) Sistema kreipiasi į Twitter serverius ir pradeda rinktis žinutes.
5) Sistema kaupia žinučių srautą į naują duomenų bazės lentelę.

2.1.7. Kiti naudojimo atvejai

2.1.7.1. Paprastų raktažodžių įvedimas

1) Atidaroma programėlė ir pasirenkamas 1 naudojimo atvejis.

2) Atsidaro raktažodžių įvedimo langas. Vartotojas įveda raktažodį ir spaudžia „Įtraukti“. Kai baigia įvedinėti žodžius tuomet paspaudžia „Paleisti“.

3) Atsidaro bendras vizualizavimo langas su valdymo skydeliu. Vizualizacija keičiasi kas 1 sekunde, jeigu duomenis pasiekia mobilųjį įrenginį. Jei duomenų nėra, rodoma žinutė „Ieškomi raktažodžiai, prašome palaukti. Gauta *n* žinučių“.

4) Norint pakeisti veikimą vartotojas naudoja valdymo skydelio mygtukus.

5) Paspaudus ant konkretaus raktažodžio stačiakampio atsidaro langas su žinučių sąrašu, kuris kas 3 sekundes atsinaujina.

2.1.7.2. Kategorijos iš failo

1) Atidaroma programėlė ir pasirenkamas 2 naudojimo atvejis.

2) Atsidaro langas failui pasirinkti. Vartotojas pasirenka .csv failą ir spaudžia „OK“. Jeigu failas netinkamo formato, tuomet parodomas klaidos pranešimas: „.CSV failo formatas neteisingas, žiūrėti instrukciją!“. Jei failas gero formato, tuomet failas nuskaitomas.

3) Atsidaro bendras vizualizavimo langas su valdymo skydeliu. Vizualizacija suskirstyta į kategorijas. Diagrama keičiasi kas 1 sek., jeigu duomenis pasiekia mobilųjį įrenginį. Jei duomenų nėra, rodoma žinutė „Ieškomi raktažodžiai, palaukite. Gauta *n* žinučių“.

4) Norint pakeisti veikimą vartotojas naudoja valdymo skydelio mygtukus.

5) Paspaudus ant konkrečios kategorijos atsidaro langas su raktažodžių Treemap vaizdu, kuris kas 3 sekundes atsinaujina.

6) Paspaudus ant raktažodžio stačiakampio atsidaro langas su žinučių sąrašu, kuris kas 3 sekundes atsinaujina.

2.1.7.3. Kategorijos pagal kontinentus ir šalis

1) Atidaroma programėlė ir pasirenkamas 3 naudojimo atvejis.

2) Atsidaro bendras langas ir rodamas vaizdas su visu Treemap ir linijine diagrama apačioje. Treemap bus suskirstytas į dalis pagal didžiausius gyvenamus žemynus (Š. Amerika, P. Amerika, Europa, Afrika, Azija, Australija).

3) Pirštu paspaudus ant konkretaus žemyno stačiakampio bus priartinta prie būtent to žemyno Treemap vizualizacijos, t.y. bus parodoma vizualizacija tik tos kategorijos.

4) Paspaudus ant konkretaus stačiakampio bus atidarytas atskiras langas su Twitter žinutėmis, kurios susijusios tik su to stačiakampio tema. Naujas žinutes galima bus pamatyti paspaudus mygtuką viršuje korio pavadinimas bus „N naujos žinutės“, kur ***N*** tuo metu neparodytų naujų žinučių skaičius.

5) Paspaudus mygtuką „atgal“, bus grįžtama į praeitą konkretaus žemyno Treemap vizualizaciją su jau perpieštais stačiakampiais priklausomai nuo naujų žinučių.

6) Paspaudus mygtuką „atgal“ dar kartą, vaizdas bus atitolinamas ir grįžtama į pradinį puslapį.

2.1.7.4. Konkreti teritorija

1) Atidaroma programėlė ir pasirenkamas 4 naudojimo atvejis.

2) Atsidaro koordinačių įvedimo langas. Vartotojas įveda koordinates į du atskirus langelius ilgumai ir platumai. Toliau spaudžia „Įtraukti“. Kai vartotojas baigia įvedinėti dvi koordinačių poras tuomet mygtukas „Įtraukti“ išjungiamas ir vartotojas paspaudžia mygtuką „Paleisti“.

3) Atsidaro bendras vizualizavimo langas su valdymo skydeliu. Vizualizacija keičiasi kas 1 sekunde, jeigu duomenis pasiekia mobilųjį įrenginį. Jei duomenų nėra, rodoma žinutė „Ieškomi raktažodžiai, prašome palaukti. Gauta *n* žinučių“.

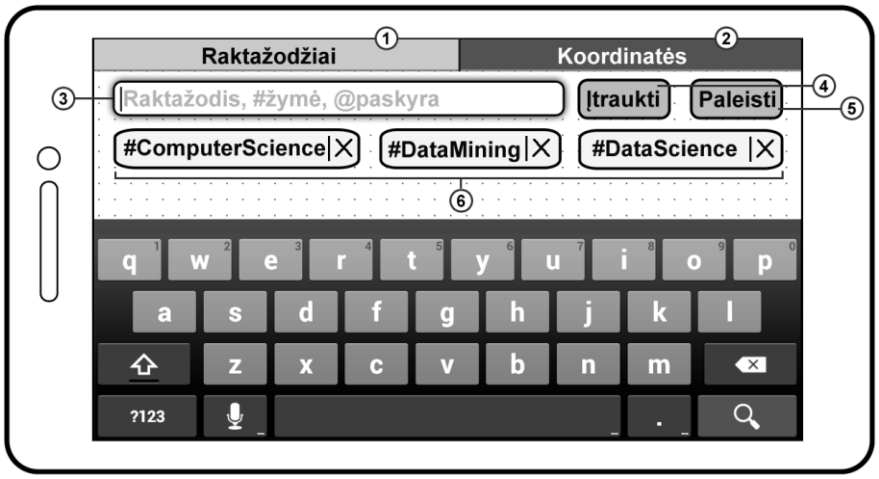
4) Norint pakeisti veikimą vartotojas naudoja valdymo skydelio mygtukus.

5) Paspaudus ant konkretaus raktažodžio stačiakampio atsidaro langas su žinučių sąrašu, kuris kas 3 sekundes atsinaujina.

2.2. Vizualizavimo sprendimai

2.2.1. Vartotojo sąsaja

Duomenų įvedimo sąsaja



20 pav. TreemapViz raktažodžių įvedimo vartotojo sąsaja

1) **Raktažodžiai**. Įjungta kortelė, kur vartotojas įveda raktažodžius.

2) **Koordinatės.** Išjungta kortelė, kur vartotojas įveda koordinates (20 pav.).

3) **Įvesties laukas**.

4) **Įtraukti** – mygtukas skitas įtraukti raktažodį į sąrašą apačioje.

5) **Paleisti** – mygtukas pabaigti raktažodžių įvedimą ir pradėti piešti vizualizaciją. Toliau rodomas vizualizavimo vaizdas (21 pav.) su valdymo skydeliu.

6) **Raktažodžiai** – sąrašas įvestų raktažodžių, kuriuos galima išmesti paspaudus ant „X“.

Valdymo skydelis

1) **Linijinė diagrama** vaizduoja kiek skirtingų raktažodžių gaunama realių laiku (21 pav.).

2) **Stabdyti** – mygtukas skirtas stabdyti vizualizavimą ir nutraukti ryšį.

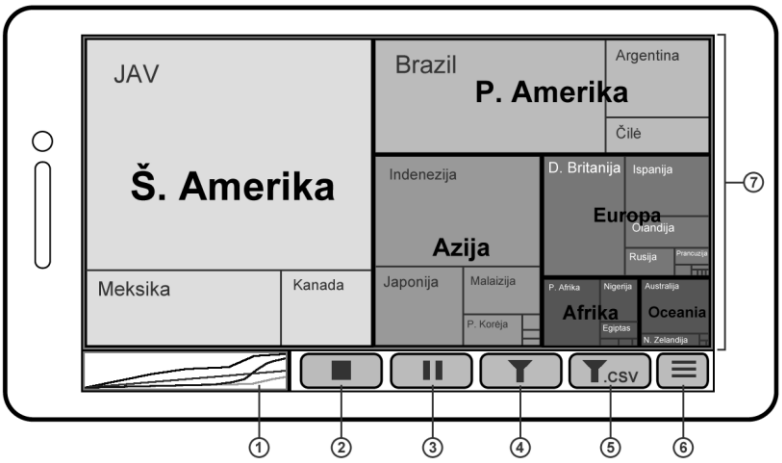
3) **Pauze/Paleisti** – mygtukas skirtas stabdyti, o po to tęsti vizualizavimą.

4) **Filtruoti** – mygtukas skirtas atidaryti langą skirtą raktažodžių įvedimui (20 pav.).

5) **Filtruoti iš .csv failo** – mygtukas skirtas atidaryti langą skirtą .csv failo parinkimui.

6) **Nustatymai**– mygtukas skirtas atidaryti numatytą nustatymų langą.

7) **Medžio pavidalo žemėlapis** (angl. *Treemap*).

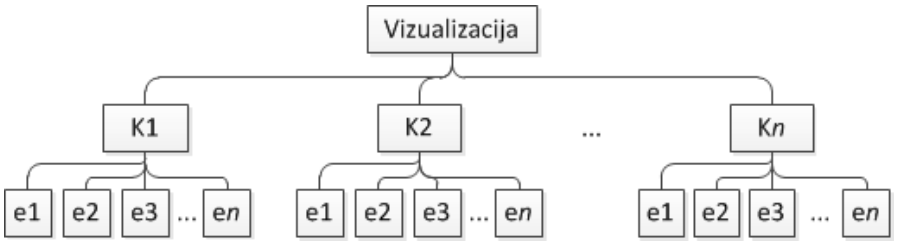


21 pav. TreemapViz vartotojo sąsaja su vizualizacija ir skydeliu

2.2.2. Medžiais paremtas žemėlapis

Skirstymas į kategorijas

Esant daugybei ieškomų raktažodžių Treemap vizualizaciją bus galima suskirstyti į kategorijas. Kiekviena kategorija turės atskirus elementus (lapus), taigi duomenys bus suskirstyti pagal medžio principą. Medžio šaknis tai visa vizualizacija (22 pav.).



22 pav. Medžio pavidalo žemėlapio duomenų struktūra paremta medžiu su kategorijom

Skirstyti į kategorijas bus įmanoma filtruojant Twitter srautą naudojant .csv failą. Failo turi būti parašytas CSV formatu ir pirmoje eilutėje turėti kategorijų pavadinimus, o apatiniuose stulpeliuose tom kategorijom skirtus raktažodžius, kaip pavaizduota šiame pavyzdyje:

```
Kategorija-1,Kategorija-2,Kategorija-3
e1,e1,e1
e2,,e2
e3,,
```

Naudojamas algoritmas

Naudosime Bruls-Huizing-Wijk pagerintą „Squarified Treemap“ [BHW00] algoritmą, kuris yra paremtas Johnson ir Shneiderman darbu apie medžio pavidalo žemėlapius [JS91].

2.3. Galimi sistemos architektūros sprendimai

2.3.1. Skaičiavimai mobiliajame įrenginyje

Šiame poskyryje aprašome bendri komponentai kurie bus naudojami sprendimuose 1 ir 2.

Mobili Android programėlė

Realizuojant mobiliąją programėlę bus naudojama Android SDK, todėl būtina mokėti programuoti Java kalba. Taip pat reikia suprasti tokių failų struktūrą: XML, JSON, CSV.

Mobilus sprendimas yra skirtas žmonėm, kurie neturi priėjimo prie stacionaraus ar nešiojamo kompiuterio. Pavyzdžiui, dalyvaujant kokiam nors renginyje (sporto varžybos, koncerte ar vakarėlyje) arba esant ekstremaliom sąlygom (evakuacijos metu dėl žemės drebėjimo, audros, tsunamio ar pan.). Sprendimas mobiliąja įrenginyje gali būti patogesnis tais atvejais, kai vartotojas nori naudoti programėlę šalia sėdint prie personalinio kompiuterio. Pasirenkame mobiliosios programėlės kelią, o ne tinklalapio, nes:

- Naudojant programėlę galimą sukurti patogesnę sąveiką ir gestais paremtą valdymą (priartinti, atitolinti ir pan.).
- Tarkime programėlė yra užsakyta komerciniams tikslams ir vienas iš reikalavimų yra galimybė ją parduot naudojant elektroninę programėlių parduotuvę (Google Play, iStore ir pan.).
- Kai kurie senesni išmanieji telefonai nepalaiko HTML5, todėl negalime naudoti JavaScript vizualizavimo bibliotekų (D3.js ir pan.).

Pasirinktas Android, o ne iPhone, nes norint programuoti ant iPhone išmaniojo telefono būtina turėti kompiuterį su Mac operacinę sistema (OS). Nors VU MIF suteikia galimybę programuoti ant tokių kompiuterių, deja tai nėra visą laiką prieinama. Kadangi norima dirbti ir namie, buvo pasirinkta Android OS, nes įrenginiams su šia OS programuoti tinka ir Windows aplinka. Programuoti verta naudoti Google suteiktą programavimo aplinką „Android Studio“.

OAuth ir Hosebird Client Java biblioteka

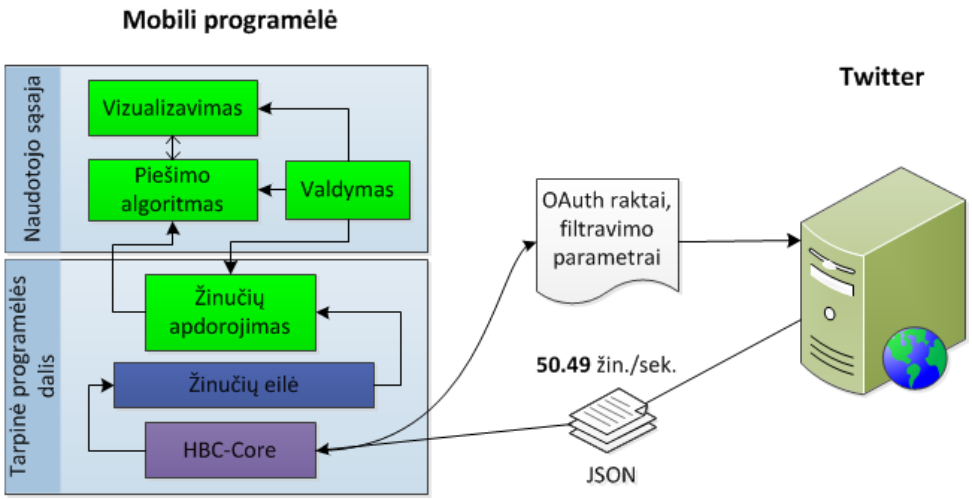
Autentikacijai su Twitter serveriu naudosime OAuth prisijungimo raktus. OAuth sąsaja jau yra įtraukta į Hosebird Client (HBC) Java biblioteką pavadinimu HBC-Core, taigi mums nereikės jos programuoti. HBC-Core turi visas mums reikiamas funkcijas ir failus, kurias naudosime duomenų srauto atsisiuntimui iš Twitter serverio. Ši biblioteka pasirinkta todėl, kad parašyta Java kalba, o tai leidžia ją naudoti Android mobiliosios programėlės kūrime.

Vizualizavimas naudojant android.graphics

Norint sukurti vizualizacijas Android mobiliajame įrenginyje, galima naudoti standartinę 2 dimensijų piešimo sąsają Andriod 2D „Drawing API“, kuri pasiekiama importuojant `android.graphics.*` klases į projektą. Piešimui reikėtų mažiausiai keturių klasių:

- *Bitmap* – klasė, kuriame keičiami piešinio aukščio, pločio, pikselio parametrai ir pan.,
- *Canvas* – per kurį kviesime piešimo funkcijas ir procedūras,
- *Rect, Path, Point* ar pan. – bet kuri klasė apibūdinanti piešiamą objektą (stačiakampį, atkarpą, tašką ar pan.),
- *Paint* – klasė su kuria aprašysime šrifto dydį ir kuriomis spalvomis bus piešiama.

2.3.2. Pirmas sprendimas: mobili programėlė



23 pav. Sprendimo nr. 1 architektūros diagrama

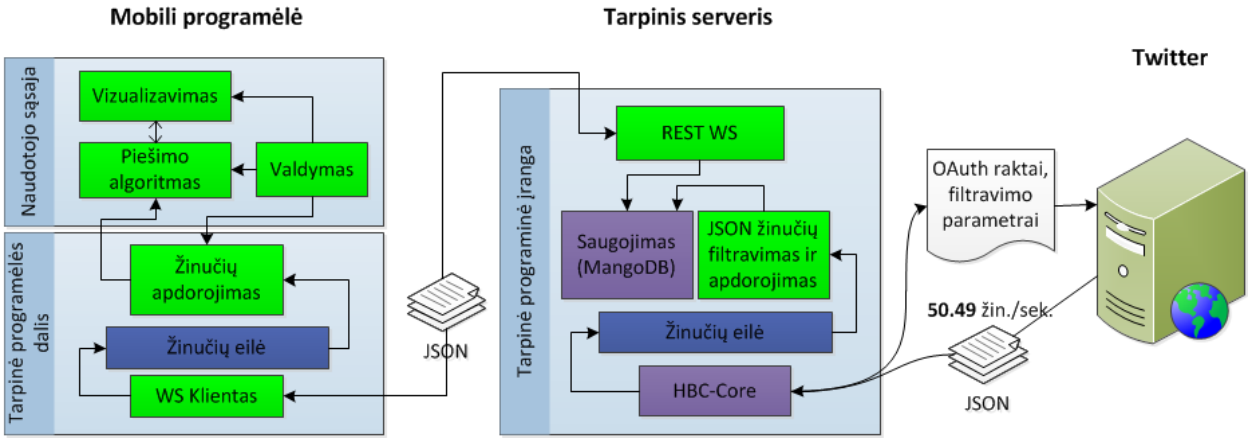
Privalumai

- Nebūtina turėti asmeninį kompiuterį, todėl galima naudotis visur kur yra mobilus ryšys.
- Galima siųsti Twitter serveriui filtravimo parametrus, o tai reiškia, kad mus pasieks tik tie duomenys, kurie turi apibrėžtus raktažodžius ir/arba koordinates.

Trūkumai

- Šia programėle galės naudotis tik vienas vartotojas vienu metu, kadangi prisijungimas su OAuth raktais apribojamas vienai Twitter programėlei ar programinei įrangai.
- Dėl nuolatinio duomenų srauto ir pastovaus apdorojimo mobilusis įrenginys bus apkrautas, todėl tikėtina, kad programėlė išnaudos daug įrenginio energijos.

2.3.3. Antras sprendimas: mobili programėlė su skaičiavimais tarpiniame serveryje



24 pav. Sprendimo nr. 2 architektūros diagrama

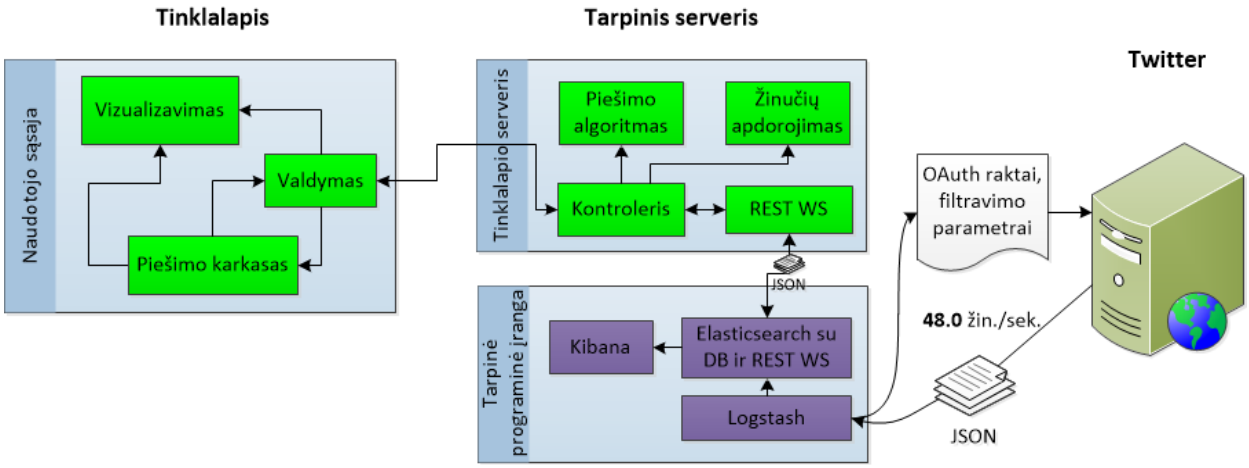
Privalumai

- Naudojant šį sprendimą bus galima apeiti reikalavimą, kad reikia registruoti atskirą Twitter programėlę kiekvienam vartotojui, nes tik tarpinis serveris naudos prisijungimo raktus. Taigi šiuo sprendimu galės naudotis daugiau nei vienas mobilus įrenginys.
- Galima bus sukurti žinučių (1-24val.) archyvą. Tai leis pateikti daugiau žinučių klientui, o to pasėkoje vizualizacija bus turiningesnė ir reikšmingi rezultatai atsiras greičiau.
- Mobilus įrenginys neturės atlikinėti sudėtingo filtravimo ir skaičiavimo. Tai sumažins mobiliojo įrenginio apkrovą, o to pasėkoje sutaupys vartotojo laiką ir mobiliojo įrenginio bateriją.

Trūkumai

- Serveris negalės siųsti užklausos su filtravimu ir teks priimti visą Twitter duomenų srautą, nes skirtingi vartotojai norės skirtingų duomenų.
- Dėl filtravimo stokos, nėra žinoma, kad bus gaunamos visas galimos žinutės. Twitter serveris siunčia tik dalį visų gautų žinučių, nėra žinoma kiek procentų iš tikrųjų yra siunčiama. Taigi sumažės vizualizacijos tikslumas (angl. *accuracy*).
- Reikės skirti nemažai laiko tarpinės programinės įrangos programavimui.
- Android diagramų piešimo karkasai neturi tokių sprendimų ir galimybių, kokius turi Javascript karkasai, tokie kaip D3 ir Highcharts. Trūksta mokomosios medžiagos.

2.3.4. Trečias sprendimas: prisitaikančio dizaino tinklalapis su skaičiavimais tarpiniame serveryje



25 pav. Sprendimo nr. 3 architektūros diagrama

Aprašymas

Trečias sprendimas ypatingas tuo, kad naudojami įrankiai palengvinantys įgyvendinimą ir sutaupantys laiko. Vietoj to, kad pačiam programuoti Twitter srauto nuskaitymo funkcionalumą, naudojamas Logstash įrankis su Twitter įskiepiu. Žinutės nukreipiamos į Elasticsearch, kuris turi dokumentų duomenų bazę. Panašiai kaip ir MangoDB, čia nereikia iš anksto apsirašyti schemos, nes indeksai (lentelių atitikmuo) saugomi JSON formatu. Elasticsearch duomenų bazės žinučių lygis įgyvendintas kaip REST saityno paslauga, o tai yra patogu tiek testuojant, tiek įgyvendinant vizualizacijos sistemą. Kibana įrankis pateikia vartotojo sąsaja su paieška ir galimybe kurti savo diagramas. Taigi galima tyrinėti gautas žinutes, palyginant testuoti ar sistemos tinklalapio vizualizacijos piešiamos teisingai. Tinklalapio serveris bus skirtas siųsti tinklalapio puslapius vartotojams (HTML + JavaScript + CSS). Serveryje apdoros vartotojų užklausas ir jame bus atliekami skaičiavimai reikalingi vizualizavimams generuoti. Serveris taip pat bendraus su Elasticsearch duomenų baze per REST saityno paslaugą. Galiausiai, vizualizacijos bus piešiamos tinklalapyje naudojant Javascript karkasus D3 ir Highcharts.

Privalumai

- Nebūtina turėti būtent Android mobilųjį įrenginį, nes tinklalapio programėlė veikia naršyklėje ir todėl galima naudoti tiek Android, tiek iOS, tiek kitų operacinių sistemų mobiliuosius telefonus, planšetes ir personalinius kompiuterius.
- Naudojant šį sprendimą bus galima apeiti reikalavimą, kad reikia registruoti atskirą Twitter programėlę kiekvienam vartotojui, nes tik tarpinis serveris naudos prisijungimo raktus. Taigi šiuo sprendimu galės naudotis daugiau nei vienas įrenginys.

- Galima bus sukurti žinučių (1-24val.) archyvą. Tai leis pateikti daugiau žinučių klientui, o to pasėkoje vizualizacija bus turiningesnė ir reikšmingi rezultatai atsiras greičiau.
- Logstash, Elasticsearch ir Kibana yra tarpusavyje suderinami atviro kodo įrankiai, pagaminti tos pačios įmonės. Yra galimybė naudoti įskiepius, kuriuos pagamino bendruomenė.
- Elasticsearch automatiškai išanalizuoja žinutės JSON struktūra ir dokumentus indeksuoja, todėl paieškos užklausos atliekamos greitai.
- Ateityje bus galima lengviau išskirstyti ir praplėsti sistemą į papildomus serverius. Tokia architektūra leis paleisti po atskirą Logstash paiešką per serverį, turint omeny, kad bus keli Twitter aplikacijos vartotojo paskyros. Kuo daugiau paieškų, tuo daugiau pasirinkimo turės vartotojas. Pvz.: vienas Logstash serveris kaups Twitter srauto duomenis iš Europos, o antras – iš JAV. Vartotojas galės pasirinkti, kuriuos duomenis nori analizuoti ir tam neprireiks stabdyti Logstash serverio ir jį perkrauti su naujais nustatymais.

Trūkumai

- Logstash serveris galės siųsti užklausą į Twitter serverį tik su iš anksto numatytais filtravimo parametrais (raktažodžiai, žymės ar koordinatės). Norint pakeisti paieškos žodžius, sistemos administratoriui reikės išjungti serverį, pakeisti konfigūracijos failą ir iš naujo paleisti Logstash serverį. Taigi teks priimti sprendimą dėl paieškos nustatymų, kurie tiktų daugeliui vartotojų. Tai nėra gerai, nes skirtingi vartotojai norės skirtingų duomenų. Norint kokybiškų ir tikslių duomenų vizualizacijai, reikia konkrečios temos, tarkim susijusios su vienu įvykiu, pvz.: Oskarų apdovanojimai, rinkimai, sporto varzybos.
- Tinklalapis nėra toks patogus valdymui, kaip programėlė, nes kažkurių vartotojo gestų tinklalapis pilnai nepalaiko ar palaiko ne taip gerai kaip programėlė, pvz.: piršo laikymas kelias sekundes (skirtingai nei piršto paspaudimas), dviejų ar trijų pirštų kombinacijos. Naudojant programėlę galimą sukurti patogesnę sąveiką ir gestais paremtą valdymą (priartinti, atitolinti ir pan.).

2.4. Sprendimo pasirinkimas

Sukūrus pirmojo sprendimo prototipą (6 priedas, 51 pav., 52 pav.) ir atlikus trijų sprendimų palyginimą ir įvertinimą (6 lentelė) buvo pasirinktas trečias sprendimas, nes tenkino daugiausiai kriterijų. Kriterijai buvo pasirinkti pagal tai, kas autoriaus nuomonei atrodo svarbiausia tokio tipo sistemos sukūrimui. Aparatinė įranga nebuvo lyginama, nes būtų naudojama ta pati visuose sprendimuose..

6 lentelė. Trijų sprendimų palyginimas ir įvertinimas pagal kriterijus

	1-as sprendimas	**2-as sprendimas**	**3-čias sprendimas**
Vartotojo sąsajos rūšis (technologijos)	Mobili programėlė (Android)	Mobili programėlė (Android)	Tinklalapis (HTML5, Javascript, CSS)
Vartotojo sąsajos programavimo karkasai, bibliotekos	Android graphics, 2D Drawing API	Android graphics, 2D Drawing API	Angular, D3, Highcharts, Bootstrap
Tarpinės programinė įranga	HBC-Core	HBC-Core, MangoDB, Tomcat	ELK įrankiai, Tomcat
Gali turėti daugiau nei 1 vartotoją	**Ne**	**Taip**	**Taip**
Žinučių archyvas	**Ne**	**Taip**	**Taip**
Numatytas serverio išplečiamumas	**Ne**	**Ne**	**Taip**
Tinka visoms OS	**Ne**	**Ne**	**Taip**
Sprendimas veikia mobiliame įrenginyje	**Taip**	**Taip**	**Taip**
Gali išnaudoti visus mobiliojo įrenginio gestus	**Taip**	**Taip**	**Ne**
Duomenų bazė palaiko JSON objektų saugojimą	**Ne**	**Taip**	**Taip**
Ar vizualizacijos kūrimas paprastas ir technologijos turi išsamią dokumentaciją su pavyzdžiais?	**Ne**	**Ne**	**Taip**

3. Prototipo įgyvendinimas

Šioje dalyje aprašoma sistemos įgyvendinimas ir architektūra (26 pav.). Pradžioje aprašomi taip vadinami ELK įrankiai ir kaip jie buvo pritaikomi. ELK – tai trijų atviro kodo įrankių derinys: Elasticsearch, Logstash ir Kibana. Logstash surenka duomenis arba failus, Elasticsearch juos saugo ir indeksuoja, o Kibana pateikia paieškos ir vizualizavimo vartotojo sąsają. Naršyklėje naudojamas Angular Javascript karkasas skirtas darbui su duomenimis ir komunikacija su saityno paslauga. Diagramos piešimui naudojamas D3 ir Highchart Javascript karkasai. Puslapio išdėstymui ir prisitaikančiam dizainui (angl. *responsive design*) – Bootstrap CSS karkasas.

3.1. Praktinės dalies eiga

Ketvirto semestro metu buvo pasirinktas sprendimas nr. 3 ir pagal jį buvo sukurta Twitter vizualizavimo sistema. Tai buvo atlikta sekančia seka:

1. Sukurti naudotojo sąsajos puslapių eskizai (20 pav., 21 pav. 5 priedo, 49 pav., 50 pav.).
2. Sukurta Twitter žinučių duomenų bazė Elasticsearch serveryje ir sukurtos lentelės su Bafta ir Oskarų apdovanojimų įvykių žinutėmis.
3. Atliktas tyrimas norint surasti sprendimą kaip suderinti Angular JS, D3 ir Highcharts karkasus.
4. Atliktas tyrimas norint surasti sprendimą kaip sukurti komunikacijos sluoksnį tarp Angular JS ir Elasticsearch duomenų bazės.
5. Integruotas Elasticsearch serveris su savo sukurtu funkcionalumu.
6. Sukurta detali sistemos architektūros diagrama prijungiant ELK įrankius.
7. Sukonfigūruotas vietinis tinklas ir serverių nustatymai leidžiantys sistemos tinklalapį pasiekti HTTP protokolu per internetą. Sukurta dislokacijos diagrama.
8. Sukurtas funkcionalumo klasės ir D3 vizualizavimo algoritmas (Javascript).
9. Suprogramuotas prototipas su vartotojo sąsaja, kurioje: valdymo panelė su parametrų įvedimo elementais, diagramos ir žinučių dalys.
10. Ištestuotas tinklalapio valdymas ir piešiamos diagramos, kurios buvo aprašytos naudojimo atvejų skyriuje. Įgyvendinti sekantys scenarijai (6 priedas):
    a. NA1. Scenarijus nr. 1. Surasti Twitter populiariausias žymes.
    b. NA3. Scenarijus nr. 1. Linijinė diagrama – populiariausi aktoriai
    c. NA3. Scenarijus nr. 2. Plokštuminė diagrama – populiariausi filmai
    d. NA5. Scenarijus nr. 1. Naujos DB lentelės sukūrimas
    e. NA5. Scenarijus nr. 2. Rinkimo parametrų pakeitimas

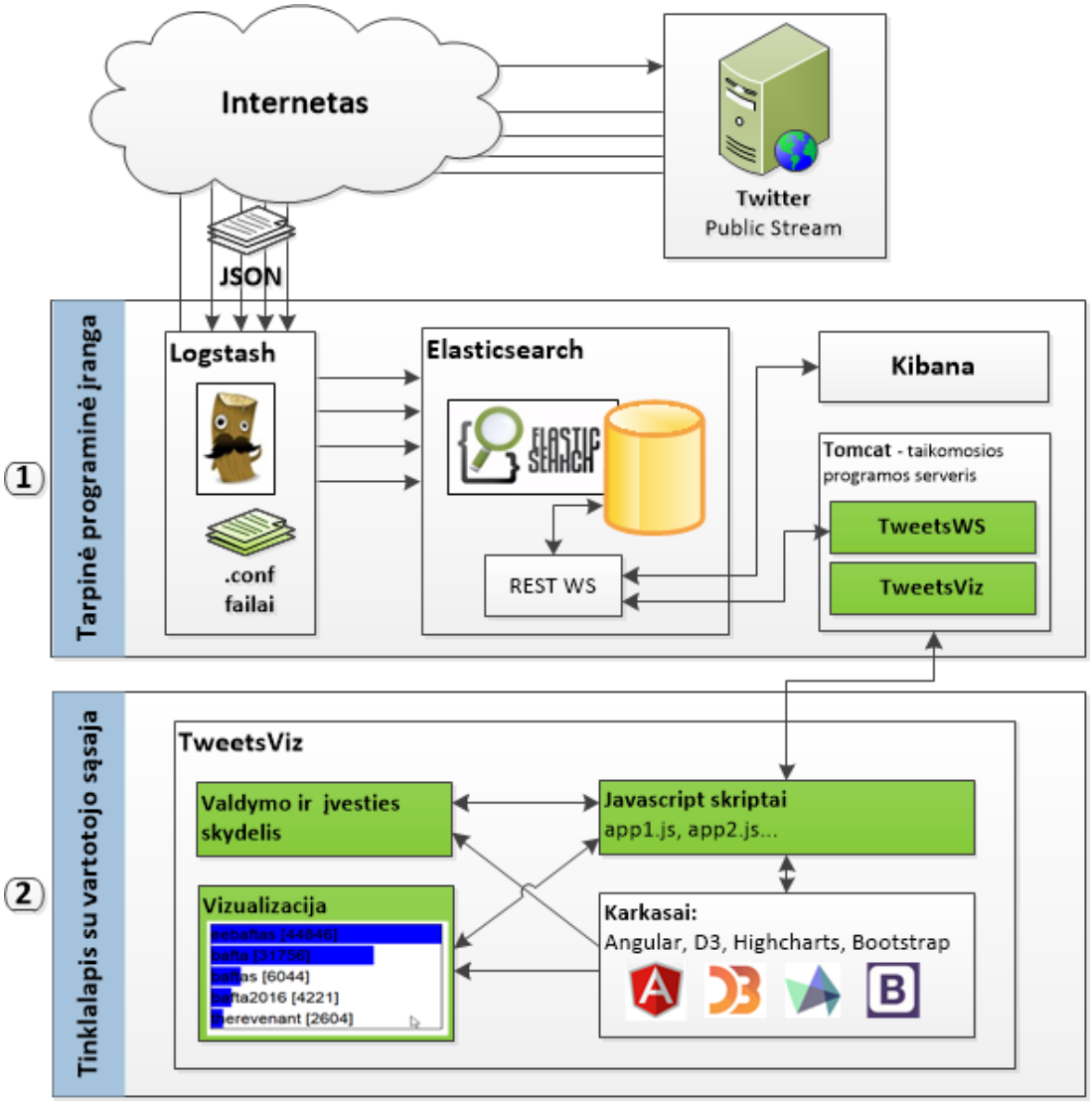
11. Vizualizavimo rezultatai pratestuoti palyginant su Elasticsearch duomenų bazės duomenimis (per REST WS) ir su Kibana paieškos rezultatais ir grafikais.
12. Aprašyta kaip ELK įrankiai buvo konfigūruoti ir pritaikyti sistemoje.

3.2. Sistemos įgyvendinimo architektūra

Sistemos architektūra sudaro dvi pagrindinės dalys (26 pav.):

1) Tarpinė programinė įranga – tai viename fiziniame serveryje (kompiuteryje) esantys paslaugų serveriai. Šie serveriai yra: Logstash, Elasticsearch, Kibana ir Tomcat taikomųjų programų serveris. Jame paleista šio darbo metu sukurta paslauga TweetsWS ir tinklalapis TweetsViz (nuspalvinta žaliai).
2) Tinklalapis su vartotojo sąsaja – tai sukurtas TweetsViz tinklalapis, kuris turi valdymo skydelį, vizualizavimo dalį ir naudoja karkasus: Angular, D3 Highcharts ir Bootstrap.

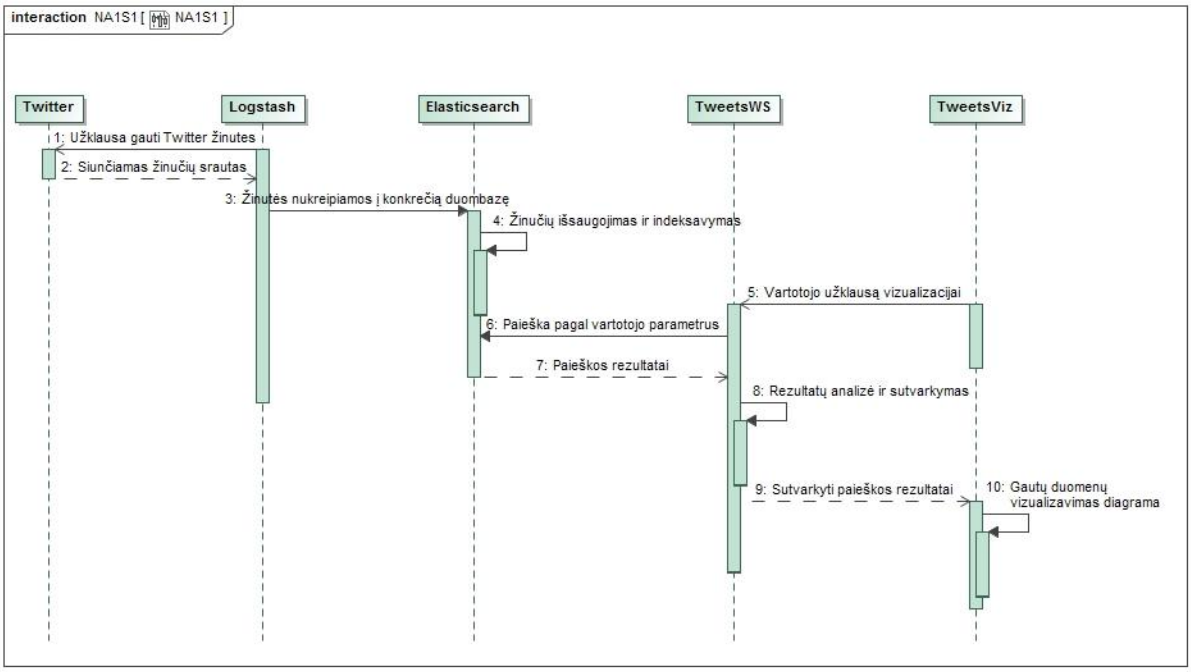
Detaliau apie išvardintas architektūros dalis aprašoma sekančiuose poskyriuose.



26 pav. Sistemos įgyvendinimo architektūra

3.3. Naudojimo atvejo NA1 sekų diagrama

Kadangi pagrindinių naudojimų atvejų (NA1 – NA4) sekų diagramos nesiskiria viena nuo kitos, o NA5 scenarijai apima tik pirmus du sekos žingsnius, todėl pateikta tik NA1S1 sekų diagrama kaip pavyzdys.

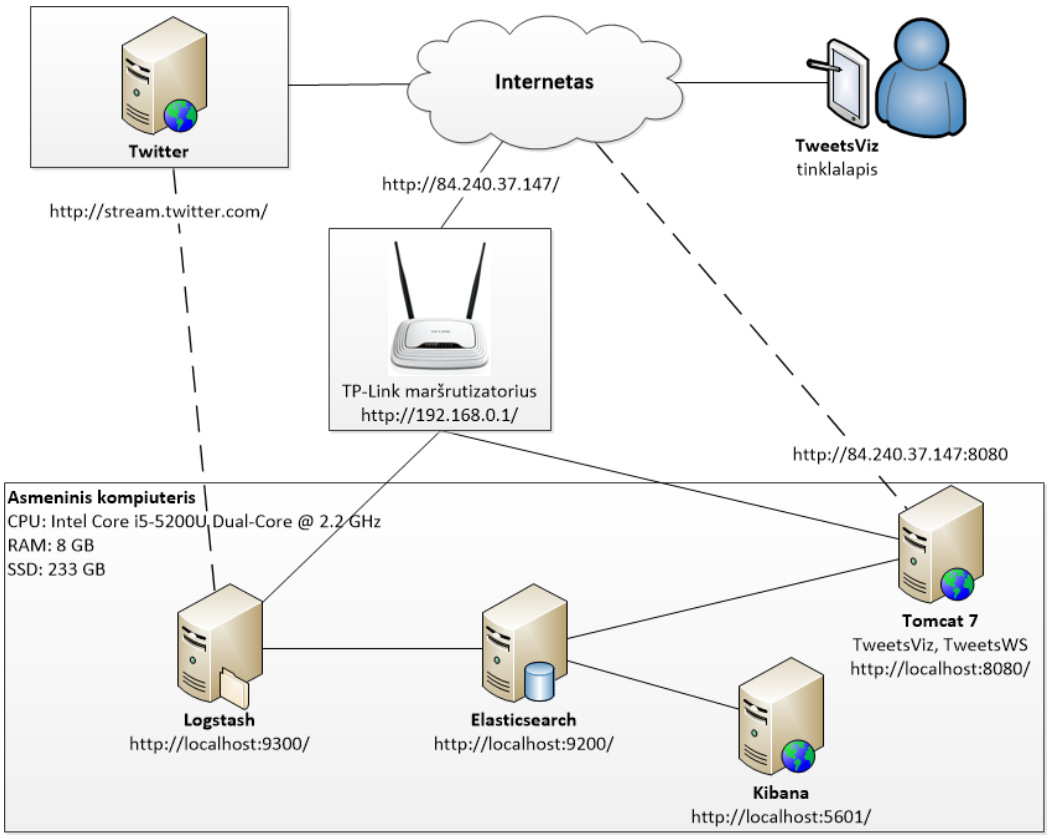


27 pav. NA1S1 naudojimų atvejo sekų diagrama

Sekų diagramos žingsniai:

1. Logstash siunčia užklausą Twitter serveriui, kad gauti Twitter žinutes pagal konkrečius raktažodžius, žymes ir/arba teritoriją.
2. Twitter paslauga siunčia žinutes (JSON formatu) pastoviu srautu (angl. *stream*) tol, kol Logstash serveris nesustabdomas arba tol, kol pasiekia Twitter nustatytą dienos limitą.
3. Logstash pastoviai perduoda Twitter žinutes Elasticsearch serveriui.
4. Elasticsearch serveris gautas žinutes išsaugo ir indeksuoja.
5. Vartotojas naudoja TweetsViz tinklalapio vartotojo sąsają ir per ją siunčia užklausą vizualizacijai sukurti. Kiekvienas panaudojimo atvejis turės savo užklausą su parametrais, kuriuos pasirinks vartotojas.
6. TweetsWS perduoda paieškos užklausą su vartotojo parametrais Elasticsearch serveriui.
7. Elasticsearch serveris suranda konkrečias žinutes ir siunčia paieškos rezultatų JSON objektą atgal TweetsWS paslaugai.
8. TweetsWS atlieka rezultatų analizę ir sutvarko duomenis JSON formatu.
9. TweetsWS siunčia sutvarkytus paieškos rezultatus JSON formatu.
10. TweetsViz sukuria vizualizaciją pagal gautus duomenis.

3.4. *Dislokavimo diagrama*



28 pav. Tinklo ir serverių diagrama

Lokaliame tinkle esantys serveriai sukonfigūruoti taip, kad galėtų tarpusavyje komunikuoti. Elasticsearch duomenų bazės REST paslauga prieinama iš lokalaus tinklo tam, kad užtikrinti saugumą. Todėl TweetsWS klientas talpinamas lokalaus tinklo serveryje. TweetsViz ir TweetsWS taikomosios programos paleidžiamos ant Tomcat 7 serverio ir tik šitas tinklalapio serveris yra prieinamas iš interneto. Logstash serveris naudoja internetą, bet jo IP adresas neprieinamas iš interneto. Mobilaus įrenginio IP adresas reikšmės neturi.

7 lentelė. Įgyvendintos sistemos IP adresai su prievado (angl. port) numeriais pagal serverio tipus

Serverio vardas	Lokalaus tinklo IP adresas	Išorinis IP addresas
TP-Link maršrutizatorius	http://192.168.0.1/	http://84.240.37.147/
Tomcat 7 (TweetsWS, TweetsViz)	http://localhost:8080/	http://84.240.37.147:8080/
Kibana	http://localhost:5601/	neprieinamas
Elasticsearch	http://localhost:9200/	neprieinamas
Logstash	http://localhost:9300/	neprieinamas

3.5. Logstash – Twitter žinučių srauto surinkimas

Twitter žinutes galima gauti naudojant REST ir Stream API. Žinučių srautui apdoroti pasirinktas Twitter Stream API. Tačiau kaip panaudoti Stream API? Galima suprogramuoti įgyvendinimą patiems arba pasinaudoti jau sukurtu įrankiu pavadinimu Logstash. Tai yra atviro kodo įrankis skirtas rinkti, apdoroti ir saugoti sistemos išvesties (angl. *log*) failus, tam kad galima būtų juos panaudoti ateityje. Logstash dažniausiai naudojamas su Elasticsearch dokumentų duomenų baze, tačiau Logstash gali saugoti duomenis ir į paprastus duomenų formatus pvz.: .txt, .csv, . ar saugoti kitais būdais, pvz.: siųsti el. paštą, saugoti į MangoDB ir kitaip.

Visi ELK įrankiai diegimo failai pasiekiami elasticsearch puspalyje[14] ir paleidžiami kaip serveriai naudojant komandinę eilutę. Norint paleisti Logstash procesą, iš pradžių atsisiunčiamas „logstash-2.1.1.zip“ archyvas ir išpakuojamas. Tuomet sukuriamas konfigūracijos `.conf` failas pagrindiniame `logstash-2.1.1` kataloge. Katalogo pavadinimas skirsis nuo įrankio versijos numerio.

```
input {
    twitter {
        # add your data
        consumer_key => "[redacted]W5jNWGn1UxS67X"
        consumer_secret => "[redacted]fXXWlQ516dEhpXvQmCLXos7ZCQ0u8lXzsYWDva2Ow7"
        oauth_token => "2926052357-Fo[redacted]k0bUCfBQv3BGPPHNMwUUuC7Ly9A"
        oauth_token_secret => "sGY[redacted]CjtDJ7UnfdSSVujg0QEj9bWMcg0a"
        keywords => ["bafta","eebaftas","#bafta","#eebaftas"]
        # languages => ["lt"]
        # ignore_retweets => true
        full_tweet => true
    }
}
filter{
}
output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "tweets-bafta"
    }
}
```

29 pav. Žinučių gavimas nurodant raktažodžius, failo „twitter-bafta.conf“ fragmentas

Konfigūracijos failas susideda iš trijų dalių: `input`, `filter` ir `output` (29 pav.). Naudojamas „logstash-input-twitter-2.2.0“ įskiepis, kuris jau yra įdiegtas kartu su Logstash. Dalyje `input` įrašomi Twitter užregistruotos programėlės prisijungimo raktai: `consumer_key`, `consumer_secret`, `oauth_token`, `oauth_token_secret`. Visi šie raktai yra būtini norint prisijungti prie Twitter (žiūrėti punktą 1.4.1). Naudojant Logstash būtina apibrėžti bent vieną paieškos parametrą, tai gali būti raktažodis(iai) (paramentras

[14] https://www.elastic.co/products

`keywords`) ar koordinatės (paramentras `locations`). Išeities kodo pavyzdyje 29 pav. panaudoti tokie raktažodžiai kaip „bafta“, „eebaftas“, „#bafta” ir “#eebaftas” skirti surasti Twitter žinutes susijusęs su BAFTA (angl. *British Academy Film Awards*) apdovanojimais. Komentarai prasideda grotelių simboliu “#”, išskirus raktažodžius tarp kabučių, tokius kaip „#bafta”, nes jie laikomi žymėmis. Užkomentuotose eilutėse yra papildomi parametrai tokie kaip kalba `language` ir ignoruoti pasidalintas žinutes `ignore_retweets`. Šiame pavyzdyje jie nedaro įtakos, tačiau kai kuriais atvejais jie gali praversti, tarkim jei norima iš Twitter gauti žinutes parašytas konkrečia kalba. Papildomai filtruoti duomenų srautą galima `filter` dalyje, kad išvengume per didelio kiekio nereikalingų duomenų. Šiame paprastame pavyzdyje filtras nenaudojamas. Kodo bloke `output` nurodoma konkreti Elasticsearch duomenų bazė, kurios adresas `host` yra „localhost:9200“ ir duomenų bazės lentos pavadinimas `index` yra „tweets-bafta“.

Norint gauti Twitter žinutes iš konkrečios vietoves būtina apibrėžti paiešką pagal koordinates pateikiant taškų aibę parametre `locations`, iš kurių susidarytų geometrinė figūra (30 pav. kairė). Paprasčiausias pavyzdys yra dvi stačiakampio kraštinių koordinatės formatu „I-1, P-1, I-2, P-2“, kur ilguma I-*n*, o platuma P-*n*. Pavyzdžiui „-20.0, 40.0“ ir „30.0, 67.0“ koordinates apima didžiąją dalį Europos (30 pav. dešinė). Kuo didesnė teritorija, tuo daugiau žinučių galima tikėtis.

```
1  input {
2      twitter {
3          ...
4          # Europos teritorijos koordinates
5          locations => "-20.0,40.0,30.0,67.0"
6      }
7  }
8  filter{}
9  output {... index => "tweets-eu2" } }
```

30 pav. Kairėje – žinučių gavimas nurodant dvi koordinates, dešinėje – Europos teritorijos dalis apibrėžta stačiakampiu naudojant dvi koordinates

Taigi turint „twitter-bafta.conf“ failą ir Logstash serveris paleidžiamas sekančia komandinės eilutės komanda: `"bin\logstash.bat" -f twitter-bafta.conf`. Jeigu naudojama Elasticsearch duombazė, tuomet Elasticsearch serverį reikia paleisti pirma (žiūr. Elasticsearch poskyrį), o tik poto paleisti Logstash. Serveris sustabdomas kaip ir bet koks Windows komandinės eilutės procesas, t.y. su klavišų kombinacija „Ctrl + C”.

3.6. Elasticsearch – žinučių saugojimas ir indeksavimas

Elasticsearch – tai paieškos ir indeksavimo serveris, turintis JSON (angl. *Javascript Object Notation*) dokumenų domenų bazę kaip pagrindą. Atliekant indeksavimą ir paiešką JSON dokumentai atitinka lentelės eilutės, o atributai – kaip lentelės stulpeliai. Elasticsearch nereikalauja iš anksto apsibrėžti indekso (lentelės) schemos, nes nauji atributai (stulpeliai) atrandami automatiškai. Taip pat yra REST saityno paslaugos sąsaja, kuri skirta įtraukti naujus dokumentus ir ieškoti jau esančius naudojant srities specializuotąją kalbą Query DSL (angl. *domain specific language*), kuri rašoma JSON formatu. Elasticsearch veikia realiu laiku, t.y. ji gali vienu metu saugoti dokumentus, juos indeksuoti ir tuo pat metu atsakyti į vartotojų paieškos užklausas. Taigi šis įrankis labai naudingas analizuoti tekstinius srauto duomenis realiu laiku. Didėjant duomenų kiekiui ir norint išlaikyti sistemos greitį galima išskaidyti ir dubliuoti duomenų bazę į kelis serverius ir klasterius be papildomų įrankių.

3.6.1. Diegimas ir paleidimas

Elasticsearch diegimui atsisiunčiamas „elasticsearch-2.1.1.zip“ archyvas, jis išpakuojamas, ir Windows operacinėje sistemoje paleidžiamas naudojant komandinės eilutės komandą „`bin\elasticsearch.bat`“. Serveris bus pasiekiamas per HTTP adresą „http://localhost:9200/”. Galima pakeisti prievado numerį `http.port` konfiguraciniame faile „`conf\elasticsearch.yml`“. Kiekvieno indekso (lentelės) duomenys saugomi specialiuose kataloguose adresu „`elasticsearch\nodes\0\indices`“.

3.6.2. Paieška GET metodu

Elasticsearch turi labai detalizuotą paieškos sąsają kartu su DSL kalba, kurios panaudojimas išsamiai paaiškintas dokumentacijoje[15]. Paieškos ir kitos užklausos siunčiamos per HTTP protokolą į REST saityno paslaugą. GET metodo užklausos naudoja paprastą sintaksę: `/`**`<duombazė>`**`/`**`<lentelė>`**`/_search?q=`**`<atributas>:<reikšmė>`**. 31 pav. pavyzdyje esantis parametras `user.scren_name` yra `tweets-bafta` atributas (stulpelis), kuris skirtas Twitter vartotojo vardui saugoti, o `MashableNews` yra to atributo reikšmė ir šiuo atveju paieškos frazė. Elasticsearch duomenų bazė gavusi tokią užklausą grąžins visas Twitter žinutes parašytas MashableNews vartotojo.

```
GET /tweets-bafta/logs/_search?q=user.screen_name:MashablNews
```

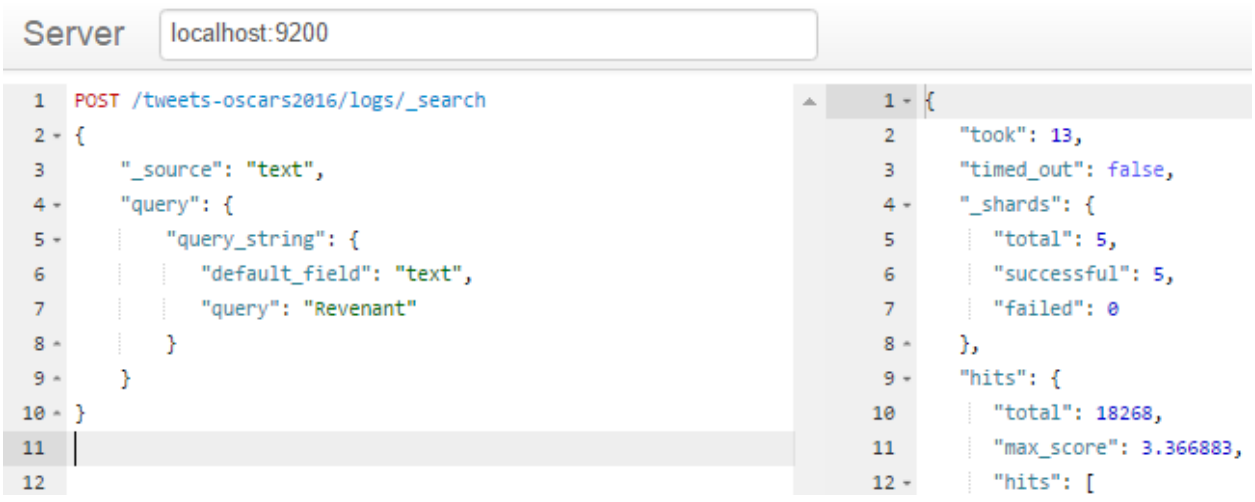
31 pav. Elasticsearch GET užklausa

[15] https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

3.6.3. Paieška POST metodu

POST metodas siunčia sudėtingesnę užklausą, kuri aprašyta JSON formatu. 32 pav. užklausos pavyzdys skirtas rasti visas Twitter žinutes esančias `tweets-bafta` duombazėje, kurios autorius „MashableNews“. Elasticsearch grąžina JSON dokumentą su užklausos rezultatais, kuriame yra `hits` objektas su 20 dokumentų su visais žinučių atributais.

Pavyzdys (32 pav.) yra užklausa skirta dokumentų paieškai naudojant reikšminius žodžius lentelėje `tweets-oscars2016`, kuriame yra naudojamas filmo pavadinimas „Revenant“. Naudojamas HTTP POST metodas norint sukurti sudėtingesnę užklausą, todėl papildomai prie užklausos yra siunčiamos duomenų objektas JSON formatu. Paieškai naudojamas objektas `query`, kuriame aprašomi tokie atributai kaip `default_field` (nurodo kuriame dokumento atribute ieškoti) ir `query` – ieškomas žodis. Jei vartotojas nori matyti tik tekstines žinutes be papildomų atributų (pvz. be `id`, `user`, `entities`, `lang` ir kitų Twitter žinutę sudarančių atributų), tuomet parametre `_source` nurodomas norimas atributas, šiuo atveju tai `text`. Tai patogu, nes gaunami tik tie duomenys, kurie reikalingi analizei, be papildomų nereikalingų duomenų.

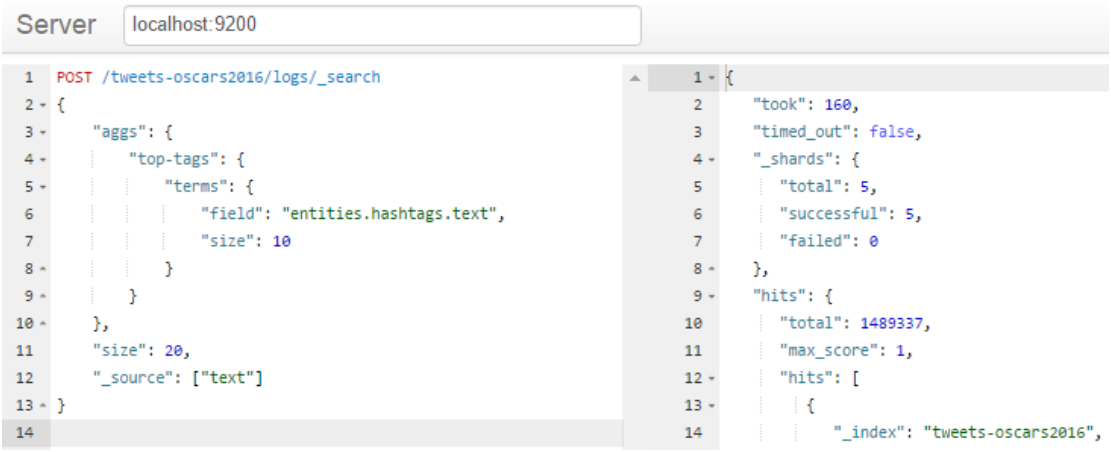


32 pav. Elasticsearch POST užklausa naudojant Chrome įskiepį „Sense“

3.6.4. Paieška naudojant surinkimo objektą

Užklausa keičiasi priklausomai nuo to, ko norima pasiekti. Sekantis pavyzdys (33 pav.) naudoja surinkimo objektą `aggs`, kuris atlieka skaičiavimus, pateikia sudėtingų santraukų statistiką ir duomenis. Ši užklausa skirta 1-am naudojimo atvejui – t.y. gauti dešimt dažniausiai pasitaikančių žymių naudojant atributą `entities.hashtags.text`. Užklausos atsakymas turi sąrašą su 10 žymių bei 20 Twitter žinučių, kurios turi tas žymes. Šie dydžiai nurodomi per atributą `size`. Naudoti `aggs` objektą naudinga, nes nebereikia patiems kurti algoritmo, kuris

įteruotų kiekvieną duombazės dokumentą ir skaičiuotų populiarias žymes, taigi sutaupomas laikas. Pilną užklausos ir atsakymo tekstą galima pamatyti 8 priedo 63 pav.



33 pav. POST užklausa su aggs surinkimo objektu – populiariausios žymės

Surinkimo objektas `aggs` taip pat buvo naudojamas 3-iam naudojimo atvejui norint sugrupuoti paieškos blokus pagal laiką, kad išeitų histograma. X ašyje data ir laikas, o Y ašyje – skaičius kiek kartų paieškos frazė buvo rasti žinutėse. Užklausa pavaizduota 34 pav. skirta tam, kad rasti paieškos frazės „Matt Damon“ pasikartojimų skaičių kas 1 valandą (`"interval":"1h"`), kur laikas didesnis ar lygus 2016-02-28 23:00:00 (milisekundėmis: `"gte":1456693200000`) ir mažesnis ar lygus 2016-02-29 09:00:00 (`"lte": 1456729200000`). Šis laikotarpis yra tuomet kai vyko Oscarų apdovanojimai ir keletas valandų po to. Laiko intervalą galima nurodyti ir minutėmis, tarkim 10min., tuomet atsakyme gausime daugiau sugrupuotų rezultatų, o tai reiškia daugiau taškų ir didesnį tikslumą histogramoje. Šiame pavyzdyje `size` atributui nustatytas nulis, nes mus domina tik agregracijos rezultatai, o ne pačios žinutės. Atributas `group_by_time_interval` yra specialiai sukurtas pavadinimas šiai užklausai ir nėra dalis elasticsearch sintaksės.

```
POST /tweets-oscars2016/logs/_search
{
    "query": {
      "filtered": {
      "query": {
        "query_string": {
          "query": "\"Matt Damon\"",
          "analyze_wildcard": true
        }
      },
      "filter": {
        "range": {
          "@timestamp": {
            "gte": 1456693200000, "lte": 1456729200000,
            "format": "epoch_millis"
          }
        }
      }
```

```
    }
  },
  "size": 0,
  "aggs": {
    "group_by_time_interval": {
      "date_histogram": {
        "field": "@timestamp",
        "interval": "1h",
        "time_zone": "Europe/Helsinki",
        "min_doc_count": 1,
        "extended_bounds": {
          "min": 1456693200000, "max": 1456729200000
        }
      }
    }
  }
}
```

34 pav. Tekstinės paieškos užklausa su surinkimu pagal laiką

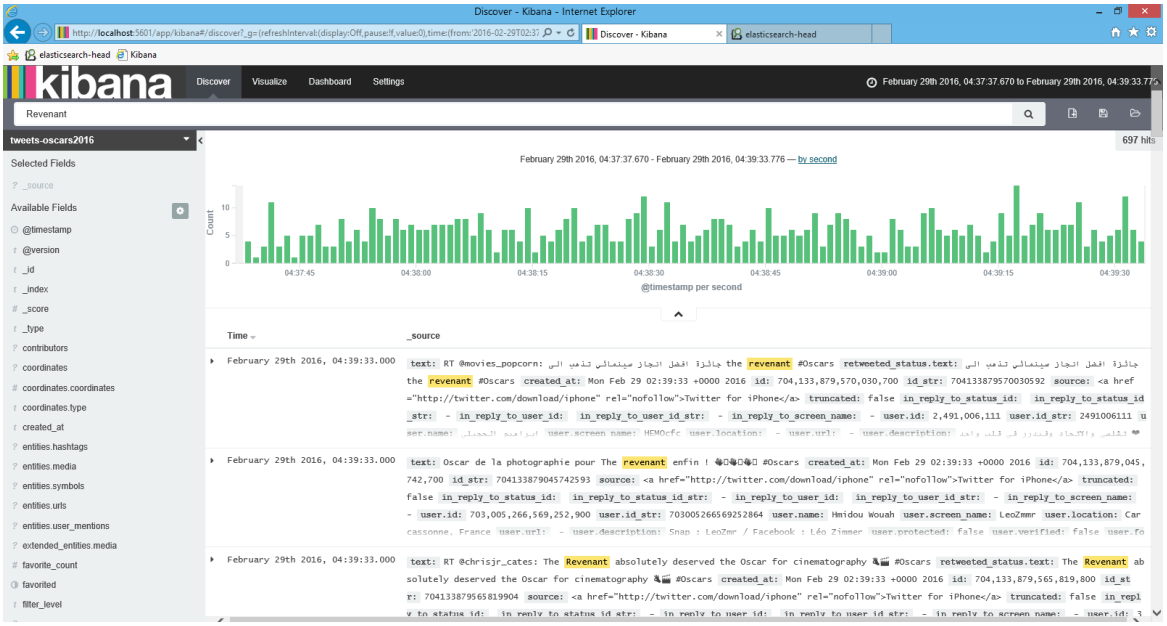
Surinkimo užklausos atsakymas atrodo maždaug taip, kaip parodyta 35 pav., kuriame turėtų būti 9 `buckets` objektai, tačiau pateikti tik 3 taupant vietą. Iš atsakymo matome, kad užklausa užėmė 7ms (atributas `took`), buvo surast 2413 dokumentai (`hits.total`) ir pagrindiniai rezultatai yra masyve pavadinimu `buckets`, kurio pilna JSON „nuoroda“ yra `aggregations.group_by_time_interval.buckets`. Per pirma valandą frazė „Matt Damon“ surandama Twitter žinutėse 518 kartų (atributas `"doc_count"`). Antrą valandą – 292 kartų. O paskutinę valandą (tarp 7 ir 8val.) – 82 kartų. Atributas key nurodo datą ir laiką milisekundžių formatu, o atributas `key_as_string` – standartiniu formatu. Pilną užklausos ir atsakymo tekstą galima pamatyti 8 priedo 64 pav.63 pav.

```
{
  "took": 7,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 2413,
    "max_score": 0,
    "hits": []
  },
  "aggregations": {
    "group_by_time_interval": {
      "buckets": [{
        "key_as_string": "2016-02-29T00:00:00.000+02:00",
        "key": 1456696800000,
        "doc_count": 518
      },
      {
        "key_as_string": "2016-02-29T01:00:00.000+02:00",
        "key": 1456700400000,
        "doc_count": 292
      },
      ...
      {
        "key_as_string": "2016-02-29T08:00:00.000+02:00",
        "key": 1456725600000,
        "doc_count": 82
      }]
    }
  }
};
```

35 pav. Elasticsearch tekstinės paieškos atsakymas į užklausa su surinkimu pagal laiką

3.7. Kibana – paieškos ir vizualizacijos vartotojo sąsaja

Kibana yra vartotojo sąsaja pasiekiama per naršyklę, kuri naudojama siekiant ypač greitai ieškoti ir peržiūrėti suindeksuotus Elasticsearch duomenų bazės dokumentus (36 pav.). Kibana padeda analizuoti duomenis įvairiais pjūviais (pagal kiekvieną JSON atributą, pagal kiekį, pagal datos intervalą ir pan.) ir vizualizuoti duomenis skirtingais būdais (plokštumine diagrama, lentele, histograma, skrituline ir stulpeline diagrama (7 priedo 62 pav.).



36 pav. Kibana paieškos puslapis, kuriame: paieškos laukas (viršus), duombazės JSON dokumentų atributai (kairėje), paieškos rezultatų vizualizacija ir patys dokumentai (vidurys)

Kibana įdiegiama atsisiunčiant ir išarchyvuojant „kibana-4.3.1-windows.zip“ archyvą. Kaip ir Elasticsearch, Kibana pasiekiama per HTTP protokolą, o tai reiškia, kad paleidus Kibana serverį galimą vartotojo sąsają atsidaryti mobiliajame ekrane. Tačiau autoriaus nuomone Kibana turi keletą trūkumų. Visų pirma, vartotojo sąsaja yra nepritaikyta mobiliajam ekranui, todėl naudojimas yra sunkesnis nei galėtų būti. Norint pasiekti kažkokį tikslą, pvz.: sukurti ar peržiūrėti vizualizaciją, vartotojui reikia nuolat priartinti ir atitolinti ekraną (angl. *zoom-in*, *zoom-out*), kad pataikyti ant įvesties lauko, mygtukų ar išsiskleidžiantį meniu (angl. *dropdown*). Antra, šiame darbe norima sukurti vizualizaciją iš nuotraukų, tačiau tokios galimybės Kibana įrankyje nėra. Kibana įrankis naudingas darant tyrimus, norint geriau suprasti turimos Elasticsearch dokumentų duomenų bazės turinį, ir būtent taip buvo naudojamas.

3.8. Saityno paslauga komunikacijai tarp Elasticsearch ir TweetsViz

3.8.1. CORS blokavimo problema ir sprendimas

Kuriant prototipą laiko taupymo tikslais komunikacijai su Elasticsearch buvo naudojamos Angular AJAX užklausos tiesiai iš Javascript kodo. Tačiau naršyklės komandinėje eilutėje buvo gaunama štai tokia klaida (37 pav.).:

```
"Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote
resource at http://localhost:9200/tweets-bafta/logs/AVLhiV9V0SqlBzByfVQl. (Reason:
CORS header 'Access-Control-Allow-Origin' missing)."
```

37 pav. Javascript CORS klaidos pranešimas

Elasticsearch serveris blokuoja REST saityno paslaugos užklausas iš Javascript kodo, nebent tik tais atvejais, kai adresas, iš kurio išsiunčiama užklausa, yra iš anksto užregistruotas „elasticsearch.yml" konfiguracijos faile naudojant sekančius parametrus:

```
http.cors.enabled: true
http.cors.allow-origin: http://localhost:8000
http.cors.allow-credentials: true
```

38 pav. Konfigūracijos failo "elasticsearch.yml“ fragmentas

Čia užregistruotas adresas `http://localholsts:8000` yra papratas HTTP serveris. Jame yra Javascript failas „app.js", iš kurio siunčiamos REST užklausos. Toks būdas tinka paprastam, lokaliame tinkle esančiam, prototipui naudojamam per naršyklę, kai Elasticsearch ir programėlės serveris yra tam pačiam kompiuteryje. Tačiau norint panaudoti vizualizacijos programėlę mobiliajame įrenginyje platesniame tinkle reikalingas atskiras HTTP serveris kartu su atskira saityno paslauga, kuri tarpininkaus tarp Elasticsearch ir TweetsViz programėlės. Tokiu būdu priėjimas nebus apribotas tik vienam vartotojui.

3.8.2. TweetsWS – saityno paslauga

Norint naudotis vizualizavimo sistema ne tik lokaliame tinkle buvo sukurta TweetsWS paslauga. Tai tarpinė komunikacijos paslauga tarp Elasticsearch duomenų bazės ir TweetsViz tinklalapio. Šios paslaugos prototipe buvo įgyvendintos dvi funkcijos: `getTop10Hashtags()` ir `sendDSLQuery()`. Įgyvendinimui buvo naudojamas REST saityno paslaugos biblioteka Jersey (versija 1.14) skirta Java programavimo kalbai.

8 lentelė TweetsWS saityno paslaugos `/top10hastags` užklausos dokumentacija

Paslaugos pavadinimas	**Elasticsearch 10 populiariausių žymių užklausa**
Aprašymas	Paprasta užklausa, kuri pagal nurodytą Elasticsearch duomenų bazę užklausia 10 populiariausių tos Twitter žinučių žymių. Ši užklausa skirta NA1 vizualizacijos įgyvendinimui.
URL	**/top10-hashtags/:databaseName** :databaseName – tai duomenų bazės pavadinimas.
HTTP metodas	GET
URL parametrai	**Būtini:** databaseName=[String] pvz.: databaseName = “tweets-bafta”.
Duomenų parametras	Nėra
Sėkmės atveju	HTTP kodas: **200** Duomenys: `{ "code": 200, "message": "OK"}` JSON dokumentas su duomenimis.
Nesėkmės atveju	HTTP kodas: **404** Duomenys: `{ "code": 404, "message": "Database not found" }`
Užklausos pavyzdys	`http://localhost:8080/TweetsWS/top10-hashtags/tweets-bafta`
Užklausos atsakymo pavyzdys	JSON objektas: `{"took": 19, "timed_out": false, "_shards": {"total": 5, "successful": 5, "failed": 0}, "hits": {"total": 134690, "max_score": 0.0, "hits": []}, "aggregations": {"top-tags": {"doc_count_error_upper_bound": 170, "sum_other_doc_count": 40183, "buckets": [{"key": "eebaftas", "doc_count": 44846}, {"key": "bafta", "doc_count": 31756}, {"key": "baftas", "doc_count": 6044}, {"key": "bafta2016", "doc_count": 4221}, {"key": "therevenant", "doc_count": 2604}, {"key": "dakotajohnson", "doc_count": 1446}, {"key": "leonardodicaprio", "doc_count": 1301}, {"key": "katewinslet", "doc_count": 788}, {"key": "dicaprio", "doc_count": 741}, {"key": "elrenacido", "doc_count": 737}]}}}`

9 lentelė. TweetsWS saityno paslaugos `/DSLquery` užklausos dokumentacija

Paslaugos pavadinimas	**Elasticsearch paieškos užklausa**
Aprašymas	Ši užklausa skirta sukurti Elasticsearch duomenų bazės užklausą naudojant DSL kalbą. Pati TweetsWS paslauga neinterpretuoja duomenų objekto, o tik perduodą ją į Elasticsearch. Vienintelis parametras kuris yra tikrinamas yra duomenų bazės pavadinimas, nes be jo negalima sėkmingai sukurti užklausos į Elasticsearch.
URL	**/DSLquery/:databaseName** :databaseName – tai duomenų bazės pavadinimas.
HTTP metodas	POST
URL parametrai	**Būtini:** databaseName=[String] pvz.: databaseName = “tweets-oscars2016”.
Duomenų parametras	**Struktūra:** JSON objektas. Naudojama ta pati sintaksė kaip ir Elasticsearch užklausose pagal Elasticsearch DSL. **Pavyzdys:** `{"_source": ["text"], "aggs": {"top_tags": {"terms": {"field": "entities.hashtags.text", "size": 5}}}}`
Sėkmės atveju	HTTP kodas: **200** Duomenys: JSON dokumentas su tekstiniais ir skaitiniais duomenimis.
Nesėkmės atveju	HTTP kodas: **404** Duomenys: `{"code": 404, "message": "Database not found"}`
Užklausos pavyzdys	`http://localhost:8080/TweetsWS/DSLquery/tweets-oscars2016` `{"_source": ["text"], "aggs": {"top_tags": {"terms": {"field": "entities.hashtags.text", "size": 5}}}}`
Užklausos atsakymo pavyzdys	JSON objektas: `{ "took": 87, "timed_out": false, "_shards": { "total": 5, "successful": 5, "failed": 0 }, "hits": { "total": 1489337, "max_score": 0, "hits": [] }, "aggregations": { "top_tags": { "doc_count_error_upper_bound": 3162, "sum_other_doc_count": 506609, "buckets": [{ "key": "oscars", "doc_count": 1157813 }, { "key": "oscars2016", "doc_count": 86995 }, { "key": "therevenant", "doc_count": 32746 }, { "key": "oscarssowhite", "doc_count": 23382 }, { "key": "leonardodicaprio", "doc_count": 17195 }] } } }`

3.9. *TweetsViz – kuriamo vizualizacijos tinklalapio prototipas*

TweetsViz tai tinklalapio prototipas su prisitaikančiu dizainu (tinka ir mobiliojo įrenginio ir personalinio kompiuterio ekranui), kuris buvo suprogramuotas naudojant Angular, Bootstrap, D3, Highcharts ir jQuery Javascript karkasus.

3.9.1. AngularJS – Javascript karkasas

AngularJS[16] (dar vadinamas Angular ar Angular.js) tai Javascript karkasas skirtas kurti vieno puslapio tinklalapius (angl. *SPA – single page application*). Pagrindinė šio karkaso ypatybė yra dvišalis duomenų suršimas (angl. *two-way data binding*). Angular paslauga `$scope` pastebi pokyčius modulyje (angl. *model*) ir automatiškai padaro atnaujinimus vartotojo sąsajos sluoksnyje (angl. *View layer*) t.y. HTML. Ir atvirkščiai, kai pasikeičia vartotojo sąsajos reikšmės, tuomet atnaujinami modulio kintamieji. Taigi nereikia atskirai programuoti HTML DOM objektų pokyčių, tarkim naudojant jQuery karkasą. Šiame projekte Angular naudojamas, nes tai pagreitina programavimą ir padaro vartotojo sąsają reaktyvia.

3.9.2. AngularJS direktyvos kūrimas

Kad naudoti Angular kartu su diagramų karkasu D3, reikia sukurti specialią direktyvą (angl. *directive*), specialų HTML elementą su atributais ir suprogramuoti kaip jis turi veikti. Sprendimui sukurti buvo vadovaujamasi šiais darbais: [Hil13] [Men13].

```
app.directive('barsChart', function () {
    var directiveDefinitionObject = {
        restrict: 'E',
        replace: false,
        link: function (scope, element, attrs) {
            /* kodas */
        },
        scope: {
            chartDataArray: '=chartData', numberOfChartBars: '=numberOfBars',
            canvasWidth: '=width', isResponsive : '='
        }
    };
    return directiveDefinitionObject;
});
```

39 pav. Angular kodas skirtas specialiam HTML elementui sukurti

Pateikta kodo ištrauka (39 pav.) su direktyva pavadinimu `barsChart`. Direktyvos įgyvendinimo bevardė funkcija gražina Javascript objektą `directiveDefinitionObject`, kuriame yra 4 parametrai:

[16] https://angularjs.org/

1. Parametras `restrict` (39 pav. 3 eilutė) pažymi, kad direktyva galioja HTML elementui su specialiu pavadinimu `<bars-chart>` (40 pav.), o ne `<div>` elementui, todel žymima raide `'E'`.
2. Parametras `replace` su reikšme `false` skirtas pažymėti, kad nenorima perrašyti direktyvos deklaraciją HTML kode.
3. Parametras `link` aprašo bevardžią funkciją, kurioje programuojama didžiausia dalis direktyvos funkcionalumo. Šioje funkcijoje galima aprašyti kintamuosius ir papildomas funkcijas. Bevardė funkcija turi tris kintamuosius `scope`, `element` ir `attrs`.
    a. Kintamasis `scope` turi `<bars-chart>` atributų reiškmes, pvz.: `scope.chartDataArray` yra masyvas iš Angular modulio su duomenimis skirtiems diagramai nubrėžti.
    b. Kintamasis `element` veikia kaip HTML elementų nuoroda, tarkim `element[0]` yra `<bars-chart>` objektas kartu su papildomais kintamaisiais, pvz.: `element[0].attributes[3]` grąžins atributą su reikšme: `chart-data="responseDataArray"`. Tas naudinga norint pridėti `<svg>` ir `<rect>` HTML elementus piešiant su D3.js.
    c. Kintamasis `attrs` turi direktyvos atributų masyvą.
4. Paskutinis parametras `scope` skirtas sukurti Javascript kintamuosius kiekvienam iš `<bars-chart>` atributų, pvz.: HTML atributas `chart-data` Javascript kode konvertuojamas į „camelCase“ formatą `chartData` ir priskiriamas į Javascript `scope` kintamąjį `chartDataArray` (39 pav. 9 eilutė). Toks kintamasis bus pasiekiamas kaip `scope.chartDataArray` bevardėje funkcijoje. Naudojant tik lygybės ženklą tintamasis turi turėti tokį pat pavadinimą kaip ir atributas tik „camelCase“ formatu, pvz.: `is-responsive` (40 pav. 46 eilutė) atributas „camelCase“ formatu tampa `isResponsive` (39 pav. 10 eilutė) ir todėl priskiriamas `isResponsive` kintamąjam tik su lygybės ženklu, o ne `isResponsive: '=isResponsive'`.

```
43      <div class="row">
44          <div id="chart" class="col-xs-11">
45              <bars-chart chart-data="responseDataArray" number-of-bars="numberOfBars"
46                  width="canvas_width" is-responsive="isResponsive"></bars-chart>
47          </div>
48      </div>
```

40 pav. Sukurto <bars-chart> HTML elemento naudojimas

3.9.3. Kintamųjų reikšmių kitimo stebėjimas

Angular turi specialią funkciją `$watch()`, kuri skirta stebėti kintamajam ir kai kintamasis pasikeičia nurodyta ar bevardė funkcija kviečiama. 41 pav. pavyzdyje stebimas `selectedDatabase` kintamasis, todėl kai vartotojas pasirenka kitą „dropdown“ meniu reikšmę bus iškviečiama bevardė funkcija, kurioje vykdoma funkcija pavadinimu `$scope.callElasticsearchService()`.

```
$scope.$watch('selectedDatabase', function(){
    $scope.callElasticsearchService();
});
$scope.$watchCollection('checkBoxArr', function (newCollection, oldCollection) {
    /* išvesties kodas skirtas perpiešti diagramą */
});
```

41 pav. Funkcijos $watch panaudojimo pavyzdys

Funkcijos `$watch()` privalumas tame, kad nereikia patiems programuoti įvykių valdymo funkcionalumo su Javascript funkcija `element.addEventListener()`. Taip pat Angular padaro DOM kintamųjų atnaujinimą be naršyklės perkrovimo, todėl vartotojas turės patogesnę valdymo programėlės patirtį. Funkcija `$watchCollection` skirta tai pačiam tikslui, bet čia stebimas ne vienas kintamasis, o jų rinkinys, t.y. masyvas. Ši funkcija panaudota stebėti kada vartotojas nuima varneles. Tarkime yra 5-ios paieškos frazės su varnelėm ir vartotojas nuima varneles nuo 3-ų, tuomet diagrama bus perpiešta tik su 2 paieškos frazių rezultatais. Bevardėje funkcijoje galima turėti kintamuosius `newValue` ir `oldValue`, bet tai nėra būtina.

3.9.4. Asinchroninių užklausų formulavimas

Kad kviesti TweetsWS saityno paslaugą buvo sukurta funkcija `callESOnce()` (42 pav.). Asinchroninė Ajax tipo užklausa atliekama naudojant Angular `$http.post()` funkciją, joje perduodamas JSON objektas su užklausos parametrais, kurių pavyzdžiai aprašyti elasticsearch dalyje. Ajax užklausa yra asinchroninė ir neblokuoja likusio kodo vykdymo, todėl reikia nurodyti kas atsitiks kai užklausa bus sėkmingai įvykdyti (funkcija `successCallback`) ir nesėkmingai (funkcija `errorCallback`).

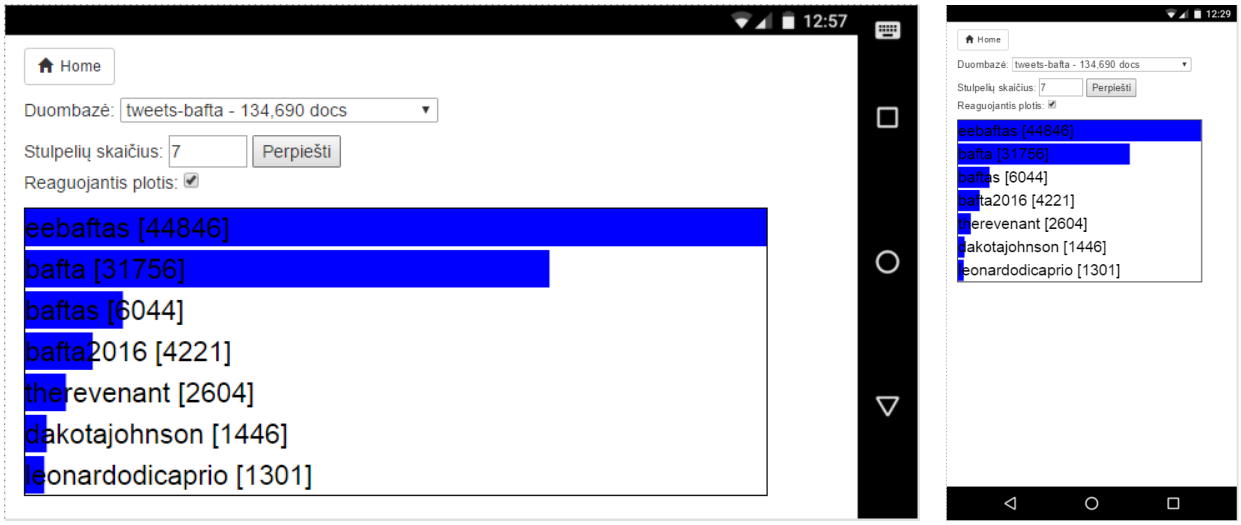
```
function callESOnce(request_data_str, index) {
    return $http.post(url, request_data_str)
        .then(function successCallback(response1) {
            return response1;
        }, function errorCallback(response2) {
            console.log("callESOnce request error for index: " + index);
            console.log(response2);
            return response2;
        };
}
```

42 pav. Asinchroninės ajax užklausos funkcija `callESOnce()`

3.10. Vizualizavimo karkasas - D3

D3.js[17] (toliau D3) – tai Javascript karkasas skirtas kurti dinamiškas ir interaktyvias duomenų vizualizacijas naršyklėje. D3 pagristas HTML5, CSS ir SVG (angl. *scalable vector graphics*). Karkasas skiriasi nuo kitų diagramas piešiančių karkasų (pvz.: Highcharts) tuom, kad programuotojui suteikiama didelė galimybė pritaikyti ir keisti kiekviena vizualizacijos detalę. Tačiau to pasekoje norint sukurti paprasčiausią histogramą prireikia nemažai eilučių kodo. Visgi šis karkasas tinka kurti išskirtinėms ir precedento neturinčioms vizualizacijoms.

43 pav. pateikta vizualizacija, kuri įgyvendina pirmąjį naudojimo atvejį (NA1S1). Vartotojas pasirenka duomenų bazę iš meniu ir diagrama perpiešiama. Galima padidinti stulpelių skaičių iki 10. Interaktyvumas pasireiškia kai ekrano rezoliucija pasikeičia, tuomet diagramos plotis padidėja arba sumažėja atitinkamai, kad išnaudoti 11-ika 12-ųjų erdvės. Likusi 1-a 12-toji dalis skirta pirštui, kad galima būtų slankioti ekraną aukštyn ir žemyn.



43 pav. Naudojimo atvejis 1: kairėje ekranas gulsčias, o dešinėje – stačias

D3 diagramos piešimą galima suskirstyti į tris pagrindines dalis (65 pav.):

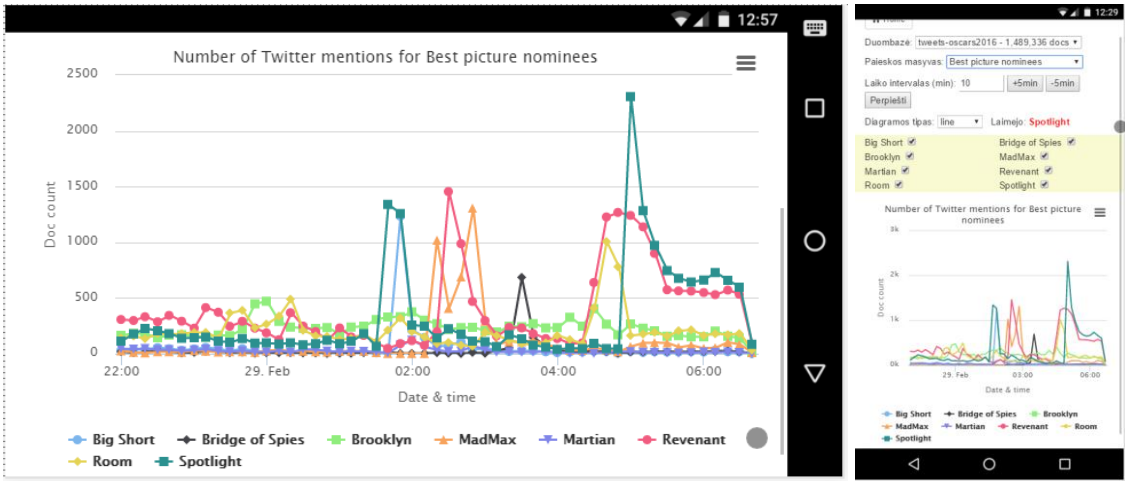
1. Paruošiami duomenys (dažniausiai skaičių ir teksto masyvai) pagal konkretų diagramos tipą, pvz.: stulpelinė diagrama (66 pav.).
2. Nupiešiamas `<svg>` elementas su konkrečiu pločiu, aukščiu ir riba (67 pav.).
3. Į paruoštą `<svg>` elementą pridedamas kiekvienas diagramos stulpelis (el. `<rect>`) su tekstu (el. `<text>`) naudojant D3 komandą `append` (68 pav.). Pridedant kiekvieną stulpelį būtina nurodyti konkrečias koordinates kur stulpelis turi prasidėti, koks jo ilgis, aukštis ir koks tarpas po juo. Pradžios koordinatės nurodamos ir tekstui.

[17] https://d3js.org/

D3 reikalauja kruopštumo ir apskaičiavimo kiekvienai detalei, todėl reikėjo sukurti daug įvairių kintamųjų ir atlikti apskaičiavimų, pvz.: atskiri kintamieji stsulepiams ir svg elementui, svg elemento aukštis priklausomai nuo stulpelio skaičiaus, svg elemento kintamųjų perskaičiavimas, kai pasikeičia rezoliucija ir pan. Taigi D3 naudojimas reikalauja daugiau pastangų palyginus su Highchart. Jį geriausia naudoti išskirtinėms vizualizacijoms.

3.11. Diagramų karkasas - Highcharts

Dar vienas Javascript karkasas, kuris buvo panaudotas yra Highcharts[18]. Jis taip pat leidžia kurti dinamiškas ir interaktyvias duomenų vizualizacijas naršyklėje. Šis karkasas skiriasi nuo D3, nes su juo lengviau sukurti paprastas diagramas ir sprendimo kodas užima mažiau eilučių. Karkaso oficialiame puslapyje pateikta po keletą pavyzdžių (su kodu fragmentais) kaip naudoti Highcharts norint sukurti standartines diagramas ir netgi jų kombinacijoms (pvz.: histogramos ir stulpelinės diagramos kombinacija).



44 pav. NA3 įgyvendinimas: kairėje ekranas gulsčias, o dešinėje –stačias

44 pav. pateikta vizualizacija, kuri įgyvendina trečiąjį naudojimo atvejį (NA3) apie Oskarų apdovanojimus. Vartotojas pasirenka duomenų bazę iš meniu ir histograma perpiešiama. Vartotojas taip pat gali pasirinkti konkretų paieškos masyvą, pvz.: geriausi aktoriai, arba geriausias filmas, ir diagrama vėl perpiešiama atitinkamai. Vartotojas gali nuimti varneles nuo konkrečių paieškos žodžių ir tuomet tos linijos nebus brėžiamos, pvz.: galima palyginti tik dviejų filmų populiarumą.

Highcharts diagramos piešimą galima suskirstyti į dvi pagrindines dalis (69 pav.):

1. Paruošiami duomenys (skaičių ir teksto masyvai) pagal konkretų diagramos tipą, pvz.: linijinė diagrama (70 pav.). Čia būtina pateikti duomenis (kintamasis `output` atitiks

[18] http://www.highcharts.com/

kintamąjam `series`) remiantis specialia struktūra: `series = { [name: "frazė", data: ["data_laikas", skaičius] ] }`, kur objektas turi masyvą iš dviejų atributų `name` ir `data`. Atributas frazė yra paieškos žodis, o atributas data turi masyvą iš dviejų kintamųjų `data_laikas` ir `skaičius` (71 pav.). Skaičiaus reikšmė atitinka kiek frazė pasikartojo per konkretų laiką, pvz.: valandos bėgyje nuo `data_laikas1 = "2016-02-29T00:00:00"` iki `data_laikas2 = "2016-02-29T01:00:00"`. Su esamais naudojimo atvejais galima tikėtis, kad kiekviena paieškos frazė turės apie 20-40 `data` masyvų su `data_laikas` ir `skaičių` poromis. Kuo daugiau laiko atkarpų, tuo tikslesnė diagrama.

2. Sukuriamas naujas Highchart.chart objektas, kuris suranda konkretų HTML elementą pagal `id` atributą, pvz.: `<div id="chart2" …/>` ir toje vietoje sugeneruoja diagramą pagal parametrus. Ne visus parametrus būtina nurodyti, tačiau svarbiausi yra: diagramos tipas, pavadinimas, X ir Y ašių duomenų tipai ir patys duomenys (71 pav.).

Taigi Highcharts naudojimas reikalauja mažiau pastangų palyginus su D3. Pagal nutylėjimą Highcharts turi įdiegtą papildomą funkcionalumą, kuris pravertė šiame darbe: a) prisitaikantis diagramos plotis ir ilgis, b) konkrečios linijos nerodymas paspaudžiant ant linijos pavadinimo apačioje esančiame aprašė (angl. *legend*).

3.12. Bootstrap CSS stilių karkasas

Twitter Bootstrap [19] yra atviro kodo CSS (HTML stiliaus failas) karkasas, t.y. standartizuotas stilių, klasių ir elementų rinkinys skirtas palengvinti ir paspartinti web puslapių kūrimą. Bootstrap susideda iš CSS, Javascript ir šrifto failų, o taip pat turi HTML šablonų. Šiame darbe buvo naudojama Bootstrap tinklelio sistema (angl. *grid*), kuri padalina puslapį ir elementus į 12 lygių dalių. Tai palengvina sudėlioti elementus į reikiamas vietas ir suteikti jiems norimą dydį.

Kadangi vienas šio darbo tikslų kurti vizualizaciją, kuri tiktų ir mobiliajam įrenginiui, todėl buvo atsižvelgta į skirtingas rezoliucijas. Projektuojamo puslapio išmatavimai keisis priklausomai nuo ekrano dydžio (mobilusis, planšetė, asmeninio kompiuterio monitorius ir pan.), todėl naudojamos specialios Bootstrap CSS klasės stulpeliams: `.col-xs-N` ir `.col-sm-N`, kur N yra skaičius nuo 1-o iki 12-kos. Kuo didesnis skaičius, tuo platesnis stulpelis, taigi 12-to dydžio stulpelis užima visą ekrano plotį. Pirma `.col-xs-N` klasė skirta labai mažiems ekranams (angl. ***e**xtra **s**mall devices*) su mažiau nei 768 pikselių pločio, o `.col-sm-N` klasė mažiems ekranams (angl. ***sm**all devices*) skirta planšetėms ir ekranamas nuo 768 iki 992 piskelių

[19] http://getbootstrap.com/css/

pločio. Labai naudinga yra tai, kad galima naudoti kelias klases vienu metu ir ekrano elementai pakeičia plotį priklausomai nuo ekrano dydžio. Tas aktualu tokais atvejais, kai vartotojas pakeičia mobiliojo įrenginio padėtį pasukdamas 90 laipsnių ir ekrano plotis padidėja. Taigi naudojant Bootstrap buvo sutaupytas laikas tinklalapio vartotojo sąsajos struktūrai programuoti.

REZULTATAI IR IŠVADOS

Šio darbo tikslas buvo sukurti Twitter srauto duomenų vizualizavimo sistemą mobiliam įrenginiui skirtą pavaizduoti duomenų tematikų apibendrinimą pagal raktinius žodžius, efektyvinti informacijos paiešką bei analizę ir padėtų išskirti tendencijas. Sprendimas turi veikti tiek asmeniniame kompiuteryje, tiek mobiliuose įrenginiuose.

Šio darbo tikslas buvo įgyvendintas ir pateikiami sekantys rezultatai:

1. Atlikta esamų vizualizavimo būdų literatūros analizė ir palyginimas, aprašytas vizualizavimo procesas.
2. Atliktas Twitter srauto duomenų paslaugos tyrimas ir įrankių palyginimas. Pasirinktas labiausiai tinkamas įrankis – Logstash.
3. Sukurti vizualizavimo sistemos naudojimo atvejai ir aprašyti naudojami algoritmai.
4. Sukurti trys galimi sprendimai. Atliktas palyginimas ir pasirinkta, bei vėliau detaliau aprašyta, taikomosios programos sistema su tinklalapiu ir tarpiniu serveriu.
5. Sukurtas sistemos prototipas, kuris gali vizualizuoti Twitter srauto duomenis naudojant stulpelinę ir linijinę diagramas realiuoju laiku (6 priedas).
6. Sukurta prototipo vartotojo sąsaja su prisitaikančiu dizainu, todėl sprendimas veikia tiek ant asmeninio kompiuterio, tiek ir ant mobilaus įrenginio, nes prisitaiko prie skirtingų rezoliucijų (6 priedo, 53 pav., 54 pav., 55 pav.).

Šio darbo išvados:

1. Darbo pradžioje literatūros analizės metu dėmesys buvo skirtas labiau į vizualizavimo būdus. Buvo norima surasti sudėtingesnį vizualizavimo būdą, kuris išsiskirtų iš kitų. Tačiau, kuriant sistemos architektūrą prireikė papildomo tyrimo įrankiams surasti, kurie galėtų palengvinti žinučių surinkimą ir paiešką. Galiausiai buvo nuspręsta, kad darbo rezultatas nėra vizualizavimo algoritmas, o visos vizualizavimo sistemos modelis – architektūra.
2. Sukurti vizualizavimo sistemą vien tik mobiliajame įrenginyje be tarpinio serverio įmanoma, tačiau yra techniškai sudėtinga. Todėl šio darbo sprendimas naudojo tarpinį skaičiavimų serverį kartu jau sukurtais įrankiais Logstash ir Elasticsearch.
3. Darbo pradžioje buvo norima darbą pritaikyti tik mobiliesiems įrenginiams. Tačiau vėliau nuspręsta pasirinkti tokį sprendimą, kuris tiktų ir mobiliems įrenginiams ir asmeniniams kompiuteriams, t.y. tinklalapio programa pasiekiama per naršyklę.
4. Twitter vizualizavimo įrankio kūrimas yra techniškai sudėtingas uždavinys. Pradedant tiriamąjį darbą nebuvo suprasta, kad nėra įrankių skirtų patogiai analizuoti ir kurti Twitter žinučių vizualizavimus pagal norimus parametrus. Todėl reikėjo sukurti tokį

įrankį patiems kombinuojant esamas atviro kodo technologijas. Sukurtas prototipas skirtas analizuoti Twitter žinučių srauto tendencijas. Tyrėjas nustato temą ir paleidžia žinučių paieškos procesą, sukurtas vizualizavimo įrankis rodo proceso būseną.

Tolimesni darbai galėtų būti tokie:

1. Siūloma pilnai įgyvendinti vizualizavimo sistemą pagal sukurtus naudojimo atvejus ir naudojant tokius vizualizavimo būdus: medžio pavidalo žemėlapis, skritulinė diagrama, žemėlapis.
2. Siūloma atlikti tyrimus norint sužinoti kaip greitai sistema veikia, kokius vartotojo kiekius ir žinučių srautus gali atlaikyti.
3. Siūloma atlikti apklausą norint įvertinti vizualizavimo sistemos patogumą, naudingumą.

ŠALTINIAI

[ACZ+11] J. Alsakran, *et al*. *STREAMIT: Dynamic visualization and interactive exploration of text streams*. Pacific Visualization Symposium (PacificVis), IEEE, 2011, psl. 131-138.

[Amb08] A. Ambrazevičiūtė. *Analizės modelio kūrimas ir modeliavimas*. Vilniaus Universitetas, 2008.
[žiūrėta 2016-02-13]. Prieiga per internetą:
< www.mif.vu.lt/~ragaisis/PSI_mag2008/StudMedziaga/Modeliavimas_analizeje-7gr_AisteAmbrazeviciute.doc>

[BFJ+13] J. Bradley, N. Fung, I. Julien, M. Malu, M. Mauriello. *ViralViz: Visualizing Temporal Content Flow in Social Networks*. University of Maryland. October, 2013.

[BHW00] M. Bruls, K. Huizing, J.J. van Wijk. *Squarified Treemaps*. In: W. de Leeuw, R. van Liere (eds.), Data Visualization 2000, Proceedings of the joint Eurographics and IEEE TCVG Symposium on Visualization, 2000, Springer, Vienna, p. 33-42.

[BKK+12] S. Bhulai, *et al*. *Trend Visualization on Twitter: What's Hot and What's Not?*. In: DATA ANALYTICS 2012, The First International Conference on Data Analytics, 2012, pp. 43-48.

[DGJ14] V. Dagienė, G. Grigas, T. Jevsikova. *Enciklopedinis kompiuterijos žodynas.* 4-as leidimas. Vilnius: VU MII, 2014.

[DKŽ08] G. Dzemyda, O. Kurasova ir J. Žilinskas. *Daugiamačių duomenų vizualizavimo metodai*. MII, Mokslo Aidai, Vilnius, 2008, 206 psl.

[GKG+10] M.M. Gaber, S. Krishnaswamy, B. Gillick, N. Nicoloudis, J. Liono, H. AlTaiar, A. Zaslavsky. *Adaptive Clutter-Aware Visualization for Mobile Data Stream Mining*, Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI vol. 2, IEEE Computer Society, 2010, psl. 304-311.

[Gup15] Yuvraj Gupta. *Kibana Essentials*. *Chapter 6: Real-Time Twitter Data Analysis*. Packt Publishing Ltd, 2015, psl. 145-167.

[Hil13] G. Hilkert. *D3 on AngularJS*. Ng-newsletter, September, 2013.
[žiūrėta 2016-04-17]. Prieiga per internetą:
<http://www.ng-newsletter.com/posts/d3-on-angular.html>

[IDC13b] IDC, *Android Pushes Past 80% Market Share While Windows Phone Shipments Leap 156.0% Year Over Year in the Third Quarter, According to IDC*. International Data Corporation, November 12, 2013.
[žiūrėta 2013-12-08]. Prieiga per internetą:
<http://www.idc.com/getdoc.jsp?containerId=prUS24442013>

[JS91] B. Johnson and B. Shneiderman. *Treemaps: a space-filling approach to the visualization of hierarchical information structures*. In: Proc. of the 2nd International IEEE Visualization Conference, pages 284–291, October 1991.

[LE11] A. Leimuller, M. Ebner. Treemap Visualization of the Semantic Twitter Analysis Tool. Graz University of Technology, August, 2011, 32 psl.

[Men13] E. Mendoza. *Proper use of D3.js with Angular directives*. September 27, 2013.
[žiūrėta 2016-04-17]. Prieiga per internetą:
<http://odiseo.net/angularjs/proper-use-of-d3-js-with-angular-directives>

[MKL+13] F. Morstatter, S. Kumar, H. Liu, R. Maciejewski. *Understanding Twitter Data with TweetXplorer*. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, psl. 1482–1485.

[Mou14] A. Moujahid. *An Introduction to Text Mining using Twitter Streaming API and Python*. July, 2014.
[žiūrėta 2014-12-31]. Prieiga per internetą:
<http://adilmoujahid.com/posts/2014/07/twitter-analytics/>

[PCB+11] M. Panka, M. Chlebiej, K. Benedyczak, P. Bala. Visualization of Multidimensional Data on distributed mobile devices using interactive video streaming techniques, Proceedings of the 34th International Convention, MIPRO, 2011, psl. 246-251.

[RLC+13] G. C. Rotta, V. S. de Lemos, A. L. M. da Cunha, I. H. Manssour, M. S. Silveira, A. F. Pase. *Exploring Twitter Interactions through Visualization Techniques: Users Impressions and New Possibilities*. Knygoje: Human-Computer Interaction–INTERACT 2013, Springer Berlin Heidelberg, 2013, psl .700-707.

[ROR13] O. de Rooij, D. Odijk, M. de Rijke. *ThemeStreams: Visualizing the Stream of Themes Discussed in Politics*. Proceedings of SIGIR'13: 36th International ACM SIGIR conference on Research and development in information retrieval, vol. 13, ACM, 2013, psl. 1-2.

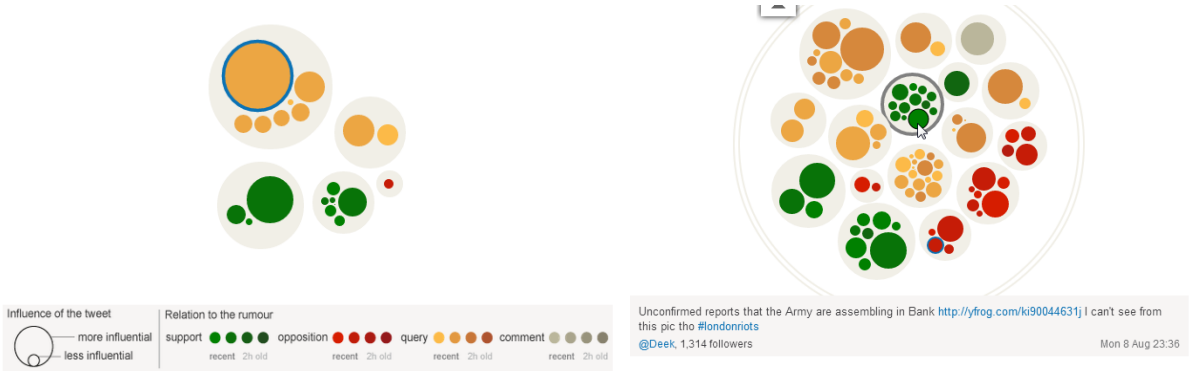
[SM08] Z. Shen, K.L. Ma. *MobiVis: A Visualization System for Exploring Mobile Data*. Proceedings of IEEE Pacific Visualisation Symposium 2008, IEEE Computer Society, 2008, psl 175-182.

[ZK13] P. Zadrozny, R. Kodali. *Chapter 12: Analysing Tweets*. Knygoje: Big Data Analytics Using Splunk: Deriving Operational Intelligence from Social Media, Machine Data, Existing Data Warehouses, and Other Real-Time Streaming Sources. Apress, 2013, psl. 211-229.

PRIEDAI

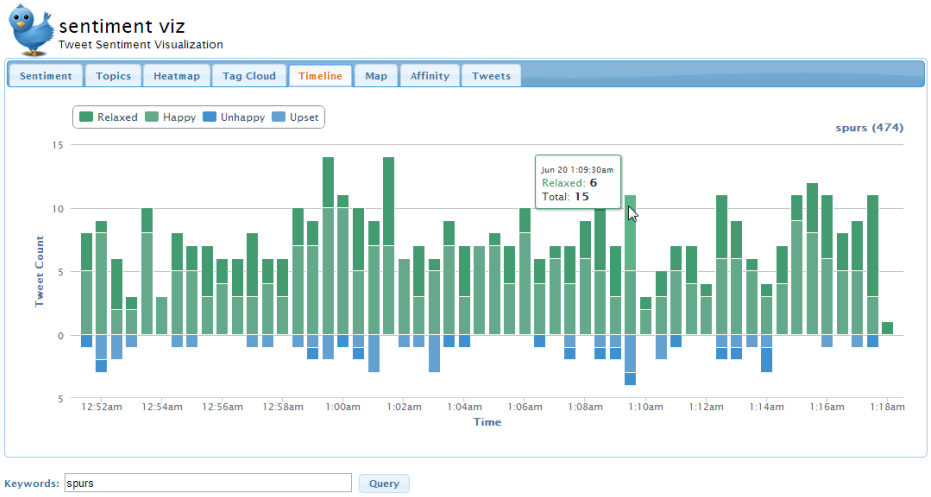
1 priedas. Twitter duomenų vizualizavimo pavyzdžiai

Traukos taškais paremtas grafas:



45 pav. Riaušių Londone Twitter žinučių analizė[20]. Žalia – palaikymas, raudona – nepritarimas, geltona – užklausa, pilka – komentaras

Sukrautų stulpelių diagrama (angl. *Stacked column diagram*):



46 pav. *Sentiment viz* Twitter žinučių vizualizacija sukrautų stulpelių diagrama [HR13]

[20] http://www.theguardian.com/uk/interactive/2011/dec/07/london-riots-twitter

2 priedas. Twitter žinučių vizualizavimas žemėlapyje

47 pav. Pavaizduotos vietovės iš kurių buvo išsiųstos Twitter žinutės apdorojus 3 minučių srautą (2015-01-12, 22:28-22:31) naudojant geojson.io[21]

3 priedas. Pilnas Twitter žinutės JSON failas

Twitter serveris pilną žinutę su meta duomenimis atsiunčia JSON formatu vienoje eilutėje. Pateikta žinute apačioje buvo suformatuota tam, kad atskiros žinutės dalys būtų įskaitomos. Geltonai paryškintos dalys yra duomenys, kurie bus naudojami paieškai.


```
{
   "created_at":"Sun Jan 04 13:10:19 +0000 2015",
   "id":551727323297624064,
   "id_str":"551727323297624064",
   "text":"Testuojame Twitter Stream API magistriniam darbui. #ComputerScience #SoftwareEngineering #DataMining
#TwitterAPI #python #datascience",
   "source":"\u003ca    href=\"http:\/\/twitter.com\/download\/android\"    rel=\"nofollow\"\u003eTwitter    for
Android\u003c\/a\u003e",
   "truncated":false,
   "in_reply_to_status_id":null,
   "in_reply_to_status_id_str":null,
   "in_reply_to_user_id":null,
   "in_reply_to_user_id_str":null,
   "in_reply_to_screen_name":null,
   "user":{
      "id":2926052357,
      "id_str":"2926052357",
      "name":"Eugenijus S",
      "screen_name":"eugenijus_s",
      "location":"",
      "url":null,
      "description":null,
```


[21] http://geojson.io/

```
      "protected":false,
      "verified":false,
      "followers_count":0,
      "friends_count":1,
      "listed_count":0,
      "favourites_count":0,
      "statuses_count":2,
      "created_at":"Wed Dec 17 00:36:10 +0000 2014",
      "utc_offset":null,
      "time_zone":null,
      "geo_enabled":true,
      "lang":"en",
      "contributors_enabled":false,
      "is_translator":false,
      "profile_background_color":"C0DEED",
      "profile_background_image_url":"http:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png",
      "profile_background_image_url_https":"https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png",
      "profile_background_tile":false,
      "profile_link_color":"0084B4",
      "profile_sidebar_border_color":"C0DEED",
      "profile_sidebar_fill_color":"DDEEF6",
      "profile_text_color":"333333",
      "profile_use_background_image":true,
      "profile_image_url":"http:\/\/abs.twimg.com\/sticky\/default_profile_images\/default_profile_4_normal.png
",
      "profile_image_url_https":"https:\/\/abs.twimg.com\/sticky\/default_profile_images\/default_profile_4_nor
mal.png",
      "default_profile":true,
      "default_profile_image":true,
      "following":null,
      "follow_request_sent":null,
      "notifications":null
   },
   "geo":{
      "type":"Point",
      "coordinates":[54.719770,25.260526]
   },
   "coordinates":{
      "type":"Point",
      "coordinates":[25.260526,54.719770]
   },
   "place":{
      "id":"01cce83d8638e5f7",
      "url":"https:\/\/api.twitter.com\/1.1\/geo\/id\/01cce83d8638e5f7.json",
      "place_type":"city",
      "name":"Vilnius",
      "full_name":"Vilnius, Lietuva",
      "country_code":"LT",
      "country":"Lietuva",
      "bounding_box":{
         "type":"Polygon",
         "coordinates":[
            [[25.0245985,54.5689032],[25.0245985,54.8322118],[25.48146,54.8322118],[25.48146,54.5689032]]]
      },
      "attributes":{}
   },
   "contributors":null,
   "retweet_count":0,
   "favorite_count":0,
   "entities":{
      "hashtags":[
```

```
            {"text":"ComputerScience","indices":[51,67]},
            {"text":"SoftwareEngineering","indices":[68,88]},
            {"text":"DataMining","indices":[89,100]},
            {"text":"TwitterAPI","indices":[101,112]},
            {"text":"python","indices":[113,120]},
            {"text":"datascience","indices":[121,133]}
        ],
        "trends":[],
        "urls":[],
        "user_mentions":[],
        "symbols":[]
    },
    "favorited":false,
    "retweeted":false,
    "possibly_sensitive":false,
    "filter_level":"medium",
    "lang":"lt",
    "timestamp_ms":"1420377019129"
}
```


48 pav. Twitter žinutė JSON formatu

4 priedas. Twitter JSON žinutės struktūra

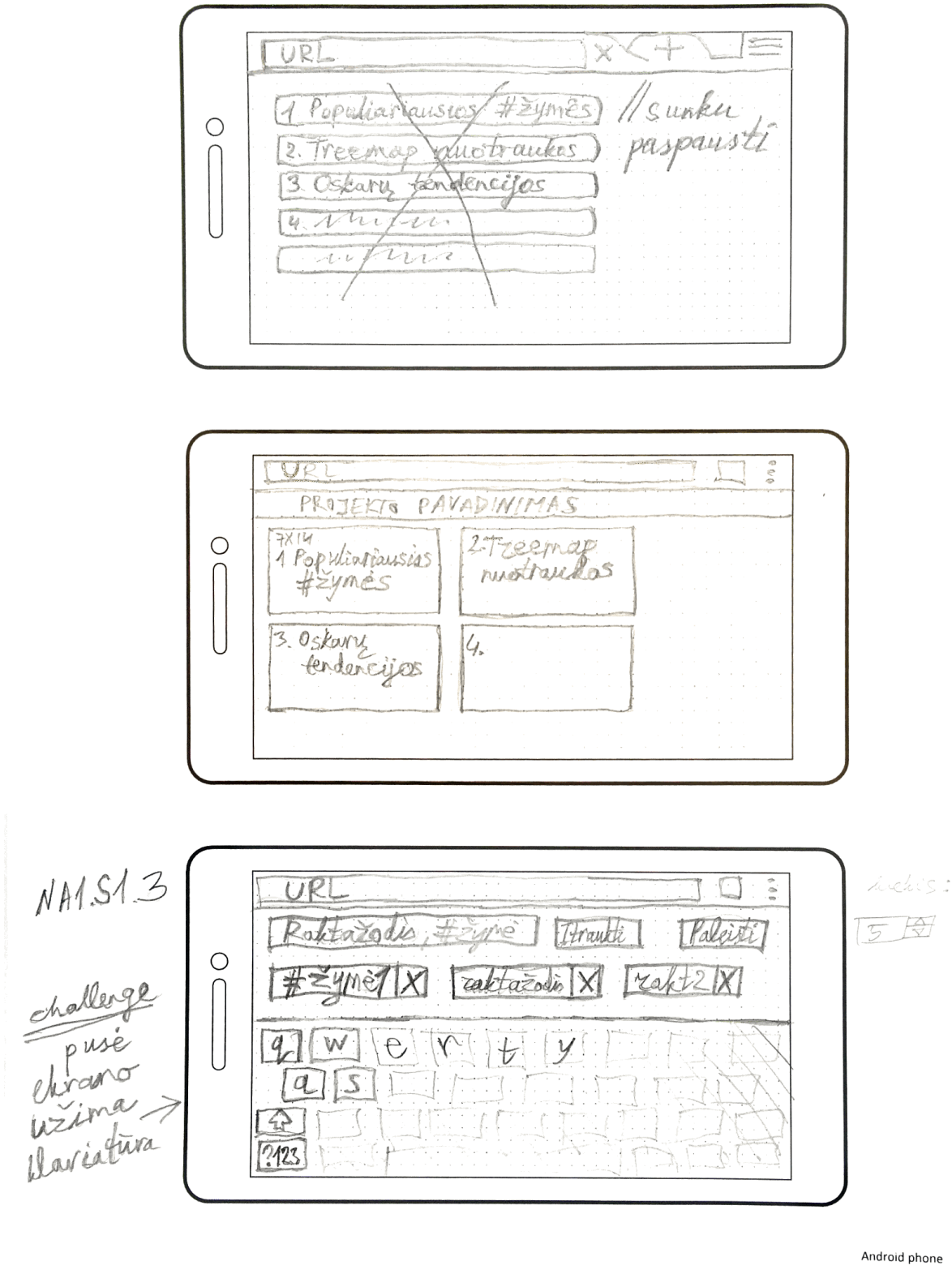
10 lentelė. Twitter JSON žinutės struktūra

Parametras	**Aprašymas, vertimas**
created_at	Konkreti data ir laikas kada Twitter žinutė buvo išsiųsta, sekundžių tikslumu.
id, id_str	Žinutės unikalus identifikavimo kodas sudarytas iš 18 skaitmenų.
text	Žinutės tekstas.
user	Atskiras vartotojo duomenų blokas atskirtas riestiniai skliaustais. Toliau user.<elem> reikš vartotojo duomenų elementą.
user.id, user.id_str	Vartotojo unikalus identifikavimo kodas sudarytas iš 8 skaitmenų.
user.name	Vartotojo pilnas vardas.
user.screen_name	Paskyros pavadinimas (gali būti vartotojo vardas, bet nebūtinai tikras).
user.followers_count	Kiek pasekėjų turi vartotojas, kurie skaito vartotojo žinutes.
user.friends_count	Kiek vartotojas yra prisinumeravęs kitų Twitter vartotojų.
user.statuses_count	Kiek iš viso vartotojas yra parašęs žinučių.
user.created_at	Kada buvo sukurta vartotojo paskyra (užregistruota) Twitter sistemoje.
user.utc_offset	Skirtumas nuo GMT sekundėmis. Pvz.: „-21600“ yra -6 valandos.
user.time_zone	Kurioje laiko juostoje vartotojas gyvena.
user.geo_enabled	Ar vartotojas pateikia savo geografinės koordinates {true, false}.
user.lang	Kalba, pvz.: anglų būtų „en“, lietuvių – „lt“.

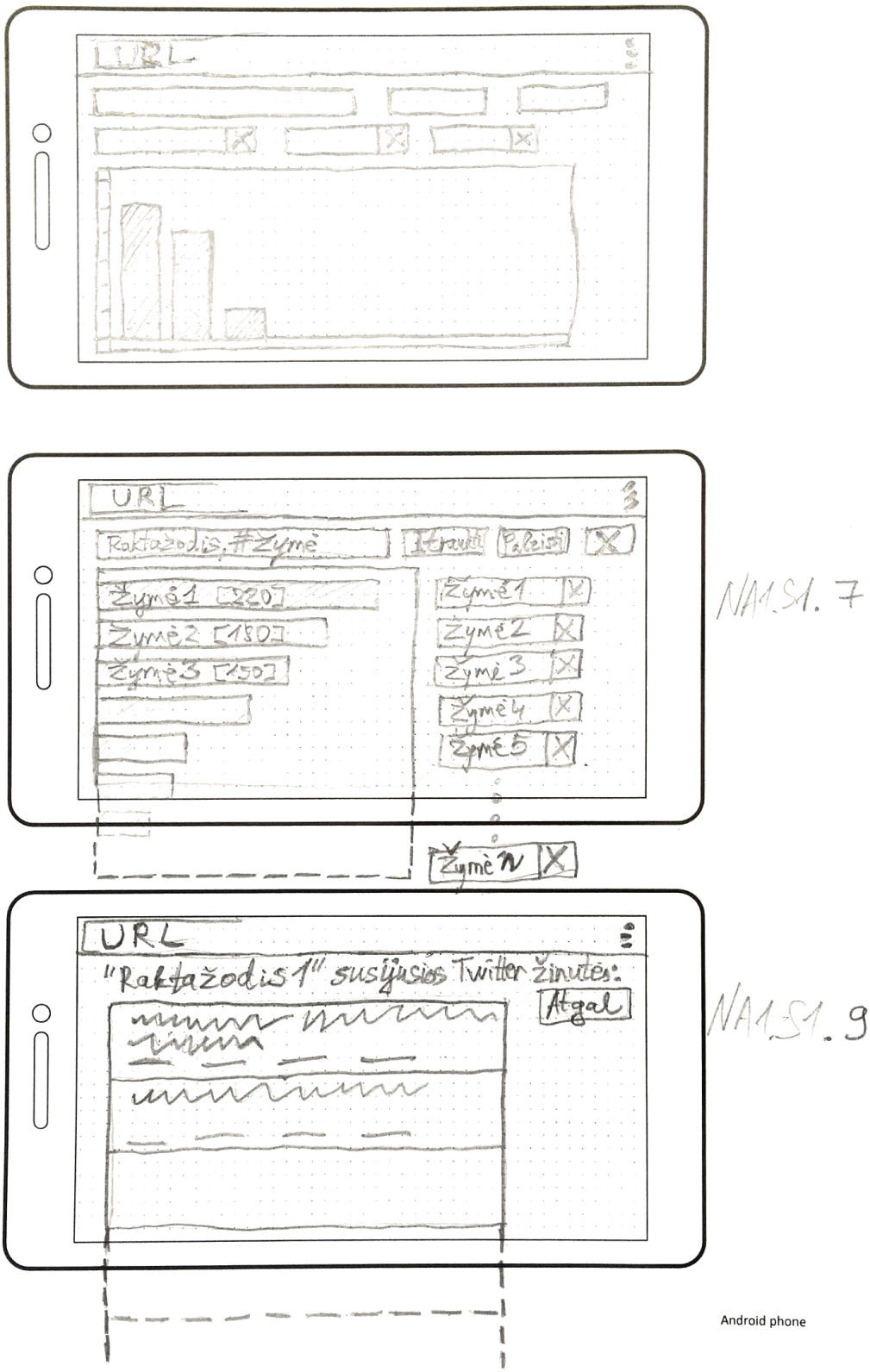
user. profile_image_url	Vartotojo paskyros nuotraukos arba piešinėlio (dažniausiai JPEG formatu) URL adresas.
geo.coordinates	Tikslios vartotojo koordinatės iš kur buvo išsiųsta žinutė [±*nn.nnnnnn*, ±*nn.nnnnnn*] formatu [ilguma, platuma].
place	Atskiras vietovės duomenų blokas.
place.place_type	Geografinės teritorijos rūšis, pvz.: miestas yra „city“.
place.name	Miesto, miestelio ar rajono pavadinimas.
place.full_name	Miesto, miestelio ar rajono pavadinimas su apygardos.
place.country_code	Šalies kodas, pvz.: JAV yra „US“, Lietuva yra „LT“.
place.country	Pilnas šalies pavadinimas.
entities	Atskiras papildomų duomenų blokas.
entities.hashtags	Grotelinė žymė, kreipė (t.y. nukreipiamoji žymė), etiketė.
entities.trends	Žodis apibūdinantis tendenciją, populiarus žodis ar kreipė.
entities.urls	Žinutėje esanti nuoroda ar jų sąrašas.
entities.user_mentions	Duomenų blokas atskirtas laužtiniais skliaustais apie žinutėje paminėtus kitus Twitter vartotojus. Jame yra tokie parametrai kaip jau minėti „screen_name“, „name“, „id“ ir kiti.
entities.media	Duomenų blokas atskirtas laužtiniais skliaustais apie žinutėje esančius papildomus failus (paveikslėlius ar video). Jame yra tokie parametrai kaip „media_url“, „url“ (sutrumpina t.co nuoroda), „display_url“ (tikroji nuoroda), „type“, „sizes“ ir kiti.
favorited	Ar kas nors iš kitų vartotojų pasižymėjo šią žinutę kaip mėgstamą.
retweeted	Ar kas nors iš kitų vartotojų persiuntė (angl. *retweeted*) šią žinutę savo paskyroje.

Tokie laukai kaip ir „favorited“ ir „retweeted“ Twitter duomenų srauto vizualizacijoje bus sunkiai panaudojami, nes jei žinutė yra tik ką publikuota, tuomet jos dar nespėjo niekas persiųsti ar pažymėti kaip mėgstamą.

5 priedas. Vartotojo sąsajos eskizai

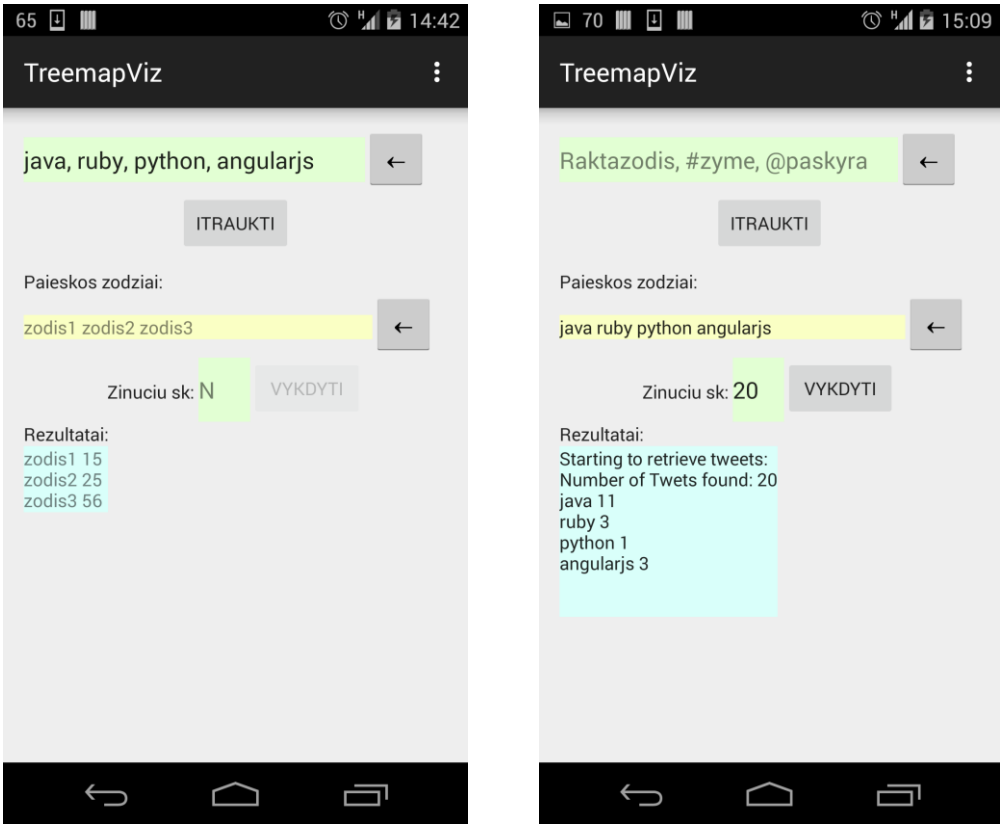


49 pav. Vizualizavimo sistemos vartotojo sąsajos eskizai: viršuje – meniu langas, viduryje – atnaujintas meniu langas, apačioje – NA1S1 raktažodžių ir žymių įvedimo langas

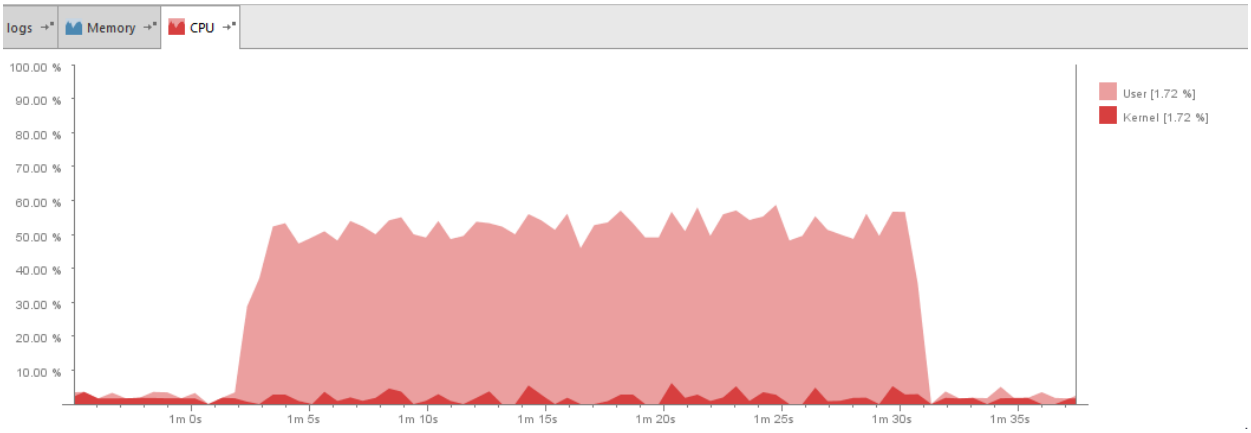


50 pav. Vizualizavimo sistemos vartotojo sąsajos eskizai: viršuje – NA1S1 langas su histograma, viduryje – sistemos langas su vertikalių stulpelių diagrama, apačioje – Twitter žinučių langas

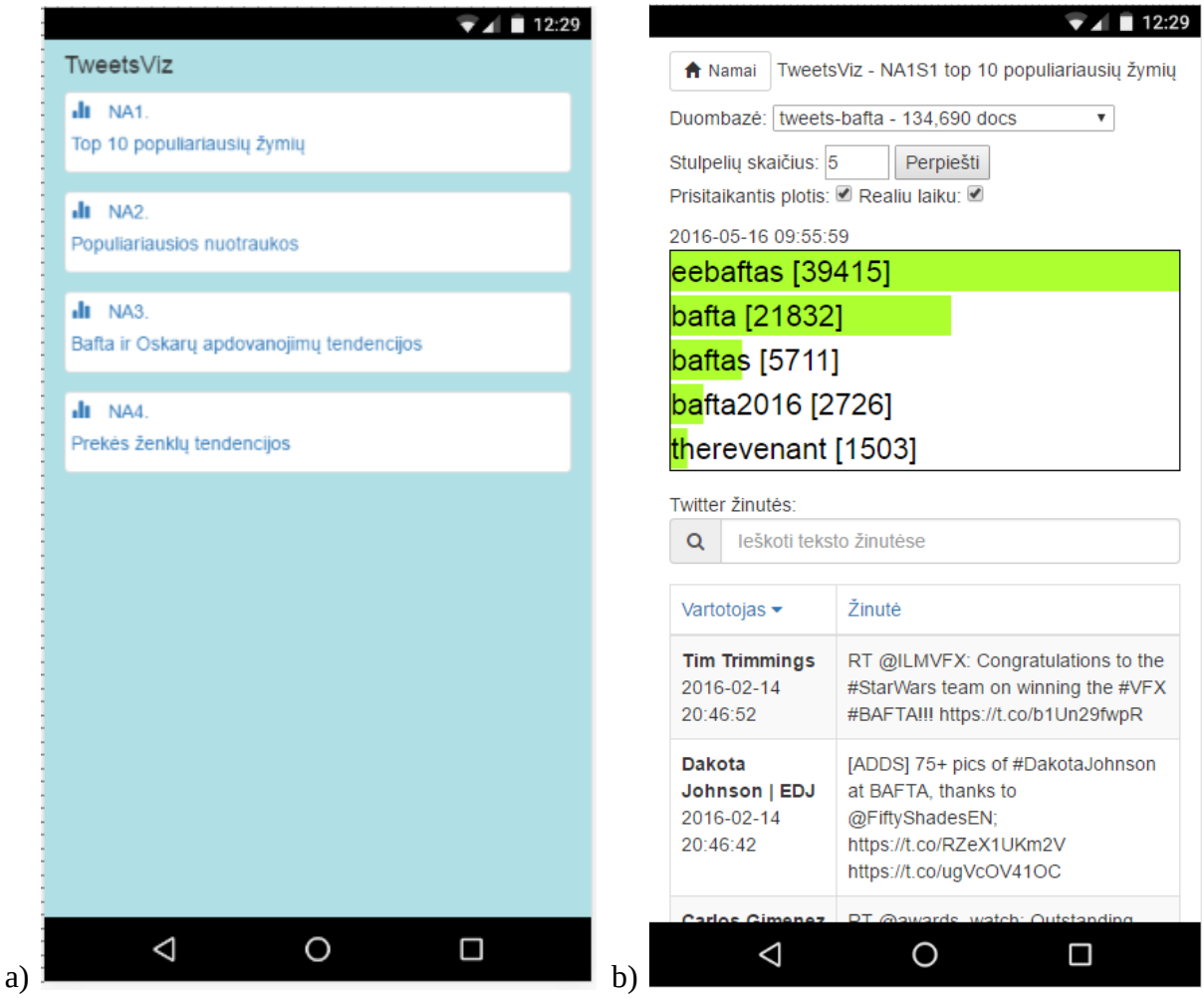
6 priedas. Prototipų nuotraukos



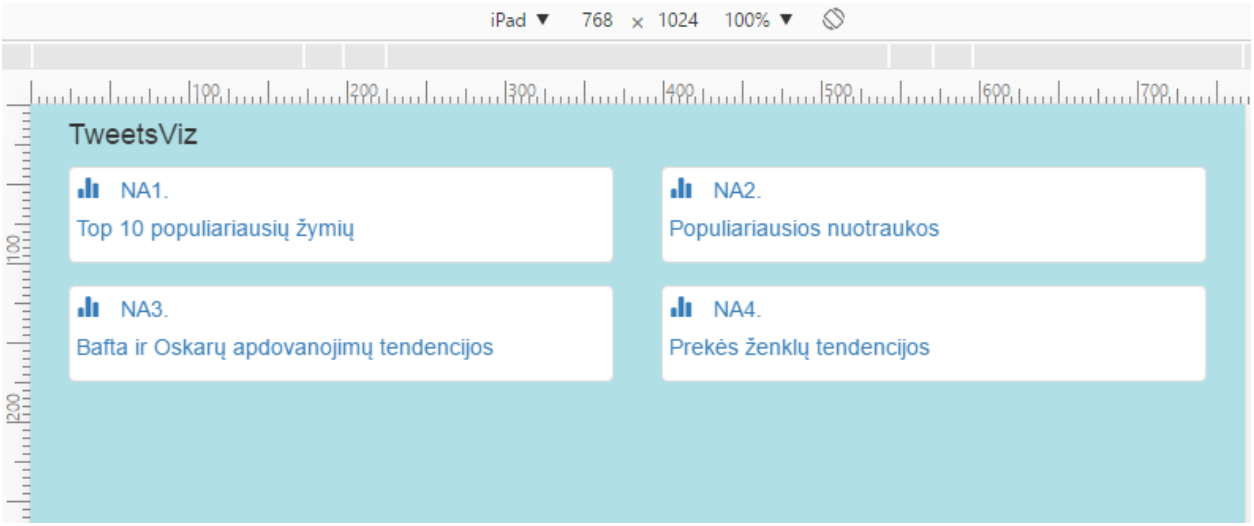
51 pav. Ankstyvasis pirmo sprendimo prototipas, mobili programėlė TreemapViz, kuriame įgyvendintas tiktai Twitter srauto duomenų skaičiavimo funkcionalumas. Kairėje pradinis langas, dešinėje – pradinis langas su rezultatais



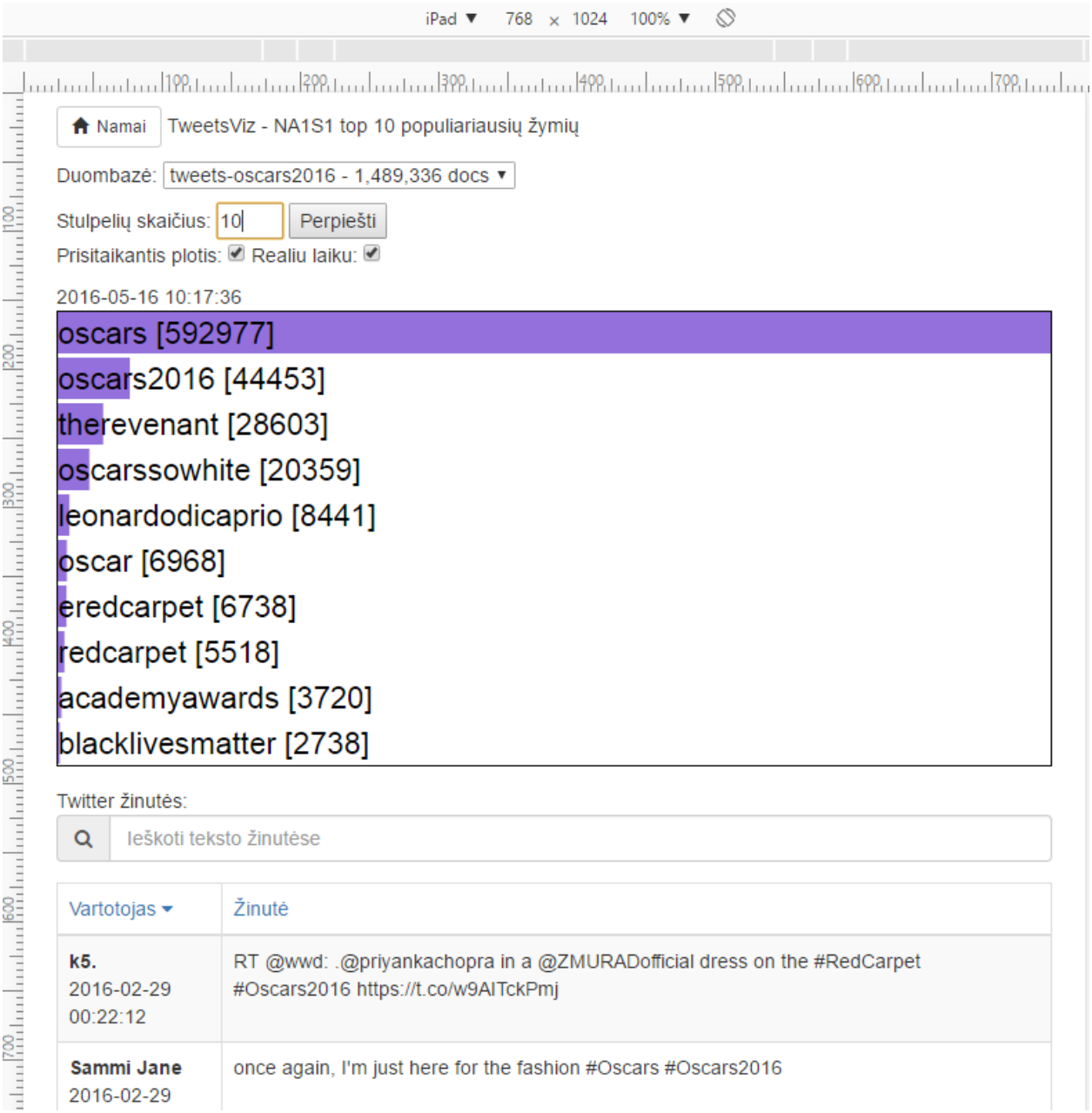
52 pav. Pirmojo sprendimo mobili programėlė skaičiavimams naudojo 50-60% procesoriaus galios, kas yra pakankamai daug



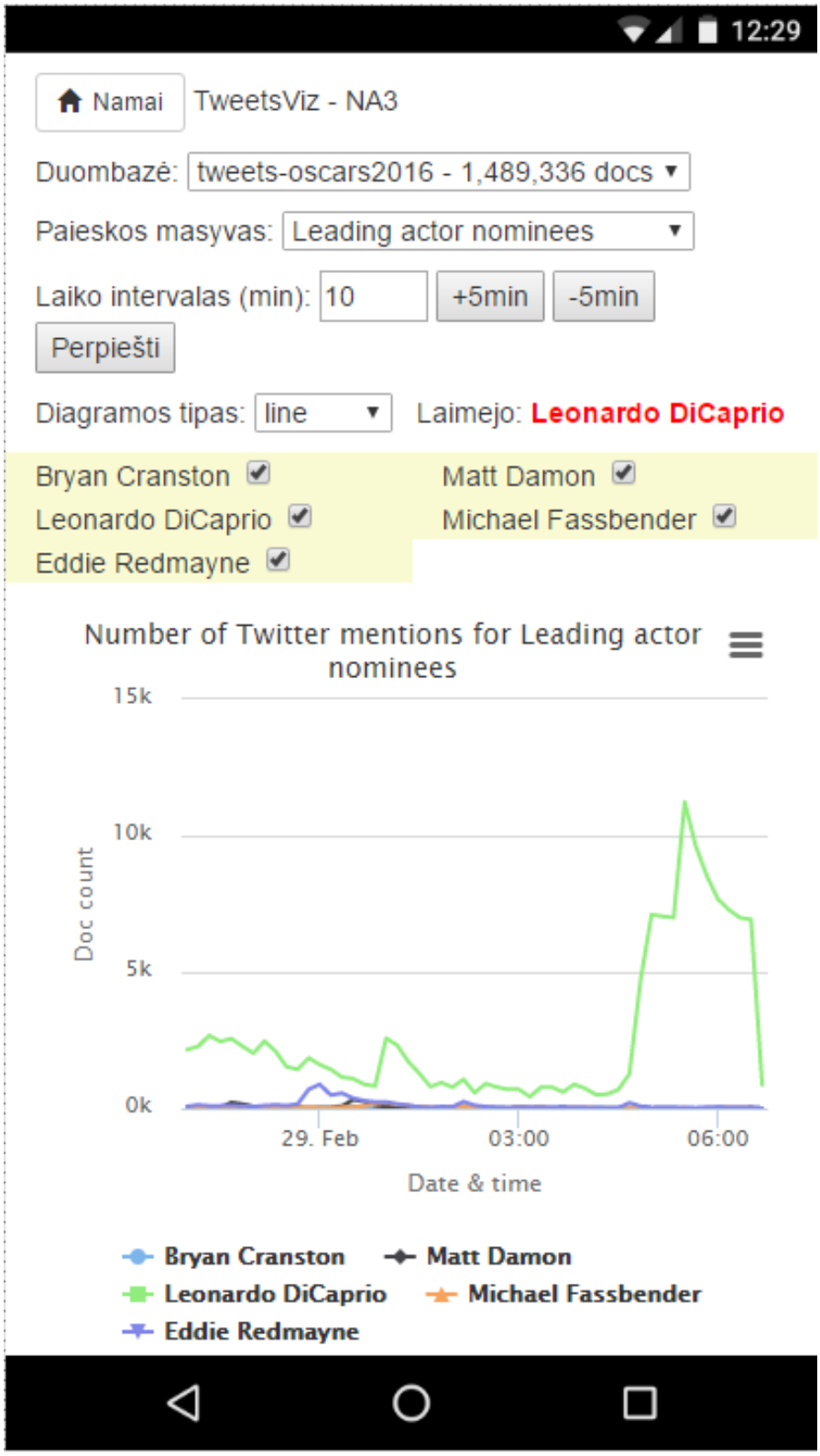
53 pav. Mobiliojo įrenginio Nexus 5X ekrano nuotraukos Iš kairėjs į dešinę a) TweetsViz tinklalapio pagrindinis puslapis su meniu, b) naudojimo atvejo NA1S1 „top 10 populiariausių žymių“ puslapis



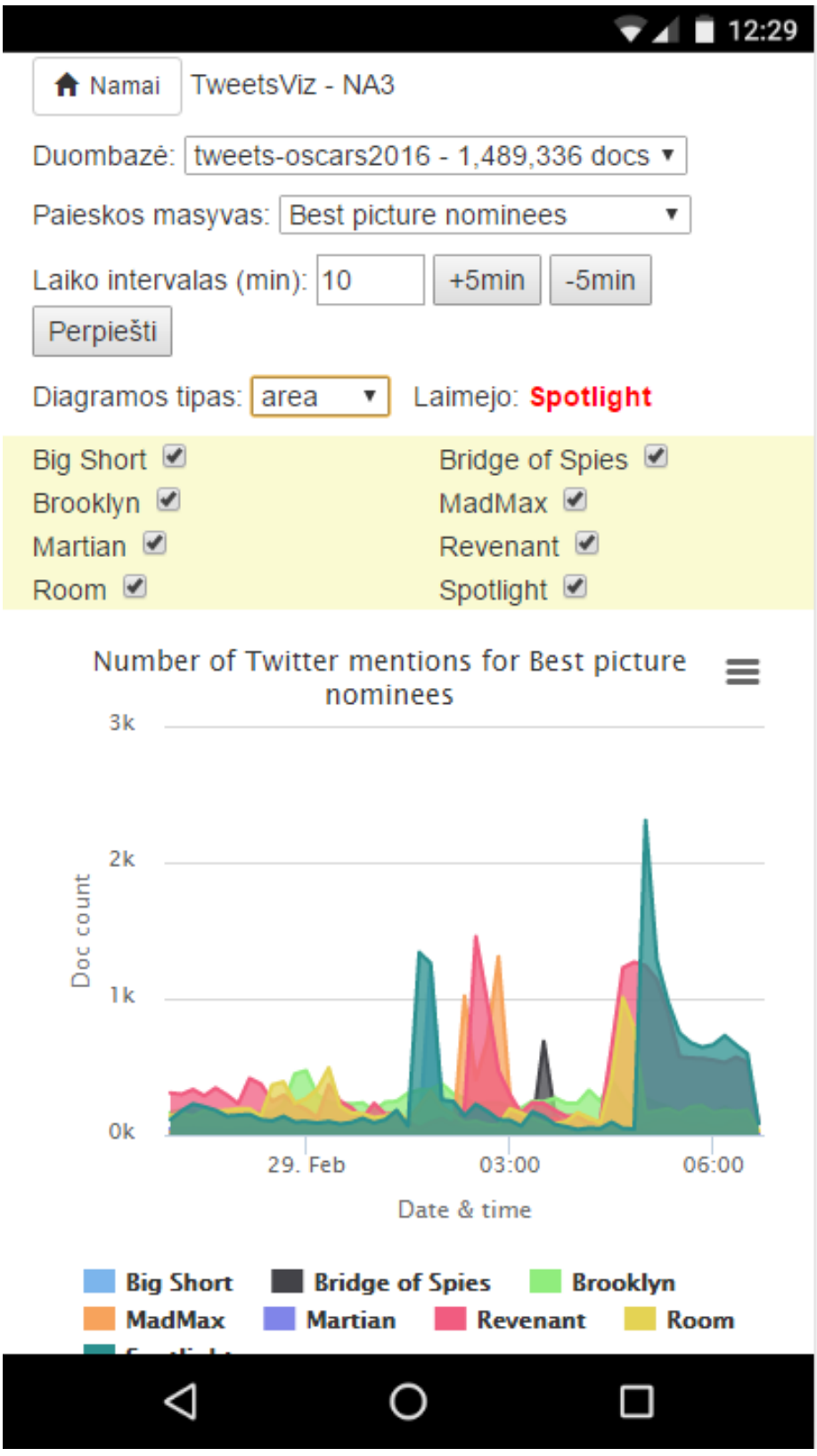
54 pav. Planšetės iPad ekrano nuotrauka su TweetsViz tinklalapiu, kuriame mygtukai prisitaiko prie rezoliucijos ir jų pateikiama po 2 per eilutę



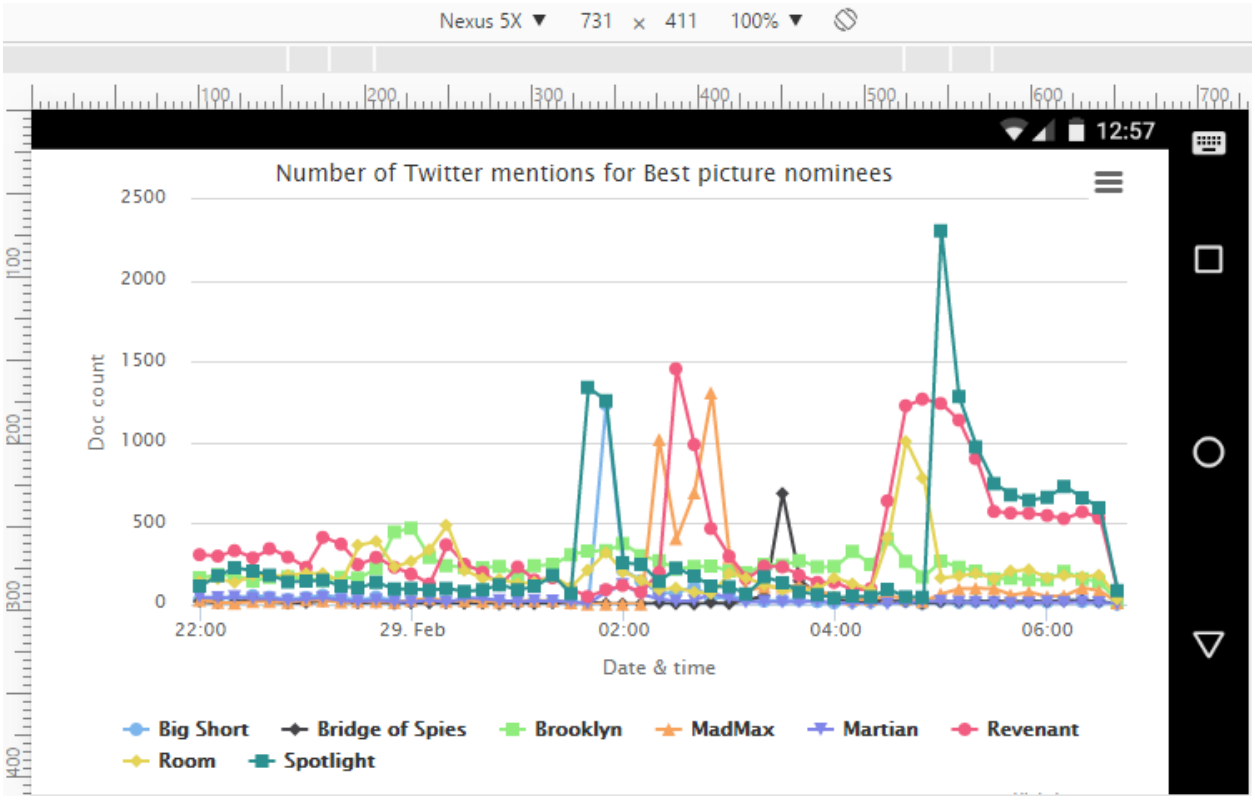
55 pav. Planšetės iPad ekrano nuotrauka su TweetsViz tinklalapiu, kuriame pateikiamas naudojimo atvejo NA1S1 „top 10 populiariausių žymių“ puslapis



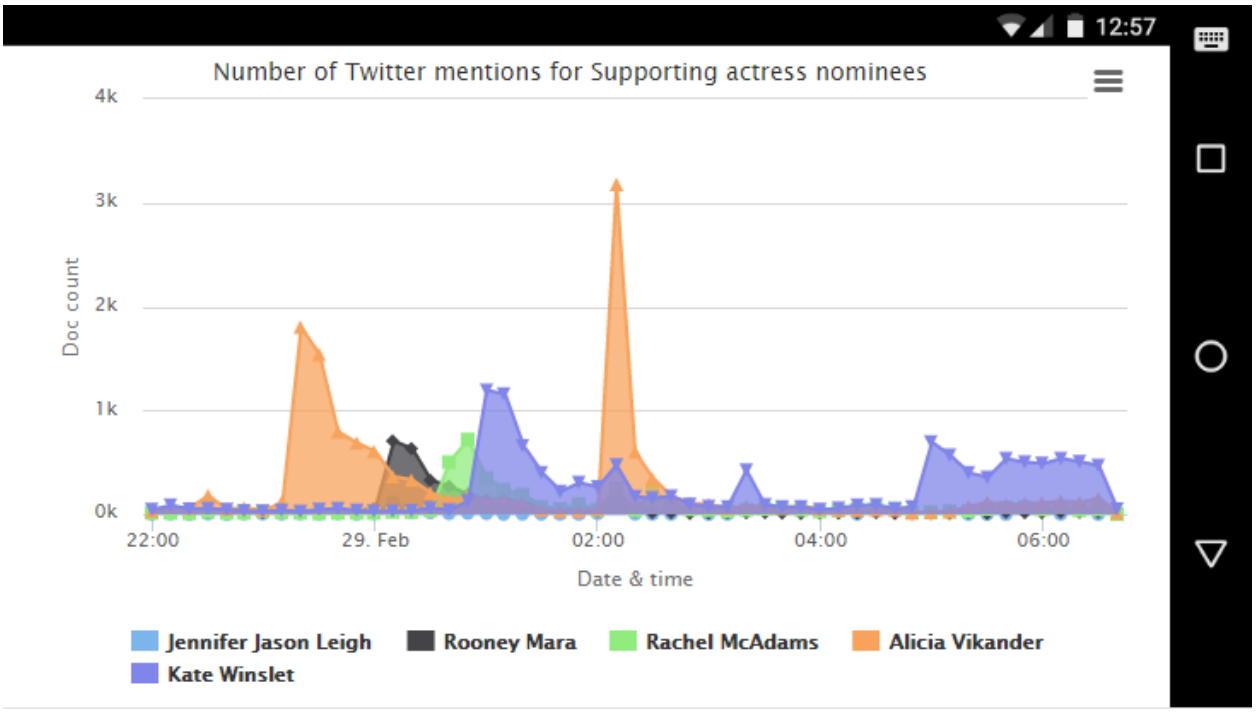
56 pav. Naudojimo atvejo NA3S1 scenarijus „linijinė diagrama – populiariausi aktoriai“



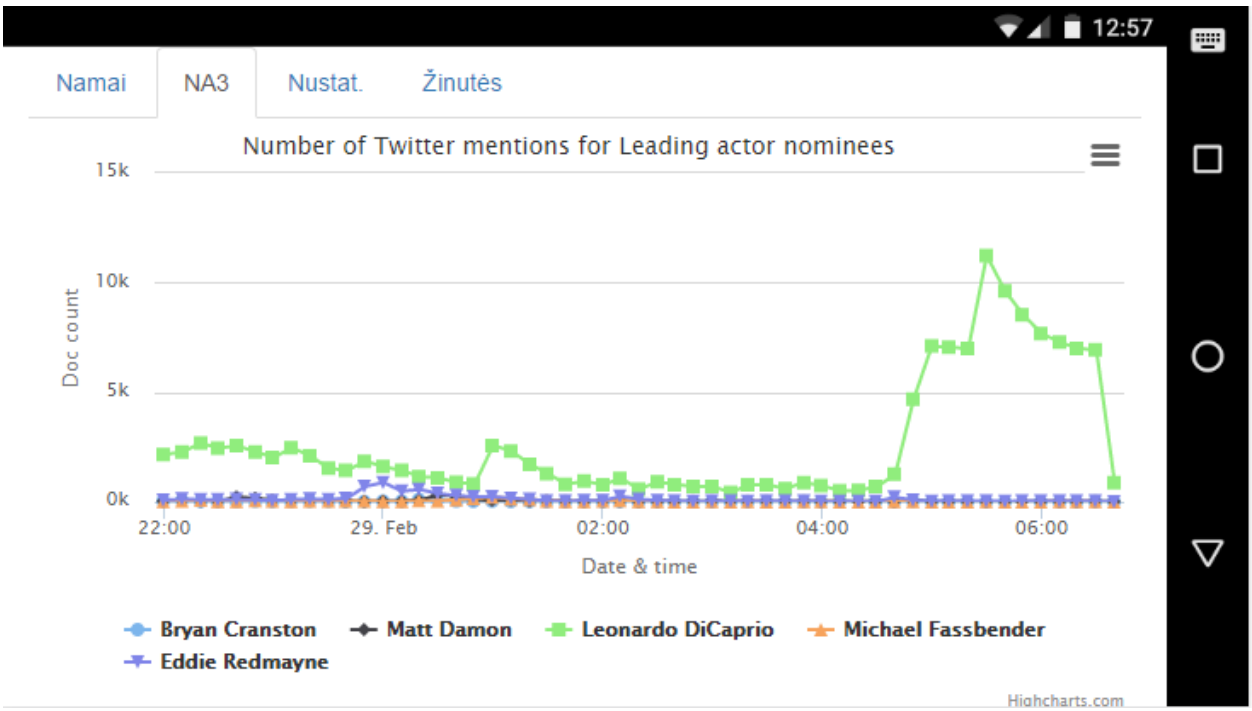
57 pav. Naudojimo atvejo NA3S2 scenarijus „plokštuminė diagrama – populiariausi filmai“



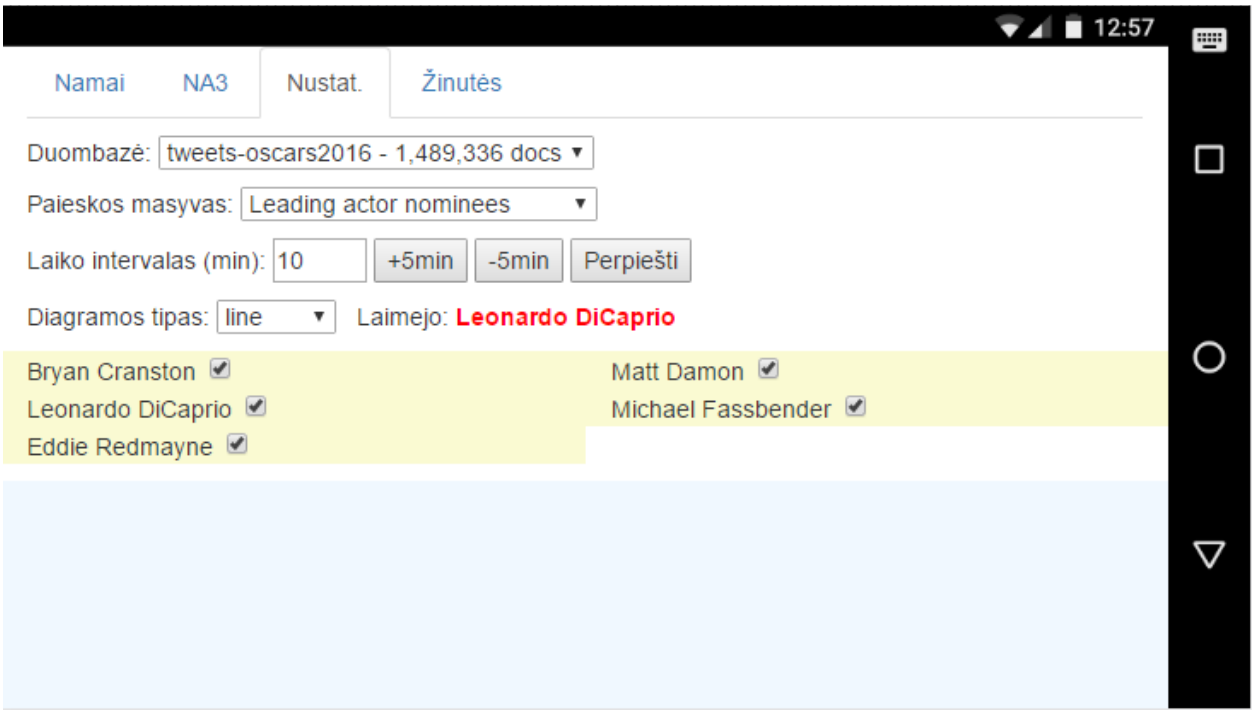
58 pav. Naudojimo atvejo NA3S2 scenarijus „plokštuminė diagrama – populiariausi filmai“ naudojant linijinę diagramą, kai mobiliojo irenginio ekranas gulsčias



59 pav. Naudojimo atvejo NA3S1 scenarijus naudojant plokštuminę diagramą ir paieškos masyvą „geriausios pagalbinės aktorės“, kai mobiliojo irenginio ekranas gulsčias

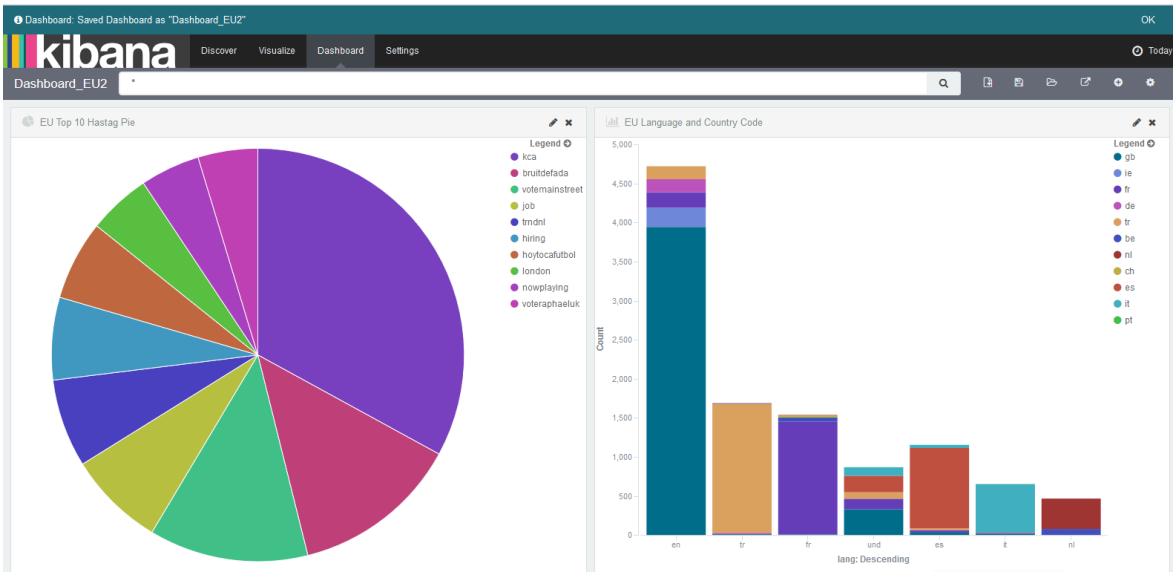


60 pav. Vartotojo sąsajos versija su kortelėm, kurioje pavaizduotas naudojimo atvejo NA3S1 scenarijus „linijinė diagrama – populiariausi aktoriai“



61 pav. Vartotojo sąsajos versija su kortelėm, kurioje nustatymai perkelti į atskirą kortelę, kad pagerinti naudojamumą ir padaryti daugiau vietos vizualizacijai

7 priedas. Kibana vartotojo sąsaja



62 pav. Kibana Dashboard puslapis, kuriame 2 vizualizacijos: kairėje skritulinė diagrama su 10 dažniausiai pasitaikančių žymių, o dešinėje – stulpelinė diagrama su žinutėse naudojamų kalbų pasiskirstymu

8 priedas. Elasticsearch užklausos ir atsakymų pavyzdžiai

```
================================================================================
UŽKLAUSA:
================================================================================
POST http://localhost:9200/tweets-oscars2016/logs/ search
{
    "aggs": {
        "top-tags": {
            "terms": {
                "field": "entities.hashtags.text",
                "size": 5
            }
        }
    },
    "size": 4,
    "_source": ["text"]
}
================================================================================
ATSAKYMAS:
================================================================================
{
   "took": 115,
   "timed out": false,
   " shards": {
      "total": 5,
      "successful": 5,
      "failed": 0
   },
   "hits": {
      "total": 1489337,
      "max score": 1,
      "hits": [
         {
            "_index": "tweets-oscars2016",
            " type": "logs",
            " id": "AVMqLGzDK-lZK6jpm3Fx",
            " score": 1,
            "_source": {
```

```
               "text": "@Palm4Beach io mi sto destreggiando tra #gazebo e gli #Oscars ... Non so cosa guardare
????????"
            }
         },
         {
            "_index": "tweets-oscars2016",
            "_type": "logs",
            "_id": "AVMqLGzDK-lZK6jpm3F3",
            " score": 1,
            " source": {
               "text": "My local news is hosting an Oscar party featuring insights from Lights Camera Jackson.
#fml #oscars"
            }
         },
         {
            " index": "tweets-oscars2016",
            "_type": "logs",
            "_id": "AVMqLGzDK-lZK6jpm3F6",
            "_score": 1,
            "_source": {
               "text": "RT @DivergenteBR: Será que nossa Jeanine ganha um #Oscars hoje? A Kate Winslet foi
indicada! Torcendo aqui já ??"
            }
         },
         {
            "_index": "tweets-oscars2016",
            " type": "logs",
            " id": "AVMqLGzDK-lZK6jpm3F7",
            " score": 1,
            "_source": {
               "text": "?? ??????? ???  https://t.co/yu6f4FKlio"
            }
         }
      ]
   },
   "aggregations": {
      "top-tags": {
         "doc_count_error_upper_bound": 3162,
         "sum_other_doc_count": 506609,
         "buckets": [
            {
               "key": "oscars",
               "doc_count": 1157813
            },
            {
               "key": "oscars2016",
               "doc_count": 86995
            },
            {
               "key": "therevenant",
               "doc_count": 32746
            },
            {
               "key": "oscarssowhite",
               "doc_count": 23382
            },
            {
               "key": "leonardodicaprio",
               "doc_count": 17195
            }
         ]
      }
   }
}
================================================================================
```


63 pav. POST užklausa ir atsakymas su surinkimo objektu `aggs` – populiariausios žymės

```
================================================================================
UŽKLAUSA:
================================================================================
POST http://localhost:9200/tweets-oscars2016/logs/_search
{
    "query": {
      "filtered": {
      "query": {
        "query string": {
          "query": "\"Matt Damon\"",
          "analyze_wildcard": true
        }
      },
      "filter": {
        "range": {
          "@timestamp": {
            "gte": 1456693200000,
            "lte": 1456729200000,
            "format": "epoch_millis"
          }
        }
      }
```

```
    }
  },
  "size": 0,
  "aggs": {
    "group_by_time_interval": {
      "date_histogram": {
        "field": "@timestamp",
        "interval": "1h",
        "time zone": "Europe/Helsinki",
        "min doc count": 1,
        "extended bounds": {
          "min": 1456693200000,
          "max": 1456729200000
        }
      }
    }
  }
}
================================================================================
ATSAKYMAS:
================================================================================
{
   "took": 29,
   "timed_out": false,
   "_shards": {
      "total": 5,
      "successful": 5,
      "failed": 0
   },
   "hits": {
      "total": 2413,
      "max_score": 0,
      "hits": []
   },
   "aggregations": {
      "group by time interval": {
         "buckets": [
            {
               "key_as_string": "2016-02-29T00:00:00.000+02:00",
               "key": 1456696800000,
               "doc count": 518
            },
            {
               "key_as_string": "2016-02-29T01:00:00.000+02:00",
               "key": 1456700400000,
               "doc_count": 292
            },
            {
               "key_as_string": "2016-02-29T02:00:00.000+02:00",
               "key": 1456704000000,
               "doc_count": 808
            },
            {
               "key as string": "2016-02-29T03:00:00.000+02:00",
               "key": 1456707600000,
               "doc_count": 257
            },
            {
               "key_as_string": "2016-02-29T04:00:00.000+02:00",
               "key": 1456711200000,
               "doc count": 131
            },
            {
               "key_as_string": "2016-02-29T05:00:00.000+02:00",
               "key": 1456714800000,
               "doc count": 163
            },
            {
               "key_as_string": "2016-02-29T06:00:00.000+02:00",
               "key": 1456718400000,
               "doc_count": 112
            },
            {
               "key_as_string": "2016-02-29T07:00:00.000+02:00",
               "key": 1456722000000,
               "doc_count": 50
            },
            {
               "key as string": "2016-02-29T08:00:00.000+02:00",
               "key": 1456725600000,
               "doc_count": 82
            }
         ]
      }
   }
}
================================================================================
```

64 pav. POST statistikos surinkimo užklausa pagal laiką ir užklausos atsakymas

9 priedas. Išvesties kodo fragmentai

```
177             function drawBarChart(){
178                 if(!scope.chartDataArray){
179                     console.log("drawBarChart() FAILED!");
180                 } else {
181                     if( typeof scope.chartDataArray !== 'undefined' ) {
182                         if(redraw_counter > 0){
183                             color_number = Math.floor(Math.random() * colors.length);
184                             d3.select("svg").remove();
185                         }
186                         //1) Prepare Data with hashtag count
187                         var hashtag_count_array = prepareData();
188                         //2) create SVG element
189                         createSVG();
190                         //3) Drawing all the bars one by one
191                         hashtag_count_array.forEach(drawEachBar);
192                         redraw_counter = redraw_counter + 1;
193                     }
194                 }
195             }
```

65 pav. Stulpelinės diagramos piešimo funkcija

```
134             //copying each Hashtag name and number of hastags found to separate arrays
135             function prepareData(){
136                 var hashtag_count_array = [];
137                 //hashtag_name_array is initializes before
138                 for (var i = 0; i < scope.chartDataArray.length; i++) {
139                     hashtag_count_array[i] = scope.chartDataArray[i].doc_count;
140                     hashtag_name_array[i] = scope.chartDataArray[i].key;
141                 }
142                 max_hashtag_count = d3.max(hashtag_count_array);
143                 return hashtag_count_array;
144             }
145
```

66 pav. Duomenų paruošimo funkcija

```
146             //Creating SVG element
147             function createSVG(){
148                 if(scope.numberOfChartBars){
149                     number_of_bars = scope.numberOfChartBars;
150                     svg_height = number_of_bars * (bar_height + bar_gap);
151                 }
152                 svg = d3.select(element[0]).append('svg')
153                     .attr({width: svg_width, height: svg_height})
154                     .style('border', '1px solid black');
155             }
```

67 pav. Funkcija SVG elemento kūrimui naudojant D3 karkasą

```
157             //Draws one bar depending on variables and bar's length value *element*
158             function drawEachBar(element, index, array) {
159                 //TODO explain this formula
160                 var element_w = (element * svg_width / max_hashtag_count) - 2;
161                 // drawing 1 bar
162                 svg.append('rect').style('fill', colors[color_number])
163                     .attr("x", 0)
164                     .attr("y", index * (bar_height + bar_gap)) //2 pikseliu tarpas tarp stulpeliu
165                     .attr("width", (element_w) )
166                     .attr("height", bar_height);
167                 //adding 1 text label on the bar
168                 svg.append('text')
169                     .attr("x", 0)
170                     .attr("y", index * (bar_height + bar_gap) + (3 * bar_height / 4))
171                     .text( hashtag_name_array[index] + " " + "[" + element + "]")
172                     .attr("font-family", "sans-serif")
173                     .attr("font-size", bar_height-8+"px")
174                     .attr("fill", "black");
175             }
```

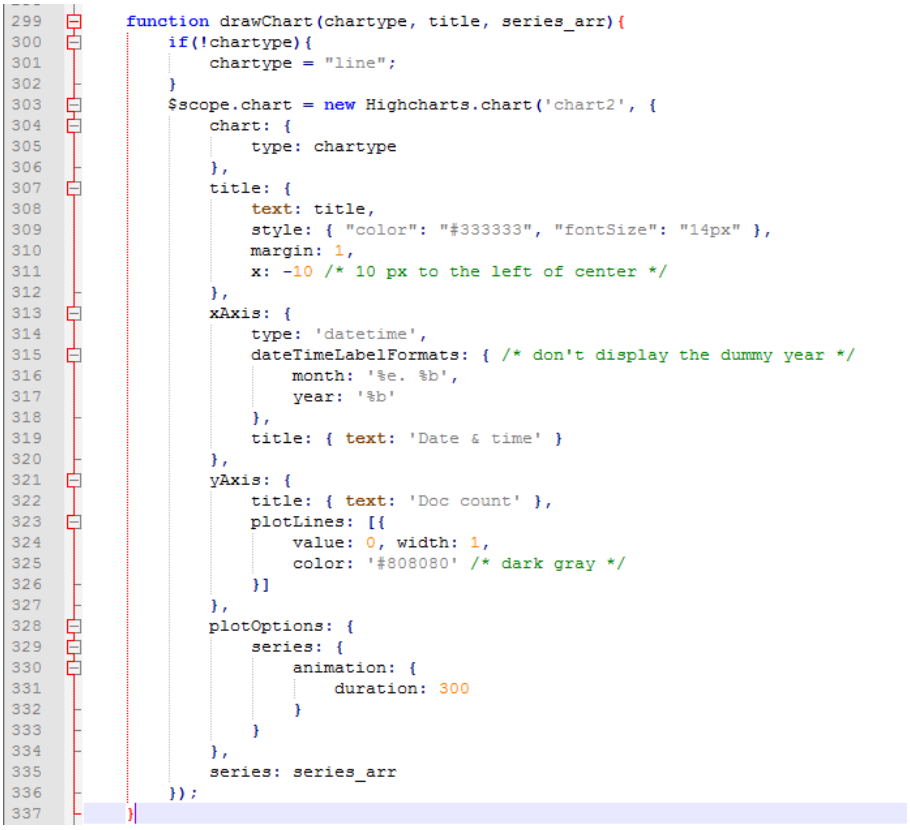
68 pav. Funkcija skirta vienam stulpeliui nupiešti naudojant D3 karkasą

```
171     $scope.$watchCollection('responseObjectsArray', function () {
172         console.log(">>>>>>> responseObjectsArray changed! Length: " + $scope.responseObjectsArray.length);
173         //console.log($scope.responseObjectsArray );
174         var number_of_phrases = $scope.search_phrases.length;
175         var number_of_elements = 0;
176         for(var i = 0; i < number_of_phrases; i++){
177             if($scope.responseObjectsArray[i] instanceof Object){
178                 number_of_elements++;
179             }
180         }
181         //console.log("number_of_elements = " + number_of_elements);
182         if(number_of_elements == number_of_phrases){
183             console.log("LETS parseAllResponsesIntoArrays();");
184             $scope.responseSeries = parseAllResponsesIntoArrays($scope.responseObjectsArray);
185             drawChart($scope.charttype.selectedOption.value, $scope.chartTitle, $scope.responseSeries);
186         }
187     });
```

69 pav. Duomenų masyvo responseObjectArray stebėjimo ir atnaujinimo funkcija

```
266     function parseAllResponsesIntoArrays(responseObjects){
267         var number_of_phrases = $scope.search_phrases.length;
268         //output will consist of [name: phrase, data: [timestamp, doc_count]}
269         var output = new Array(number_of_phrases);
270         var responseTimestampAndDocCount = new Array(number_of_phrases);
271         var number_of_docs_received_arr = new Array(number_of_phrases);
272 
273         for (var i = 0; i < number_of_phrases; i++) {
274             if(!responseObjects[i].data || !responseObjects[i].data.aggregations
275                 || (!responseObjects[i].data.hits && responseObjects[i].data.hits > 0)
276                 || !responseObjects[i].data.aggregations.group_by_time_interval.buckets
277             ){
278                 console.log("parseAllResponsesIntoArrays(): SOMETHING IS WRONG WITH responseObjects["+i+"]");
279                 console.log(responseObjects[i]);
280             } else {
281                 var buckets = responseObjects[i].data.aggregations.group_by_time_interval.buckets;
282                 responseTimestampAndDocCount[i] = new Array(buckets.length);
283                 output[i] = new Array(2);
284 
285                 for (var j = 0; j < buckets.length; j++) {
286                     var timestamp_and_doc_count = new Array(2);
287                     timestamp_and_doc_count[0] = buckets[j].key;
288                     timestamp_and_doc_count[1] = buckets[j].doc_count;
289                     number_of_docs_received_arr[i] += buckets[j].doc_count;
290                     responseTimestampAndDocCount[i][j] = timestamp_and_doc_count;
291                 }
292                 output[i] = { name: $scope.search_phrases[i], data: responseTimestampAndDocCount[i] };
293             }
294         }
295         return output;
296     }
```

70 pav. Funkcija skirta duomenų apdorojimui prieš piešiant linijinę diagramą



```
function drawChart(chartype, title, series_arr){
    if(!chartype){
        chartype = "line";
    }
    $scope.chart = new Highcharts.chart('chart2', {
        chart: {
            type: chartype
        },
        title: {
            text: title,
            style: { "color": "#333333", "fontSize": "14px" },
            margin: 1,
            x: -10 /* 10 px to the left of center */
        },
        xAxis: {
            type: 'datetime',
            dateTimeLabelFormats: { /* don't display the dummy year */
                month: '%e. %b',
                year: '%b'
            },
            title: { text: 'Date & time' }
        },
        yAxis: {
            title: { text: 'Doc count' },
            plotLines: [{
                value: 0, width: 1,
                color: '#808080' /* dark gray */
            }]
        },
        plotOptions: {
            series: {
                animation: {
                    duration: 300
                }
            }
        },
        series: series_arr
    });
}
```

71 pav. Funkcija skirta linijinės diagramos piešimui naudojant Highcharts karkasą