KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Darius Lekavičius

APYTIKSLIO TEKSTO PAIEŠKA ŠALTINIO RADIMUI
APPROXIMATE STRING SEARCH FOR TEXT SOURCE RETRIEVAL

Baigiamasis magistro projektas

Vadovas
dr. Mantas Lukoševičius

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas
(parašas) Doc. dr. Tomas Blažauskas
(data)

APYTIKSLIO TEKSTO PAIEŠKA ŠALTINIO RADIMUI
APPROXIMATE STRING SEARCH FOR TEXT SOURCE RETRIEVAL

Baigiamasis magistro projektas
Programų sistemų inžinerija (621E16001)

Vadovas
(parašas) dr. Mantas Lukoševičius
(data)

Recenzentas
(parašas)
(data)

Projektą atliko
(parašas) Darius Lekavičius
(data)

KAUNAS, 2015

Turinys

Įžanga

Šiais laikais nieko jau nebestebina planšečių paplitimas, mobilieji telefonai didžiuliais ekranas ir elektroninės knygų skaityklės. Skaitmeninių knygų versijų skaitymas sparčiai populiarėja, tai yra labai patogi knygų pateikimo forma, tekste galima pridėti aktyvių nuorodų į medija turinį, nuolat atnaujinti tekstinius duomenis, rinktis iš kelių knygos kalbos variantų. Tačiau ką daryti žmonėms, kuriems reikalingas popierinis variantas, ne kiekvienam patogu ar malonu skaityti knygas kokio nors įrenginio ekrane.

„Spausdintos knygos niekada nemirs“ taip teigia dalis žmonių [1]. Popierinės knygos jų savininkams turi išliekamąją vertę. Daugelis spausdintų knygų propaguotojų pasakys, kad jos yra patogesnės, mielesnės ir apskritai knygos skaitymas tai tokia patirtis ir pojūčiai, kurių niekaip neįmanoma perteikti virtualiose versijose. Tad kaipgi galėtume suteikti tas pačias papildomas šiuolaikines galimybes spausdintoms knygoms? Tam gali pagelbėti mobilieji įrenginiai su kameromis. Kameros dėka knygos vaizdas konvertuojamas į tekstą, o pagal tekstą galima surasti ir visą papildomą informaciją bei mobiliajame įrenginyje pateikti naujausią informaciją apie susijusius leidinius ar papildomą medija turinį.

Kad tai pasiekti reikia sukurti mobiliąją aplikaciją vaizdo apdorojimui ir duomenų bazę leidinių informacijos kaupimui.

Kuriama elektroninių leidinių ir medija turinio kaupimo sistema bus pritaikyta popierinių leidinių mėgėjams. Šios sistemos dėka, paprastų knygų naudotojai, kaip ir elektroninių knygų skaitytojai, galės išmaniojo telefono ekrane peržvelgti su knygos tekstu susietą skaitmeninę informaciją, medijos turinį, aktyvias nuorodas.

Kameros dėka knygos vaizdas konvertuojamas į tekstą, o pagal tekstą galima surasti ir visą papildomą informaciją bei mobiliajame įrenginyje pateikti naujausią informaciją apie susijusius leidinius ar papildomą medija turinį. Kad tai pasiekti reikia sukurti mobiliąją aplikaciją vaizdo apdorojimui ir duomenų bazę leidinių informacijos kaupimui. Šią sistemą sudaro dvi dalys: mobilioji aplikacija ir serverio duomenų bazė. Mobiliojoje aplikacijoje bus realizuojama vaizdo apdorojimo ir teksto atpažinimo dalis, o serverio duomenų bazėje duomenų įkėlimo sąsaja ir apytikslio teksto paieška.

Analitinė dalis

Projekto tikslas ir aktualumas

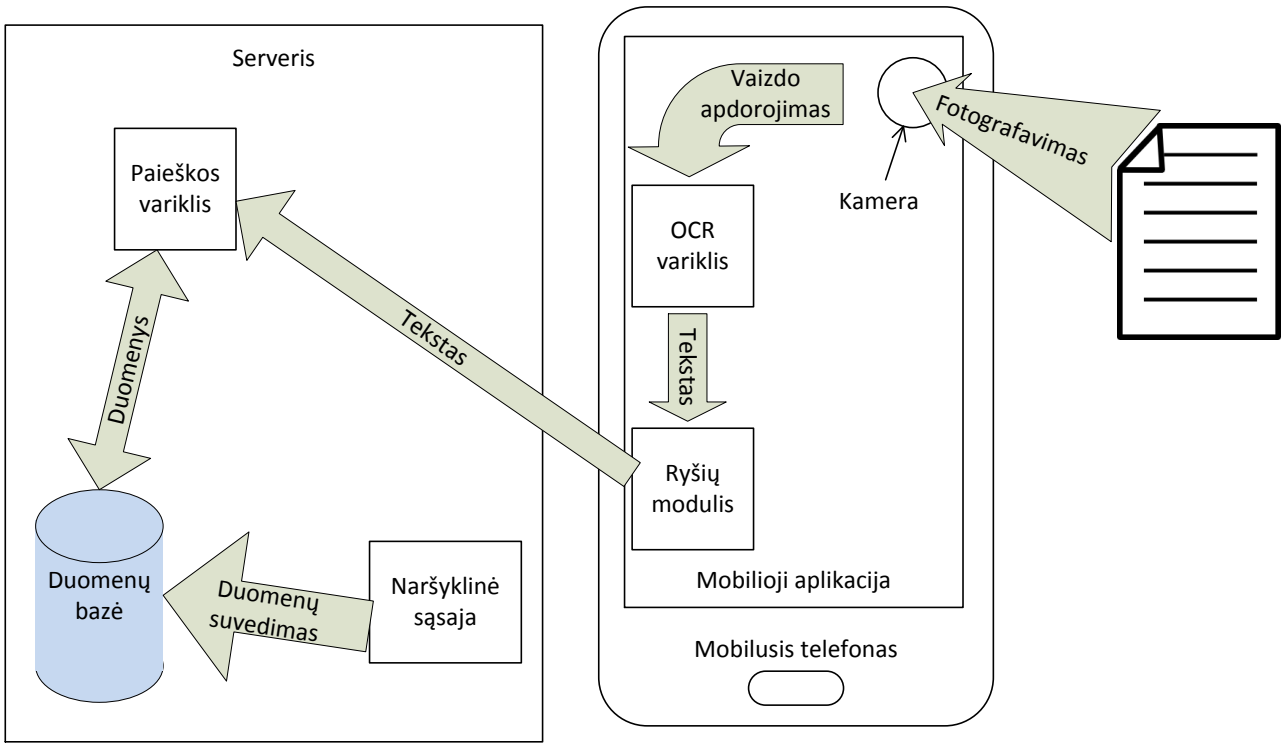
Tikslas ištirti programinės įrangos skirtos teksto atpažinimui iš paveikslėlio ir apytikslio teksto paieškos duomenų bazėje specifikas bei problemas. Šiame darbe koncentruojamasi ties

pagrindinėmis šios sistemos funkcijomis. Viena iš jų yra surasti ir pritaikyti optimalų teksto atpažinimo algoritmą mobiliajam telefonui, norint išgauti kuo tikslesnį pradinį tekstą. Antroji dalis yra serverio dalies apytikslio teksto paieškos algoritmo radimas ir pritaikymas paieškai naudojant teksto atpažinimo dalies pateiktą tekstą.

Sukurta programine įranga naudosis popierinių leidinių skaitytojai, norėdami gauti papildomos informacijos apie skaitomą tekstą ar peržvelgti prie teksto prikabintą medija turinį.

Aprašytą projekto tikslą išskirstome į keletą skirtingų užduočių:

1. Išanalizuoti galimybes pagal optinio teksto atpažinimo (OCR) būdu apdorotą tekstą, surasti šio tekstinio fragmento šaltinį elektroninių leidinių duomenų bazėje.
2. Surasti OCR apdoroto teksto vietą skaitmeniniame leidinio variante.
3. Sukurti apytikslės paieškos algoritmų veikimą iliustruojantį prototipą.
4. Sukurti mobiliąją aplikaciją, kuri nuskaitytų tekstą iš kameros vaizdo.



Paveikslėlis 1 – principinė projekto komponentų schema

Temos aktualumas

Toks sprendimas galėtų stipriai sumažinti knygos spausdinimo kaštus, nes iliustruotos knygos yra žymiai brangesnės už tekstinę spaudą. Visą spalvotą vizualinę informaciją ir papildomą video informaciją naudotojai galėtų gauti iš mobilaus įrenginio. Įvertiname tai, kad dabar didžioji dalis naudotojų turi išmaniuosius telefonus su vaizdo kameromis ir prieiga prie interneto.

Temos perspektyvumas

Šios sistemos perspektyvos didelės, nes identifikavus skaitomą skaitmeninį turinį ar skaitmenines funkcijas galima plėsti beveik neribotai. Kalbant apie pačias knygas, identifikavus knygą būtų galima rasti tą pačią knygą tik kita kalba, neverčiant viso teksto, o ieškant tarp sistemoje esančių egzempliorių. Toliau ši sistema galėtų pasitarnauti ne vien knygoms ar spausdintam tekstui. Bet koks tekstas bet kur atvaizduotas galėtų turėti susijusį skaitmeninį turinį, kurį galima būtų peržiūrėti telefono ekrane, panašiu principu kaip dabar plinta 2D brūkšninio kodo (kitaip QR kodas) naudojimas.

Panašūs egzistuojantys sprendimai

Tokio tipo ar tokios paskirties programinės įrangos rasti nepavyko. Panašu, kad tai yra pakankamai nauja idėja ir niekas dar nėra sukūręs programinio sprendimo. Yra tik panašias funkcijas turinčių programinių sistemų, kurios vykdo paiešką pagal nufotografuotą vaizdą arba realiu laiku vaizde pateikia papildytosios realybės objektus.

Atlikus rinkos analizę, kaip artimiausius programinius sprendimus būtų galima įvardinti šiuos:

- Google Goggles – tai mobili aplikacija, leidžianti atlikti paiešką pagal nufotografuotą vaizdą. Taip pat ji gali kokybiškai konvertuoti matomą vaizdą į tekstą, atlikti interneto paiešką naudojant atpažintą tekstą [1] [2].
- Google Search – tai galingas paieškos įrankis, turintis daug papildomų funkcijų lengvinančių paiešką internete. Šio projekto kontekste dominantis šio paieškos variklio funkcionalumas yra labai gerai išvystytas klaidų aptikimas ir taisymas [2].
- TinyEye – idėja kažkuo panaši - ieško paveikslėlio savo paveikslėlių duomenų bazėje, tačiau tai tik naršyklėje veikiantis sprendimas [3].
- Amazon books, Google books – tai didelės knygų parduotuvės leidžiančios ieškoti knygų pagal daugelį kriterijų, tačiau neturi šiam projektui aktualios teksto paieškos knygų turinyje [4] [5].
- Layar – papildytosios realybės programa, atvaizduojanti papildomą informaciją telefono ekrane ant paruoštų teksto ar paveikslėlių fragmentų, tai yra geras pavyzdys kaip galima pateikti skaitmeninį ar medija turinį nukreipus kamerą į paprastą informacinį leidinį [4].

Taigi kažkokių panašių įrankių rasti galima, tačiau vieno patogaus telefone veikiančio sprendimo, kuris apjungtų visas šių servisų savybes, rasti nepavyko. Galima teigti, kad tai perspektyvi niša ir tokio produkto išpopuliarinimas rinkoje duotų naudos ir vartotojams.

Technologijos

Mobili platforma

Šiuo metu rinkoje karaliauja trys pagrindinės mobiliosios platformos Google Android, Apple iOS ir Microsoft Windows Phone. Palyginus šias platformas tarpusavyje buvo pasirinkta Android platforma kaip patraukliausia šiam projektui. Pagrindiniai Android platformos koziriai lėmę jos pasirinkimą aptarti žemiau.

Android atvira ir nemokama platforma.

Android išmaniojo telefono turėtojui praktiškai nereikia papildomų kaštų norint pradėti kurti šiai platformai skirtus sprendimus. Nėra nei brangių licencinių mokesčių nei kūrimo įrankių, už kuriuos reikėtų pakloti nemažą sumelę, tik vienkartinis registracijos mokestis [5] norint programėles talpinti Google Playparduotuvėje, kai tuo tarpu Apple ir Microsoft iš kūrėjų reikalauja susimokėti už metinį planą [6] [7]. Taip pat kadangi Android sistemos pagrindas ir pats programinės įrangos kūrimo įrankių rinkinys (SDK) yra atviro kodo, galima stebėti kiekvieną naują versiją ir jos kūrėjus informuoti kokie įspūdžiai su naująja versija.

Programavimo kalba yra Java.

Android aplikacijos rašomos Java kalba su gausiu bibliotekų rinkiniu. Kiekvienas turintis praktinių Java programavimo žinių gali nesunkiai pradėti programuoti Android aplikacijas. Java programuotojų rinkoje yra tikrai nemažai, o Android dokumentacija yra labai gausi. Šiuo metu pradėti programuoti Android platformai yra pats geriausias laikas.

Galingas kūrimo karkasas.

Android suteikia galimybę kurti vienos aplikacijos modeliu, pateikiant ją šimtams milijonų vartotojų, įvairiausiems įrenginiams, pradedant telefonais ir neapsiribojant planšetėmis [8]. Android taip pat pateikia įrankius kurti gerai atrodančias aplikacijas, suteikia galimybes išnaudoti technines kiekvieno įrenginio galimybes. Programos išvaizda automatiškai prisitaiko prie kiekvieno įrenginio ekrano galimybių, tuo pat metu leisdama programuotojui valdyti visus išvaizdos aspektus.

Tam, kad pagelbėti kuriant efektyvias aplikacijas, Android kūrimo įrankių komplektas (angl. ADT) siūlo pilną Java Integruotą kūrimo aplinką (angl. IDE) su išsamiais nustatymais skirtais kūrimui, klaidų šalinimui ir Android programėlių pakavimui [9]. Naudojantis integruotąja kūrimo aplinka, galima kurti ir bandyti programas ant bet kurio prieinamo Android įrenginio arba susikurti virtualius įrenginius, kurie imituotų pasirinktą techninę įrangą.

Atvira rinka programėlių platinimui ir pelno siekimui.

Google neriboja programėlių prekybos ir platinimo tik per jos sukurtą parduotuvę. Android programėles galima platinti per šimtus įvairių parduotuvių ar net susikurti savo parduotuvę. Ką pats sukuri, tą pats gali ir platinti bei tai daryti sau palankiausia forma, apribojimų šioje vietoje nėra. Apribojimai gali atsirasti tik tuo atveju jei naudojamos trečių šalių elektroninės parduotuvės, kurios tuos apribojimus ir sukuria, tačiau pasirinkimas yra didelis net ir tokių parduotuvėlių, tad viskas priklauso tik nuo pasirinkimo.

Google Play yra pagrindinė parduotuvė Android programėlių pardavimui ir platinimui. Joje paskelbta programėlė yra pasiekiama milžiniškam Android vartotojų kiekiui. Kaip atvira prekyvietė, Google Play leidžia pačiam kūrėjui spręsti kaip jis nori parduoti savo produktus. Galima skelbti programėles kada tik nori ir kokiems klientams nori. Galima rinktis skirstyti produktus visoms rinkoms arba taikytis į specifinius sektorius kaip įrenginių tipai, techninės galimybės ir t.t.

Apmokestinimo būdus galima naudoti įvairiausius, kurie tenkina kūrėjo verslo modelį. Galima platinti programėles nemokamai ir uždirbti iš jose atvaizduojamų reklamų, parduoti už vienkartinį mokestį, arba užsiimti prekyba pačios programėlės viduje, parduodant jos dalis kaip papildymus ar siūlant prenumeratas – visa tai skirta generuoti pajamas.

Google Play parduotuvė gali padėti ne vien tik didinti savo vartotojų bazę, tačiau ji padeda programėlėms pasiekti matomumą ir pripažinimą, kūrėjo ženklo įvertinimą. Programėlėms, kurių populiarumas auga, Google Play skiria vietas savaitės geriausių ir populiariausių sąrašuose, taip dar labiau skatindama programėlių plitimą.

Didelis potencialių diegimui įrenginių pasirinkimas.

Kasdien yra aktyvuojama per milijoną Android įrenginių pasaulyje [10]. Tai nebūtinai vien mobilieji telefonai ar planšetės. Android naudojamas televizoriuose, kompiuteriuose, namų elektronikoje. Yra netgi žaidimų konsolė Ouya veikianti Android pagrindu [11]. Šie skaičiai stipriai įtakojami daugiau nei 300 techninės, programinės įrangos gamintojų bei mobiliųjų tinklų operatorių.

Android sistemos atvirumas padarė ją mėgstamą tiek vartotojų tiek kūrėjų, tą rodo 1,5 milijardo programėlių parsisiuntimų skaičius iš Google Play parduotuvės kiekvieną mėnesį [12]. Dėl partnerių Android nepertraukiamai plečia techninės ir programinės įrangos ribas atnešdama vis naujas galimybes vartotojams ir kūrėjams. Kūrėjams Android inovacijos leidžia kurti galingas skirtingas aplikacijas, galinčias išnaudoti naujausias technologijas.

Programavimo kalba

Šio projekto įgyvendinimui tinkamų programavimo kalbų pasirinkimą apribojo mobiliosios platformos ir paieškos variklio pasirinkimas. Pastarieji programiniai paketai parašyti Java kalbos pagrindu [13] [14] ir jų susiejimui naudoti bet kurią kitą kalbą būtų tiesiog neoptimalu. Kadangi kurti reikia tiek mobiliąją aplikaciją tiek serverio pusės dalį, tai neapsunkinant projekto keliomis kalbomis ir jų apjungimo bendravimui, pasirinkta Java kalba. Ji turi labai daug naudingų funkcijų ir savybių lengvinančių programuotojo darbą.

Java – kartą parašyta veikia visose platformose

Daugelio platformų palaikymas yra didelis Java programavimo kalbos pliusas. Verslo požiūriu tai reiškia, kad programą parašius vieną kartą, ją paleisti galima visose populiariausiose platformose [15]. Java palaikoma daugumoje platformų ir jos palaikymas plinta vis labiau, ji integruota ar integruojama praktiškai visose pagrindinės operacinėse sistemose, ji įdiegta populiariose naršyklėse, kas Java kalbą padaro naudojamą praktiškai kiekviename įrenginyje, kuris prijungtas prie interneto.

Java – viena saugiausia programavimo kalbų

Vienas svarbiausių Java požymių yra tai, kad tiek pati kalba tiek visa jai skirta platforma buvo kurta atsižvelgiant į saugumą. Java platforma leidžia vartotojams parsisiųsti nepatikimą kodą internetu ir jį vykdyti savo kompiuteryje saugioje aplinkoje, kurioje tas kodas negali padaryti jokios žalos, jis negali užkrėsti sistemos virusu, negali skaityti ar rašyti failų be vartotojo leidimo ir t.t. Šie bruožai ir išskiria Java iš kitų platformų.

Java kalbos saugumo bruožai yra atidžiai nagrinėjami saugumo ekspertų visame pasaulyje. Su saugumu susijusios spragos, kai kurios net potencialiai labai rimtos, greitai randamos ir pataisomos. Kadangi Java žada saugumą, kiekviena rasta saugumo spraga yra visuomet didelis įvykis. Nors ir nėra ši kalba tobulai saugi, tačiau nėra kitos programavimo kalbos, kuri turėtų bent panašaus stiprumo saugumo lygį kaip Java [16], ir to pakanka, kad ji tenkintų kasdienius saugumo reikalavimus.

Orientuota kompiuteriniam tinklui

Sun korporacijos moto visuomet buvo „Tinklas yra kompiuteris“. Tad ir kuriant Java kalbą buvo teiktas didelis dėmesys bendravimui kompiuteriniu tinklu. Iš programuotojo perspektyvos, Java kalba leidžia nesunkiai dirbti su tinkliniais resursai ir kurti tinklu paremtas aplikacijas naudojant serverio/kliento ar kelių pakopų architektūrą. Tai reiškia, kad Java programuotojams sudaromos geros sąlygos pradėti kurti kompiuterinio tinklo aplinkoje.

Našumas

Java programos kompiliuojamos į nešiojamą tarpinę formą, žinomą kaip baitkodą, vietoje instrukcijų rinkinio procesoriui. Java virtuali mašina vykdo Java programą interpretuodama baitkodo instrukcijas. Tai reiškia, kad Java programos yra greitesnės negu pilnai interpretuojamos kalbos, tačiau lėtesnės negu C ar C++ programos sukompiliuotos į mašininį kodą.

Nors Java kentėjo dėl našumo problemų, Java VM greitis stipriai gerėja su kiekviena laida. Virtuali Mašina daugelyje sričių buvo stipriai optimizuota. Daugelyje specializuotų sprendimų yra realaus laiko kompiliatorius, kuris Java baitkodą verčia mašininiu kodu. Tokių sudėtingų kompiliatorių dėka Java programos gali būti vykdomos labai panašiu greičiu kaip i C ar C++ programos.

Optinis simbolių atpažinimas (OCR)

Greitis, tikslumas ir didelis programų pasirinkimas – dalykai, dėl kurių optinio simbolių atpažinimo (toliau OCR) programinė įranga tapo taip paplitusi paskutiniais metais. OCR programų naudotojai yra nuo valstybinių įstaigų iki studentų. Tačiau nedaugelis žino kaip OCR programos veikia ar kokia viso šio proceso istorija.

OCR algoritmai

Didžiąją istorijos dalį OCR programos veikė paremtos algoritmais, kurie atliko veiksmus vadinamus matricos atitikimais. OCR programa lygindavo kiekvieną simbolį dokumente su raidžių šablonais ir taip atlikdavo simbolių atpažinimą. Matricos atitikimas turi vieną didelį trūkumą: kadangi jis dirba lygindamas dokumento vaizdus su šablonіniais vaizdais, jis veikia tik su tais šriftais, kurių šablonus turi.

Šiuolaikinės OCR programos naudoja bruožų išskyrimą. Bruožų išskyrimo algoritmai bando ignoruoti kintamus simbolių elementus ir analizuoti tik nekintamas jų dalis. Pavyzdžiui didžioji raidė „H“ visuomet yra sudaryta iš dviejų lygiagrečių vertikalių ir vienos horizontalios linijos, jungiančios pirmąsias dvi. Kadangi OCR programa analizuoja tik nekintančius simbolių elementus, jai reikia tik vieno prototipo sudaryto iš nekintamų elementų, kiekvienai raidei. Didžiausias šio algoritmo pliusas, kad šiuo būdu OCR programos gali atpažinti daug šriftų.

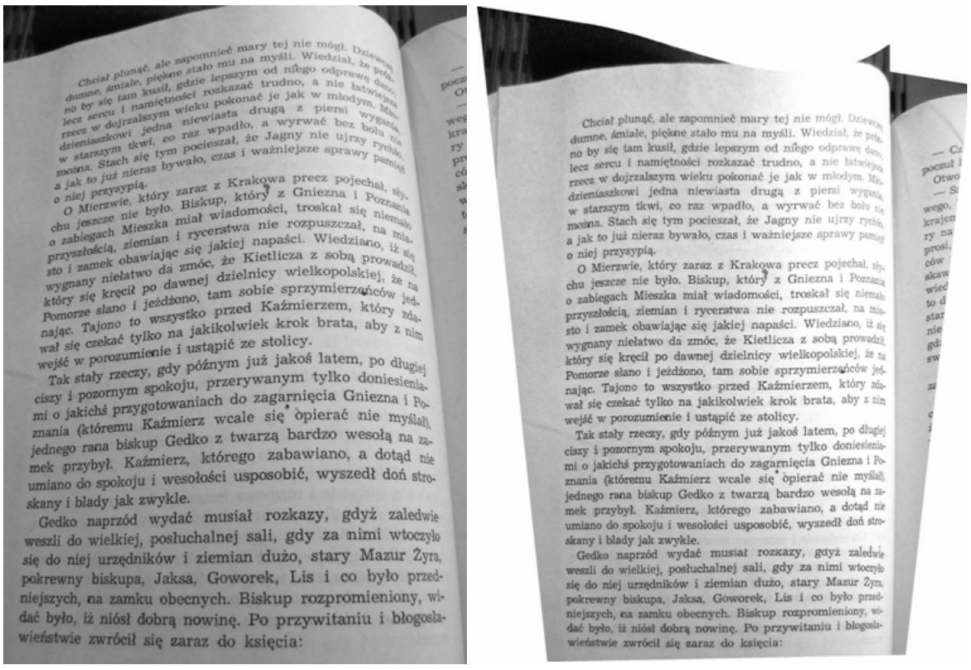
OCR programos tapo labai tikslios ir kai kuriais atvejais gali pasiekti net 99% tikslumą [17]. Žinoma tikslumas labai stipriai priklauso nuo dokumento kokybės, kurį reikia atpažinti. Švarios didelės rezoliucijos kopijos duoda geriausius rezultatus. Yra tyrimų teigiančių, kad OCR programos tiksliau nuskaito dokumentus, kurie nuskenuoti tik pilkais atspalviais [18]. Dauguma algoritmų bando optimizuoti dokumentus nuskaitymui papildomai apdorodami juos ir bandydami ištiesinti, išlyginti bei panaikinti dėmes, prieš atliekant vaizdo apdorojimą.

OCR veikimas

Skenuoto dokumento pavertimas į redaguojamą formatą yra išskirstytas į keletą žingsnių, kurių kiekvienas turi jam skirtą algoritmą atliekantį po dalį teksto atpažinimo proceso.

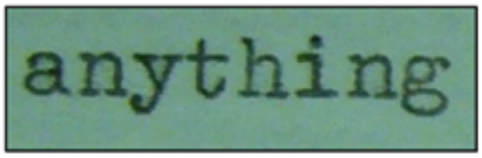
Pagrindiniai žingsniai yra šie:

- Paveikslėlio įkėlimas bitmap formatu – šaltinis gali būti failas ar nuoroda į atmintie bloką. Taip pat gera OCR programa privalo atpažinti daugelį paveikslėlių formatų, tiek vieno puslapio, tiek daugelio puslapių (tiff, pdf). PDF formato atpažinimas yra bene svarbiausias, nes daugelis dokumentų saugoma būtent šiuo formatu. Šio projekto atveju vaizdas bus imamas iš kameros, tad papildomų dokumentų formatų palaikymas yra neaktualus.
- Svarbių parametrų nustatymas ir inversija – daugelis OCR algoritmų tikisi gauti tam tikro dydžio ir tam tikrų spalvų paveikslėlius apdorojimui, tad paveikslėliui gali tekti pakeisti mastelį ar invertuoti spalvas.
- Paveikslėliai gali būti pasukti arba su daug šiukšlių – tam reikalingi tiesinimo ir švarinimo algoritmai [19].

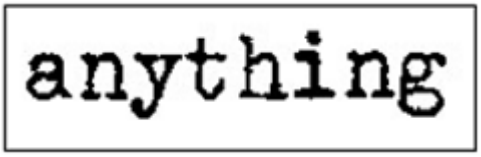


Paveikslėlis 2 – kairėje pradinis paveikslėlis, dešinėje tas pats paveikslėlis panaudojus tiesinimo algoritmą

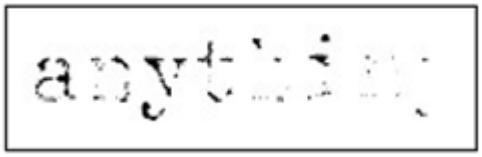
- Daugelis OCR algoritmų reikalauja dviejų spalvų paveikslėlio – tam reikia spalvotą ar pilkų atspalvių paveikslėlį padaryti juodai baltu. Šis procesas vadinamas binarizacija ir yra labai svarbus žingsnis, nes neteisinga binarizacija gali sukelti labai daug problemų [20] [21].



Paveikslėlis 3 – originalus paveikslėlis

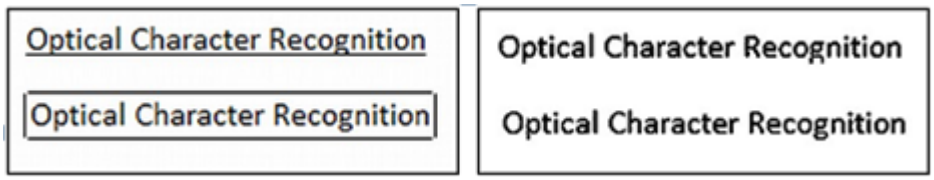


Paveikslėlis 4 – teisinga binarizacija



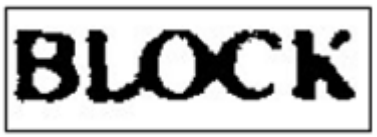
Paveikslėlis 5 – neteisinga binarizacija

- Linijų aptikimas ir šalinimas – šis žingsnis reikalingas pagerinti puslapio išdėstymo analizę, aptikti lenteles, supaprastinti pabraukto teksto atpažinimą.



Paveikslėlis 6 – tekstas prieš ir po linijų šalinimo

- Puslapio išdėstymo analizė – šiame žingsnyje OCR programa nustato ir pažymi visų paveikslėlio zonų tipus.
- Teksto eilučių ir žodžių aptikimas. Kartais nėra lengva nustatyti žodžių ar eilučių dėl skirtingų šriftų dydžių ir tarpų tarp žodžių.
- „Kombinuotų-sulūžusių“ simbolių analizė – dažnai simboliai yra išsiskaidę į kelias dalis arba liečiasi su kitais simboliais. Svarbu atpažinti tokius atvejus ir teisingai nustatyti kiekvieno simbolio ribas.



Paveikslėlis 7 - „sulūžusių“ ir sukibusių simbolių pavyzdys

- Simbolių atpažinimas – tai yra pagrindinis OCR programos algoritmas. Kiekvieno simbolio paveikslėlis konvertuojamas į atitinkamą simbolio kodą. Kartais šis algoritmas sugeneruoja kelis simbolių kodus neaiškiems simboliams. Pavyzdžiui simbolio „I“ atpažinimas gali sugeneruoti „I“, „|“, „l“, „1“ simbolių kodus, o galutinis bus parinktas vėliau.

- Žodyno palaikymas – šis žingsnis gali pagerinti atpažinimo kokybę, atvejais kai keli simboliai atrodo labai panašiai („I“ raidės atveju) žodynas gali padėti nuspręsti kuris simbolis turėtų būti [22].

Šis sąrašas neapima visko ką reikia atlikti, tai yra tik pagrindiniai veiksmai tinkamam atpažinimui atlikti. Kiekvienas šių žingsnių yra svarbus ir visas atpažinimo procesas gali žlugti jei bent vienas žingsnis nesuveiks kaip privalo. Kiekvienas algoritmas turi sugebėti teisingai atpažinti kuo didesnį kiekį paveikslėlių, todėl rinkoje yra tik keletas gerų universalių teksto atpažinimo programų. Norint pasiekti geriausius rezultatus OCR sistema turi sugebėti priderinti svarbiausius parametrus kiekvienam algoritmui, kartais tai vienintelė galimybė pagerinti atpažinimo kokybę. Deja šiuo metu nėra OCR programos, kuri sugebėtų konkuruoti su žmogaus akimis ir panašu, kad artimoje ateityje tokia nebus sukurta [23].

OCR variklis Tesseract

Teksto atpažinimui mobiliojoje aplikacijoje bus naudojamas vienas tiksliausių atvirojo kodo variklių – Tesseract [24]. Šis variklis naudojamas daugelyje projektų, jis pritaikytas veikti populiariausiose platformose ir jį nuolat tobulina Google programuotojai bei pateikia paruoštas bibliotekas pritaikytas naudoti Android programose [25].

Tesseract variklis nors ir atviro kodo yra ne ką prastesnis nei komerciniai šiuo metu rinkoje esantys teksto atpažinimo algoritmai. Šio variklio stiprioji dalis yra didelis funkcijų pasirinkimas. Tesseract variklį galima apmokyti atpažinti vieną ar kitą kalbą, kas labai pagerina tikslaus teksto atpažinimo statistiką, o apsimokymo sukurtais failais galima dalintis tarp kelių projektų [26].

Silpnoji šio variklio dalis yra ta, kad jis naudoja daugiakampę aproksimaciją, vietoje neapdorotų kontūrų, dėl to simboliai, kurių dalys skenavimo metu tampa atitrūkusiomis, atpažįstami neteisingai arba vienas simbolis išskaidomas į kelis.

Teksto paieška duomenų bazėje

Tekstinio atpažinimo algoritmai nėra tiek pažangūs, kad galima būtų iš jų gauti 100% teisingą tekstą. Dar pridėkime ir tai, kad teksto nuotraukos šiame projekte bus daromos mobiliuoju telefonu, o retas mobilusis telefonas gali užtikrinti tinkamą nuotraukos kontrastą ar aiškumą. Visa tai susumavus gauname tikrai mažą tikimybę gauti tikslų tekstą paieškai. Todėl norint atlikti paiešką turint tekstą su galimai neteisingais žodžiais ar raidėmis, reikia rasti tinkamą algoritmą paieškai, kuris galėtų leisti suklysti duodant tekstinę užklausą bei reitinguoti rezultatus pagal tai kiek rezultatas atitinka užklausą.

Duomenų bazė – paieškos variklis

Pradinis tikslas yra turėti kuo daugiau skaitmeninių knygų, kurių viduje ir bus atliekama paieška. Tam tikslui reikalinga duomenų bazė su vidiniu paieškos varikliu. Atlikus paiešką internete buvo rasti ir apsvarstyti kaip potencialūs sprendimai šie programiniai paketai:

- Solr [27] – labai greita atviro kodo, korporacinio lygio paieškos platforma, galinti atlikti pilno teksto paiešką, rezultatų reitingavimą, duomenų bazių integravimą, daugelio dokumentų formatų palaikymą.
- OpenFTS [28] – tai atviro kodo sudėtingas PostgreSQL duomenų baze grįstas paieškos variklis, kuris gali atlikti duomenų indeksavimą ir duomenų reitingavimą pagal pateiktą užklausą. Nebeatnaujinamas nuo 2009 metų.
- SphinxSearch [29] – atviro kodo pilno teksto paieškos serveris, parašytas C++ kalba. Gali naudoti duomenų bazes ar tiesiog failus duomenų kaupimui, palaiko indeksavimą.
- Xapian [30] – atviro kodo paieškos variklio biblioteka, parašyta C++ kalba. Palaiko reitinguotus paieškos rezultatus.
- MG4J [31] – nemokamas pilno teksto paieškos variklis pritaikytas didelėms dokumentų kolekcijoms ir parašytas Java kalba.
- Terrier [32] – atviro kodo paieškos variklis pritaikytas didelėms dokumentų kolekcijoms ir parašytas Java kalba.

Apache solr – atviro kodo paieškos platforma

Iš nagrinėtų paketų geriausias variantas yra Apache Software Foundation kuriama ir plėtojama Solr atvirojo kodo paieškos platforma [27]. Ši platforma yra nurodoma kaip labai patikima, tinkama tiek mažiems tiek labai dideliems duomenų kiekiams ir yra atspari klaidoms. Taip pat Solr užtikrina paskirstytą indeksavimą, replikavimą ir balansuotos apkrovos užklausų aptarnavimą, automatinį perjungimą ir atstatymą gedimo atveju bei kitus paieškos varikliams aktualius sprendimus. Šio variklio paieškos ir navigacijos savybės naudojamos daugelyje pasaulio didžiausių interneto svetainių.

Solr yra parašyta Java kalba ir veikia kaip savarankiškas pilno teksto paieškos serveris. Solr naudoja to paties Apache sukurtą Lucene paieškos biblioteką kaip branduolį pilno teksto paieškoms ir ją papildo aplikacijų programavimo sąsaja [33]. Solr programinio paketo galingos išorinės konfigūracijos galimybės leidžia ją pritaikyti naudojimuisi praktiškai bet kokia programa be papildomo Java programavimo, taip pat Solr turi išplėstą papildinių architektūrą, kurią galima panaudoti sudėtingiems platformos pakeitimams.

Pagrindiniai Solr paketo bruožai:

- Išplėstinės pilno teksto paieškos galimybės
- Optimizacija dideliems interneto srautams
- Standartinės atviros sąsajos (XML, JSON, HTTP)
- Beveik realaus laiko indeksavimas
- Lankstus ir pritaikomas XML konfigūracijomis
- Išsamūs įrankiai administravimui ir veiklos stebėjimui

Apytikslio teksto paieškos metodas

Apytikslio teksto paieškos sistema kuriama iš dviejų pagrindinių komponentų. Duomenų bazė duomenų kaupimui ir analizei bei mobiliojo telefono programėlė teksto atpažinimui ir duomenų bazės rezultatų atvaizdavimui.

Mobilioji programėlė daroma Android platformoje. Teksto atpažinimui naudojamos atviro kodo Tesseract optinio simbolių atpažinimo variklio bibliotekos.

Duomenų bazės pagrindas Linux operacinė sistema su Apache Solr duomenų bazė skirta greitam duomenų indeksavimui ir paieškai.

Šių komponentų tarpusavio bendravimui palengvinti ir atlikti papildomų rezultatų įterpimui naudojamas papildomas web servisas. Jame atliekamos užduotys, kurių nėra galimybės atlikti Solr duomenų bazėje ir tos, kurios reikalauja per daug resursų, kad būtų tinkamos atlikti mobiliojoje aplikacijoje.

Solr duomenų bazės deklaruojamos išplėstinės pilno teksto paieškos galimybės yra pagrindinis svertas jos naudojimui šiame projekte. Paieškos galimybėse apibrėžiamos projektui naudingos funkcijos rašybos tikrinimas, sinonimų parinkimas, galimybė tobulinti paieškos rezultatus keičiant paieškos dalių svarbumą, taikant logines sąlygas arba įtraukiant tik aktualią informaciją į paieškos rezultatus.

Solr duomenų bazė neturi galimybės grąžinti ieškomo teksto koordinačių, todėl koordinačių paieškos dalis atliekama tarpiniame web servise. Mobilioji aplikacija bendrauja tik su tarpiniu web servisu. Gauta užklausa iš mobiliosios aplikacijos yra siunčiama į Solr serverį, kad būtų surastas tikslus teksto atitikmuo. Web servisas gavęs patikslintą ieškomą tekstą atlieka to teksto paiešką knygos tekste skaičiuodamas raidžių, žodžių bei sakinių kiekį esantį nuo knygos pradžios iki ieškomo teksto vietos.

Sistemos prototipo architektūra

Reikalavimai

Tikslas ištirti programinės įrangos skirtos teksto atpažinimui iš paveikslėlio ir apytikslio teksto paieškos duomenų bazėje specifikas bei problemas. Šiame darbe bus koncentruojamasi ties pagrindinėmis šios sistemos funkcijomis. Viena iš jų yra surasti ir pritaikyti optimalų teksto atpažinimo algoritmą mobiliajam telefonui, norint išgauti kuo tikslesnį pradinį tekstą. Antroji dalis yra serverio dalies apytikslio teksto paieškos algoritmo radimas ir pritaikymas paieškai naudojant teksto atpažinimo dalies pateiktą tekstą.

Sukurta programine įranga naudosis popierinių leidinių skaitytojai, norėdami gauti papildomos informacijos apie skaitomą tekstą ar peržvelgti prie teksto prikabintą medija turinį.

Sukurtas produktas bus sudarytas iš dviejų programinių dalių. Mobiliosios aplikacijos ir serverio duomenų bazės.

Mobilioji aplikacija bus skirta telefono kamera užfiksuoti vaizdą, jį apdoroti, nuskaityti tekstą ir jį nusiųsti į serverį duomenų paieškai. Taip pat atvaizduoti iš serverio gautą medija turinį bei leisti jį atsidaryti.

Serverio duomenų bazėje bus realizuotas duomenų indeksavimas ir apytikslio teksto paieška. Gautas teksto fragmentas ieškomas duomenų bazėje ir pagal labiausiai atitinkantį rezultatą mobiliajai aplikacijai yra siunčiamas medija turinys ar nuorodos į susijusią informaciją. Taip pat serveryje realizuota internetinė prieiga turinio įkėlėjui sukelti elektroninius leidinius bei suvesti susijusią informaciją.

Vartotojo charakteristikos

Vartotojai norintys naudotis mobiliąja aplikacija turi turėti išmanųjį Android telefoną ir mokėti juo naudotis. Aplikacijos naudojimosi principas labai panašus į telefono kameros aplikacijos, tad papildomų reikalavimų aplikacijos naudojimuisi nėra.

Turinio įkėlėjai turi turėti naudojimosi kompiuteriu ir interneto naršykle pagrindus. Turinio kėlimui į sistemą bus naudojama interneto naršyklė.

Vartotojo problemos

Popierinių leidinių vartotojai norėdami gauti su tekstu susijusią skaitmeninę informaciją aplikacijos dėka, gali tiesiog nufotografuoti skaitomą tekstą ir gauti visą papildomą informaciją neįdedant daugiau pastangų. Skaitmeninių leidinių naudotojai turi galimybę spustelti aktyvias

nuorodas ir atsidaryti interneto šaltinius, kopijuoti tekstą citavimui ir t.t. Ši aplikacija tokią galimybę suteiktų ir popierinių leidinių vartotojams.

Vartotojo tikslai

Vartotojai gali pasiekti papildomą skaitmeninį turinį, kuris paprastai pasiekiamas tik skaitmeninių knygų skaitytojams.

Turinio įkėlėjas gali įkelti elektroninius leidinius ir jiems priskirti skaitmeninį turinį, nuorodas.

Bendri apribojimai

Šio projekto programinės įrangos apribojimai yra mobiliajai aplikacijai, ji veiks tik Android platformoje. Serverio dalis gali veikti tiek Linux, tiek Windows platformose, serveris turės būti pasiekiamas internetu.

Informacija apie užsakovo organizaciją

Projekto užsakovas – Sigitas Limontas, atstovaujantis įmonei UAB „Novitas“. Įmonė užsiima įvairios programinės kūrimu ir konsultavimu. Ši įmonė gauna teisę neribotą laiką naudotis sukurtu produktu, užsiimti produkto kopijų platininiu ir naudojimu kituose projektuose.

Projektavimas

Duomenų bazė

Galutiniam projektui įgyvendinti bus reikalingos kelios duomenų bazės. Pirminiams sistemos variantams naudojama tik viena, Solr programinio paketo, duomenų bazė. Ji optimizuota tekstinės informacijos kaupimui ir indeksavimui bei greitų atsakymų pateikimui. Papildomos informacijos, kuri būtų atvaizduojama radus tam tikrą knygos puslapį, kaupimui bus reikalinga atskira duomenų bazė su informacijos įkėlimo sąsaja. Sistemos vystymas pradžioje vykdomas iteraciniu būdu. Pirmojoje ir antrojoje iteracijose naudojama tik Solr paketo vidinė duomenų bazė.

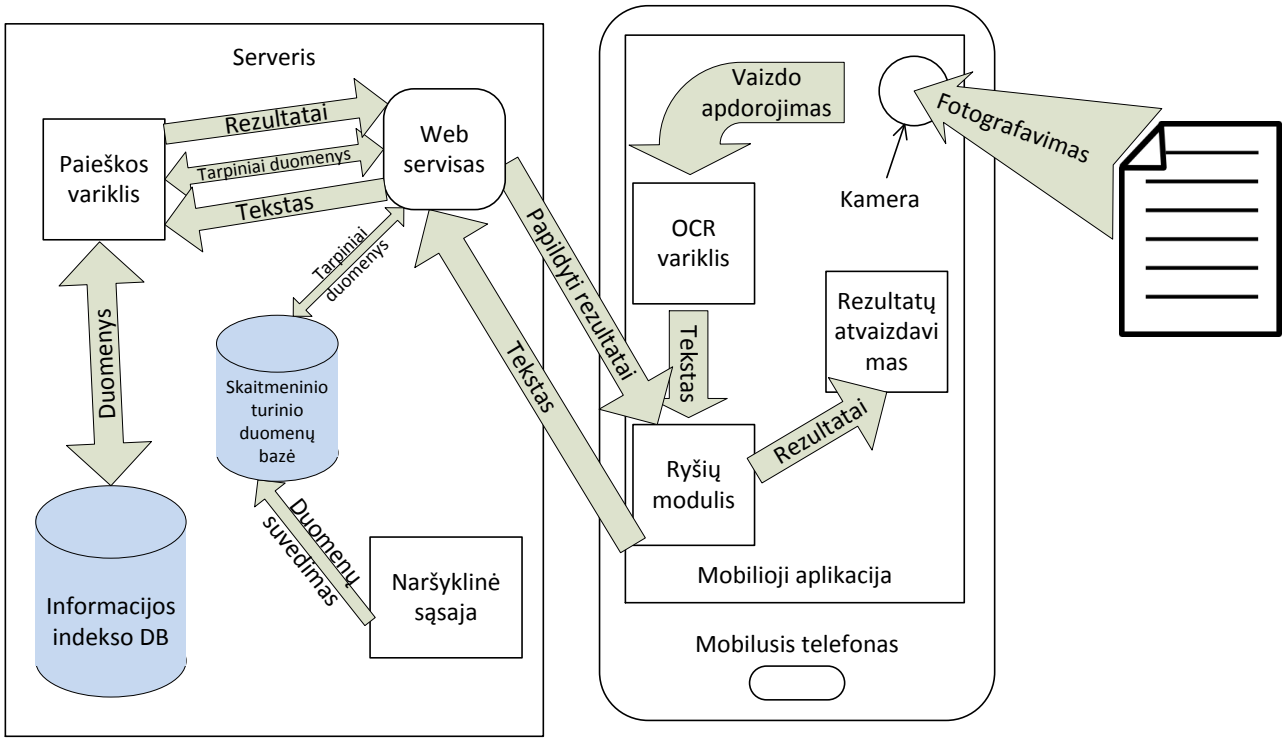
Projektinis modelis

Sistema susideda iš trijų funkcinių paketų:

- Duomenų bazė serveryje – skirta indeksuoti skaitmenines knygas bei pateikti atsakymus pagal paieškos užklausas.
- Tarpinis web servisas – pagalbinis servisas papildomų duomenų pateikimui, bei papildomiems teksto paieškos veiksmams atlikti

- Mobilioji aplikacija – pagrindinė vartotojo sąsaja, teksto atpažinimo modulis, duomenų siuntimo ir rezultatų atvaizdavimo sąsaja.

Komponentų diagrama



Paveikslėlis 8 principinė komponentų diagrama

Pagrindiniai sistemos paketai susideda iš smulkesnių komponentų.

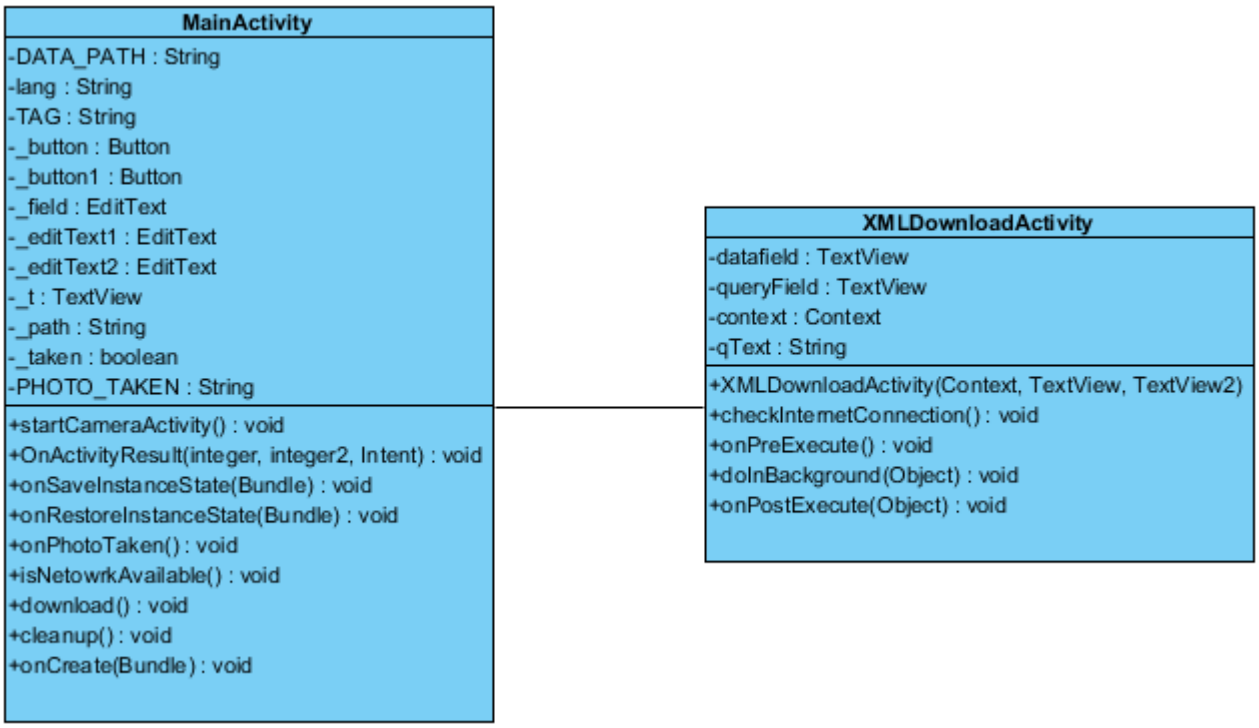
Mobiliosios aplikacijos pagrindiniai komponentai yra šie:

- Telefono kameros aplikacija nuotraukos darymui
- Teksto atpažinimo bibliotekos iš Tesseract projekto
- Vidinis ryšių modulis teksto siuntimui ir rezultatų gavimui
- Rezultatų ekrane atvaizdavimo modulis

Serverio duomenų bazių numatoma struktūra susidės iš:

- Paieškos variklio Solr, kuris indeksuoja skaitmenines knygas ir pateikia rezultatus pagal ieškomus žodžius
- Vartotojo sąsajos skirtos įkelti skaitmenines knygas bei papildomo turinio susiejimui su leidinių tekstais
- Atskira duomenų bazė papildomo skaitmeninio turinio kaupimui

Klasių diagramos



Paveikslėlis 9 Mobilios aplikacijos klasių diagrama

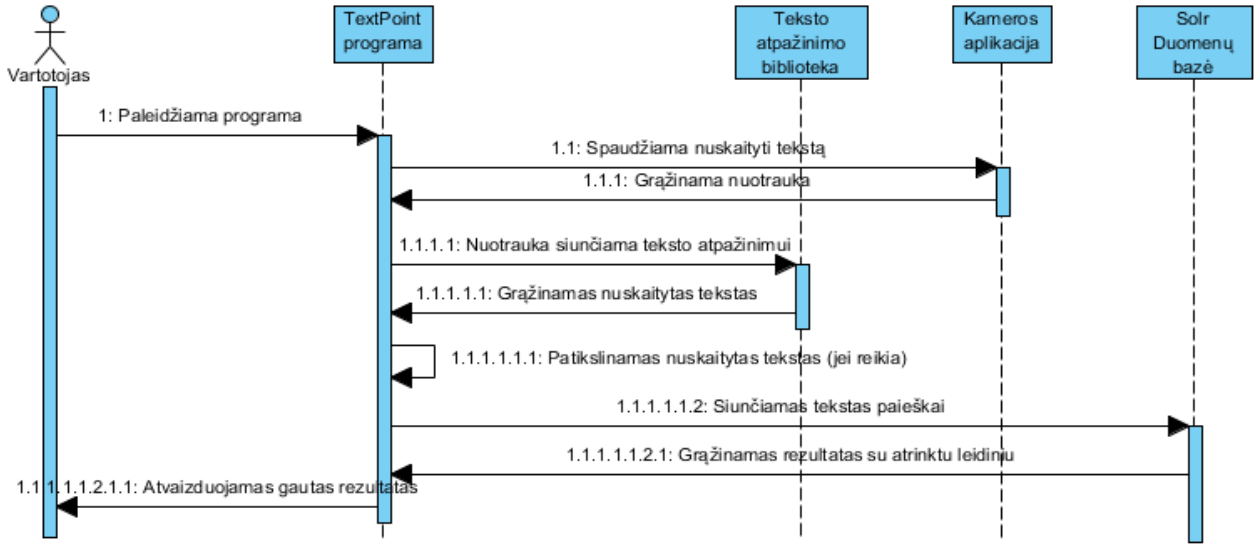
Klasė „MainActivity“ yra pradinė, iškviečiama klasė.

Joje realizuota šie metodai:

- startCameraActivity() – iškviečia telefono kameros aplikaciją
- onPhotoTaken() – nuskaito tekstą iš kameros padarytos nuotraukos
- OnActivityResult(int, int, Intent) – tikrina ar sėkmingai padaryta nuotrauka
- onSaveInstanceState(Bundle) – vykdomas perjungimas tarp šios ir kameros programų
- onRestoreInstanceState(Bundle) – vykdomas perjungimas tarp šios ir kameros programų
- isNetworkAvailable() – tikrina ar įjungtas bent vienas tinklas ir yra prisijungta
- download() –iškviečiamas XML rezultato parsiuntimo metodas siunčiant nuskaitytą tekstą
- cleanup() – valomas nuskaityto teksto laukas
- onCreate() – iniciavimo metu priskiriami kintamieji, programa paruošiama darbui

Sekų diagrama

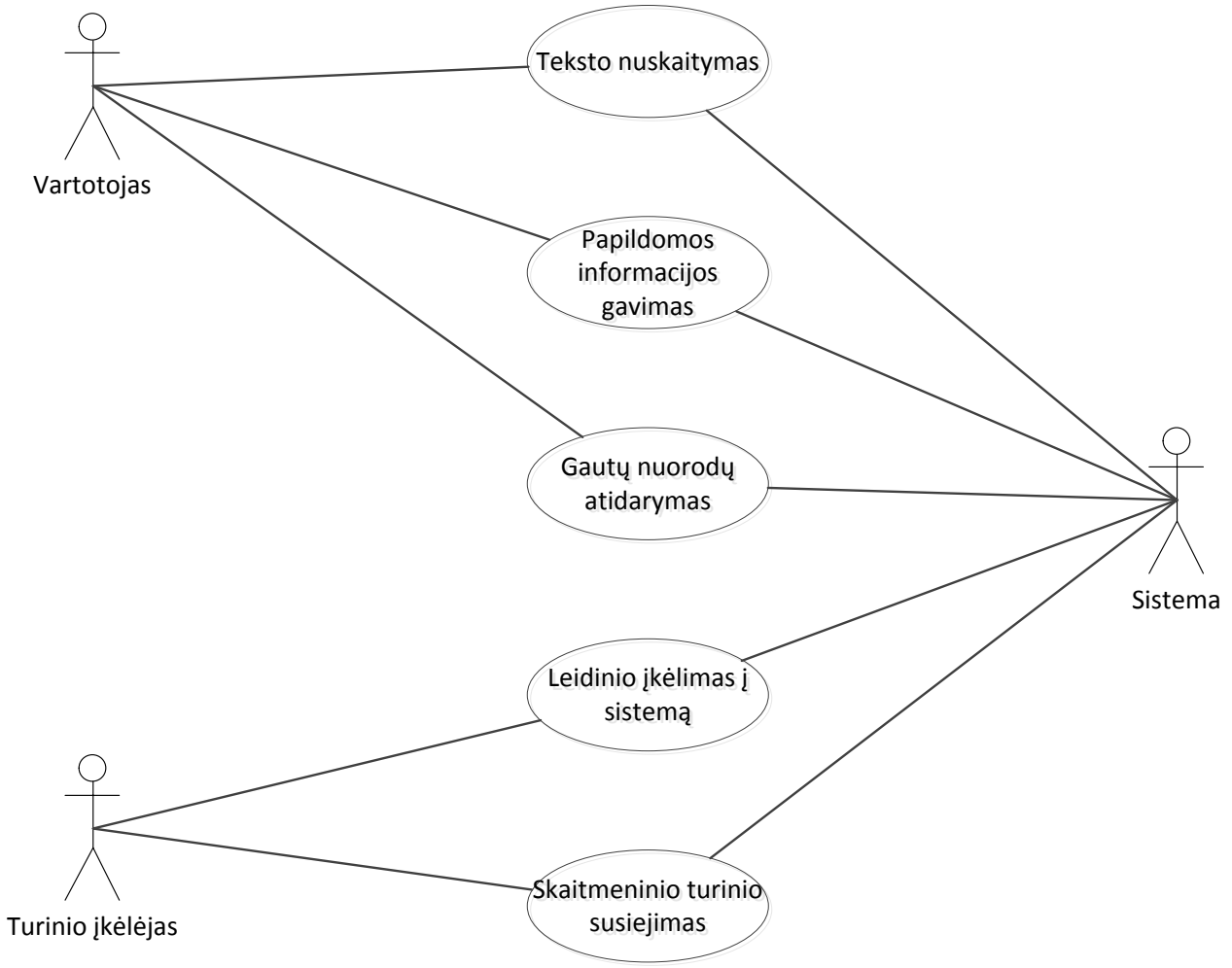
Ši sekų diagrama atvaizduoja knygos teksto nuskaitymo ir rezultatų pagal nuskaitytą tekstą gavimo procesą.



Paveikslėlis 10 Sekų diagrama, kaip gaunamas rezultatas pagal nuskenuotą tekstą

Panaudos atvejų diagrama

Sistemos veikimui reikalingi du veikėjai: vartotojas ir turinio įkėlėjas. Vartotojas skenuos tekstą mobilia aplikacija ir mobili aplikacija jam pateiks informaciją iš serverio, informacija gali būti nuorodos į interneto šaltinius. Video, audio ar tekstinė medžiaga atvaizduojama tiesiai ekrane. Turinio įkėlėjas turės sąsają per naršyklę elektroninių leidinių įkėlimui į sistemą bei skaitmeninio turinio priskyrimui prie leidinių teksto. Programinės įrangos panaudojimo atvejai pateikti paveikslėlyje.



Paveikslėlis 11 – Panaudos atvejų modelis

Prototipo kūrimas

Prototipo kūrimas pradėtas kurti iteracinio proceso modeliu. Tokiu modeliu iš karto kuriama veikianti sistema su minimaliais reikalavimais, tačiau stengiantis realizuoti kuo daugiau jos dalių. Vėliau kiekvienos iteracijos metu tos dalys tobulinamos kol sukuriamas pilnas sąlygas tenkinantis produktas.

Pirma iteracija

Aprašymas ir užduotys

Pirmosios iteracijos tikslas yra sukurti principinę sistemą atliekančią esminius veiksmus, reikalingus galutiniam produktui.

Iteracijos užduotys:

- Paleisti Solr serverį ir testinę OCR programėlę ant mobilaus telefono
- Užpildyti Solr knygų duomenų bazę keletu knygų. Sukonfigūruoti sistemą, kad būtų randamas leidinys pagal įvestą tekstą.

- Mobiliosios aplikacijos nuskaityto teksto paieška tiesiai duomenų bazėje bei rezultatų atvaizdavimas mobiliojoje aplikacijoje.

Realizavimas

Sudiegtas Linux serveris su Solr programiniu paketu skirtu knygų skaitmeniniu formatu kaupimui ir indeksavimui. Solr programinis paketas turi įrankių rinkinį duomenų bazės ir indeksavimo valdymui, šių įrankių dėka sukonfigūruotas knygų PDF formatu nuskaitymas ir užkrovimas į duomenų bazę. Sukonfigūruota sistema geba grąžinti knygos pavadinimą, autorių bei teksto fragmentus, kuriuose randami ieškomi žodžiai. Taip pat rezultatuose pateikiamas skaitinis kiekvieno rezultato įvertinimas palyginimui, kuris iš rezultatų labiausiai atitinka ieškomą tekstą.

Sukurta Android mobili aplikacija skirta nufotografuoti knygos lapą, jį nuskaityti, nuskaitytą tekstą siųsti serveriui kaip užklausą, gauti atsakymą iš serverio ir jį atvaizduoti.

Nufotografuoto lapo nuskaitymas vyksta naudojantis atvirojo kodo bibliotekomis iš Tesseract OCR projekto. Nuskaitytas tekstas siunčiamas serveriui, iš serverio gaunamas atsakymas JSON duomenų apsikeitimo formatu. JSON formatas pasirinktas dėl to, kad jis palaikomas Android aplinkoje be papildomų bibliotekų bei yra lengviau skaitomas nei XML kai gaunamas neformatuotas atsakymas.

Antra iteracija

Aprašymas ir užduotys

Antrosios iteracijos tikslas yra gauti iš serverio informaciją apie ieškomos frazės vietą knygoje.

Iteracijos užduotys:

- Pakeisti Solr serverio konfigūraciją arba sukurti jam priedėlį, kuris indeksuotų visų leidinių žodžių vietas įkeliant leidinius į sistemą ir galėtų pateikti ieškomų žodžių vietas vieno leidinio atžvilgiu.
- Sukurti tarpinį servisą tarp mobilios aplikacijos ir duomenų serverio Solr, kuris dalinai apdorotų užklausas iš mobilios aplikacijos ir būtų tarpinis komponentas duodantis papildomą turinį susijusį su pateiktomis užklausomis.

Realizavimas

Sukurtas tarpinis web servisas užklausų iš mobiliosios aplikacijos perdavimui į Solr serverį, bei serverio atsakymo apdorojimui ir pateikimui mobiliajai aplikacijai. Jo paskirtis yra papildyti serverio atsakymą vartotojui naudingu turiniu, kuris būtų talpinamas atskiroje duomenų bazėje tekstiniu arba kuriuo nors medija formatu.

Šiuo metu servisas atlieka dalį teksto paieškos funkcijų ir nepateikia jokio papildomo turinio. Papildomo turinio pateikimas atidedamas vėlesnėms sistemos iteracijoms. Dalis paieškos funkcijos atliekama šiame servise, nes Solr serveris neturi galimybės pateikti ieškomos frazės koordinačių ieškomai frazei. Solr serveris turi galimybę kaupti ir grąžinti pavienių žodžių koordinates, tačiau, norint kad jis teiktų reikalingą informaciją apie frazę ir tik joje naudojamus žodžius, reikia kurti priedėlį pačiam Solr serveriui, tai yra sudėtingesnis procesas, nei paieškos funkciją perkelti ant web serviso todėl apsiribota paprastesniu sprendimu. Solr serveris atlieka leidinio atitikimo funkciją, suranda ieškomą frazę leidinyje, grąžina leidinio pavadinimą, autorių bei patikslintą frazę. Web servisas atlieka patikslintos frazės paiešką leidinyje pateikdamas simbolių ir žodžių kiekį prieš rastą ieškomą frazę ir visą informaciją pateikia mobiliajai aplikacijai.

Sukurto prototipo veikimas

Sukurtas prototipas, kuris atlieka pagrindines funkcijas priskirtas kiekvienam iš trijų sistemos dalių. Prototipo sistema sugeba telefonu pasinaudoti mobiliojo telefono kamera ir padaryti nuotrauką. Padarytą nuotrauką nuskaityti pasinaudojant atviro kodo teksto atpažinimo paketu, nuskaitytą tekstą išsiųsti tarpiniam web servisui. Servisas atlieka teksto paiešką.

Gautą tekstą servisas siunčia Solr duomenų bazei, Solr grąžina rastą knygos pavadinimą ir teksto fragmentą, kuriame rastas ieškomas tekstas. Tarpinis servisas užklausia pilno knygos teksto iš Solr duomenų bazės pagal gautą knygos pavadinimą. Servisas gavęs knygos tekstą ir patikslintą teksto frazę atlieka simbolių ir žodžių skaičiavimus kiek nuo knygos pradžios yra nutolęs ieškomas tekstas. Atlikus skaičiavimus siunčiamas atsakymas mobiliajai aplikacijai.

Mobilioji aplikaciją iš gauto rezultato atvaizduoja knygos pavadinimą ir ieškomos frazės vietą toje knygoje simbolių skaičiumi.

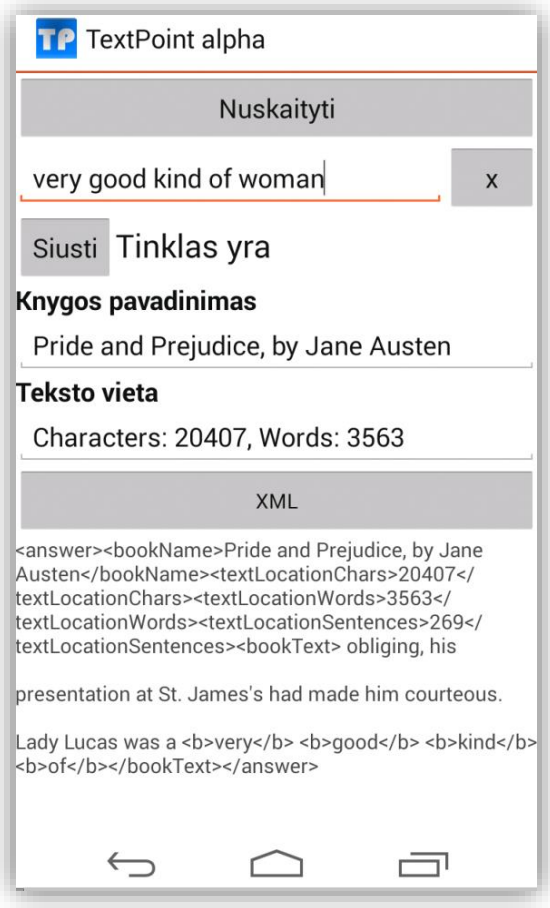
Mobilioji aplikacija

Mobilioji aplikacija buvo realizuota tik iki vystymui skirto lygio, ji nebuvo pritaikyta galutiniam vartotojui. Svarbiausios jos funkcijos buvo integruoti atviro kodo Tesseract optinio simbolių atpažinimo bibliotekas ir jomis pasinaudoti nuskaitant tekstą iš kamera padarytos nuotraukos.

Mobiliosios programos pagrindiniai elementai:

- Mygtukas Nuskaityti – perjungia į kameros režimą ir leidžia padaryti nuotrauką
- Pirmas teksto įvedimo laukas – jame padarius nuotrauką yra automatiškai įkeliamas iš nuotraukos nuskaitytas tekstas. Tekstą pagal poreikį galima įvesti ar pataisyti ranka.
- Mygtukas Siųst – įvedimo lauko tekstas siunčiamas web servisui.

- Knygos pavadinimo ir teksto vietos laukai – juose atvaizduojama dalis informacijos gauta iš tarpinio web serviso.
- Mygtukas XML – jį paspaudus žemiau atvaizduojama visa iš serviso gauta informacija.



Paveikslėlis 12 mobiliosios aplikacijos darbinis langas

Tarpinis web servisas

Tarpinis web servisas buvo sukurtas atliekti užduotis, kurių negali atlikti Solr duomenų bazė ir kurios yra per daug resursų reikalaujančios, kad galėtų būti atliekamos mobiliajame telefone. Servisas iš mobiliosios aplikacijos gautą tekstą perduoda duomenų bazei, iš jos gauna rezultatus su labiausiai tikėtinomis teksto vietomis duomenų bazėje esančiose knygose. Servisas pasirenka rezultatą su geriausiu įvertinimu ir panaudoja jame įrašytą knygos pavadinimą bei patikslintą frazę iš knygos. Knygos pavadinimas siunčiamas kitu užklausimui Solr serveriui, kad gautų visą knygos tekstą.

Gautą knygos tekstą servisas naudoja atlikti patikslintos frazės paieškai. Turint abu tekstus atliekamas žodžių ir simbolių skaičiavimas knygos tekste iki tos vietos kurioje randamas patikslintas teksto fragmentas. Atlikus skaičiavimus gauti duomenys siunčiami mobiliajai aplikacijai.

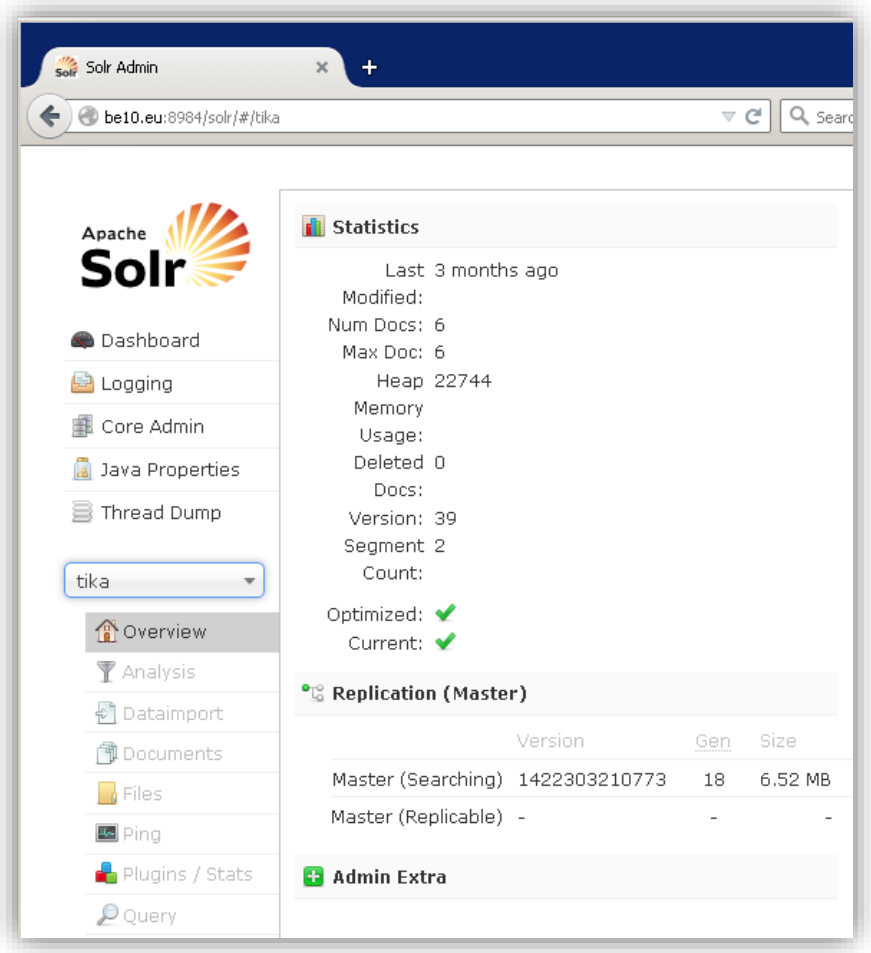
Iš serviso sąsajos lango galima matyti, kad atsakyme siunčiamame mobiliajai aplikacijai yra siunčiama informacija apie teksto koordinates, knygos pavadinimas ir teksto fragmentas, kuriame yra paryškinti ieškomoje frazėje rasti žodžiai.

Paveikslėlis 13 tarpinio web serviso langas skirtas pateikti rezultatus mobiliajai aplikacijai

Solr duomenų bazė

Solr duomenų bazė buvo sudiegta ir sukonfigūruota indeksuoti knygų tekstus ir automatizuotai įtraukti knygas pdf formatu. Solr konfigūracijoje atlikti nustatymai, skirti išskirstyti knygų autorius, pavadinimus, tekstu į atskirus skirtingų tipų laukus, kiekvienam pritaikant indeksavimą bei knygos tekstui kiekvieno žodžio koordinačių duomenis. Knygų įkėlimui į Solr sistemą naudojamas Apache Tika papildinys, galintis apdoroti didesnį kiekį duomenų failų tipų.

Solr turi naršyklės sąsają duomenų bazės stebėjimui ir valdymui. Joje galima atlikti dalį konfigūracijos, inicijuoti knygų įkėlimą į sistemą, stebėti sistemos apkrovimą ir resursų naudojimą. Joje yra įrankiai sukauptos informacijos analizei ir paieškos optimizavimui naudojant grafinę sąsają paieškos parametrų keitimui.



Paveikslėlis 14 Solr serviso valdumo langas

Prototipo kokybė

Prototipas buvo sukurtas ir veikiantis. Kiekviena jo dalis sugebėjo atlikti suplanuotas užduotis. Tačiau galutinis veikimo rezultatas neteikia tinkamų rezultatų tolimesniam sistemos plėtojimui. Mobiliąja programa daromos nuotraukos reikalauja labai gero apšvietimo, norint kad tekstas būtų atpažįstamas kokybiškai, Paieškos duomenų bazė Solr ne visada tinkamai suranda ieškomą teksto fragmentą. Siekiant išsiaiškinti kaip galima patobulinti gaunamų rezultatų kokybę nuspręsta atlikti tyrimą naudojant Solr bazės nustatymų analizę. Tyrimo metodas plačiau aprašytas 4 skyriuje.

Prototipo tolesnio tobulinimo galimybės

Perspektyvų Lietuvos rinkoje kol kas nematyti. Bent jau ne prototipo stadijoje. Kadangi tikslinė rinka yra anglų kalba šnekantys ir angliškų turiniu besinaudojantys žmonės, tai Lietuvoje toks daiktas pradinėse stadijose tikrai dar nebus paklausus.

Lietuvių kalba yra mažai paplitusi pasaulyje ir sudėtingesnė nei anglų, nors teksto atpažinimui tai didelės įtakos neturi, tačiau lietuviškų raidžių turimi papildymai varnelėmis ar

nosinėmis, palyginus su anglų kalba, tikrai apsunkina jų atpažinimą. Anglų kalboje tiek žodžiai, tiek jų variacijos yra paprastesnės ir lengviau perkandamos programiniams algoritmams, o kuriant prototipą svarbiausia sukurti veikiantį pavyzdį, kurio funkcionalumą kitomis kalbomis bus galima išplėsti vėliau, kai bus pasiektas tenkinamas bandomųjų duomenų apdorojimo lygis.

Apytikslio teksto paieškos tyrimas

Tyrimo metodas

Idėja: sugadintų tekstų paiešką automatizuotai atlikti su skirtingais paieškos parametrais.

Apytikslio teksto paieškos analizei reikalingų duomenų surinkimas realizuojamas atrinkus keletą frazių iš naudojamų knygų, atliekama tų frazių paieška naudojant skirtingus paieškos parametrus, išnaudojant žinomas Solr serverio teksto paieškos funkcijas pritaikant skirtingus paieškos parametrus ir lyginant gautus rezultatus. Iš rezultatų palyginimo turėtų paaiškėti, koks paieškos algoritmas yra optimalus ir ar papildomų Solr funkcijų naudojimas paieškos optimizavimui gali pagerinti netikslaus teksto paiešką duomenų bazėje.

Programinė įranga

Netikslaus teksto paieškos eksperimentui reikalingų nuoseklių duomenų generavimui ir apdorojimui buvo sukurtos dvi programos. Pirmoji programa skirta turimo frazių rinkinio laipsniškam gadinimui, programos veikimas detaliau aprašytas 4.1.2 poskyryje. Antroji programa skirta atlikti paiešką naudojant pirmosios programos paruoštą duomenų rinkinį ir su kiekvienu to rinkinio tekstu atlikti paiešką Solr duomenų bazėje, taip pagal gautus rezultatus galima įvertinti kiekvieno algoritmo pranašumą lyginant su pradiniu variantu. Antrosios programos veikimas detaliau 4.1.3 poskyryje.

Naudoti tekstai

Tekstinės frazės testams buvo atrinktos atsitiktiniu būdu imant 4-5 frazes iš knygų, kurios yra užkrautos į Solr duomenų bazę. Parinktos frazės yra 1-3 sakinių ilgio – tai yra maždaug tiek kiek pagal projekto idėją turėtų aprėpti mobiliojo telefono kamera.

Programa sugeneruoja po 20 failų kiekvienai frazei. Kiekvienas iš tų failų turi pradinę frazę su tam tikru kiekiu atsitiktinai pakeistų simbolių. Norint imituoti skirtingą teksto sugadinimo laipsnį bei užtikrinti nuoseklų ir didėjantį teksto išgadinimą buvo parinktas toks algoritmas:

Pradinė frazė apdorojama cikle, kiekvieno ciklo metu atsitiktinai pakeičiama dalis raidžių į taip pat atsitiktai sugeneruotus simbolius, taip pakeista frazė įrašoma į laikiną failą. Šiuo atveju ciklas kartojamas 20 kartų, kiekvienu ciklu frazėje pakeičiant po 5 simbolius, taip po 20 ciklų frazėje

būna pakeista apie 100 simbolių ir kiekvieno ciklo rezultatas yra failas su n*5 pakeistais simboliais (n – iteracijos numeris). Reikia pastebėti, kad cikluose keičiami simboliai nėra stebimi atskiromis sąlygomis, tad galutiniame variante, gali būti pakeista mažiau nei 100 simbolių, nes tas pats simbolis gali būti pakeistas kitu kelis kartus.

Įvertinus tai, kad naudojant optinį teksto atpažinimą yra mažesnė tikimybė, kad tarpas ar skyrybos ženklas bus atpažinti kaip kitokie simboliai, buvo sugeneruotas antras duomenų rinkinys su šiek tiek pakeistu atsitiktiniu raidžių keitimo algoritmu. Pirminis variantas teksto frazėse atsitiktinius simbolius keičia kitais nepriklausomai nuo to ar pasirinktas simbolis iš teksto yra raidė ar tarpas. Todėl atsitiktinių raidžių keitimo algoritme buvo įvesta papildoma sąlyga tikrinti ar planuojamas keisti simbolis yra raidė arba skaičius. Jei sąlyga netenkinama, iš naujo parenkamas atsitiktinis frazės simbolis ir taip tol, kol keitimas atliekamas raidei arba skaičiui.

Naudoti paieškos algoritmai (SOLR nustatymai)

Solr duomenų bazė buvo pasirinkta dėl deklaruojamų teksto paieškos galimybių ir daugybės nustatymų palengvinančių paieškos vykdymą. Su pirmąja programa sugeneruotus duomenų failus naudojame patikrinti keletą Solr paieškos nustatymų. Naudojamų funkcijų sąrašas:

- „Fuzzy“ – tai nustatymas, pagal kurį kiekvienam pažymėtam žodžiui yra ieškoma panašių žodžių, kurie skiriasi viena dviem raidėmis nuo ieškomojo.
- „Boost“ – tai nustatymas, pagal kurį pažymėtiems žodžiams pridedama „svorio“, t.y. žodžiai su šiuo požymiu turi didesnę reikšmę, nei kelių mažareikšmių žodžių aptikimas. Ši reikšmė bandant paiešką naudojama dviem būdais – reikšmė dedama tik žodžiams ilgesniems nei 4 simboliai, arba trumpesniems nei 5, rezultatų palyginimas vėlesniuose skyriuose.
- „Proximity“ – pagal šį nustatymą ieškoma pažymėtos frazės žodžių tam tikru atstumu vienas nuo kito, pagal šią sąlygą rezultatas yra tinkamas tik jeigu ieškomi žodžiai yra apibrėžtu atstumu vienas nuo kito.
- „Fuzzy+Boost“ – pirmų dviejų kombinacija, bandoma ieškoti duodant „svorio“ ilgesniems nei 4 simbolių žodžiams tuo pačiu ieškant į juos panašių.

Tyrimo rezultatai ir įžvalgos

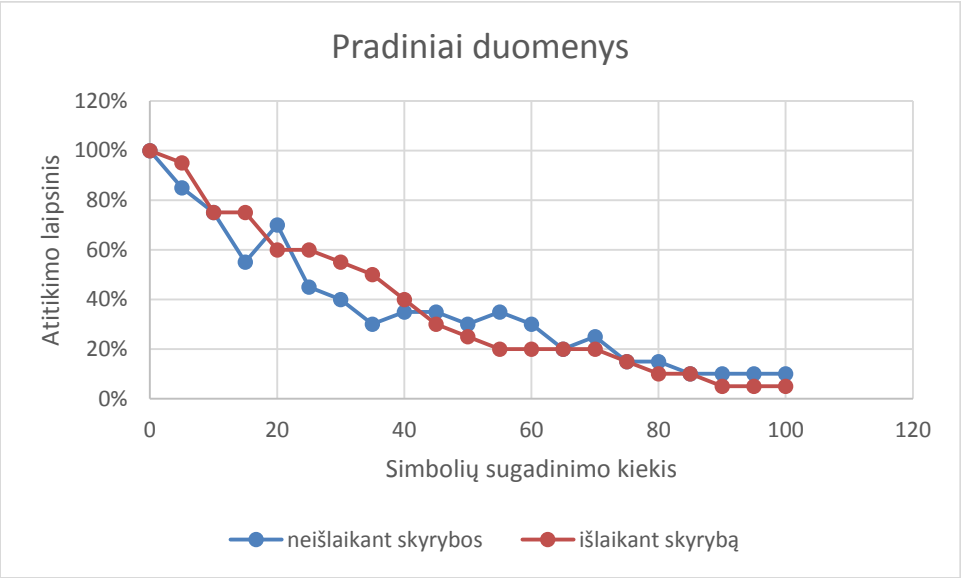
Rezultatų apdorojimas

Atlikus paieškas su anksčiau minėtomis frazėmis ir paieškos sistemos nustatymais, gaunamos matricos su rasto teksto koordinatėms kiekvienam iš paieškos algoritmų. Vienos matricos pavyzdys yra priede pavadinimu „Pavyzdinė frazių paieškos rezultatų matrica“. Teksto koordinatės yra matomų simbolių kiekis nuo knygos pradžios iki rastos teksto frazės. Tokių duomenų palyginimui

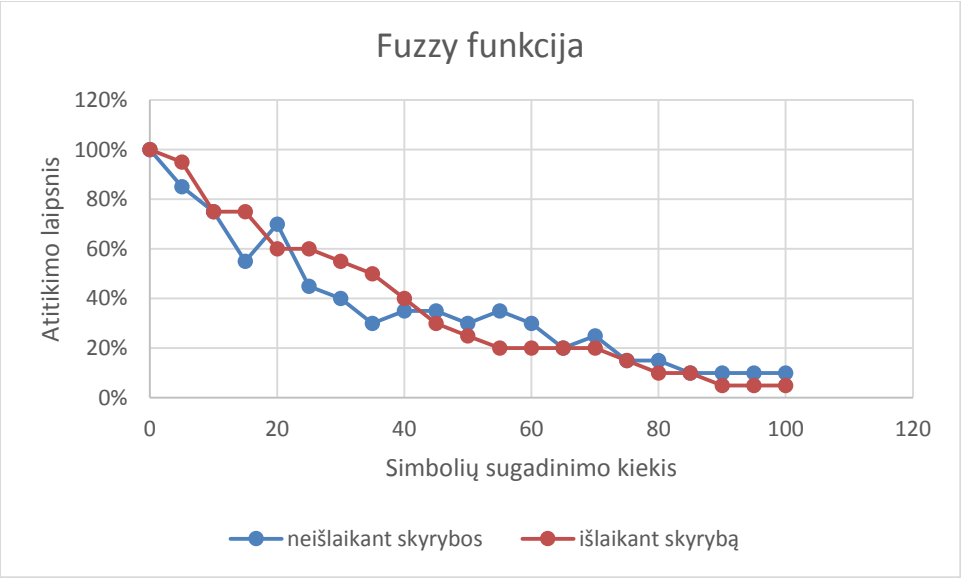
matrica konvertuojama į dvejetainę matricą, kur 1 reiškia, kad frazė rasta arti arba reikiamoje vietoje, o 0 reiškia, kad frazės rasti nepavyko. Suskaičiavę tokios dvejetainės matricos stulpelių vidurkius gausime įverčius kiek kurio sugadinimo lygio frazių pavyko teisingai aptikti naudojantis pasirinktą metodą. Naudojant šiuos vidurkius nubraižome grafikus ir taip galėsime vizualiai palyginti algoritmų efektyvumą tarpusavyje.

Rezultatai

Rezultatai skirstomi į du duomenų rinkinius. Pirmasis duomenų rinkinys nuo antrojo skiriasi tuo, kad antrajame simuliuojant duomenų gadinimą artimesnį optinio teksto atpažinimo algoritmui buvo išlaikyta skyryba ir tarpai tarp žodžių, keičiant tik raides.

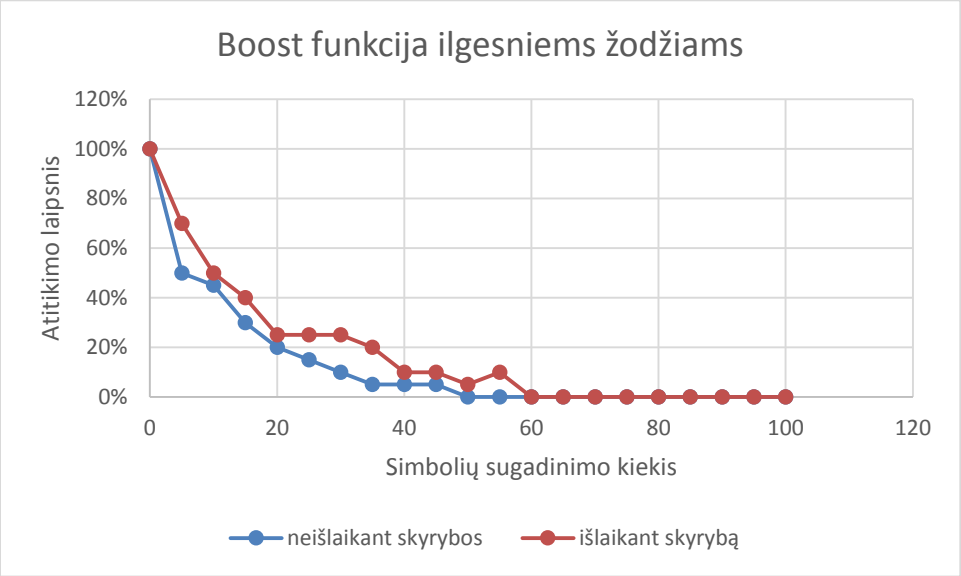


Paveikslėlis 15 pradinių duomenų paieškos sėkmingumo grafikas

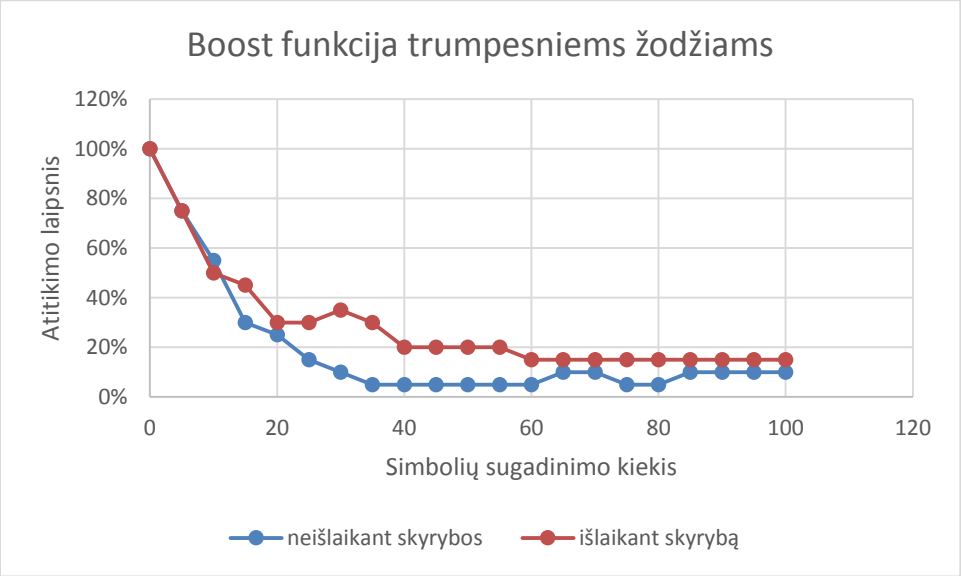


Paveikslėlis 16 Fuzzy nustatymo duomenų paieškos sėkmingumo grafikas

Šiuose grafikuose matyti, kad Fuzzy algoritmo taikymas paieškoje nedavė jokios naudos, paieškos rezultatai yra identiški pradiniams duomenims, kai paieškai netaikomi jokie papildomi nustatymai. Lyginant duomenų rinkinių paieškos rezultatus, vertesnis yra tas rezultatas, kuris rodo daugiau sėkmingų paieškų esant labiau sugadintam tekstui. Šiuo aspektu antrasis duomenų rinkinys turi prastesnius rezultatus.



Paveikslėlis 17 ilgesniųjų žodžių svorio nustatymo duomenų paieškos sėkmingumo grafikas



Paveikslėlis 18 trumpesniųjų žodžių svorio nustatymo duomenų paieškos sėkmingumo grafikas

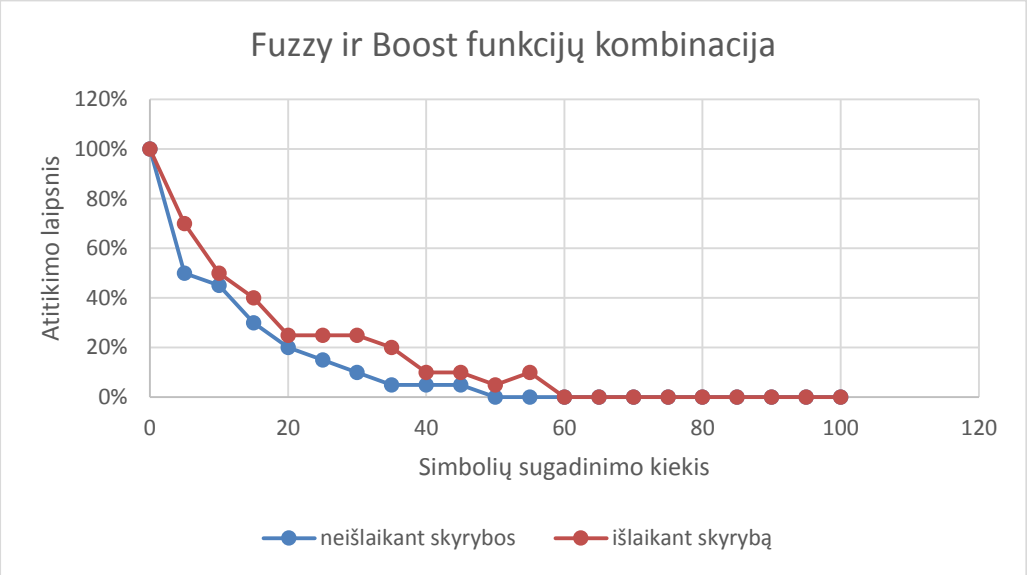
Tolimesnis tyrimas buvo daliai žodžių pridedant papildomo „svorio“ paieškos rezultatams. Bandyti buvo du variantai:

- Svorio suteikiant žodžiams ilgesniems negu 4 simboliai, su mintimi, kad ilgesni žodžiai yra unikalesni ir jų paieška turėtų grąžinti tikslesnius rezultatus.

- Svorio suteikiant žodžiams trumpesniems negu 5 simboliai, su mintimi, kad atsitiktinio gadinimo metu yra didesnė tikimybė sugadinti ilguosius žodžius nei trumpuosius.

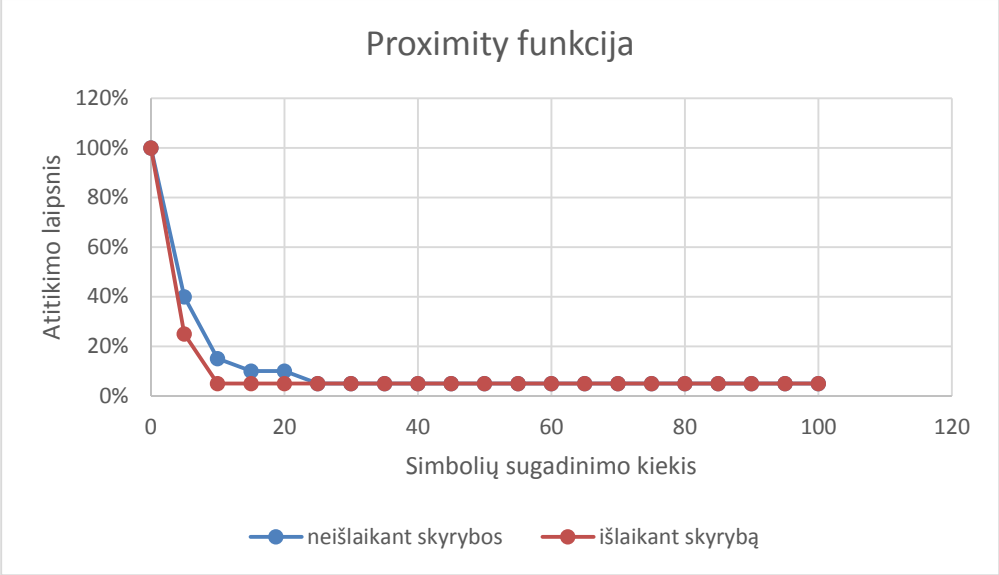
Lyginant aukščiau esančius grafikus matyti, kad trumpųjų žodžių vertės didinimas davė apčiuopiamų rezultatų. 18 paveikslėlio grafike matyti, kad tiek su nedideliu, tiek ir su dideliu klaidų kiekiu, teksto radimo rezultatai yra geresni, negu 17 paveikslėlio grafike. Trumpų žodžių vertės didinimas labiausiai įtakojo rezultatų dalį atspindinčią didelį teksto sugadinimo laipsnį.

Lyginant šių lentelių duomenų rinkinių skirtumus aiškiai matyti, kad tekstų su teisingais tarpais paieškos rezultatai yra geresni visais atvejais.



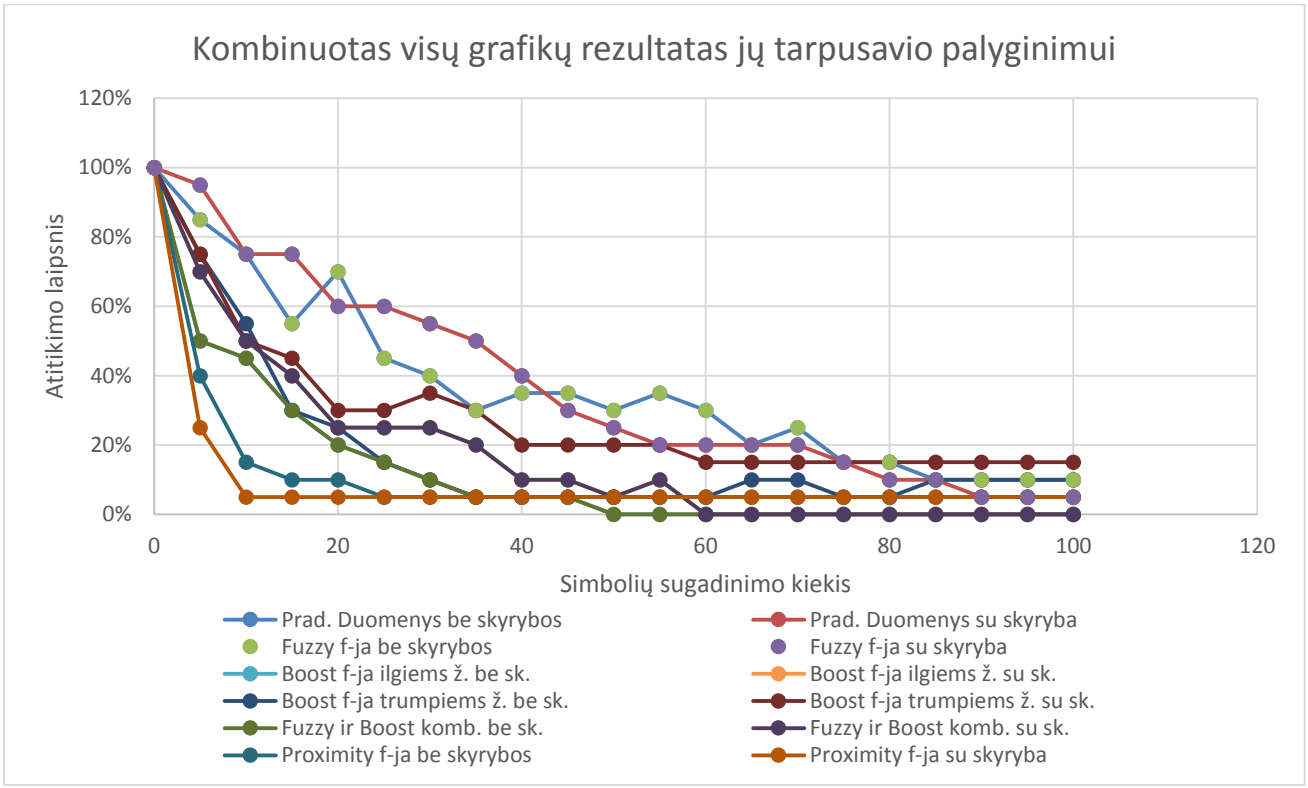
Paveikslėlis 19 kombinuotų funkcijų nustatymo duomenų paieškos sėkmingumo grafikas

Atliekant kelių nustatymų kombinaciją, kaip Fuzzy ir Boost nustatymo atveju buvo tikėtasi, kad šie nustatymai vienas kitam padės pasiekti tinkamų rezultatų, tačiau kaip ir pastebėta su pavieniais Boost ir Fuzzy nustatymais jokios naudos ar pagerėjimo nebuvo.



Paveikslėlis 20 Proximity nustatymo duomenų paieškos sėkmingumo grafikas

Proximity nustatymai paieškos rezultatams nedavė jokios naudos, netgi priešingai, rezultatai yra stipriai prastesni negu paieškos visai be parametrų.



Paveikslėlis 21 kombinuotas visų nustatymų duomenų paieškos sėkmingumo grafikas. Svarbu – pradinių duomenų ir Fuzzy funkcijos grafikai sutampa, tad grafike jos matosi kaip viena linija.

Fuzzy ir pradinių duomenų grafikai sutampa, tai reiškia, kad Fuzzy algoritmas nieko nekeičia atliekant netikslaus teksto paiešką Solr duomenų bazėje. Galbūt Šiame grafike yra sudėti visi bandytų nustatymų rezultatai, kad juos būtų galima lengviau patikrinti tarpusavyje. Pagal šį grafiką matyti, kad geriausi rezultatai lieka paieškos be papildomų nustatymų. Pagal tai galima spręsti, kad

arba šie nustatymai naudojami netinkamai, arba apytikslio teksto paieškai jie negali pagelbėti ir norint naudoti patikslintą tekstą paieškai reikia ieškoti kitų būdų kaip pataisyti sugadintus žodžius.

Išvados

Apie prototipą

Projekto tikslas yra mobiliuoju telefonu nufotografavus teksto fragmentą, gauti informaciją apie fotografuojamą leidinį, prie to leidinio priregistruotą skaitmeninį turinį. Tą galima pasiekti pasitelkus mobiliajam telefonui suprogramuotą aplikaciją, kuri nufotografavusi vaizdą iš jo ištraukia tekstą ir jį siunčia serveriui. Serveris turi didelę skaitmeninių knygų duomenų bazę, kurioje ir atlieka paiešką pagal programėlės siųstą tekstą, ir per programėlę pateikia papildomą informaciją vartotojui.

Kaip aptarta anksčiau, problema yra ta, kad tikslaus teksto nuskaityti nepavyks dėl netobulų algoritmų, tačiau tuos netikslumus galima švelninti dviem būdais: vienas iš būdų yra gerinti teksto atpažinimą pačioje programėlėje, optimizuoti paveikslėlį, kad atpažinimo algoritmams būtų kuo mažiau dvejonių dėl pačių simbolių; kitas būdas yra optimizuoti patį apytikslio teksto paieškos algoritmą serverio pusėje, tikslaus teksto paieškai gauti nepavyks, tačiau galima išmokyti paieškos algoritmą ieškoti teksto su klaidomis. Kuo labiau bus tobulinami aplikacijos ir serverio algoritmai, tuo tikslesnis ir greitesnis bus visos sistemos veikimas.

Projekto vystymo metu, projektavimo procesas buvo pakeistas iteraciniu. Darbas buvo daromas iteracijomis kuriant sistemą su minimaliomis veikiančiomis funkcijomis, vėliau funkcijas plečiant bei užtikrinant jų tikslesnį veikimą. Tačiau esamų programavimo žinių ir gebėjimų kiekis šio projekto kūrime yra nepakankamas, todėl nepavyko jo pilnai įgyvendinti.

Esamas veikiantis minimumas patvirtina teoriją, kad tokia sistema gali veikti, tačiau, kad ji veiktų ir veiktų gerai reikia papildomų laiko ir žinių resursų.

Apie tyrimą

Prototipo kūrimo metu buvo pastebėta, kad kokybiškam sistemos veikimui reikalinga optimaliai veikianti sistema, kurioje daromos kokybiškos nuotraukos ir atliekama tiksli ieškomo teksto paieška. Optimaliai veikiančios sistemos ingredientai yra du: kokybiškos nuotraukos ir daug teksto neatitikimų toleruojanti teksto paieška.

Nuotraukų apdorojimo algoritmai, galintys pagerinti teksto iš nuotraukų atpažinimą kaip ir teksto klaidas eliminuojantys yra sudėtingi ir reikalauja daug papildomų tyrimų norint juos pritaikyti esamai sistemai. Tačiau esamą sistemą reikėjo patobulinti gerinant atpažįstamo teksto arba paieškos rezultatų grąžinimą, kad būtų galima tęsti jos vystymą. Projekte naudojama Solr duomenų bazė turi

labai daug tekstinę paiešką keičiančių nustatymų, todėl tyrimas atliktas norint išsiaiškinti ar šių nustatymų naudojimas gali pagerinti paieškos rezultatų kokybę.

Atlikus Solr sistemos nustatymų analizę buvo pritaikyti keli nustatymai, kurie gali keisti teksto paieškos rezultatų kokybę šio projekto atveju. Tyrimui buvo sukurta programinė įranga imituojanti nekokybišką teksto simbolių atpažinimą keliais lygiais bei automatizuotam rezultatų palyginimui tarpusavyje. Prieduose yra pavaizduotas pavyzdinis duomenų apdorojimo procesas.

Solr nustatymų poveikio apytikslio teksto paieškos rezultatams analizė nedavė norimo rezultato. Kiekvienas paieškos sistemos funkcijos variantas ar jų kombinacija davė daugiau ar mažiau prastesnius rezultatus nei paieška nenaudojant papildomų nustatymų. Tokius rezultatus gali lemti kelios priežastys: naudoti paieškos nustatymai nepritaikyti naudoti su tekstais, kuriuose negali būti tiksliai apibrėžtos teksto sugadinimo laipsnis arba netinkamai išnaudotos šių funkcijų galimybės. Galutinė rezultatų išvada, kad reikia ieškoti kitų sistemos būdų norint pagerinti teksto atpažinimo, jo paieškos ir visų rezultatų kokybę.

Apie perspektyvą

Įvykdžius šį projektą, sukurta sistema popierinių knygų naudotojams pasidžiaugti ir technologijų teikiamais privalumais, be poreikio visą tekstą skaityti varginančiame elektroninio prietaiso ekrane. Serveryje saugoma informacija pasiekiama keliais programėlės paspaudimais. Tai leistų sutaupyti knygų leidybai reikalingus kaštus, visi spalvoti paveiksliukai bei video ar audio medžiaga galės būti talpinama serveryje ir vartotojas ją galės prieiti kada panorėjęs.

Konkurentus ši sistema lenks jau vien tuo, kad turės galimybę atlikti paiešką kūrinių viduje. Ir iš dalies leidinio teksto bus galima atrasti koks tai leidinys.

Ateityje sistemą galima plėsti, kad palaikytų daugiau mobiliųjų platformų ar įvairesnius popierinius ir skaitmeninius leidinius. Informacijos pateikimas galėtų būti pasiekiamas ne vien tik užfiksavus tekstą, bet ir įkeliant kitus paveikslėlius parsiųstus iš interneto ar nuotraukų albumų.

Literatūros sąrašas

[1] „Why Printed Books Will Never Die“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://mashable.com/2013/01/16/e-books-vs-print/

[2] „Google Goggles“ [žiūrėta 2014-01-27]. Prieiga per internetą: https://play.google.com/store/apps/details?id=com.google.android.apps.unveil

[3] „TinEye is a reverse image search engine“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://www.tineye.com/about

[4] „Layar augmented reality application“ [žiūrėta 2014-01-27]. Prieiga per internetą: https://www.layar.com/products/app/

[5] „Android developer registration“ [žiūrėta 2014-01-27]. Prieiga per internetą: https://support.google.com/googleplay/android-developer/answer/113468?hl=lt

[6] „Choosing iOS development program“ [žiūrėta 2014-01-27]. Prieiga per internetą: https://developer.apple.com/programs/start/ios/

[7] „Windows phone development account tyoes, locations and fees“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://msdn.microsoft.com/en-us/library/windows/apps/jj863494.aspx

[8] „Android phone and tablet development“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://mobisoftinfotech.com/services/android-phone-tablet-app-development-usa/

[9] „Android developer tools“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://developer.android.com/tools/index.html

[10] „Daily Android activations grow to 1.5 million, Google Play surpasses 50 billion downloads“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://bgr.com/2013/07/20/android-activations-app-downloads/

[11] „Android java and native applications“ [žiūrėta 2014-01-27]. Prieiga per internetą: https://devs.ouya.tv/developers/docs/android

[12] „Visibility for Your Apps“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://developer.android.com/distribute/googleplay/about/visibility.html#reach

[13] „Android and Java“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://news.cnet.com/8301-13580_3-9815495-39.html

[14] „Apache Solr“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://lucene.apache.org/solr/

[15] „Java in a nutshell“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://docstore.mik.ua/orelly/java-ent/jnut/ch01_02.htm

[16] Nathanel Paul, David Evans „Comparing Java and .NET Security: Lessons Learned and Missed“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://web.eecs.utk.edu/~pauln/papers/computers_and_security-net-java.pdf

[17] Eugene Borovikov, Ilya Zavorin, Mark Turner „A filter based post-OCR accuracy boost system“, p-5, [žiūrėta 2014-01-27]. Prieiga per internetą: http://www.researchgate.net/publication/234813125_A_filter_based_post-OCR_accuracy_boost_system/file/3deec524990a595b1d.pdf

[18] Stephen V. Rice, Frank R. Jenkins, ir Thomas A. Nartker „The Fourth Annual Test of OCR Accuracy“ p-15, [žiūrėta 2014-01-27]. Prieiga per internetą: http://stephenvrice.com/images/AT-1995.pdf

[19] Wojciech Bieniecki, Szymon Grabowski ir Wojciech Rozenberg „Image Preprocessing for Improving OCR Accuracy“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://wbieniec.kis.p.lodz.pl/research/files/07_memstech_ocr.pdf

[20] „Optical Character Recognition (OCR) – How it works“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/

[21] William B. Lund, Douglas J. Kennard ir Eric K. Ringger „Why Multiple Document Image Binarizations Improve OCR“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://faculty.cs.byu.edu/~ringger/CS679/papers/Lund-HIP2013.pdf

[22] Juan C. Perez-Cortes, Juan C. Amengual, Joaquim Arlandis „Stochastic Error-Correcting Parsing for OCR Post-processing“ [žiūrėta 2014-01-27]. Prieiga per internetą: https://prhlt.iti.upv.es/demos/demo_hwforms/doc/icpr00.pdf

[23] Line Eikvil „Optical Character Recognition“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.217.4980&rep=rep1&type=pdf

[24] „Tesseract OCR engine that was developed at HP Labs between 1985 and 1995... and now at Google.“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://code.google.com/p/tesseract-ocr/wiki/ReadMe

[25] Chirag Patel, Atul Patel, PhD., Dharmendra Patel „Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://www.researchgate.net/publication/235956427_Optical_Character_Recognition_by_Open_source_OCR_Tool_Tesseract_A_Case_Study/file/50463516fa43a64739.pdf

[26] Ray Smith „An Overview of the Tesseract OCR Engine“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://static.googleusercontent.com/media/research.google.com/lt//pubs/archive/33418.pdf

[27] „Apache Solr“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://lucene.apache.org/solr/

[28] „OpenFTS (Open Source Full Text Search engine)“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://openfts.sourceforge.net/

[29] „Sphinx is an open source full text search server“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://sphinxsearch.com/about/sphinx/

[30] „Xapian is an Open Source Search Engine Library“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://xapian.org/

[31] „MG4J (Managing Gigabytes for Java) is a free full-text search engine for large document collections written in Java“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://mg4j.di.unimi.it/

[32] „Terrier is a highly flexible, efficient, and effective open source search engine“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://terrier.org/

[33] „Apache Lucene“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://lucene.apache.org/

[34] „Lucene's FuzzyQuery is 100 times faster in 4.0“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://blog.mikemccandless.com/2011/03/lucenes-fuzzyquery-is-100-times-faster.html

[35] „OCR – How it works“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/

[36] „Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://www.dlib.org/dlib/march09/holley/03holley.html

[37] „TIOBE Index“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[38] „The Android source code“ [žiūrėta 2014-01-27]. Prieiga per internetą: http://source.android.com/source/index.html

Priedai

Priedas nr. 1.

Pavyzdinės frazių paieškos rezultatų matricos konvertavimo į grafiką procesas.

	Originali teksto vieta	Teksto sugadinimo laipsnis simboliais																			
		5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Skirtingos frazės	4 259	4259	4259	4259	4259	4259	4259	4259	4259	756	756	756	0	0	0	0	0	0	2680	2680	0
	11 225	11225	11225	11225	11225	11225	11225	11225	11225	11225	11225	11225	11225	11225	11225	11225	11225	11225	4021	4021	4021
	1 274	1274	1274	1274	1195	1274	1274	1363	1363	1195	1363	1363	1363	1363	23452	23452	23452	19488	19488	19488	19488
	4425	4425	4425	4425	4425	4425	4425	4425	4425	4425	4425	4425	4425	4425	4425	3547	3547	3547	3547	3547	3547
	11 034	11034	11034	11121	11121	11121	11121	11034	11034	11034	11034	11034	4736	4736	4736	4736	4736	4736	10958	10958	10958
	19 654	19654	19572	19572	19572	19738	19572	18520	18520	18520	18520	19572	19572	19572	19572	21255	21255	21255	18520	18520	18520
	9 628	9547	9547	9628	9628	9628	9547	9547	9547	9547	9142	9547	6786	6786	29680	29680	25883	25883	25883	25883	25883
	9553	9553	9553	9553	895	895	895	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10748	10748	971	971	971	971	971	971	971	11202	1677	1677	1677	1677	0	1677	1677	1677	1677	1677	0
	22609	22609	22609	22609	5519	5519	4504	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	37110	37110	37184	37184	37110	37110	5665	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	35 355	35355	35355	35355	35355	35355	35355	35355	35355	35355	35355	35355	35355	35355	35355	35355	20698	20698	20698	794	794
	28705	28705	28705	28705	28705	28705	28705	28705	3085	3085	4581	29475	29475	29475	25339	25339	25339	25339	25339	25339	25339
	25301	25301	21310	25301	25301	25301	1602	1602	1436	1436	0	0	0	0	0	0	0	0	0	0	0
	20 697	20697	20697	20697	9792	9792	20697	20697	20697	20697	20697	9792	9390	9390	4581	4581	4581	4581	9244	9244	9244
	9 628	9628	9628	9628	9628	9628	9628	9628	9628	9142	9142	9142	9142	9142	9142	9142	9142	9142	9142	9142	9142
	11 225	11225	11225	11307	11383	11383	11383	11383	11383	11383	11383	11383	11225	11225	11225	11225	11225	11225	11225	11225	11225
	20 486	20486	20486	20486	20486	20567	20486	20567	20567	20486	152	152	8924	8924	8924	8924	152	152	152	152	152
	32 067	32067	32067	32067	32067	32067	32067	32067	32067	32150	32150	21310	5886	5886	6186	6186	6186	6186	6186	5886	19767
	20 486	20486	20486	20486	20486	20486	20486	20486	5959	5959	5959	5959	5959	5959	5959	5959	5959	5959	5959	5959	5959

Lentelė 1

Pavyzdinę paieškos rezultatų matricą konvertavus į dvejetainę pagal tai, kurie skaičiai atitinka originalią teksto vietą gauname tokį vaizdą:

0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Lentelė 2

Iš šios lentelės duomenų išskaičiavę vidurkius ir juos pavertę procentine išraiška galime sudaryti lentelę grafiko generavimui.

0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
100 %	95 %	75 %	75 %	60 %	60 %	55 %	50 %	40 %	30 %	25 %	20 %	20 %	20 %	20 %	15 %	10 %	10 %	5 %	5 %	5 %

Lentelė 3

Iš trečiosios lentelės braižomas grafikas, toks grafikas braižomas kiekvienam iš paieškos funkcijos nustatymų.

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Darius Lekavičius

(Studento vardas, pavardė)

Programų sistemų inžinerija (621E16001)

(Studijų programos pavadinimas, kodas)

APYTIKSLIO TEKSTO PAIEŠKA ŠALTINIO RADIMUI

APPROXIMATE STRING SEARCH FOR TEXT SOURCE RETRIEVAL

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano **Dariaus Lekavičiaus** baigiamasis projektas tema „APYTIKSLIO TEKSTO PAIEŠKA ŠALTINIO RADIMUI“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka)	*(parašas)*

Lekavičius D. APYTIKSLIO TEKSTO PAIEŠKA ŠALTINIO RADIMUI. *Magistro* baigiamasis projektas / vadovas dr. Mantas Lukoševičius; Kauno technologijos universitetas, Informatiko fakultetas, Programų inžinerijos katedra.

Kaunas, 2015. 1 psl.

SANTRAUKA

APPROXIMATE STRING SEARCH FOR TEXT SOURCE RETRIEVAL

This project is about the system that was created during studies of master's degree. Purpose for the system was to analyze and find a best way to find the best algorithm for searching approximate text to find the source of the mentioned text. Main audience for the final product is intended to be people who like reading books, but would benefit from the digital supplements that are available only for readers of digital media. The supplements could be reachable using a smartphone with a camera. Idea behind this is that a person reading a book would like to get some additional information about the text he is reading. Taking a photo of the mentioned page should return additional digital data about the book or the exact page giving access to online reviews, related pictures, videos and so on.

To reach a projected goal a system of three initial parts had to be created and tested. Three parts consisted of three different applications working together to provide the service for the project idea.

First a mobile application for Android system was created, which would be the main interface for the user. This application is able to take a photo, perform an optical character recognition and send the text to database and processing servers to get a digital data from them to show the information for the user.

Processing and database servers would be running on a server machines and split the tasks accordingly. The intermediate service is the main point to talk to for the database and mobile applications. This service receives data from mobile application and uses it to calculate the source location using Solr database server.

Solr database server is optimized for fast indexing and search capabilities. This server is able to perform approximate string search and return data related to it. The project goal requires to calculate the specific location of the text in search, this cannot be done using only Solr server. This is a task for intermediate service to talk to Solr server and using data from mobile application to send a respoonce with required details.

Project has been implemented up to a development version of getting all parts to work together. As the complexity of the project is high it was not completed to full user friendly and full featured application. In the end a research has been performed to find the issues why it is not working the way it was intended.