TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Arvutiteaduse instituut

Võrgutarkvara õppetool

Turismiobjektide tagide semantika analüüs, otsing ja visualiseerimine.

Bakalaureusetöö

Üliõpilane: Margus Sibrits

Üliõpilaskood: 103673

Juhendaja: Tanel Tammet

Tallinn

2015

Autorideklaratsioon

Deklareerin, et käesolev lõputöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud.

…….. …….

(kuupäev) (lõputöö kaitsja allkiri)

Annotatsioon

Lõputöös analüüsiti turismiobjektide tagide semantikat, et neid jagada kategooriateks. Saaduid kategooriaid kasutati filtreerimiseks ning tagide järgi otsimisel. Liiga üldised ja vigased tagid eemaldati ja kohti, millel oli vähe tage, prooviti täiendada sobivate wikipedia ja foursquare tüüpidega. Valmis algoritm, et sorteerida filtreerimise ja otsingud tulemused kas rohkem populaarsuse või otsitavate tagide osa järgi vastavas turismiobjektis. Tagide põhjal joonistati tagipilv ning tagide andmete esitust optimeeriti, et vähendada nende mahtu.

Annotation

In this thesis tourism sites tags semantics were analyzed for dividing them into categories. Resulted categories were used for filtering and for tags autocompletion in search box. Tags that were too generic or incorrect were removed and places with few tags replenished with appropriate wikipedia and foursquare types. Algorithm was constructed for sorting search and filter result by popularity and relevance. Tag clouds were made for better visualization and tags data was optimized for saving storage space.

Sisukord

1 Sissejuhatus

Käesoleva lõputöö ülesanne on turismiobjektide tagide semantika analüüs, tagide abil otsimine ning filtreerimine ja tagide visuaalne esitamine. Lõputöö koosneb tagide analüüsist, mille abil moodustatakse kategooriad. Tagide abil otsimine ning filtreerimine kasutades nende kategooriaid. Saadud tulemuse sorteerimine vastavalt populaarsusele ja leitud tagide tähtusele. Lisaks tagide esitamine tagipilvena ning tagide andmete hoidmise optimeerimine, et vähendada andmete mahtu.

2 Ülevaade valdkonnast

MoreTourism[7] süsteem, kus kasutajad annavad ise tage kohtadele, mille järgi kujuneb koha tagipilv ja samuti ka kasutaja enda tagipilv. Mõlemas pilves antakse tagidele kaalud tema esinemiste järgi. Tagide vahel luuakse ka seosed, kui nad esinevad koos mõne koha juures, mida rohkem koos esinevad seda suurem ka seose kaal on. On tehtud hübriidne soovituse süsteem, mis liidab sisu põhist filtreerimist ja koostööl põhinevat filtreerimist. Sisu põhine filtreerimine võrdleb kasutaja tagipilve koha tagipilvega, võtes arvesse ka tagide suhet. Koostööl põhinevat filtreerimiseks luuakse igale kohale uus tagipilv nende kasutajate tagipilvedest, kellel see koht meeldis. Loodud tagipilve võrreldaks siis kasutaja tagipilvega, samuti arvestatakse tagide vahelisi suhteid.

Liangliang Cao *et al.* [8] kirjeldab süsteemi, kus kasutatakse tagituid pilte Flickr ja Google Earth keskkonnast. Koordinaatide abil rühmitatakse pildid ja nende tagid kokku, eemaldades seejuures vähem esinevad tagid ning väiksema populaarsusega pildid. On loodud turimisi soovituse süsteem, mis lisaks tagide abil soovitamisel oskab otsida ka kohti pildi järgi. Sisendiks antud pildile leitakse sarnaseid pilte ja siis selle järgi pakutakse välja lähimaid linnu.

Meetod nimega ContextRank[9], kus kasutatakse Panoramio keskkonnast pärit pilte ja nende andmeid(tage, koordinaate). Turismikohtasid tuvastatakse piltide koordinaatide järgi, analüüsides pilte ja tage tuvastatkse neist antud kohale kõige iseloomulikumad. Soovituse süsteem kasutab kasutaja enda reisimise ajalugu, et välja pakkuda talle sobivamaid kohti.

3 Tagide semantika analüüs ja üldistamine kategooriateks

Lõputöös kasutatavad turismikohtade andmed pärinevad sightsmapi[3,10] tegemise käigus kogutud andmetest. Turismiobjektide kohta on teada olevatest andmetest kasutan : nime, koordinaate, tagide listi koos kaaludega, riiki, wikipedia lühikirjeldust, wikipedia tüüp, foursquare tüüp, panoramia pildi id. Kõigile kohtadel ei ole kõiki andmeid olemas. Kokku on andmeid 174362 turismiobjekti kohta, sorteerituna nende populaarsuse järgi. Andmetega seonduvate tegevuste jaoks, nagu nende analüüsimine, jagamine riikide kaupa jne, kasutasin Pyhtoni programmeerimiskeelt.

Kuna kohtadel on palju erinevaid tage ning paljud neist on sarnase tähendusega, siis oli mõistlik proovida jaotada need erinevateks kategooriateks, et neid kasutada filtreerimiseks ja tagide järgi otsimisel.

Esmalt tuli saada ülevaade rohkem esinevatest ning kaalukamatest tagidest, kuna tage on palju ja nende kategooriateks jagamine toimib käsitsi, siis otsustasin teha kategooriaid mõistliku koguse põhjal. Selleks tegin väikse programmi, mis arvutab ja väljastab sorteeritult tagide loetelu vastavalt sellele kui oluline on antud tag olnud temage seotud kohtades. Ehk koha juures arvutatakse iga tagi kaalu osa kogu koha tagide kaalu summast ja lisatakse see juurde antud tag olulisuse summale.

Saadud nimekirjast võtsin ligikaudu 200 esimest tagi ja jagasin need sobivateks kategooriateks. Saadud kategooriaid püüdsin omakorda rühmitada. Järgnevalt tabelid tulemustest(tabeli peal on kategooria nimi, vasakul tulbas kategooriasse kuuluvad üksikud tagid, paremal tulbas kategooriasse kuuluvad alamkategooriad). Alamkategooria tabelis on alamkategooriatesse kuuluvad tagid.

Historic	
Tagid	Alamkategooriad
old	Castle & Fortress
antique	Museum
ruin	Palace
royal	Temple
ancient	
historic	

Alamkategooriad:

Alamkategooriad	Tagid			
Castle & Fortress	castle	fortress	fort	gate
Museum	museum			
Palace	palace			
Temple	temple			

Landmark	
Tagid	Alamkategooriad
fountain	Harbour
villa	Tower
lighthouse	Monument & Memorial
mill	Religious architecture
hall	Bridge & Dam

Alamkategooriad	Tagid							
Harbour	harbour	port	boat	ferry	ship	wharf	jetty	
Monument & Memorial	monument	memorial	statue	sculpture				
Religious architecture	church	cathedral	chapel	monastery	mosque	shrine	basilica	holy
	cemetary	templom	cami	catholic	kerk			
Bridge & Dam	dam	bridge						
Tower	tower							

Nature	
Tagid	Alamkategooriad
nature	Park
forest	Garden
green	River
trees	Sea
fog	Lake
camping	Mountain
outside	Valley
	Island
	Waterfall
	Cave
	Bay
	Lagoon

Alamkategooriad:

Alamkategooriad	Tagid						
Park	park						
Garden	garden	flowers	flower	gazebo	patio		
River	river	canal	water				
Sea	sea	beach	coast	water	marine	ocean	
Lake	lake	pond	water	reservoir			
Mountain	mountain	hill	high	peak			
Valley	canyon	gorge	valley				

Island	island			
Waterfall	waterfall	water		
Cave	cave	caves	rock	rocks
Bay	bay	cove		
Lagoon	lagoon	water		

Entertainment	
Tagid	Alamkategooriad
festival	Nightlife
	Art
	Relaxation
	Outdoors & Recreation

Alamkategooriad:

Alamkategooriad	Tagid						
Nightlife	night	evening	club				
Art	theatre	Art	gallery	culture			
Relaxation	resort	Spa	pool				
Outdoors & Recreation	camping	outside	fishing	camp	campground	hiking	golf
Shopping	market	Shop	mall				

Great view	
Tagid	Alamkategooriad
view	Sunset & Sunrise
looking	Landscape
beautiful	Sky & Clouds
panoramic	

Alamkategooriad:

Alamkategooriad	Tagid			
Sunset & Sunrise	sunset	sunrise	dawn	sun
Landscape	landscape	background	fog	
Sky & Clouds	sky	clouds	sun	moon

Accommodation	
Tagid	Alamkategooriad
housing	Hotel
	Motel & Hostel

Alamkategooriad:

Alamkategooriad	Tagid	
Hotel	hotel	
Motel & Hostel	motel	hostel

Food & Drink	
Tagid	Alamkategooriad
	Restaurant
	Pub & Bar
	Cafe
	Inn

Alamkategooriad:

Alamkategooriad	Tagid	
Restaurant	restaurant	
Pub & Bar	pub	bar
Cafe	cafe	
Inn	inn	

Education	
Tagid	Alamkategooriad
	School & University
	Library

Alamkategooriad:

Alamkategooriad	Tagid			
School & University	school	university	college	campus
Library	library			

4 Tagide saneerimine ja laiendamine

4.1 Tagide saneerimine

Probleemiks oli, et osad tagid on kas liiga üldised ega anna mingit konkreetset teavet koha kohta või on lihtsalt arusaamatud. Arusaamatuse põhjuseks võib pidada, et tagid on koostatud nii, et iga tag on ainult üks sõna ja algsest mitme sõnalisest tagist on jäänud tagiks sõna, mis ei sobi.

Arvasin ka siin, et on mõistlik mittesobivaid tage otsida ligikaudu esimese 200 enim esineva ja tähtsamate tagide seast. Selleks kasutasin algset tagide statistikat. Loetelu välja korjatud tagidest:

new,iii,house,saint,photo,building,nad,website,bij,van,square,mary,bir,nad,street,ulica,het,road,front,home,side,blue,red,white,way,close,black,john,spot,seen,end,detail,post,input,zone,low,sign,colors,door,good,direction,part,across,near,pod,hdr,pedro,around,aug,nov,just,life,good,back,low

4.2 Tagide laiendamine

Mõndades kohtadel oli algselt juba vähe tage ja osade tagide eemaldamine võis nende tagide hulka veelgi vähendada. Kuid kuna paljudel kohtadel on olemas wikipediast leitud tüüp ja samuti foursquare tüüp, siis tuli mõte proovida nende abil tage kohale juurde saada. Selleks oli vaja saada ülevaade wikipedia ja foursquare tüüpidest, mida saab kasutada tagide hulga rikastamiseks. Esmalt tegin väikse programmi, mis väljastas esinemis arvu järgi sorteeritud nimekirja wikipedia ja foursquare tüüpidest, mis on kohtadel, millel on vähem kui 10 tagi. Põhjus, miks ei ma ei proovinud täiendada kõigi kohtade tagide hulka, on see, et leitud tüüpe peab analüüsima ja kuna kõike tüüpe ei ole võimalik läbi vaadata, siis paljud kohad ei saaks selle tulemusena tage juurde. Analüüsi all mõtlen seda, et nimekirjast peab leidma sobivad tüübid, mida saab kasutada ning lisaks muuta kahe sõnalised tüübid ühe sõnalisteks. Kogu selle lõpptulemuseks oli nimekiri sobivatest tüüpidest ja antud tüübile vastavast tagist.

Näiteks väike nimekiri tulemusest:

Tüüp	Tag
Scenic Lookout	View
Harbor / Marina	Harbor
History Museum	Museum

Kokku koosnes nimekiri 50-st tüüp-tag seosest ja nende abil täiendati 8882 koha tage.

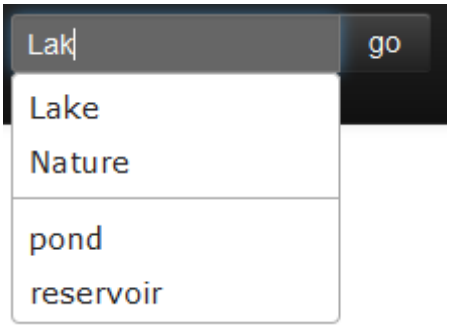
5 Tagide järgi otsimine ja tüübi filter

5.1 Tagide abil otsimine

Kuna tage on palju ja kasutaja ei pruugi täpselt teada, mis tagid kasutusel on, siis pakub rakendus jooksvalt välja nimekirja tagidest ja kategooriatest, mis algab samade tähtedega, mida kasutaja sisestas. Nii-öelda “autocomplete“. Tage pakutakse, nende populaarsuse järgi, mis saadi analüüsi käigus. Lisaks sarnastele tagidele ja kategooriatele näidatakse ka kõige sobivaima tagi ülemkategooriaid , kuna on maksimaalselt kahe tasemeline hierarhia, siis neid võib olla maksimaalselt kaks(pilt nr 1). Kui sobivaimaks osutus mingi kategooria, siis näidatakse tema ülemkategooriat. Nimekirja lõpus pakutakse kasutajale välja ka esimese ülemkategooria kahte suuremat tagi. Välja arvatud, siis kui need kaks suuremat tagi on liialt sarnased, näiteks “flower“ ja “flowers“. Seda sarnanust ei määrata automaatselt, vaid tagide ja kategooriate seoseid esitades on proovitud vältida, et kategooriate esimesed kolm tagi oleksid sarnased. Kolm selle pärast, et ühte kolmest võidakse näidata nimekirja peas ja ülejäänud kahte lõpus. Tagi suurust hindasin numbri järgi, mida sain tagide semantika analüüsist.

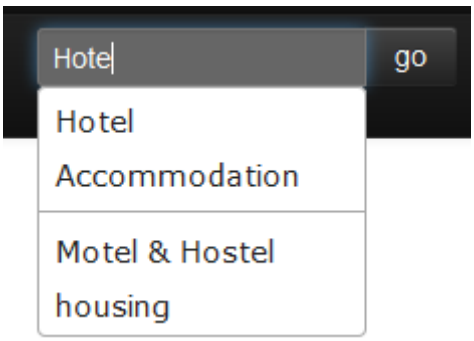
Pilt nr 1. Tavaline autocomplete, kui sobivaim on tavaline tag.

Kui sobivaim tag on sama, mis mingi kategooria, siis näidatakse seda esimesena ja lõpus kuvatakse selle kategooria kaks esimest tagi(pilt nr 2) või siis kaks suuremat alamkategooriat(pilt nr 3). Kui aga kategoorial pole kahte tagi, mis oleksid kategooriast erinevad, siis kuvatakse selle ülemkategooria kaks tagi või kategooriat(pilt nr 4).



Pilt nr 2. Autocomplete, kui sobivaim on kategooria.

Pilt nr 3. Autocomplete, kui sobivaim on kõige ülemisem kategooria.



Pilt nr 4. Autocomplete, kui sobivaim on kategooria ja sellel pole temast erinevaid tage.

Rakenduses hoitakse tage ja kategooriaid võti - väärtus paaridena. Seda selle pärast, et oleks mugavam ja kiirem saada kätte kategooriaid kuhu tag kuulub. Enamik tage kuulub ainult ühte kategooriasse seega on selle tagi väärtuseks ainult ülemkategooria nimi stringina, nendel tagide, mis kuuluvad mitmesse kategooriasse, neil on väärtuseks nimekiri, milles on ülekategooriate nimed. Kategooriate seoseid hoitakse sama põhimõttega. Lisaks on rakenduses nimekiri kõikidest tagidest, sorteerituna vastavalt semantika analüüsis käigus saadud kaalule, mis näitab esinemis sagedust ja tähtsust.

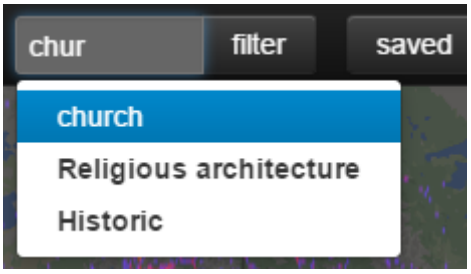
Tagide ja kategooriate pakkumise algoritm(**algoritm nr 1**) käib esmalt läbi tagide nimekirja ja proovib sealt leida kuni 10 tagi, mis algab otsitava sõnaga. Kui ühtegi tagi ei leitud, siis käiakse läbi kõik kategooriad ja proovitakse sealt leida kategooriaid, mis algavad otsitava sõnaga või selle sõna suure tähega algava variandiga. Esimene kategooria, mis leitakse märgitakse kui põhikategooria, selle abil kuvatakse lisainfot pärast. Juhul kui otsitavale sõnale leiti vastavad tagid ja kategooriad, siis esmalt vaadatakse kas esimene vastav tag ei ole sama kui tema kategooria, kui on siis muudetakse see kategooriaks. Näiteks, kui vastav tag on “lake” ja on olemas kategooria “Lake”, siis muudetakse esimene vastav tag(“lake”) ära tema kategooriaks(“Lake”). Järgnevalt käiakse läbi esimese leitud tagi või kategooria ülemkategooriad ja lisatakse need leitud tagide ja kategooriate nimekirja. Kui pole määratud veel peakategooriat, siis määratakse esimene ülemkategooria selleks. Edasi lisatakse kaks peakategooria suurimat tagi või kategooriat leitud tagide nimekirja. Kui kahte ei lisatud, siis proovitakse lisada ka peakategooria ülemkategooria suurimad tagid või kategooriad.

```
function getMatchedTagList(term){
     var matchedTags = new Array();
     var category;
     var count = 0;
     var limit = 10;
     for(var i = 0; i<tags.length && count < limit;i++){
          if(tags[i].substring(0,term.length)===term){
               matchedTags.push(tags[i]);
               count++;
          }
     }
     if(matchedTags.length<=0){
          for(var tag in categoriesTag){//see käib kategooriad
läbi
               if(tag.substring(0,term.length)===term ||
tag.substring(0,term.length)===term.capitalizeFirstLetter()){
                    matchedTags.push(tag);
                    if(!category){
                         category = tag;
                    }
               }
          }
     }
     if(matchedTags.length>0){
          var tag = matchedTags[0];
          if(tagCategories[tag] ===
tag.capitalizeFirstLetter()){
               tag = tag.capitalizeFirstLetter();
               matchedTags[0] = tag;
               if(!category){
                    category = tag;
               }
          }
          while(tagCategories[tag]){
               if(typeof(tagCategories[tag])==="string"){
```

```
                    matchedTags.push(tagCategories[tag]);
                    tag = tagCategories[tag];
                    if(!category){
                         category = tag;
                    }
               }else{
                    var categoryList = tagCategories[tag];
                    for(var i = 0; i<categoryList.length &&
i<2;i++){
                         matchedTags.push(categoryList[i]);
                         if(!category){
                              category = categoryList[i];
                         }
                    }
                    break;
               }
          }
          if(category){
               var count = 0;
               var limit = 2;
               for(var i = 0;i<categoriesTag[category].length
&& count < limit ;i++){
                    var tag = categoriesTag[category][i];
                    if(matchedTags.indexOf(tag)==-1 &&
matchedTags.indexOf(tag.capitalizeFirstLetter())==-1  ){
                         matchedTags.push(tag);
                         count++;
                    }
               }
               category = tagCategories[category];
               if(count<limit && category){
                    for(var i =
0;i<categoriesTag[category].length && count < limit ;i++){
                    var tag = categoriesTag[category][i];
                    if(matchedTags.indexOf(tag)==-1 &&
matchedTags.indexOf(tag.capitalizeFirstLetter())==-1  ){
                         matchedTags.push(tag);
                         count++;
                    }
                    }
               }
          }
     }
     return matchedTags;
}
```

Algoritm nr 1. Tagide järgi otsimise autocomplete algoritm

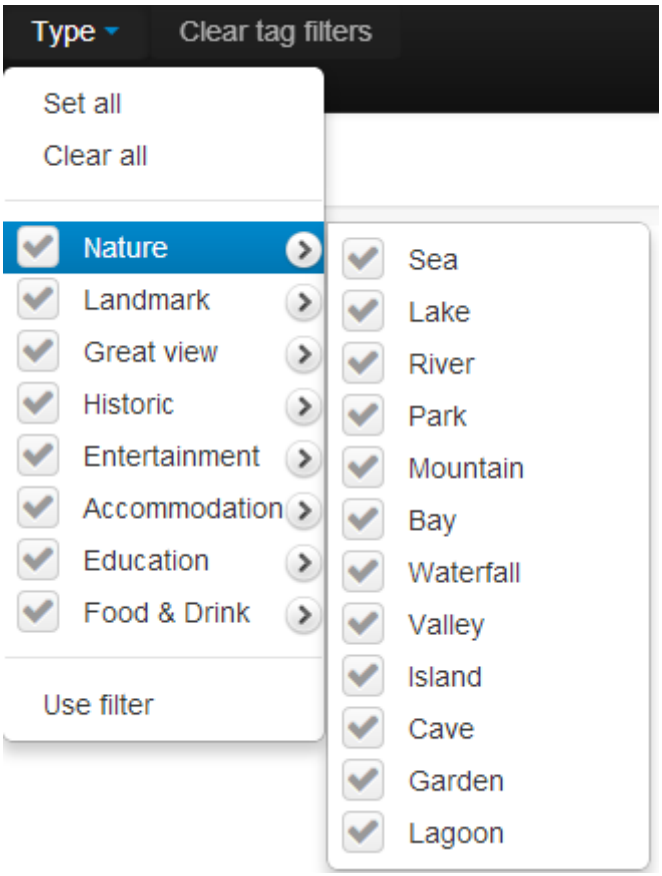
Sellisest tagide järgi otsimise süsteemist võeti eeskuju ning loodi sarnane võimalus ka sightsmapi[3] enda lehel(pilt nr 5)



Pilt nr 5. Autocomplete sightsmapi[3] lehel

5.2 Tüübi filter

Tüübi filtris on võimalik valida tagide kategooriaid, mille järgi turismiobjekte näidatakse (pilt nr 6). Kasutatakse neid samu kategooriaid, mis semantika analüüsil leidsin. Kategooriate järjestuse määrab nende kaal. Ehk kaalukamad kategooriad on eespool filtris, eeldusel, et neid kasutatakse tihedamini. Kaal tuleb sellest, et iga kategooriasse kuuluvate tagide kaalud, mis näitab sella tagi esinemise sagedust ja olulisust(tuleb semantika analüüsist), liidetakse kokku. Tagi kaal on antud juhul see, mis semantika analüüsi käigus leiti, ehk näitab tagi esinemis sagedust ja selle tähtsust. Kõige ülemiste kategooriate kaal on tema tagide ja alamkategooriate kaalude summa.



Pilt nr 6 Tüübi filter.

Rakenduses hoitakse tüübi filtri jaoks võti - väärtus paare, milles võtmeks on kategooria ja väärtuseks nimekiri, mis koosneb antud kategooriasse kuuluvatest tagidest ja alamkategooriatest. Selline hoidmine võimaldab mugavalt ja kiirelt kätte saada kõik tagid, mis kuuluvad valitud kategooriatesse. Kui rakendatakse seda filtrit, siis luuakse valitud kategooriate järgi nimekiri tagidest, mis kuuluvad nendesse kategooriatesse ja nende järgi filtreeritakse.

6 Filtreerimine ja otsingu tulemuste sorteerimine

Kui rakendatakse tüübi filtrit või otsitakse tagi järgi, siis luuakse nimekiri tagidest, mille järgi filtreeritakse. Iga koha kohta, mis on rakendusse laaditud, arvutatakse tema kõigi tagide kaalude summa ja lisaks antud kohas olevate otsitud tagide kaalude summa. Turismiobjekt, milles oli mõni otsitav tag, salvestatakse koos otsitud tagide kaalude summa jagatise antud objekti kõigi tagide kaalude summaga, edaspidi nimetan seda otsingu tulemuse kaaluks. Lisaks salvestatakse turismiobjekti algne positsioon leitud objektide nimekirjas. See näitab antud koha populaarsust, sest rakendusse olid objektid laetud populaarsuse järgi, populaarsemad ennem. Pärast seda sorteeritakse leitud objektide nimekiri kaalu järgi, ehk selle järgi, et kus oli otsitud tagide kaalude summa osa suurim antud koha kogu tagide kaalude summast(**Algoritm nr 2**).

Rakenduses on slaider, mille järgi saab kasutaja valida kuidas otsingu tulemusi sorteeritakse, kas rohkem esialgse populaarsuse järgi või siis rohkem selle järgi, kus oli otsitud tagide kaalude summa osa suurem antud turismiobjekti kõigi tagide kaalude summast. Slaideri väärtus on 0 kuni 100, ehk see näitab, mitu protsenti kasutatakse otsitud tagide kaalude summa osa antud koha kogu tagide kaalude summast lõpptulemuse sorteerimiseks. Arvutatakse välja kaks konstanti algse positsiooni jaoks ja selle positsiooni jaoks, mis saadi kui sorteeriti leitud objektide nimekiri kaalu järgi. Algse positsiooni konstant arvutatakse valemiga : (100 - slaideri väärtus)/100 , teine konstant leitakse nii, et 1-st lahutatakse esimene konstant. Iga objekti kohta arvutatakse välja tema positsiooni kaal kasutades neid kahte konstanti. See võrdub algse positsiooni korrutise tema konstandi ja kaalude järgi sorteeritud nimekirja positsiooni korrutise tema konstandi summaga. Turisimiobjektide nimekiri sorteeritakse positsiooni kaalude järgi, väiksem eespool(**Algoritm nr 3**).

```
for(var i=0;i<items.length;i++){
     var weight = 0;
     var totalWeight = 0;
     for (var j = 0; j < items[i].tags.length; j+=2) {
          if(searchList.indexOf(items[i].tags[j])>=0){
               weight += parseInt(items[i].tags[j+1]);
          }
          totalWeight +=  parseInt(items[i].tags[j+1]);
     }
```

```
      if(weight>0){

      searchResult.push({item:items[i].item,weight:weight/total
Weight,origWeightPos:-1,origPos:posCount,posWeight:0});
            posCount++;
      }
}
searchResult.sort(compareWeight);
items = searchResult;
```
Algoritm nr 2. Sorteerimise algoritmi esimene pool.

```
var origPosMultiplier = (100 - sliderValue)/100;
var weightPosMultiplier = 1 - origPosMultiplier;
for(var i=0;i<items.length;i++){
      if(items[i].origWeightPos==-1){
            items[i].origWeightPos=i;
      }
      items[i].posWeight =
items[i].origWeightPos*weightPosMultiplier +
items[i].origPos*origPosMultiplier;
}
items.sort(comparePosWeight);
```
Algoritm nr 3. Sorteerimise algoritmi teine pool.

7 Tagide visualiseerimine

Et saada turismiobjekti tagidest ülevaade kuvatakse selle objekti tagide pilv(tagcloud). Tagcloudi tegemiseks oli mitu valikut. Proovisin läbi Javascriptil põhinevad teegid, nagu jQCloud[4] versioon 1.0.2 ja awesomeCloud versioon 0.2 [5] ja Java teegi wordcram[6] versioon 0.5.7.

Mõlemad Javascripit teegid on jQuery lisad. Esimene neist ei kuva tage eriti kompaktselt ja pilve suuruse piiramine toob mõningatel juhtudel kaasa selle, et mõne tagi mingi osa kaob ära(pilt nr 7).

Pilt nr 7. Tagipilv kasutades jQCloudi, osad tagid poolikud.

Teine jQuery lisa, awesomeCloud, oskas tagipilve joonistada kompaktsemalt (pilt nr 8), kui jQCloud, kuid paljude tagide korral olid väiksema kaaluga tagid loetamatud(liiga väikse kirja suurusega)(pilt nr 9), selle parandasin antud programmi lähtekoodi ise muutes nii, et kirja suurused ei läheks liialt väikseks. Põhiline probleem awesomeCloud juures on kiirus, rakenduses kuvatakse lehel kuni 10 turimisobjekti ja kõigi tagipilvede joonistmine võtis aega ja aeglustas märgatavalt veebisirvija tööd.

Pilt nr 8. Tagipilv kasutades awesomeCloud koos lähtekoodis tehtud muudatusega.

Pilt nr 9. Tagipilv kasutades awesomeCloudi.

Java teegi wordcramiga on võimalik luua tagipilve pildina (pilt nr 10) . See muidugi tähendab, et kasutaja ei saa tagipilvega interaktiivselt suhelda, kuid kuna antud rakenduse juures on tagipilv eelkõige visuaalseks esitamiseks, siis ei pidanud seda suureks miinuseks. Samuti see vähendab kasutaja veebisirvija tööd, kuna tagi pildid on kõik juba olemas. Miinuseks on muidugi see, et muudatuste tegemisel peab kõik pildid uuesti looma, mis võtab paratamatult aega. Suureks plussiks on väga kompaktne tagide esitus. Kuna wordcramiga tehtud pilved näevad kõige paremad välja ja probleeme ei leidnud, mis segaksid seda rakendamast antud juhul, siis otsustasingi seda kasutada.

Pilt nr 10. Tagipilv kasutades wordcrami.

Kuna turismiobjekte on palju ja nendele kõikidele tagipilvede tegemine võtab aega, siis tundus mõistlik proovida jaotade see töö mitme lõime peale ära. Java rakendus, mis kasutab wordcrami teeki, et tagipilveid joonistada, kasutab master/slave disaini mustrit[1,2]. Rakenduses on tööjaotaja(master), milles on BlockingQueue, kus on andmed, mida pilve joonistamiseks vaja(faili nimi,tagid,kaalud). Töö lõimed(slaves) küsivad tööjaotaja käest andmeid, mille alusel joonistavad tagipilveid. Põhjus, miks töö lõimedele algul kogu nende tööd ei anta, on see, et osad pilved on suuremad ning nende joonistamine võtab rohkem aega ja selle pärast mõnedel lõimetel võib töö otsa saada kuigi teised veel töötavad.

8 Tagide esituse optimeerimine

Tagide esitus nende algupärasel kujul on mahukas, erinevaid tage ei ole väga palju ja samuti nende kaalud korduvad tihti. Seega otsustasin esitada tagid ja kaalud kahe täheliste kombinatsioonidega. Varasemast tagide analüüsist teadsin, et erinavaid tage on 867 ja analüüsides kaalude väärtusi selgus, et neid on kokku 986 erinevat. Otsustasin kasutada suuri ja väikseid tähti kombinatsioonide jaoks, nende abil oleks võimalik luua 2500 erinevat kahe tähelist kombinatsiooni, ehk rohkem kui mul on vaja.

Tagide puhul algasid kombinatsioonid AA ja lõppesid Qi. Kaalude puhul otsustasin, et pole mõtet väärtusi 1-99 asendada, kuna see ei tooks kaasa mahu vähenemist ja 1-9 puhul oleks tulemus vastupidine. Kaalude kombinatsioonid algasid AA ja lõppesid RC

Rakenduse poole peal hoitakse optimeerimis infot tagidega seotud tegevuste jaoks tag - kombinatsioon paaridena, selleks et tagide otsimine või filtreeride tüübi järgi oleks võimalik otsitavad tagid muundada kombinatsioonideks. Kaalude puhul on see vastupidine, ehk hoitakse kombinatsioon - kaal paarides, sest on vaja kätte saada kombinatsioonile vastav kaal.

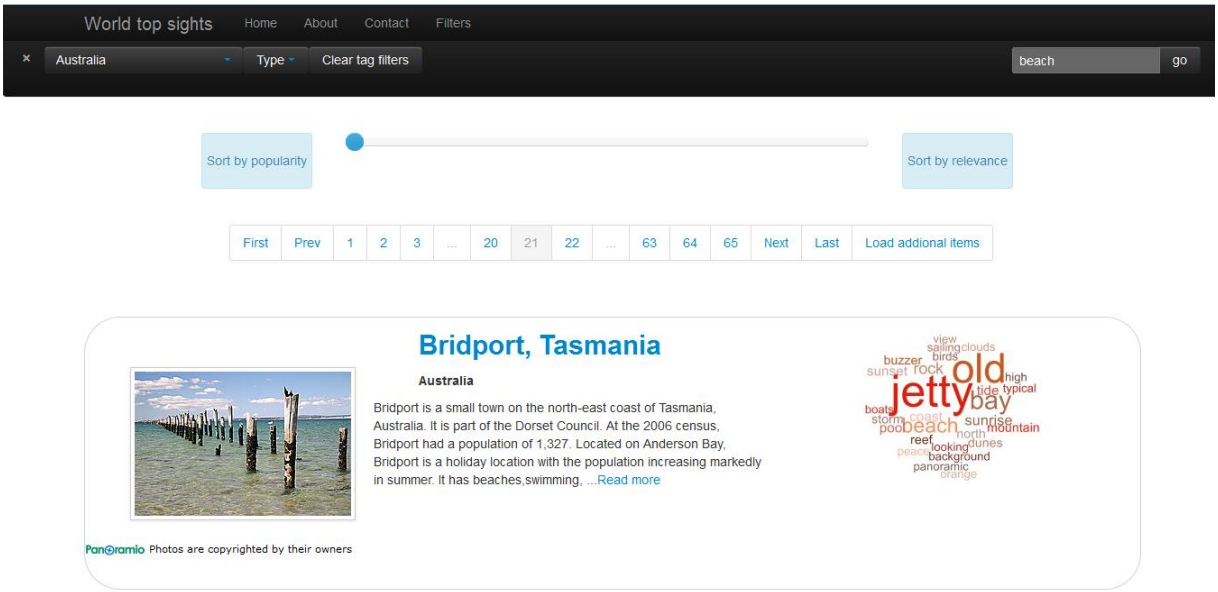
Kogu andmete tagide esitamiseks kasutati ennem 13293805 tähemärgi, kombinatsioonidega kasutatakse 4864216 tähemärki, mis on 35.59% esialgsest. Kõigi kaalude andmete esitamiseks kasutati ennem 2664560 tähemärki, kombinatsioonidega kasutatakse 2653462 tähemärki, mis on 99.58 % esialgsest. Põhjus, miks kaalude esitust ei õnnestunud palju optimeerida, on see, et enamus kaalude väärtused on 1-99.

9 Ülevaade prototüübist

Lõputöö tulemusena valmis prototüübina(pilt nr 11) lehekülg maailmas enim pildistatud turismiobjektidest.

Lehel on võimalik filtreerida üksikute tagide kaupa või ka kategooriate kaupa. Tüübi filter võimaldab valida mitut erinevat kategooriat näitamiseks. Vastavalt valitud riigile ja valitud tüüpidele või tagidele kuvatakse kasutajale vastavad turismiobjektid. Kui on rakendatud, mingit filtreerimist, kas siis tagide või kategooriatega, siis kuvatakse sorteerimise jaoks slaider, mis võimaldab sorteerida kas siis rohkem populaarsuse järgi või rohkem otsitava tagi või kategooria tähtsuse järgi turismikohas. Igal lehel kuvatakse kuni 10 kohta ning kasutajal on võimalik liikuda erinevate lehtede vahel. Esmalt laetakse teatud hulk populaarseimaid kohti valitud riigis, seda selleks, et kiirendada lehe tööd. Kasutajal on võimalus jooksvalt objekte juurde laadida. Juurde laetud objektidele rakendatakse ka filtreid ja sorteerimist.

Iga turismiobjekti juures kuvatakse selle tagipilv, olemasolul wikipedia lühikirjeldus ja link sinna, pilt panoramiast. Pealkirja link viib vastavale kohale sightsmapil.



Pilt nr 11. Ekraanipilt prototüübist

Prototüüp on tehtud kasutades javascripit, html, css. Kuna server osa pole antud töös käsitletud, siis veebisirvija laeb ise alla vastavad andmefailid ja filtreerib nende põhjal. Iga riigi andmed on jaotatud väikesteks json formaadis failideks, kus on kuni 30 turimisobjekti andmed. Iga riigi kohta on olemas ka info fail, kus on kirjas palju andmefaile ja objekte on. Riigi andmete failides on turimiobjekti id,nimi, tagid ja riigikood. Iga turimisobjekti kohta on ka eraldi info fail, kus on tema koordinaadid, wikipedia artikli nimi ja lühikirjeldus(kui need on olemas) ja panoramia pildi id. Turismikoha info laetakse, siis kui teda kuvatakse kasutajale. Riigi turismiobjektide andmed laetakse asünkroonselt ehk kasutaja näeb esimesi objekte, siis kui teisi veel laetakse.

10 Tagide statistika

Kokku on 867 erinevat tagi. Järgnevalt toon 10 kõige rohkem esinevat tagi ja nende keskmise kaalu kaalu protsendi tagipilves.

Tag	Esinemise arv	Keskmine kaalu protsent kogu tagipilvest
view	80440	9.60
church	42127	13.32
park	35526	11.36
old	30105	6.29
sunset	28400	5.92
river	26811	11.88
beach	25343	17.37
bridge	25007	12.03
panoramic	24669	6.28
tower	22045	7.25

11 Kokkuvõte

Lõputöö tulemusena analüüsit sightsmap[3,10] tegemisel kogutud turismiobjektide tage. Tagid järjestati nende esinemis sageduse ja tähtuse järgi ning lähimalt analüüsiti umbes esimest 200. Jaotades neid kategooriateks ja märkides üles liialt üldiseid ja vigaseid tage, mis eemaldati. Väikse tagi hulga kohtade wikipedia ja foursquare tüüpe analüüsiti sarnaselt, et nende abil rikastada tagide nimekirja. Leitud kategooriate abil loodi tagide järgi otsimise jaoks nii-öelda "autocomplete", kus pakutakse välja ka kategooriaid ning nende tage ja alamkategooriaid. Valmis algoritm, et filtreerimise ja tagide järgi otsimise tulemust sorteerida vastavalt kas rohkem populaarsuse või otsitavate tagide osa järgi vastavas turismiobjektis. Tagide visuaalseks esitamiseks kasutatati tagipilve, selle tegemiseks prooviti läbi kolm erinevat teeki ja valiti nendest üks. Tagipilvede joonistamiseks valmis mitmelõimeline rakendus, et seda protsessi kiirendada. Tagide ja nende kaalude esitust optimeeriti, et vähendada andmete mahtu. Prototüübina valmis lehekülg maailma enim pildistatud kohtadest, võimaldades neid filtreerida tagide, kategooriate või riigi järgi ja saadud tulemust sorteerida. Iga turismiobjekti juures kuvatakse selle wikipediast võetud tutvustus ja panoramiast leitud pilt, samuti antud koha tagi pilv.

Lõputöö raamidest jäi välja, et kuidas andmeid hoida andmebaasis ning kuidas nende hulgast leida kiirelt otsitud tagidele vastavaid turimisobjekte.

12 Viited

1. Luis Moura e Silva, Rajkumar Buyya, Parallel Programming Models and Paradigms: http://www.buyya.com/cluster/v2chap1.pdf
2. Kuo-Chan Huang, Feng-Jian Wang. Design Patterns for Parallel Computations of Master-Slave Model ---- International Conference on Information, Communications and Signal Processing ICICS '97 Singapore, 9-12 September 1997
3. Sightsmap [WWW] http://www.sightsmap.com/ (23.11.2014)
4. jQCloud [WWW] https://github.com/lucaong/jQCloud (23.11.2014)
5. awesomeCloud [WWW] https://github.com/indyarmy/jQuery.awesomeCloud.plugin (23.11.2014)
6. wordcram [WWW] http://wordcram.org/ (23.11.2014)
7. Rey-López, M.; Barragáns-Martínez, A.B.; Peleteiro, A.; Mikic-Fonte, F.A.; Burguillo, J.C., "moreTourism: Mobile recommendations for tourism," *Consumer Electronics (ICCE), 2011 IEEE International Conference on* , vol., no., pp.347,348, 9-12 Jan. 2011
8. Liangliang Cao; Jiebo Luo; Gallagher, A.; Xin Jin; Jiawei Han; Huang, T.S., "Aworldwide tourism recommendation system based on geotaggedweb photos," *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on* , vol., no., pp.2274,2277, 14-19 March 2010
9. Kai Jiang; Peng Wang; Nenghai Yu, "ContextRank: Personalized Tourism Recommendation by Exploiting Context Information of Geotagged Web Photos," *Image and Graphics (ICIG), 2011 Sixth International Conference on* , vol., no., pp.931,937, 12-15 Aug. 2011
10. Tammet, T.; Luberg, A.; Järv, P. (2013). Sightsmap: crowd-sourced popularity of the world places. *In: Information and Communication Technologies in Tourism 2013: ENTER 2013, Innsbruck, Austria, January 22-25, 2013. (Eds.)Cantoni, L.; Xiang, Z.*. Springer, 2013.