ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

გიორგი ჭიჭინაძე

მონაცემთა ბაზაში ინფორმაციის საძიებო სისტემის მოდელი

სამაგისტრო პროგრამა: ინფორმაციული ტექნოლოგიები

ნაშრომი შესრულებულია ინფორმაციულ ტექნოლოგიებში
მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: მაია არჩუაძე

თბილისი

2014

შინაარსი

ანოტაცია 3

შესავალი 4

თავი I. ინფორმაციის ძიება, საწყისი ამოცანები და რეალობა 6

1.1 ადრეული ვითარება 6

1.2 ინფორმაცია, მონაცემების ძიება 7

1.3 ძებნის თანამედროვე ამოცანა 8

1.4 მომხმარებლის ამოცანა 9

1.5 ძიების პროცესი 10

1.6 ინფორმაციული ძებნის ხარისხის შეფასება 12

1.7 ძებნის რანჟირებული შედეგების შეფასება 14

თავი II. ინფორმაციული ძებნის მოდელები 16

2.1 ბულის ძებნა 18

2.2 ვექტორული სივრცის მოდელი 19

თავი III. ბულის ძებნაზე დაფუძნებული მოდიფიცირებული მეთოდი კონცეპტების გამოყენებით 23

3.1 კონცეპტების გამოყენება ინფორმაციის ძიებაში 23

3.2 მეთოდის აღწერა 26

თავი IV. საძიებო ძრავის ალგორითმის რეალიზაცია 30

დასკვნა 34

ლიტერატურა 35

ანოტაცია

ნაშრომში განხილულია თანამედროვე საინფორმაციო ტექნოლოგიების ერთერთი ყველაზე აქტუალური და პოპულარული ამოცანა - სემანტიკური ძებნა. გაანალიზებულია არსებული პრობლემები და მათი გადაჭრის თანამედროვე მოდელები, როგორც ალგორითმულ ასევე პროგრამულ დონეზე. რელიზაციისათვის შერჩეულია თსუ-ს მკვლევართა ჯგუფის მიერ შემუშავებული სემანტიკური ძებნის ალგორითმი. ალგორითმი რეალიზებულია პროგრამული აპლიკაციის სახით, რომლის გამოყენება შესაძლებელია ისეთ მონაცემთა ბაზაში სადაც განთავსებულია ძირითადად ტექსტური ინფორმაცია. პროგრამა რეალიზებულია დაპროგრამების ენა JavaScript და Html-ის ისტრუმენტების გამოყენებით.

Annotation

The article deals with one of the most actual and popular tasks of the modern informational technologies - semantic search. There is analyzed the existing problems and modern models as at the algorithmic level as programmatic to solve them. Semantic search algorithm is chosen for the realization which is developed by the research group of Tbilisi State Uiversity. The algorithm is realized in the form of software applications, which can be used in such database where the textual information is located on the whole. The program is implemented using Javascript and HTml.

შესავალი

ინფორმაციული ძებნის ამოცანა წარმოადგენს ინფორმატიკის კლასიკურ და უაღრესად აქტუალურ ამოცანას, რომელიც ინფორმაციული ნაკადის ზრდასთან ერთად დღითიდღე სულ უფრო მნიშვნელოვანი ხდება. მიუხედავად იმისა, რომ ამ სფეროში კვლევები აქტიურად და დიდი ხნის განმავობაში მიმდინარეობს, თანამედროვე საძიებო სისტემები სრულყოფილები არ არიან. ამის გამო ყველა წამყვანი და არაწამყვანი ფირმები, რომლებიც ამ სფეროში მოღვაწეობენ ცდილობე საძიებო სისტემების სრულყოფასა და ახალი მიდგომების შემუშავებას.

ინფორმაციის ძიებისას (IR) საუბარია წარმოდგენაზე, მეხსიერებაზე, ორგანიზებაზე და ინფორმაციის ნაწილებზე წვდომაზე. ინფორმაციის ნაწილების წარმოდგენა და ორგანიზება უზრუნველყოფს მომხმარებელის მარტივი კავშირით წდომას იმ ინფორმაციზე რომელიც აინტერესებს.

დღეისათვის ინფორმაციის ძებნის ეფექტური ალგორითმების შექმნა საკმაოდ აქტუალურია. მთავარი ამოცანაა ინფორმაციის ნაკადის მზარდ მოცულობაში მომხმარებლის მოთხოვნის შესაბამისი რელევანტური ინფორმაციის მიღება. ამ პროცესში გასათვალისწინებელია შემდეგი ძირითადი ასპექტები: პასუხი მოთხოვნაზე იყოს რაც შეიძლება ზუსტი (სემანტიკური თვალსაზრისით), პასუხი იყოს სრული და პასუხი იყოს რაც შეიძლება სწრაფი.

სემანტიკური თვალსაზრისით ზუსტი პასუხის მისაღებად აუცილებელია თვით მომხმარებლის მოთხოვნა წარმოსდგეს ისეთი ფორმალიზებული სახით, რომ არ დაირღვეს მისი შინაარსობრივი მნიშვნელობა. გამომდინარე ენობრივი თავისებურებებიდან ინფორმაციის ძებნისას გასათვალისწინებელი ფაქტორია საძიებო ენა. მაგ: მოიძებნოს ყველა გვერდი შემცველი ინფორმაციისა კოლეჯის ჩოგბურთის გუნდებისა რომლებიც მონაწილეობს NCAA ჩოგბურთის ტურნირში. რელევანტურობისათვის გვერდი უნდა შეიცავდეს ინფორმაციას ნაციონალურ რეიტინგს გუნდებისა ბოლო სამი წლის განმავლობაში და გუნდის მწვრთნელის მეილს.

ნათელია, რომ ეს სრული აღწერა მომხმარებლისათვის საჭირო ინფორმაციისა არ შეიძლება გამოყენებული იქნას პირდაპირი ძებნისათვის დღევანდელ ძებნის სისტემებში.

ყველაზე მთავარია ითარგმნოს ის გასაღები სიტყვები რომელიც შეაჯამებს მომხმარებლისათვის საჭირო ინფორმაციას. მომხმარებლის საძიებო სიტყვა არის მთავარი გასღები IR სისტემისა რომელიც შეძლებს მოიძიოს მომხმარებლისათვის რელევანტურური ინფორმაცია.

მარტივი ანალიზი აჩვენებს, რომ ერთიდაიგივე საძიებო სისტემაში ქართულ და ინგლისურ ენაზე გაკეთებულ მოთხოვნაზე მიღებული შედეგების ხარისხი მკვეთრად განსხვავდება ერთმანეთისაგან. უპირველესი ამის მიზეზი არის ის, რომ ნაკლებადაა ქართულენოვან ინტერნეტ სივრცეში ვებ გვერდები წარმოდგენილი SWD დოკუმენტების სახით; სხვა მიზეზად შეიძლება დასახელდეს ქართული ენის მორფოლოგიურ-სინტაქსური სირთულე, რაც არანაკლებ მნიშვნელოვანია; ასევე პრობლემაა ქართული ენის კორპუსების სიმცირე და ხელმისაწვდომობა ინტერნეტ სივრცეში, რაც რიგი საძიებო სისტემების მიერ გამოიყენება ლექსიკონებისა და ონტოლოგიების შესაქმნელად არა მარტო ძებნის არამედ მანქანური თარგმნის უზრუნველსაყოფად.

გამომდინარე ზემოთთქმულიდან ნებისმიერი მცდელობა შეიქმნას ახალი ალგორითმი, რომელიც გააუმჯობესებს სემანტიკური ძებნის პროცესს, მეტად აქტუალურია.

თავი I. ინფორმაციის ძიება, საწყისი ამოცანები და რეალობა

1.1 ადრეული ვითარება

დაახლოებით 4000 წელი ადამიანი ახდენდა ინფორმაციის ორგანიზებას შემდეგი ძიებისთვის და გამოყენებისათვის. მას შემდეგ რაც ინფორმაციის მოცულობა საგრძნობლად გაიზარდა საჭირო გახდა სპეციალური სტრუქტურის შექმნა რათა მომხდარიყო სწრაფი წვდომა შენახულ ინფორმაციაზე. ძველი და პოპულარული მონაცემთა სტრუქტურები შექმნილი სწრაფი ინფორმაციის ძიებისათვის იყენებდა არჩეული სიტყვების კოლექციას ან კონცეპტს რომელიც ასოცირდებოდა მიზნებთან დაკავშირებულ ინფორმაციასთან - ინდექსთან. ინდექსის ესა თუ ის ფორმა არის ბირთვი ყოველი თანამედროვე საძიებო სისტემისა. ინდექსი უზრუნველყოფს სწრაფ წვდომას მონაცემებზე. საუკუნეების განმავლობაში ინდექსები იქმნებოდა ხელით , როგორც იერარქიების კატეგორიზაცია. ფაქტია. რომ დიდი ნაწილი ბიბილიოთეკებისა ჯერ კიდევ გამოიყენებენ ფორმას კატეგირიზირებული იერარქიისა იმისათვის , რომ მოახდინოს მონაცემების კლასიფიკაცია. თნამედროვე კომპიუტერები ქმნიან დიდი ზომის ავტომატურ ინდექსებს. მნიშვნელოვანია გარჩევა მოხდეს IR ის ორი განსხვავებული წარმოდგენის პრობლემისა: computer-centered და human-centered.

computer-centered მთავარი მიზანია შექმნას ეფექტური ინდექსები, დაამუშაოს მომხმარებლის შეკითხვა მაღალი სისფრაფით და შექმნას ალგორითმების რეიტინგი, რომელიც აუმჯობესებს პასუხის ხარისხს.

ბიბლიოთეკებია, სადაც მოხდა პირველად IR სისტემის გამოყენაბა ინფორმაციის ძიებისათვის. პირველი თაობის ძიებები ეყრდნობოდა წიგნის დასახელებით ან მწერლის მიხედვით ძიებას. მეორე თაობაში დაემატა სუბიექტები, გასაღები სიტყვები და სხვა კომპლექსური ინფორმაციები. მესამე თაობა ფოკუსირებულია გრაფიკულ ინტეფეისზე, ელექტრონულ ფორმებზე, ჰიპერტექტებზე და ღია სისტემის არქიტექტურებზე.

თუ დავაკვირდებით თანამედროვე ძიების მეთოდებს WEB ში , აღმოვაჩენთ რომ ის

იყენებს ინდექსებს ისეთივეს როგორსაც იყენებდნენ ბიბლიოთეკები საუკუნის წინ.

ფუნდამენტური ცვლილებები მოხდა როგორც თნამედროვე კომპიუტერულ ტექნოლოგიებში, ასევე WEB ში. პირველ რიგში გახდა იაფი გარკვეულ ინფორმაციებზე წვდომა, რამაც გამოიწვია აუდიენციის გაზრდა. მეორე, ქსელური ტექნოლოგიების

განვითარებამ შესაძლებელი გახადა ინფორმაციის გაცვლა მოშორებული ადგილებიდან სწრაფად. მესამე, ინფორმაციის გამოქვეყნების თავისუფლება.

დაბალმა ღირებულებამ, ჩინებულმა კავშირმა და გამოქვეყნების თავისუფლებამ მისცა საშუალება მომხმარებელს გამოეყენებინა WEB , როგორც ინტერაქტიული გარემო.

მომავალში სამი ძირითადი შეკითხვა იქნება აქტუალური. პირველი მაღალი ინტერაქტიულობა , რათა მომხმარებელს გაუადვილდეს მისთვის საჭირო ინფორმაციის მოძიება. მეორე ძიების სისწრაფე. მეასმე ინტერაქტიული ინტერფეისის ხარისხი.

1.2 ინფორმაცია, მონაცემების ძიება

მონაცემების ძიება IR სისტემის კონტექსტში ძირითადათ განსაზღვრავს დოკუმენტების კოლექციას შემცველ იმ გასაღები სიტყვებისაგან მომხმარებლის საძიებო სიტყვაში, რომელიც ყველაზე ხშირად არ არის საკმარისი დააკმაყოფილოს მომხმარებლს მოთხოვნები.

ფაქტობრივად IR სისტემის მომხმარებელს აინტერესებს ინფორმაცია სუბიექტზე, რომელიც შემდეგ მოიძიებს მონაცემებს დასმულ შეკითხვაზე. მონაცემების ძიების ენა იღებს ყველა ობიექტს რომელიც დააკმაყოფილებს განსაზღვრულ პირობას ან ალგებრულ გამოსახულებას. ამგვარად, მონაცემთა ძიების სიტემისათვის ერთი შეცდომითი ობიექტი ნიშნავს მთლიან შეცდომას. თუმცა მოძიებული ობიექტებისათვის შეიძლება არააკურატული და მცირე შეცდომა შეუმჩნეველი დარჩეს მონაცემთა ძიების პროცესისათვის. ამის ძირითადი მიზეზი არის ნატურალური ენის ტექსტი რომელიც შეიძლება სტრუქტურულად და სემანტიკურად არ იყოს გამართული. მეორე მხარეს კი მონაცემთა ძიების სისტემაში საუბარია კარგად ორგანიზებულ სტრუქტურაზე და სემანტიკაზე.

მონაცემების ძიება არ ჭრის პრობლემას ინფორმაციის მოძიებისა ობიექტის შესახებ ან საძიებო თემაზე, არამედ ის ცდილობს დააკმაყოფილოს მომხმარებლის ინტერესი. IR სისტემამ უნდა როგორმე წარმოადგინოს ინფორმაციის ნაწილების(დოკუმენტების) კოლექცია და რეიტინგი მათი შესაბამისი ხარისხის მიხედვით მომხმარებლის საძიებო კითხვაზე. ეს ინტერპრეტაცია დოკუმენტის კონტენტისა კრებს სინტაქსურ და სემანტიკურ ინფორმაციას დოკომენტიდან. ამგვარად ცნება არის ცენტრი ინფორმაციის ძიებისა. ფაქტიურად მთავარი მიზანი IR სისტემისა არის მოიძიოს ყველა დოკუმენტი რომელიც

არის რელევანტური მომხმარებლის საძიებო კითხვაზე იქამდე სანამ მოიძიებს არა რელევანტურ დოკუმენტებს.

რელევანტური ინფორმაციის ეფექტური ძიება პირდაპირ უკავშირდება როგორც მომხმარებლის ამოცანას ასევე ლოგიკურ დოკუმენტის სახეს ადაპრიტებულს ძიების სისტემის მიერ.

1.3 ძებნის თანამედროვე ამოცანა

ტერმინი IR გულისხმობს არასტრუქტურირებული მონაცემტა ბაზიდან ინფორმაციის დაბრუნებას. ძებნა შეიძლება მოხდეს სხვადასხვა ტიპის ინფორმაციის: გრაფიკული, ვიდეო, აუდიო. მაგრამ ძირითადად ძებნა ფოკუსირებულია ბუნებრივი ენის ტექსტებზე. რაც გულისხმობს ძებნას დოკუმენტების კოლექციაში.ძებნის პროცესი მოიცავს აგრეთვე დოკუმენტების ფილტრაციის, მოდელირების, კლასიფიკაციის, კლასტერიზაციის, ამავე დროს საძიებო სისტემის არქიტექტურის დაპროექტების, სამომხმარებლო ინტერფეისის და მოთხოვნათა ენის განსაზღვრის ამოცანებს.

ინფორმაციის ძებნა შესაძლებელია როგორც სტრუქტურირებულ, ასევე არასტრუქტურირებულ, ან ნახევრად სტრუქტურირებულ ბაზებში.

ჩანაწერი სტრუქტურირებულია, თუ იგი შეიცავს სახელდებულ კომპონენტებს, ორგანიზებულს გარკვეული სინტაქსური კანონზომიერებით. მაგ. რელაციურ მონაცემთა ბაზის ცხრილში შეიძლება იყოს ბევრი ჩანაწერი და ყველა სტრიქონს აქვთ ერთიდაიგივე ველები. გარდა ამისა ცანაწერის თითოეულ ატრიბუტს შეიძლება გააჩნდეს განსაზღვრული მნიშვნელობა და იგი იყოს ერთიდაიგივე სემანტიკის მატარებელი ყველა ჩანაწერში. მბმს ეძებს ამ კომპონენტს სინტაქსის მიხედვით და აბრუნებს მნიშვნელობას. მაგ. ,,თსუ“ მონაცამთა ბაზის ,,სტუდენტის“ ცხრილიდან დავაბრუნოთ ,,სქესი“ . მოთხვნის შესრულებისას აუცილებელია მოხდეს დაზუსტება, თუ რომელი ცხრილიდან მოხდეს ამ ჩანაწერების წამოღება, რადგან იგივე ველი შეიძლება გააჩნდეს სხვა ცხრილსაც. მაგ ,,თანამშრომელი“. როდესაც საჭირო ხდება კონკრეტული ობიექტის შესახებ ინფორმაციის მიღება, საჭიროა მოხდეს ჩანაწერების იდენტიფიცირება. მართალია ,,ასაკი“ ველი არ არის საიდენტიფიკაციო, მაგრამ რელაციური ბაზის სტრუქტურა იძლევა ჩანაწერების იდენტიფიცირების საშუალებას. მაგ ველი : ,,პირადი ნომერი“.

არასტურქტურირებული ბუნებრივი ენების დოკუმენტების კოლექციაში (ბაზაში), არ არსებობს განსაზღვრული სინტაქსური კანონზომიერება , რითაც ძებნის სისტემას ექნება საშუალება მოცემული სემანტიკის ჩანაწერების მოძებნის, მაგ ,,ასაკი“. როდესაც ბაზაში მონაცემები შემთხვევითაა დალაგებული, არ არის გარანტია იმისა, რომ ასკი, ყოველთვის შეესაბამება რეალურად ,,ასაკს“ . ანუ არ არსებობს საშუალება, რომელიც ზუსტად განსაზღვრავს თუ სად უნდა მოხდეს ძებნა, რომელ წინადადებაში, ან რომელ აბზაცში. არ არსებობს წესი, რომლის მიხედვითაც შესაძლებელია სინტაქსურად ერთნაირი ჩანაწერების ძებნა.

ზოგიერთ შემთხვევაში შესაძლებელია მოხდეს დოკუმენტების კოლექციიდან სტრუქტურის და სემანტიკის გამოყოფა. ასეთ შემთხვევაში თითოეული აბზაცისათვის განსაზღვრულია თუ რა ინფორმაცია უნდა განტავსდეს მაგ, პირველ, მეორე, მესამე და ა. შ აბზაცებში. ასეთი ბაზა არის ნახევრადსტრუქტურირებული.მართალია ამ შემთხვევაში არ არსებობს რელაციური ცხრილები კონკრეტული ველებით, მაგრამ შესაძლებელია ალგორითმით საგასაღებო სიტყვების საშუალებით მოხდეს მათი წარმოდგენა სტრუქტურირებულ ცხრილებში.

1.4 მომხმარებლის ამოცანა

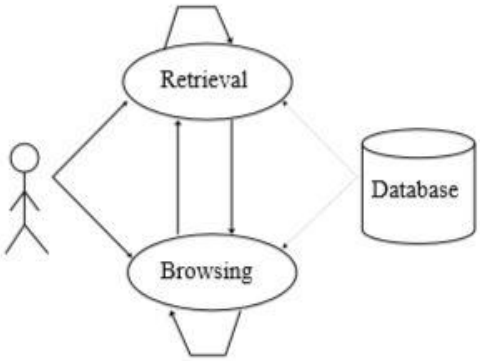
საძიებო სისტემის მომხმარებელმა უნდა თარგმნოს მისთვის საჭირო ინფორმაცია საძიებო ტექსტად(query) , სისტემის მიერ დადგენილი პირობების მიხედვით. ინფორმაციის საძიებო სისტემა ქმნის სასურველი სიტყვების კომპლექტს რომელიც გადმოცემს საჭირო ინფორმაციის სემანტიკას. ჩვენ შეიძლება ვთქვათ ,იმისათვის რომ მომხმარებელმა მოიძიოს საჭირო ინფორმაცია ასრულებს ძიების ამოცანას.

განვიხილოთ ისეთი შემთხვევა როცა მომხმარებელს აინტერესებს ცუდად განსაზღვრული ან არსებითად ფართო ინფორმაცია. მაგალითისათვის მომხმარებელს აინტერესებს დოკუმენტები მანქანების რბოლის შესახებ. ამ შემთხვევაში მომხმარებელი შეიძლება იყენებდეს ინტერაქტიულ ინტერფეისს რათა მარტივად დაათვალიეროს მანქანების რბოლის შესახებ არსებული დოკუმენტები. მაგალითად მომხმარებელს აინტერესებს დოკუმენტები ფორმულა 1 -ის ან მანქანების მწამოებლების ან „ ლე მანის 24 საათიანი რბოლის“ შესახებ. „ ლე მანის 24 საათიანი რბოლა“ ძიების შემთხვევაში მომხმარებელი შეიძლება დააინტერესოს დოკუმენტები რომელიც შეიცავს ინფორმაციას

ტურიზმის შესახებ საფრანგეთში. ამ შემთხვევაში ჩვენ ვამბობთ, რომ მომხმარებელი ათვალიერებს დოკუმენტების კოლექციას და არ ეძებს დოკუმენტს. ეს ჯერ კიდევ ინფორმაციის მოძიების პროცესია, მაგრამ ძიების ზუსტი ობიექტი ჯერ კიდევ არ არის განსაზღვრული და შეიძლება შეიცვალოს ძიების პროცესის დროს. აქედან გამომდინარე შეიძლება დავასკვნათ საძიებო სისტემის მომხმარებლის განსხვავებული ამოცანები. მომხმარებლის ამოცანა შესაძლოა იყოს ორი განსხვავებული ტიპის: ინფორმაციის ან მონაცემების ძიება და თვალიერება. კლასიკური საძიებო სისტემები ძირითადათ ანხორციელებენ ინფორმაციის ან მონაცემების ძიებას.

ჰიპერტექსტული სისტემები გამოიყენება ინფორმაციის ძიების სისწრაფისათვის. თანამედროვე ციფრული ბიბლიოთეკები და WEB ინტერფეისები ცდილობენ მოახდინონ კომბინირება ამ ორი ამოცანისა , რათა შემოგვთავაზონ გაუმჯობესებული შესაძლებლობები. თუმცა, ძიების და დათვალიერების კომბინაციის მიდგომის შექმნა არ არის დომინანტური პარადიგმა.

სურათზე გამოსახულია მომხმარებლის განხვავებული ამოცანები ძიების და მოგზაურობის პროცესი



ორივე ძიება და მოგზაურობა არის WEB -ის ენა. ეს არის მომხმარებლის მოთხოვნა ინფორმაციისა ინტერაქტიული ფორმით.

1.5 ძიების პროცესი

ახლა ჩვენ მზად ვართ რომ დეტალურად აღვწეროთ ძიების პროცესი. იმისათვის , რომ ავღწეროთ ძიების პროცესი გამოვიყენოთ მარტივი არქიტექტურა რომელიც მოცემულია სურათზე. პირველ რიგში ძიების პროცესის დაწყებამდე აუცილებლად საჭიროა შეიქმნას ტექსტური ბაზა. ეს პროცესი ძირითადათ ხორციელდება ბაზის

მენეჯერის მიერ, ვინც განსაზღვრავს შემდეგს: ა) გამოსაყენებელ დოკუმენტებს ბ) ოპერაციებს რომლებიც უნდა განხორციელდეს ტექსტზე და გ) ტექსტის მოდელს. ტექსტ ოპერაციები გარდაქმნის ორიგინალ დოკუმენტებს და აგენერირებს მათ ლოგიკურ განსაზღვრას.

მხოლოდ დოკუმენტის ლოგიკურ განსაზღვრებას აღწერს ბაზის მენეჯერი, რომელიც ქმნის ტექსტის ინდექსს. ინდექსი არის მონაცემთა სტრუქტურა , რომელიც გვაძლევს საშუალებას სწრაფი ძიებისა დიდი მოცულობის მონაცემებში. არსებობს სხვადასხვა ტიპის ინდექსის სტრუქტურა, მაგრამ მათგან ყველაზე პოპულარულია შებრუნებული ინდექსი(inverted file) და რევერსული ინდექსი(reverse index).

მას შემდეგ რაც დოკუმენტების მონაცემები დაინდექსდება შესაძლებელი ხდება ძიების პროცესის დაწყება. მომხმარებელი პირველ რიგში აღნიშნავს მომხმარებლის საჭიროებას(user need) რომელიც შემდეგ გაანალიზდება და ტრანფორმირდება იგივე ტექსტური ოპერაციების გამოყენებით ტექსტზე. ტექსტური ოპერაციების შემდეგ მიღებული შედეგი შეიძლება გამოყენებული იქნას როგორც აქტუალური საძიებო ტექსტი, რომელიც უზრუნველყოფს სისტემის წარმოდგენას მომხმარებლის საჭიროებისათვის. საძიებო ტექსტი დამუშავება იძლევა საშუალებას მოპოვებული იქნას საძიებო დოკუმენტი. სწრაფი დამუშავების საშუალებას გვაძლევს უკვე შექმნილი ინდესაციის სტრუქტურა. სანამ მომხმარებელს გაეგზავნება საძიებო დოკუმენტები ისინი დალაგდებიან ალბათობის მიხედვით. შემდეგ მომხმარებელი იკვლევს გალაგებული დოკუმენტებს იმისათვის, რომ იპოვოს საჭირო ინფორმაცია. მომხმარებლის მიერ არჩეული დოკუმენტების მიხედვით შეიძლება შეიცვალოს საძიებო ტექსტის ფორმულირება. იმედია , რომ მოდიფიცირებული საძიებო ტექსტი იქნება უკეთესი წარმოდგენა რეალური მომხმარებლის საჭიროებისა.

მიგვაჩნია, რომ ახლა მზად არის მომხმარებლის ინტეფეისი არსებული ინფორმაციის ძიების სისტემებისთვის. პირველ რიგში უნდა ავღნიშნოთ, რომ მომხმარებელი თითქმის ვერასოდეს განსაზღვრავს თავისათვის საჭირო ინფორმაციას. უმრავლესი მომხმარებელი ვერ აყალიბებს ადექვატურ საძიებო ტექსტს . ამიტომ არ იქნება მოულოდნელობა, რომ ცუდათ ფორმულირებული საძიებო ტექსტი გამოიწვევს ცუდ ძიებას, როგორც ეს ხშირად ხდება WEB-ში.

არასტრუქტურირებული დოკუმენტები სტრუქტურირებული სათაურები. დოკუმენტები ხშირად ნაწილობრივ სტრუქტურირებულია. მათ გააჩნიათ

სტრუქტურირებული ჰიდერი და არასტურქტურირებული ტანი. მაგრამ ჰიდერი ძირითადა შეიცავს მეტამონაცემებს, ინფორმაციას დოკუმენტზე. ბიბლიოგრაფიულ დოკუმენტებში წიგნები, ჟურნალები

მეტამონაცემებია ავტორის დასახელება, გამოცემის წელი. უფრო კონკრეტულად, მეტამონაცემები არის ის მონაცემები, რომელთა ძებნაც შესაძლებელია ბიბლიოთეკის კატალოგში. ელ. ფოსტაში სტრუქტურირებულია From, To, subject. ძებნის სისტემა ადვილად ეძებს მეტამონაცემებით კონკრეტულ ობიექტს.

დოკუმენტები ხშირად სტრუქტურირებულია სხვა გზითაც. მათ გააჩნიათ ,,დოკუმენტის„ სტრუქტურა , რაც გულისხმობს შემდეგს: დოკუმენტს გააჩნია სარჩევი, წინასიტყვაობა, დაყოფილია თავებად, რომლებიც შეიცავს სურათებს, ცხრილებს, გრაფიკას. პროგრამული უზრუნველყოფის ინსტრუმენტებით შესაძლებელია მოხდეს ამ სტრუქტურის ცალკეული ნაწილების ძებნა ფორმატების მიხედვით: indentation, საგასაღებო სიტყვებით: Index, Figure. შეიძლება ძებნა მოხდეს თავების, სექციების, ფიგურების, მაგრამ ძებნა სექციის, რომელიც შესატყვისია ფრაზის ,,ინფორმაციული ძებნა“ შეუძლებელია.

ინფორმაციული ძებნის მიზანი. ინფორმაცული ძებნის ამოცანა ისეთი დოკუმენტების ძებნაა, რომელთა შინაარსიც წარმოდგენილია არასტრუქტურირებული კომპონენტებით. მოთხოვნა შეიძლება შეიცავდეს როგორც სტრუქტურირებულ, ასევე არასტრუქტურირებულ კომპონენტებს. მაგ. ,,დოკუმენტი ქვანტურ ალგორითმებზე, გროვერის ალგორითმი“. ცხადია დოკუმენტის ტანი არასტრუქტურირებული, ხოლო სტრუქტურირებული ნაწილია ,,გროვერი“.

მოთხოვნა ფორმირდება მომხმარებლის მიერ. თუ ძებნის სისტემის შედეგად დაბრუნებული დოკუმენტები აკმაყოფილებს მომხმარებლის მოთხოვნას, ასეთი დოკუმენტები არის რელევანტური, ხოლო თუ არა, მაშინ არარელევანტური.

1.6 ინფორმაციული ძებნის ხარისხის შეფასება

ინფორმაციის ძებნის სისტემის მთავარი მიზანი არის მომხმარებლის ინფორმაციული მოთხოვნის შესაბამისი რელევანტური ინფორმაციის დაბრუნება. დოკუმენტი რელევანტურია, თუ სრულად შეიცავს მომხმარებლისათვის საჭირო ინფორმაციას ან რაც შეიძლება ახლოსაა მის ინფორმაციულ მოთხოვნილებასთან.

ინფორმაციული ძებნის სისტემის შეფასებისას საჭიროა შემდეგი ძირითადი კომპონენტები:

- დოკუმენტი; დოკუმენტში იგულიხმება ობიექტი, რომელიც შეიცავს ინფორმაციას ფიქსირებულ ფორმატში. დოკუმენტი შეიძლება შეიცავდეს ბუნებრივ ან ფორმალურ ენებზე დაწერილ ტექსტებს, გრაფიკულ ობიექტებს, ხმოვან ინფორმაციას.
- მოთხოვნა; მოთხოვნა ეს არის მომხმარებლის მიერ ფორმალიზებული სახით წარმოდგენილი ინფორმაციული საჭიროება. ამისათვის გამოიყენება მოთხოვნათა წარმოდგენის ენები.
- რელევანტობის კრიტერიუმის განსაზღვრა. შესაბამისობა მოთხოვნასა და დაბრუნებულ დოკუმენტს შორის.

ინფორმაციული ძებნის ეფექტურობა ფასდება ორი კრიტერიუმით:

- სისრულე (recall)
- სიზუსტე (precision)

სისრულე- ძებნის შედეგად დაბრუნებული დოკუმენტების წილი მთლიანად მოძებნილ რელებანტური დოკუმენტების რაოდენობასთან მიმართებაში.

$$\textit{სისრულე} = \frac{\textit{დაბრუნებული რელევანტური დოკუმენტების რაოდენობა}}{\textit{რელევანტური დოკუმენტების სრული რაოდენობა}}$$

სიზუსტე- მთლიანად მოძებნილ დოკუმენტებში რელევანტური დოკუმენტების წილი.

$$\textit{სიზუსტე} = \frac{\textit{დაბრუნებული რელევანტური დოკუმენტების რაოდენობა}}{\textit{მთლიანად დაბრუნებული დოკუმენტების რაოდენობა}}$$

	რელევანტური	არარელევანტური
დაბრუნდა	A	B
არ დაბრუნდა	C	D

$$PRECISION = \frac{A}{(A \cup B)}$$

$$RECALL = \frac{A}{(A \cup C)}$$

ეს ორი სიდიდე ერთმანეთთან ურთიერთ საპირისპირო დამოკიდებულებაშია. ნათელია, რომ სიზუსტის გაზრდა იწვევს მთლიანობის შემცირებას და პირიქით. მაღალი ხარისხის ძებნის სისტემა ორივე სიდიდისათვის იძლევა მაღალ მაჩვენებლებს.

ბევრ სიტუაციაში აღმოჩნდება, რომ ერთი უფრო მნიშვნელოვანია ვიდრე მეორე. მაგალითად ინტერნეტ სივრცეში ინფორმაციის ძებნისას, მომხმარებლისათვის მნიშვნელოვანია სიზუსტე, ხოლო როდესაც ხდება ინფორმაციის ძებნა დისკზე, ამ შემთხვევაში სისრულე. თუ გაიზრდება კოლექციაში დოკუმენტების რაოდენობა და შესაბამისად გაიზრდება მოძებნილი დოკუმენტების რიცხვი, ეს არ გამოიწვევს სისრულის შემცირებას, მაშინ როდესაც ამ შემთხვევაში სიზუსტე იკლებს.

ამ ორ მაჩვენებელს შორის დამოკიდებულების დასაბალანსებლად გამოიყენება F ზომა

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

სადაც α არის წონა.

$\beta^2 = \frac{1-\alpha}{\alpha}$ $\alpha \in [0,1]$ $\beta^2 \in [0,\infty]$

ბალანსირებული F (F_β) ზომა სისრულეს და სიზუსტეს ერთნაირ წონებს ანიჭებს.

როცა $\alpha = \frac{1}{2}$, მაშინ $\beta = 1$,

ფორმულა მარტივდება და იღებს სახეს:

$$F_\beta = \frac{2PR}{P+R}$$

1.7 ძებნის რანჟირებული შედეგების შეფასება

სიზუსტე, სისრულე და F ზომა გამოთვლადი სიდიდეებია არასტრუქტურირებული კოლექციებისათვის. დაბრუნებულ დოკუმენტების სიმრავლიდან თუ გამოვყობთ k დოკუმენტს და ავაგებთ მრუდს სიზუსტე-სისრულე, ამ მრუდს ექნება ,,წყვეტილი'' სახე: თუ k+1 მოძებნილი დოკუმენტი არარელევანტურია, მაშინ სისრულე არ იცვლება, რჩება ისეთივე, როგორიც იყო k დოკუმეტის შემთხვევაში. ხოლო თუ დოკუმენტი რელევანტური აღმოჩნდება, მაშინ როგორც სისრულე, ასევე სიზუსტეც იზრდება.(მრუდი აკეთებს

ნახტომს ზევით და მარჯვნივ). ამიტომ ხშირ შემთხვევაში მოსახერხებელია გამოყენებულ იქნეს ინტერპოლირებული სიზუსტე P_{interp}, (interpolated precision)

$$P_{interp}(r) = \max_{r' \geq r} p\,(r')$$

ინტერპოლირებული სიზუსტე საკმაოდ ინფორმატიულია. ხშირად სასურველია ამ ინფორმაციის წარმოდგენა ერთი ან რამდენიმე მნიშვნელობით. ტრადიციულად ამისათვის გამოიყენება 11 საშუალო სიზუსტე, ინტერპოლირებული 11 წერტილით. (eleven point interpolated average precision)

უკანასკნელ წლებში სულ უფრო ფართოდ გამოიყენება სიზუსტის სხვა ზომები. მაგ MAP(mean average Precision). იგი იძლევა საშუალებას სისტემის ხარისხის განსაზღვრის სისრულის სხვადასხვა დონეზე ერთი რიცხვით.

$$MAP = \frac{\sum_{q=1}^{Q} Ave\ P(q)}{Q}$$

სადაც Q არის მოთხოვნათა რაოდენობა.თუ კოლექციაში რელევანტური დოუმენტი არ არის ნაპოვნი, მაშინ MAP=0.

(საშუალო სიზუსტე- მაგალითად მოთხოვნის შედეგად დაბრუნებული დოკუმენტების შეფასება ასეთია: 1, 0, 0, 1, 1, 1; სიზუსტე=1/1; 0; 0; 2/4; 3/4; 4/6) Ave (p)=0,6917)

A simple way to interpret is to produce a combination of zeros and ones which will give the required AP.
For example, an AP of 0.5 could have results like
`0, 1, 0, 1, 0, 1, ...`
where every second image is correct, while an AP of 0.333 has
`0, 0, 1, 0, 0, 1, 0, 0, 1, ...`
where every third image is correct.

თავი II. ინფორმაციული ძებნის მოდელები

დოკუმენტების კოლექციაში, განსაზღვრული ტერმინების მოსაძებნად, საჭიროა ყველა დოკუმენტის თანმიმდევრობით დათვალიერება. ეს არის პირდაპირი ძებნა-grepping (ბრძანება grep, რომელიც ოპერაციულ სისტემა UNIX-ში ასრულებს ამ მოთხოვნას). მაგრამ ძებნის ამ მეთოდს აქვს გარკვეული ნაკლოვანებები:

- დოკუმენტების თანმიმდევრობით დათვალიერება, როდესაც კოლექციაში მილიონობით დოკუმენტია, გარკვეულწილად ბევრ სიძნელეს ქმნის.
- ტერმინების შედარებისას საჭიროა უფრო მეტად მოქნილი სისტემა.
- რანჟირებული ძებნა. ამ დროს საჭიროა არა მხოლოდ სესაბამისი დოკუმენტის მოძებნა, არამედ ძებნის შედეგად დაბრუნებული დოკუმენტების დალაგება რანჟირების მიხედვით.

იმისათვის რომ თავიდან იქნეს აცილებული ისეთი დიდი ოპერაცია, როგორიცაა დოკუმენტების თანმიმდევრობით დათვალიერება, ხდება დოკუმენტების ინდექსაცია. ინდექსები წარმოადგენს ტერმინებს (ტერმინი შეიძლება იყოს სიტყვა, კოდი და ა. შ ნებისმიერი სტრუქტურის). ჩნდება ბინარული მატრიცა ტერმინი-დოკუმენტი. ამის შემდეგ მიიღება ტერმინების ვექტორი, სადაც ნაჩვენებია თუ რომელ დოკუმენტში გვხვდება ტერმინი, ან მიიღება დოკუმენტის ვექტორი, რომელიც გვიჩვენებს თუ რომელი ტერმინები გვხვდება ამ დოკუმენტში.

მაგ:

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Рис. 1.1. Матрица инцидентности "термин–документ". Элемент матрицы (t,d) равен 1, если пьеса в столбце d содержит слово из строки t, и 0, если не содержит

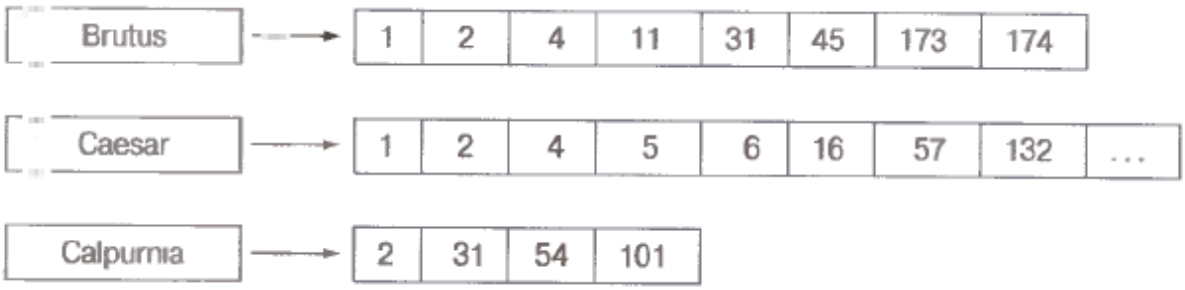
Для обработки запроса Brutus AND Caesar AND NOT Calpurnia мы берем вектор для терминов Brutus, Caesar и Calpurnia, вычисляем двоичное дополнение к последнему вектору и выполняем поразрядную операцию AND.

110100 AND 110111 AND 101111 = 100100

Ответ на запрос выглядит так: Antony and Cleopatra и Hamlet (рис. 1.2).

დოკუმენტების ჩვეულებრივი, სტანდარტული ინდექსაციის პროცესი აქ უკვე შეუძლებელი ხდება , რადგან მატრიცა ტერმინი-დოკუმენტი, უკვე ძალიან ბევრი ელემენტებისაგან შედგება. აქ უკვე ჩნდება ძებნის ახალი კონცეფცია, რომელიც ემყარება ინვერტირებული ინდექსის შექმნას.

თავიდან იქმნება ტერმინების კოლექცია(ლექსიკონი), ხოლო შემდეგ თითოეული ტერმინისათვის იქმნება დოკუმენტების სია, რომელშიც გვხვდება ეს ტერმინი. აგრეთვე იქმნება ამ ტერმინის კოორდინატი დოკუმენტში (posting) (ინვერტირებული სია).



ინვერტირებული ინდექსის აგების ძირითადი ეტაპები:

- ინდექსაციისათვის საჭირო დოკუმენტების შეკრება;
- ტექსტის დაყოფა ლექსემებად;
- ლექსემების თანმიმდევრული ლინგვისტური დამუშავება; იქმნება ნორმალიზებული ლექსემების სია, რომელიც წარმოადგენს ინდექსირებულ ტერმინებს
- ხდება დოკუმენტების ინდექსაცია ტერმინების მიხედვით, იქმნება ინვერტირებული ინდექსი, რომელიც მიუთითებს თუ სად და რა პოზიციაში გვხვდება კონკრეტული ტერმინი.

2.1 ბულის ძებნა

ძებნის სტარატეგია დაფუძნებულია მოთხოვნასა და დოკუმენტს შორის მსგავსების გაზომვაზე.

ძებნის სტრატეგიები იდენტიფიცირებულია როგორც ძებნის მოდელები:

- **ბულის მოდელი;**
- **ვექტორული სივრცის მოდელი;**
- **ალბათური ძებნის მოდელი;**
- **სემანტიკური ინდექსაცია;**
- **ენის მოდელები;**
- **არამკაფიო ძებნა.**

ბულის ძებნის მოდელი ეს არის ინფორმაციის ძებნის მოდელი, რომელის დროსაც ხდება ბულის გამოსახულებებით (AND, OR, NOT) შედგენილი მოთხოვნის დამუშავება. ძებნის შედეგი უნდა იყოს რელევანტური მოთხოვნასთან მიმართებაში. ინფორმაციული სისტემის ამოცანაა ad-hoc მოთხოვნის დამუშავება.

ბულის ძებნის მოდელში არის პრობლემა. მოთხოვნის შედეგი არ არის რანჟირებული. შედეგები შეიძლება დალაგებული იყოს თარიღის მიხედვით, ან სხვა პარამეტრით. რეიტინგს ნაკლები დატვირთვა აქვს. ამიტომ ძებნის სისტემები, რომლებიც მუშაობენ ბულის მოდელით, ნაკლებად ეფექტურ შედეგს იძლევა, ვიდრე ისი სისტემები,

რომლებიც რანჟირებაზე მუშაობენ. ისისნი დოკუმენტებისათვის განსაზღვრავენ შეფასების რიცხობრივ მაჩვენებლს-რელევანტობის ქულას, რომლის მიხედვითაც ხდება დაბრუნებულ შედეგში მათი განთავსება. ასეთი მოდელებია ვექტორული სივრცის მოდელი, ალბათური მოდელი.

ბულის ძებნის შედეგად ბრუნდება ორი ტიპის დოკუმენტების სიმრავლე: რელევანტური ან არარელევანტური. თუმცა რელურად შესაძლებელია მეტად რელევანტური ან ნაკლებად რელევანტური დოკუმენტების არსებობაც.

ბულის მოდელი უფრო მუშაობს მონაცემების ძებნაზე, ვიდრე ინფორმაციულ ძებნაზე.

ბულის მოდელი-,,ზუსტი დამთხვევა"

- მოთხოვნით ზუსტად განსაზღვრული კრიტერიუმები;
- თითოეული დოკუმენტი შეესაბამება ან არ შეესაბამება მოთხოვნას;
- შედეგი არის დოკუმენტების სიმრავლე

ვექტორული სივრცის, ალბათური მოდელი-,,საუკეთესო დამთხვევა"

- მოთხოვნის შედეგად ბრუნდება ,,კარგი" და ,,უფრო ზუსტი" დოკუმენტები;
- თითოეული დოკუმენტი შეესაბამება მოთხოვნას გარკვეული ხარისხით;
- შედეგი არის რანჟირებული დოკუმენტების სია.

ბულის მოდელში აღნიშნული პრობლემები გადაჭრილია ვექტორული სივრცის და ალბათური ძებნის მოდელებით.

2.2 ვექტორული სივრცის მოდელი

ვექტორული სივრცის მოდელში მოთხოვნა და დოკუმეტი წარმოდგენილია ვექტორების სახით. და ხდება ამ ორ ვექტორს შორის მსგავსების ზომის განსაზღვრა. მთავარი მიდგომა დოკუმენტის სტატისტიკური დამუშავებისა და ინდექსაციის არის ტექსტის წარმოდგენა ტერმინების სახით. ტექსტის ინდექსაცია ხდება ტერმინებით.

ვექტორული სივრცის მოდელი დაფუძნებულია იმ იდეაზე, რომ დოკუმენტი წარმოდგენს ტერმინების სიმრავლეს. მოთხოვნაში ხშირად მომხმარებლისათვის ზოგიერთი ტერმინი უფრო მეტად მნიშვნელოვანია, ვიდრე სხვა დანარჩენი. ამიტომ

საჭირო ხდება ტერმინების წონების განსაზღვრა, რომელიც დამოკიდებულია დოკუმენტში ტერმინის სიხშირეზე. ერთიდაიგივე ტერმინს შესაძლებელია ჰქონდეს სხვადასხვა წონა სხვადასხვა დოკუმენტში. ამგვარად ამ მოდელში ყველაზე მნიშვნელოვანი პარამეტრი არის ტერმინის წონა.

მოცემულ დოკუმენტში ტერმინების დადგენილი წონები შეიძლება განხილულ იქნეს, როგორც ამ დოკუმენტის კოორდინატები ,, დოკუმენტების სივრცეში“, ზოგჯერ სასურველია განვსაზღვროთ ტერმინების სივრცე, სადაც თითოეული ტერმინის კოორდინატი არის მისი შესაბამისი წონა.

ტერმინის სიხშირეზე დამოკიდებული წონა

$$\boldsymbol{tf - idf = tf_{td} \times itf_t}$$

- t= განსხვავებული ტერმინების რაოდენობა დოკუმენტში
- $tf_{ij} = t_j$ ტერმინის გამოჩენის სიხშირე D_i დოკუმენტში.
  - ტერმინის (keyword) სიხშირე დოკუმენტში

    <u>მაღალი სიხშირე-მეტი მნიშვნელოვნება(მაღალი წონა)</u>
- df_j = დოკუმენტის სიხშირე
  - დოკუმეტების რაოდენობა, რომლებიც შეიცავენ t_j ტერმინს.
- $idf_j = log\frac{d}{df_j}$ დოკუმენტების შებრუნებული სიხშირე (d დოკუმეტების საერთო რაოდენობა)

რაც უფრო ერთგვაროვანია ტერმინის განაწილება, მით ნაკლებად სპეციფიურია ის დოკუმეტისათვის.

დავუშვათ გვაქვს დოკუმენტების კოლექცია, რომელიც შედგება 3 დოკუმენტისაგან:

Q: “ gold silver truck”

D_1: “Shipmen of gold damaged in a fire”;

D_2: “Delivery of silver arrived in a silver truck”

D_3: “Shipment of gold arrived in truck”

კოლექციაში რადგან გვაქვს 3 დოკუმენტი, ამიტომ d=3. თუ ტერმინი გამოჩნდება მხოლოდ ერთ დოკუმენტში, მისი $idf = log\frac{3}{1} = 0.477$. ანალოგიურად , თუ ტერმინი გამოჩნდება სამიდან 2 დოკუმენტში, მაშინ მისი $idf = log\frac{3}{2} = 0.176$ და თუ ტერმინი გამოჩნდა სამივე დოკუმენტში, მაშინ მისი $idf = log\frac{3}{3} = 0$

მოცემულ მაგალითში დავთვალოთ idf

$idf_a = 0$

$idf_{arrived} = 0.176$

$idf_{damaged} = 0.477$

$idf_{delivery} = 0.477$

$idf_{afire} = 0.477$

$idf_{in} = 0$

$idf_{of} = 0$

$idf_{silver} = 0.477$

$idf_{ashipment} = 0.176$

$idf_{atruck} = 0.176$

წარმოვადგინოთ დოკუმენტის ვექტორი

docid	a	arrived	damaged	delivery	fire	gold	in	of	shipment	silver	truck
D_1	0	0	0.477	0	0.477	0.176	0	0	0.176	0	0
D_2	0	0.176	0	0.477	0	0	0	0	0	0.954	0.176
D_3	0	0.176	0	0	0	0.176	0	0	0.176	0	0.176

Q	0	0	0	0	0	0.176	0	0	0	0.477	0.176

მსგავსების კოეფიციენტი

$$SQ(Q, D_i) = \sum_{j=1}^{t} w_{qj} \times d_{ij}$$

$$SQ(Q, D_1) = (0)(0) + (0)(0) + (0)(0.477) + (0)(0) +$$

$$(0)(0.477) + (0.176)(0{,}176) + (0)(0) + (0)(0)$$

$$(0)(0.176) + (0.477)(0) + (0.176)(0) = 0{,}176^2 \approx 0{,}031$$

ანალოგიურად:

$$SQ(Q, D_2) = (0{,}954)(0{,}477) + 0{,}176^2 \approx 0{,}486$$

$$SQ(Q, D_2) = 0{,}176^2 + 0{,}176^2 \approx 0{,}062$$

დოკუმენტისა და მოთხოვნის მსგავსების შედეგებიდან გამომდინარე დოკუმენტების რანჟირებული სია ასეთი იქნება: D_2, D_3, D_1

თავი III. ბულის ძებნაზე დაფუძნებული მოდიფიცირებული მეთოდი კონცეპტების გამოყენებით

3.1 კონცეპტების გამოყენება ინფორმაციის ძიებაში

მოცემულია ობიექტების კლასი *C*, რომელიც შედგება არაიგივური სასრული რაოდენობა ობიექტებისაგან $\{C_1, C_2, \dots, C_n\}$. ყოველი ობიექტი C_i, i=1, … , n ხასიათდება ნიშანთვისებათა სასრული რაოდენობით $A = \{A_1, A_2, \dots, A_m\}$ და შეფასებით, თუ რომელ კლასს განეკუთვნება ეს ობიექტი $C^+ \subset C$ თუ $C^- \subset C$. ყოველ ნიშანთვისებას A_j, j=1,…,m შეუძლია მიიღოს მნიშნელობა $b_{jk} \subset B$, $k = 1, 2, \dots, n_j$. $A_1, A_2, \dots, A_m$ მნიშნელობათა დალაგებულ სიმრავლეს C_i, ობიექტის აღწერის შემთხვევაში ვუწოდოთ „ტრაექტორია“. C_i ობიექტი შეიძლება ჩავწეროთ „ტრაექტორიის“ სახით

$$C_i = \{b_1(i), b_2(i), \dots, b_m(i)\}, b_j(i) \in B, \; j = 1, 2, \dots, m$$

ქვეკლასებიდან ობიექტებზე დაკვირვების შედეგად სუბიექტმა უნდა ჩამოაყალიბოს ცნება, რომელიც შეესაბამება C^+ და C^- ქვეკლასებს. ანალიტიკური ევრისტიკების მეთოდი საშუალებას იძლევა შეფასებული ობიექტების საფუძველზე ავაგოთ პატერნი რომელიც შეესაბამება C^+ და C^-.

პატერნის აგების პროცესი შედგება შემდეგი ეტაპებისაგან:

I. **ნიშანთვისებათა ბინარიზაცია მათი „მნიშვნელობათა სივრცის“ ბინარიზაციით.** ბინარიზაციი ყველაზე ზოგად მეთოდად შეიძლება ავიღოთ სიმრავლის გაყოფა ორ ურთიერთშემავსებელ ქვესიმრავლედ „არის“ - „არ არის“. ამ აღნიშვნებში C_i ობიექტს შეიძლება გააჩნდეს A_k ან $\overline{A_k}$, $k = 1, 2, \dots, m$.

II. **ნიშანთვისებათა გადაკოდირება.**

შემოგვაქვს რიცხვითი სიმრავლე და ალ-სიმრავლე:

ნაცვლად ნიშანთვისებათა სიმრავლის გვექნება $A = \{A_1, A_2, \dots, A_m\} - N_A = \{1, 2, \dots, m\}$

ნაცვლად მნიშნელობათა სიმრავლისა $B = \{ b_{11}, b_{12}, \dots, b_{m\, n_m} \} - \underline{N}_B = \left\{ \overset{\vee}{1}, \overset{\vee}{2}, \dots, \overset{\vee}{n} \right\}$,

სადაც $\overset{\vee}{k} = \begin{cases} k \\ -, \\ k \end{cases} \quad k = 1, 2, \dots, n$;

„ტრაქტორიის“ ნაცვლად $C_i = \{ b_1(i), b_2(i), \dots, b_m(i) \} - \underline{N}_{C_i} = \left\{ \overset{\vee}{\alpha}_1(i), \overset{\vee}{\alpha}_2(i), \dots, \overset{\vee}{\alpha}_m(i) \right\}$,

სადაც $\overset{\vee}{\alpha}_j(i) \in \underline{N}_B$, $j = 1, 2, \dots, m$.

III. **ორთონორმირებული ბინარული მდგომარეობის ვექტორების აგება.**

შემოდის V მატრიცა განზომილებით $n \times m$ ($2n = 2^m$). ამ მატრიცის სვეტებს წარმოადგენენ მდგომარეობის ორთონორმირებული ვექტორები (ანუ ფილტერბ) ψ_i , $i = 1, 2, \dots, m$, რომელიც იქმნება $\underline{N}_B$ ელემენტებით (ცხრ.1).

IV. **ფილტრაციის ოპერაცია** - ყველა C_i ტრაექტორიის ორთონორმირებულ ფილტრებზე გატარება

ყოველ ტრაექტორიას $C_i = \left\{ \overset{\vee}{\alpha}_1(i), \overset{\vee}{\alpha}_2(i), \dots, \overset{\vee}{\alpha}_m(i) \right\}$ უთანადდება მდგომარეობათა ორთონორმირებულ ვექტორთა კონიუქციური ნამრავლი

$$\varphi(C_i) = \left(\overset{\vee}{\psi}_1 \overset{\vee}{\psi}_2, \dots, \overset{\vee}{\psi}_m \right)_i \quad , i = 1, 2, \dots, n$$

სადაც $\overset{\vee}{\psi}_j = \psi_j$, თუ ტრაექტორიის j-ური ელემენტი შედის ψ_j ვექტორში „არის“ სახით, ანუ e_j , $j = 1, 2, \dots, m$

და $\overset{\vee}{\psi}_j = \overline{\psi}_j$, თუ ტრაექტორიის j-ური ელემენტი შედის ψ_j ვექტორში „არ არის“ სახით, ანუ $\overline{e}_j$, $j = 1, 2, \dots, m$.

V. **დიზიუნქციური სუპერპოზიციის ოპერაცია.**

$$\varphi_+ = \bigcup_{C_i \in C^+} \varphi(C_i) - C^+ \text{ კონცეპტის პატერნი} \qquad (1)$$

$$\varphi_- = \bigcup_{C_i \in C^-} \varphi(C_i) - C^- \text{ კონცეპტის პატერნი} \qquad (2)$$

თუ ობიექტების რაოდენობა C^+ და C^- ქვეკლასებთან მიმართებაში საკმაოდ დიდია, ხოლო თვით ობიექტები არაიგივურია და საკმაოდ ფართოდ წარმოადგენენ შესაბამის ქვეკლასს; თუ სწორადაა გამოყოფილი საკმარისი რაოდენობის ნიშანთვისებები, ზუსტადაა განსაზღვრული მათი მნიშვნელობათა სიმრავლე და ჩატარებულია ამ სიმრავლეთა

„წარმატებული“ ბინარიზაცია, მაშინ პატერნები φ_+ და φ_- შეიცავენ სრულ ინფორმაციას C^+ და C^- შესახებ (როგორც ტიპიურის, ასევე იშვიათი წარმომადგენლების ქვეკლასიდან) და არ მოდიან ერთმანეთთან წინააღმდეგობაში.

დიდ n და m შემთხვევაში შეუძლებელია გამოისახოს პატერნი მკვეთრი ლოგიკური ფორმულირებით. ამიტომ საჭიროა შემდგომი ეტაპი - პატერნის გამარტივება.

VI. **ბულის ცვლადებზე პირობითი გადასვლის ოპერაცია.**

თუ გამოსახულებებში (1) და (2) ყოველ ψ_i ვექტორს შევცვლით x_i-ზე , ხოლო $\overline{\psi}_i - \overline{x}_i$-ზე, მათინ φ_+ და φ_- ფუნქციონალები მიიღებენ სრულ დიზიუნქციურ ნორმალურ ფორმაც [10, 36]:

$$\varphi_+ = \bigvee_{I_+(\sigma_1 \sigma_2 \dots \sigma_m)} x_1^{\sigma_1} x_2^{\sigma_2} \dots x_m^{\sigma_m}$$

$$\varphi_- = \bigvee_{I_-(\sigma_1 \sigma_2 \dots \sigma_m)} x_1^{\sigma_1} x_2^{\sigma_2} \dots x_m^{\sigma_m}$$

სადაც $\sigma_i = \begin{cases} 1 & if \quad x_i \\ 0 & if \quad \overline{x}_i \end{cases} \quad i = 1, 2, \dots, m$,

$I_+(\sigma_1 \sigma_2 \dots \sigma_m)$ – ნაკრებების სიმრავლეა , რომელიც შეესაბამება C^+ ქვეკლასის ტრაექტორიებს;

$I_-(\sigma_1 \sigma_2 \dots \sigma_m)$ – ნაკრებების სიმრავლეა , რომელიც შეესაბამება C^- ქვეკლასის ტრაექტორიებს.

ამ ნორმალური დიზიუნქციური ფორმების მინიმიზაციით მივიღებთ პატერნის ბინარულ ფორმას [10, 36]:

$$K_+ = f^+\left(\xi_1^{\sigma_1}, \xi_2^{\sigma_2} \dots \xi_l^{\sigma_l}\right) = \vee \xi_1^{\sigma_1} \xi_2^{\sigma_2} \dots \xi_l^{\sigma_l} ,$$

სადაც $l < m$ და $\xi_1^{\sigma_1}, \xi_2^{\sigma_2}, \dots, \xi_l^{\sigma_l}$-ის საშუალებით გადანიშნულია ის $x_i^{\sigma_i}, x_j^{\sigma_j}, \dots, x_k^{\sigma_k}$ ცვლადები, რომლებიც დარჩნენ φ_+ სრული დიზიუნქციური ნორმალური ფორმის მინიმიზაციის შემდეგ (ანალოგიური ფორმა მიიღება φ_- -თვის). დანარჩენი $\xi_{l+1}, \dots, \xi_m$ ცვლადები არამნიშვნელოვანნი არიან, იმ აზრით, რომ მათი მნიშნელობა არ ახდენს გავლენას ობიექტის შეფასების შედეგზე.

პატერნის ბინარული ფორმა K_+ შეიცავდ მნიშვნელოვანი ნიშანთვისებების მნიშნელოვან მნიშვნელობებს, და ასახავს იმ განსაკუთრებულობას, რომელიც დამახასიათებელია C^+ ქვეკლასის სასრული ნაკრების ობიექტებისათვის. საკმაოდ დიდი n

და m -თვის პატერნი K_+ შეკუმშული ფორმით შეიცავს იმ წესებს (ზოგად შემთხვევაშ ცოდნას), რომლითაც ხელმძღვანელობდა შემფასებელი ობიექტების სიმრავლის C^+ და C^- ქვეკლასებად დაყოფისას. შესაბამისად ბინარული K_+ პატერნის სსაშულაბეით შეიძლება შეფასდეს ახალი ობიექტები, რომლებიც არ მონაწილეობდნენ პატერნის ფორმირებაში: ახალი ობიექტის C^+-ზე მიკუთვნებისათვის საკმარისის, რომ ახალი ტრაექტორიის $\xi_1, \dots, \xi_l$ ცვლადებმა მიიღონ ზუსტად ის მნიშვნელობები, რომლებიც დაფიქსირებულია K_+ პატერნის თუნდაც ერთ იმპლიკანტში, $\xi_{l+1}, \dots, \xi_m$ ცვლადების მნიშვნელობები ნებისმიერია. შევნიშნოთ, რომ ბინარული პატერნი ადვილად წარმოდგება პროდუქციული წესის სახით.

3.2 მეთოდის აღწერა

იმისათვის, რომ შევძლოთ ანალიტიკური ევრისტიკების მეთოდის გამოყენება [4], უპირველესად უნდა მოვახდინოთ ამ მეთოდში გამოყენებული ძირითადი ელემნტებისა და ცნებების განმარტება.

ნიშან თვისებათა სიმრავლე - ამ სიმრავლეს შეუსაბამოთ ამა თუ იმ ენის სიტყვათა სიმრავლე $\mathcal{A}$. ყოველი ასეთი სიმრავლე შესაძლებელია თავის მხრივ წარმოვადგინოთ რვა-ათი A_i მტყველების ნაწილების შესაბამის სიტყვების ქვესიმრავლის გაერთიანებად (1. არსებითი სახელი, 2. ზმნა, 3. ზედსართავი სახელი, 4. რიცხვითი სახელი, 4. ნაცვალსახელი, 6. ზმნიზედა, 7. კავშირი, 8. წინდებული (ან თანდებული), 9. ნაწილაკი, 10. შორისდებული) [lexical class (noun, verb, adjective, adverb, pronoun, preposition, conjunction, and interjection]. ქვესიმრავლეთა რაოდენობა დამოკიდებულია კონკრეტული ენის სპეციფიკაზე (მაგ. ქართული ენისთვის - ათი). ყოველი $A_i, (i = \overline{1,10})$, არის $a_{i,j}, j = \overline{1,N_i}$ ელემენტებისაგან შემდგარი სიმრავლე, სადაც N_i არის სიტყვათა რაოდენობა კონკრეტულ A_i მეტყველების ნაწილში.

კონკრეტული C „ცნების“ ერთი რომელიმე „აღმწერი x ტექსტი“ წარმოვადგინოთ $T_C^x = \{w_{i,j}\}, \ i = \overline{1,10} \ , j = \overline{1,M}$, სიმრავლის სახით , სადაც $w_{i,j}$ არის ყველა განსხვავაებული სიტყვა ამ ტექსტში, ხოლო M არის განსხვავაბული სიტყვების რაოდენობა ამ ტექსტში. რა თქმა უნდა არსებობს კონკრეტული C „ცნების“ აღმწერი სხვა ტექსტიც C^y. ყველა ასეთი

სიმრავლე წარმოადგენს საწყისი $\mathcal{A}$ სიმრავლის ქვესიმრავლეს. ჩვენ შეგვიძლია ეს სიმრავლე გავაფართოო ალ-სიმრავლემდე. ასეთ შემთხვევაში შევძლებთ სრულად გამოვიყენოთ ანალიტიკური ევრისტიკების მეთოდი.

როგორც აღვნიშნეთ, Explicit Semantic Analysis (ESA) მეთოდი მონაცემების წყაროდ იყენებს ვიკიპედიის საცავს [9]. ძირითადად გამოიყენება ამა თუ იმ ცნების (აღვნითშოთ ეს ცნება, როგორც C^{wik} განმმარტავი ტექსტი (როგორც წესი ერთ ცნებას შეესაბამება ერთი ტექსტი; აღვნითშოთ ეს ტექსტი, როგორც T_c^{wik}). ყოველი ასეთი ტექსტი წარმოდგება წონითი ნაკრებების სახით, სახელდობრ ეგრეთწოდებული TF-IDF scheme [10] საშუალებით. სემანტიკური გადამყვანი ახდენს ტექსტის სიტყვების იტერაციას, იღებს ინვერტირებული ინდექსებისაგან შემდგარ შესაბამის ჩანაწერს და აერთიანებს მას ცნების ვექტორში, რომელიც აღწერს ტექსტს.

ნაშრომი [9]-ის შესაბამისად წარმოვიდგინოთ C კონცეპტის აღმწერი ტექსტი w_i სიტყვების სიმრავლის სახით $T_c^{wik} = \{w_i\}$, $i=1,\ldots,M^{wik}$(M^{wik} არის ვიკიპედიის საცავში ამ ცნების შესაბამის ტექსტში შემავალი სიტყვების რაოდენობა), რომელსაც შეესაბამება TF-IDF$\langle v_i^{wik}\rangle$ ვექტორი, სადაც ყოველ w_i სიტყვას შეესაბამება v_i^{wik} წონა.

სიტყვის (ტერმინის) წონა დამოკიდებულია მის სიხშირეზე.

ტერმინის წონების დასადგენად წარმატებით გამოიყენება წონების ავტომატური გენერაციის სქემა, რომელიც შემდეგნაირად გამოისახება:

$$\text{tf- idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

ასევე იქმნება $\langle k_i^{wik}\rangle$ ვექტორი, რომელშიც k_i^{wik} წარმოადგენს w_i სიტყვის ინვერტირებულ ინდექსს C კონცეპტის შესაბამისი ტექსტისათვის ვიკიპედიის საცავიდან. საცავში არსებული C კონცეპტის შესაბამისი T_C^{wik} ტექსტისათვის გვექნება V_C^{wik} წონების ვექტორი რომელიც განისაზღვრება როგორც:

$$\sum_{w_i \in T_C^{wik}} v_i^{wik} \cdot k_i^{wik}$$

აღვწეროთ მიღებული C კონცეპტი ანალიტიკური ევრისტიკების მეთოდისათვის [11] მისაღებ ფორმაში. ჩავთვალოთ, რომ ყოველი C ზოგადად აღიწერება $w_1^{wik}, w_2^{wik}, \dots, w_N^{wik}$ "სიტყვებით". აღწერაში მონაწილე სიტყვების რაოდენობა მკვლევარის (კონცეპტის შემმუშავებლის) შეხადულებაზეა დამოკიდებული. ამ რაოდენობის განსასაზღვრად შესაძლებელია გამოყენებულ იქნას სხვადასხვა მიდგომა, რომელიც უმთავრესად დამოკიდებული იქნება ტექსტის მოცულობაზე, შინაარსობრივად განსხვავებული სიტყვების რაოდენობაზე და სხვა. ყოველი w_i^{wik} სიტყვის მოხვედრა აღწერაში განისზღვრება C კონცეპტის შესაბამისი T_j^{wik} ტექსტის V_j^{wik} წონების ვექტორით. სიტყვის მოხვედრას C კონცეპტის აღწერაში წონის გარდა განსაზღვრავს, ის თუ მეტყველების, რომელ ნაწილს განეკუთვნება ეს სიტყვა (ანუ $\mathcal{A}$ სიმრავლის რომელ A_i ქვესიმრავლის ელემენტია). როგორც წესი მეტყველების ნაწილის კავშირის შესაბამისი ქვესიმრავლის ელემენტები ასეთ აღწერაში თითქმის არ იღებენ მონაწილეობას. მკვლევართა უმეტესი ნაწილი უპირატესობას ანიჭებს არსებით სახელებს, ან სიტყვათა წყვილს „ზედსართავი სახელი"+„არსებითი სახელი", „არსებითი სახელი"+"ზმნა". სიტყვათა წყვილის გამოსაყოფად მხოლოდ ცალკეული სიტყვების წობენის ვექტორი საკმარისი არაა. ეს შემთხვევა სხვა ნაშრომში განიხილება.

გავაფართოოთ კონცეპტების შესამუშავებელი სივრცე და გამოვიყენოთ ვიკიპედიის მსგავსი სხვა საცავებიც (AllRefer.com, bartleby.com, Britannica.com, infoplease.com, Encyclopedia.com, techweb.com/encyclopedia, libraryspot.com/encyclopedias.htm#science, education.yahoo.com/reference/encyclopedia). ყველა ამ საცავშიც არსებობს C კონცეპტის შესაბამისი T_C^x ტექსტი. შესაბამისად ამ საცავისთვისაც შეგვიძლია გამოვიყენოთ ESA მეთოდი და შემდგომ აღვწერთ $w_1^x, w_2^x, \dots, w_L^x$"სიტყვებით" ($L$ არის ამ საცავში შემავალი სიტყვების რაოდენობა). თუ ამ პროცედურას ჩავატარებთ ჩვენს ხელთ არსებული ყველა საცავისათვის, მივიღებთ ერთი C კონცეპტის შესაბამის რამდენიმე, შესაძლოა განსხვავებულ, აღეწრას. C კონცეპტის ყოველი აღწერის შეესაბამისი T_C^x ტექსტისათვის გვექნება V_C^x წონების ვექტორი.

ცხრილი 1.

საცავები	c_j კონცეპტის აღწერა
Wikipedia	$w_1^{wik}, w_2^{wik}, \dots, w_N^{wik}$
x	$w_1^x, w_2^x, \dots, w_L^x$
…	…
y	$w_1^y, w_2^y, \dots, w_K^y$

ცხადია, რომ ყოველი C კონცეპტის აღმწერ სხვადასხვა „ვექტორში“ შემავალი სიტყვები მეორდება. გავაერთიანოთ ეს სიტყვები და მივიღოთ საცავების სიტყვების საერთო სიმრავლე $W = \{w_1, w_2, \dots, w_{max}\}$, max - არის ყველა საცავში არსებული მაქსიმალური განსხვავებული სიტყვის რაოდენობა. ჩავთვალოთ, რომ ეს რიცხვია N. ამ აღნიშვნებში ჩვენ შეგვიძლია C კონცეპტის აღმწერი ყველა ვექტორი წარმოვადგინოთ ერთი და იგივე N სიგრძის ვექტორის სახით, რომლის ელემენტებია $\widetilde{w_i}$, *i=1, …N*

$$\widetilde{w_i} = \begin{cases} w_i & \text{მონაწილეობს } C \text{ კონცეპტის აღწერაში;} \\ \overline{w}_i & \text{არ მონაწილეობს } C \text{ კონცეპტის აღწერაში.} \end{cases}$$

შესაბამისად ცხრილი 1, მიიღებს უნიფიცირებულ სახეს:

ცხრილი 2

საცავი	$\mathbf{w_1}$	$\mathbf{w_2}$	**…**	$\mathbf{w_i}$	**…**	$\boldsymbol{w_N}$
R^1	$\widetilde{w}_{1,1}$	$\widetilde{w}_{2,1}$	…	$\widetilde{w}_{i,1}$	…	$\widetilde{w}_{N,1}$
R^2	$\widetilde{w}_{1,2}$	$\widetilde{w}_{2,2}$	…	$\widetilde{w}_{i,2}$	…	$\widetilde{w}_{N,2}$

…	…	…	…	…	…	…
R^k	$\breve{w}_{1,k}$	$\breve{w}_{2,k}$	…	$\breve{w}_{i,k}$	…	$\breve{w}_{N,k}$
…	…	…	…	…	…	…
R^m	$\breve{w}_{1,m}$	$\breve{w}_{2,m}$	…	$\breve{w}_{i,m}$	…	$\breve{w}_{N,m}$

სადაც

$$\breve{w_{i,k}} = \begin{cases} w_i \text{ მონაწილეობს } C \text{ კონცეპტის აღწერაში} R^k \text{ სივრცეში;} \\ \bar{w}_i \text{ არ მონაწილეობს } C \text{ კონცეპტის აღწერაში} R^k \text{ სივრცეში.} \end{cases}$$

რადგან ყოველი C კონცეპტის აღწერა სასრულია, სასრულია $W = \{w_1, w_2, \dots, w_N\}$ სიმრავლეც, ჩვენ შეზღუდვის გარეშე შეგვიძლია ეს სივრცე ალ-სიმრავლედ წარმოვიდგინოთ [12], და მასში განვმარტოთ ყველა ის ოპერაციები, რაც განმარტებულია ასეთი ტიპის სიმრავლეებში. თუ დავუბრუნდებით ანალიტიკური ევრისტიკების მეთოდის აღნიშვნებს და განმარტებებს სრულად, ჩვენ შეგვიძლია ვთქვათ, რომ C კონცეპტი იგივეა,

რაც ობიექტი და ყოველი w_i ამ ობიექტის აღმწერი ნიშანთვისებების A_i-ის შესაბამისია. ეს საშუალებას გვაძლევს სრულად გამოვიყენოთ ეს მოთოდი. აქედან გამომდინარე C-ს ყოველი რეალიზაცია წარმოდგება ჩვეულებრივი იმპლიკანტის სახით, და კონცეპტის პატერნის მისაღებად მოვახდენთ ნორმალური დიზიუნქციური ფორმის მინიმიზაციას.

თავი IV. საძიებო ძრავის ალგორითმის რეალიზაცია

პირობითად წარმოვადგინოთ რაღაც C კონცეპტის სხვადასხვა აღწერები. დავუშვათ გვაქვს 5 სხვადასხვა აღწერა, რომელშიც მონაწილეობას ღებულობს 4 განსხვავებული სიტყვა:

საცავი	C კონცეპტის *იმპლიკანტი*

R^1	$w_1 \& w_2 \& \overline{w}_3 \& w_4$
R^2	$w_1 \& \overline{w}_2 \& \overline{w}_3 \& w_4$
R^3	$w_1 \& w_2 \& \overline{w}_3 \& w_4$
R^4	$w_1 \& w_2 \& w_3 \& w_4$
R^5	$\overline{w}_1 \& w_2 \& w_3 \& w_4$

ჩავწეროთ ეს რეალიზაციები დიზიუნქციური ნორმალური ფორმის სახით:

$$(w_1 \& w_2 \& \overline{w}_3 \& w_4) \vee (w_1 \& \overline{w}_2 \& \overline{w}_3 \& w_4) \vee (w_1 \& w_2 \& \overline{w}_3 \& w_4) \vee (w_1 \& \overline{w}_2 \& \overline{w}_3 \& w_4)$$
$$\vee (w_1 \& w_2 \& \overline{w}_3 \& w_4)$$

ალგორითმმა უნდა მოახდინოს მიღებული დიზიუნქციური ფრომის მინიმიზაცია.ამ შემთხვევაში მივიღებთ c კონცეპტის აღწერის განზოგადოებულ სახეს, რომელიც ყველა საცავის ტექსტებს ეფუძნება.

$$w_1 \& \overline{w}_3 \& w_4$$

რა თქმა უნდა რეალურად კონცეპტის აღწერაში ბევრად უფრო დიდი რაოდენობის სიტყვები იღებენ მონაწილეობას, მაგრამ ჩვენ შეგვიძლია შევარჩიოთ ყველაზე მაღალი წონის შესაბამისი სიტყვები Explicit Semantic Analysis (ESA) მეთოდით მიღებული ვექტორზე დაყრდნობით. რაოდენობის განსაზღვრა შეიძლება დამოკიდებული იყოს ტექსტში სიტყვების რაოდენობის და განსხვავებული სიტყვების რაოდენობის თანაფარდობაზე. ასეთი მიდგომა გააძლიერებს საბოლოოდ მიღებული კონცეპტის აღწერის სემანტიკურ მნიშვნელობას. რაც უფრო მეტი სიტყვა მიიღებს მონაწილეობას აღწერაში, მით უფრო სემანტიკიურად ადექვატური იქნება შედეგი. მეორეს მხრივ სიტყვების დიდმა რაოდენობამ შესაძლოა გაართულოს კონცეპტის გამოყენება ინფორმაციის ძებნისათვის.

ზემოთ აღწერილი ალგორითმის ერთერთ ძირითად ნაწილს წარმოადგენს დიზიუნქციური ნორმალური ფორმის სახით წარმოდგენილი იმპლიკანტების

მინიმიზაცია. ამ პროცესის წარმატებულობაზეა დამოკიდებული ამათუ იმ ცნების შესაბამისი კონცეპტ პატერნის სისრულე და ადეკვატურობა. ამიტომ არსებული ალგორითმების კომბინირების საფუძველზე შეიქმნა პრპოგრამული აპლიკაცია, რომელიც ახდენს შერჩეული ტექსტებიდან ამოკრეფილი იმპლიკანტების (კონიუქციების) დიზიუნქციის ფუნქციით შეერთებული ფორმის მინიმიზაციას.

აპლიკაციის რეალიზაციისათვის პროგრამულ ინსტრუმენტად გამოყენებულია დაპროგრამების ენა JavaScript და Html-ის ისტრუმენტები.

ამ ეტაპზე პრორამული აპლიკაცია წარმოდგენილია განზოგადოებული სახით, მისი მიბმის შედეგად კონკრეტულ ენის კორპუსთან ან სხვა რაიმე ტექსტების საცავთან შესაძლებელია მოხდეს მორგება, წაყენებული მოთხოვნების შესაბამისად.

დიზიუნქციური ნორმალური ფორმის მინიმიზაცია

შემთხვევითი მაგალითი

შემავალი სიტყვების(ცვლადების) რაოდენობა: 3

სისწორის ცხრილი:

	w_2	w_1	w_0	y
0:	0	0	0	0
1:	0	0	1	0
2:	0	1	0	0
3:	0	1	1	0
4:	1	0	0	0
5:	1	0	1	0
6:	1	1	0	0
7:	1	1	1	0

მინიმალური ლოგიკური გამოხატვა:

$y = 0$

აპლიკაციის საწყისი გაშვებისას მომხმარებელს შეუძლია შეარჩიოს იმპლიკანტში კომპონენტების რაოდენობა (ეს რაოდენობა თანადია ტექსტის ძებნისთვის გამოყოფილი მაღალწონიანი სიტყვების რაოდენობის).

შემდგომ ბიჯზე მომხმარებელს შეუძლია მოახდინოს ავტომატური გენერირება კომბინაციების ან ჩასვას მისთვის სასურველი ტრაექტორიები.

დიზიუნქციური ნორმალური ფორმის მინიმიზაცია

შემთხვევითი მაგალითი

შემავალი სიტყვების(ცვლადების) რაოდენობა: 5

სისწორის ცხრილი:

	w_4	w_3	w_2	w_1	w_0	y
0:	0	0	0	0	0	1
1:	0	0	0	0	1	0
2:	0	0	0	1	0	0
3:	0	0	0	1	1	1
4:	0	0	1	0	0	1
5:	0	0	1	0	1	1
6:	0	0	1	1	0	0
7:	0	0	1	1	1	0
8:	0	1	0	0	0	0
9:	0	1	0	0	1	0
10:	0	1	0	1	0	0

იმპლიკანტები (დალაგება 0):

	w_4	w_3	w_2	w_1	w_0	
0:	0	0	0	0	0	→
3:	0	0	0	1	1	→
4:	0	0	1	0	0	→
5:	0	0	1	0	1	→
11:	0	1	0	1	1	→
12:	0	1	1	0	0	→
14:	0	1	1	1	0	→
15:	0	1	1	1	1	→
18:	1	0	0	1	0	→
19:	1	0	0	1	1	→
20:	1	0	1	0	0	→

იმპლიკანტები (დალაგება 1):

	w_4	w_3	w_2	w_1	w_0	
0, 4:	0	0	-	0	0	✓
3, 11:	0	-	0	1	1	→
3, 19:	-	0	0	1	1	→
4, 5:	0	0	1	0	-	✓
4, 12:	0	-	1	0	0	✓
4, 20:	-	0	1	0	0	✓
11, 15:	0	1	-	1	1	→
11, 27:	-	1	0	1	1	→
12, 14:	0	1	1	-	0	✓
14, 15:	0	1	1	1	-	✓
15, 31:	-	1	1	1	1	→

შედეგად მომხმარებელი ღებულობს საწყისი ტრაექტორიის მინიმალურ ნორმალურ ფორმაც ორი სახის წარმოდგენით - გრაფილული და ანალიტიკური სახით.

შემცირებული გამარტივებული იმპლიკანტების გრაფიკი (იტერაცია 0):

	w_4	w_3	w_2	w_1	w_0	12	14	18	20	22	
12, 14:	0	1	1	-	0	●	●				$(\bar{w}_4 w_3 w_2 \bar{w}_0)$
18, 22:	1	0	-	1	0			●		○	$(w_4 \bar{w}_3 w_1 \bar{w}_0)$
20, 22:	1	0	1	-	0				●	○	$(w_4 \bar{w}_3 w_2 \bar{w}_0)$

მინიმალური ლოგიკური გამოხატვა:

$y = (\bar{w}_4 \bar{w}_3 \bar{w}_1 \bar{w}_0) \vee (\bar{w}_2 w_1 w_0) \vee (\bar{w}_4 \bar{w}_3 w_2 \bar{w}_1) \vee (w_4 w_3 \bar{w}_2 \bar{w}_0) \vee (w_3 w_1 w_0) \vee (\bar{w}_4 w_3 w_2 \bar{w}_0) \vee (w_4 \bar{w}_3 w_1 \bar{w}_0) \vee (w_4 \bar{w}_3 w_2 \bar{w}_0)$

დასკვნა

ნაშრომში დასმულიძირითადი ამოცანის - მონაცემთა ბაზაში საძიებო სისტემის მოდელის რეალიზაციისათვის, შერჩეულ იქნა ერთერთი ყველაზე აქტუალური საკითხი სემანტიკური ძებნა. ამ საკითხის შესწავლამ აჩვენა, რომ თანამედროვე საძიებო სისტემები (უმთავრესად ვებ ორიენტირებული) უმნიშვნელოვანესად მიიჩნევენ სემანტიკური ძებნის სრულყოფას ყველა პარამეტრების მიხედვით (სისწრაფე, შინაარსობრივი სიზუსტე და ა.შ.).

ჩვეულებრივ მონაცემთა ბაზებში (რომელთაგან უმთავრესი რელაციური ტიპისაა), ძებნა ხორციელდება სტანდარტული ალგორითმებით. ძებნა ხორციელდება ეგრეთ წოდებული ატრიბუტების მნიშვნელობების მიხედვით და ნაკლებად გამოიყენება ისეთი საძიებო ძრავები, რომლებიც ძებნას განახორციელებენ ტექსტური ტიპის ჩანაწერების შიგთავსისი მიხედვით. ნაშრომში აქცენტი გაკეთებულია ასეთი ტიპის საძიებო ძრავის მოდელზე.

რელიზაციისათვის შეირჩა თსუ-ს მკვლევართა ჯგუფის მიერ შემუშავებული სემანტიკური ძებნის ალგორითმი. ალგორითმი რეალიზებულია პროგრამული აპლიკაციის სახით, რომლის გამოყენება შესაძლებელია ისეთ მონაცემთა ბაზაში სადაც განთავსებულია ძირითადად ტექსტური ინფორმაცია. პროგრამა რეალიზებულია დაპროგრამების ენა JavaScript და Html-ის ისტრუმენტების გამოყენებით.

ლიტერატურა

1. Chavchanidze, V. (1974) , "Towards the General Theory of Conceptual Systems: (A New Point of View)", Kybernetes, Vol. 3 Iss: 1, pp.17 – 25.
2. Chavhcanidze, V. (1970), “Heuristic Analysis of Artificial Intelligence in the Formation of Concepts, Pattern Recognition and Classification of Objects”, p.20 Institute of Cybernetes, Georgian Academy of Sciences,dep. 2080-70, Tbilisi.
3. Khachidze M. (1998), “Artinformatic Knowledge and Some Ways of Its Presentation”.Bulletin of Georgian Academy of Scienes, vol. 165, no. 6, pp. 60-65.
4. Kvinikhidze K.S., Chavchanidze V.V. (1976). “Applocation of Conceptual Approach to Describe the Evolution of Protein Structure”.Reporp of the 8-th International Congress on Cybernetics, Namur, French.
5. Mikeladze M., Khachidze M. (2000), “Modified Conceptual-Probabilistic Method of Formation Rules for Medical Diagnostic Expert Systems”, Proceedings of the XIV International Symposium “Large System Control”, Tbilisi, 2000, pp. 162-163.
6. Khachidze M., Archuadze M., Besiashvili M., The Method of Concept Formation for Semantic Search. 7th International Conference on APPLICATION of INFORMATION and COMMUNICATION TECHNOLOGIES, 23-25 October 2013, Baku, Azerbaijan.
7. Salton G., Buckley C. (1988), “Term-weighting approaches in automatic text re trieval”, Information Processing and Management, vol. 24 (5),pp. 513–523
8. Davis R. And Lenat D. (1982), Knowledge-Based Systems in Artificial Intelligence. McGraw-Hill Advanced Computer Science Series.
9. Egozi O., Markovitch S. and GabrilovichE. (2011), “Concept-Based Information Retrieval using Explicit Semantic Analysis”, ACM Transactions on Information Systems, Vol. 29, No. 2, Article 8, Publication date: April 2011.
10. Gabrilovich, E. and Markovitch, S. (2007), “Computing Semantic Relatedness Using Wikipedia-Based Explicit Semantic Analysis. In 20th International Joint Conference on Artificial Intelligence (IJCAI’07) procedings of international conference in Hyderabad, India, January 6-12, 2007, Morgan Kaufmann Publishers, pp. 1606–1611.
11. Budanitsky A. and Hirst G. (2006), “Evaluating Wordnet-Based Measures of Lexical Semantic Relatedness”, Computational Linguistics,Vol. 32 Issue 1, pp. 13-47.
12. Deerwester S., Dumais S., Furnas G., Landauer T. and Harshman R (1990), “Indexing by latent Semantic Analysis”, Journal of the American Society for Information Science, Vol. 41 Num 6, pp. 391–407.

13. Strube M., Ponzetto S. P. (2006), “Wikirelate! Computing semantic relatedness using Wikipedia”, Proceedings of the 21st National Conference on Artificial Intelligence, Vol. 2, AAAI Press, Boston, MA, pp.1419-1424.

14. Чавчанидзе В.В. К началам теории принятия концептуальных решений в системе искусственного интеллекта. Сообщения АН ГССР, т.70, №2, 1973.

15. Чавчанидзе В.В. К проблеме распознавания образов и об универсальной природе концептуальной интеллектуальной активности. Материалы коллоквиума по "концептуальному системному анализу естественных и искусственных систем". (Медицина, наука, техника), Батуми, 24-28 апреля, 1973.

16. Manning D., Raghavan P., Schütze H. (2008), ” Boolean Retrieval”, in Introduction to Information Retrieval,Cambridge University Press, pp. 1-18.