თბილისის ივ. ჯავახიშვილის სახელობის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

ბესიკ სახიაშვილი

სამაგისტრო ნაშრომი:

გარე ძებნის ალგორითმები დიდ მონაცემებზე

(Search Engine)

ნაშრომი შესრულებულია

კომპიუტერული მეცნიერების მაგისტრის

აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელები:

პროფესორი: მანანა ხაჩიძე

დოქტორანტი: კონსტანტინე მურადოვი

თბილისი 2015 წელი

The computer revolution is a revolution in the way we think and in the way we express what we think. The essence of this change is the emergence of what might best be called procedural epistemology--the study of the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects.

H. Abelson and G. Sussman (in "The Structure and Interpretation of Computer Programs)

ანოტაცია

21–ე საუკუნეში ტექნოლოგიების როლი სულ უფრო და უფრო იზრდება, თითქმის შეუძლებელია მოიძებნოს სფერო, რომელშიც მზირე დოზით მაინც არ ხდება კომპიუტერული მოწყობილობების გამოყენება, კომპიუტერები ცდილობენ ადამიანების შრომა თავიანთ თავზე აიღონ და უფრო და უფრო ღრმად შეიჭრან ადამიანურ ყოველდღიურობაში.

ტექნოლოგიების ასეთი სწრაფი ტემპით განვითარებას თან ახლავს გარკვეული პრობლემებიც, მათ განვითარებასთან ერთად იზრდება ვირტუალური მონაცემების მოცულობაც და შესაბამისად იქმნება მათი მოძიების თუ დახარისხების პრობლემა, რადგან ნებისმიერი ინფორმაცია ნორმალიზაციის გარეშე არანაირ ფასეულობას არ წარმოადგენს. სწორედ ამიტომ, მსოფლიოში არსებული გიგანტი კომპანიები ყოველდღიურად ცდილობენ შეიმუშაონ ისეთი ალგორითმები, რომლებიც საშუალებას მოგვცემს დროის მოკლე მონაკვეთში ჩვენთვის საჭირო ინფორმაციის მოძიებასა თუ დახარისხებას.

მოცემული ნაშრომის მიზანიც სწორედ ამ პრობლემას შეეხება, შევეცდებით ნათლად წარმოვაჩინოთ არსებული პრობლემები და შემოგთავაზოთ მათი გადაჭრის გზები და მათი უპირატესობები არსებულ მეთოდებსა თუ ალგორითმებთან მიმართებაში. ასევე მათ პრაქტიკულ რეალიზებას ძებნის ძრავში.

Abstract

In 21st century, the role of technology is increasingly growing. It's almost impossible to find some area where the computer is not get involved and used. Computers try to shoulder human's work and more and more deeply intrude into daily life.

The fast development of technology is accompanied by certain problems, with the development of them the volume of virtual data is getting large and therefore the problem of sorting and searching in big data is caused, because not any information does not have any cost without normalization. So, because of this the giant companies in the world try to develop such algorithms, which let us to find and sort the information in the short period of time.

The main goal of this work is exactly to describe this problem, we will try to promote the actual problems and propose some solutions which can be used to develop the better approaches in this sphere. Also we will develop them practically in building of Search Engine.

ს ა რ ჩ ე ვ ი

შესავალი..10

ინფორმაციის მოძიება...11

ვებ გვერდებიდან ინფორმაციის დასწავლა 12

დასწავლის პროცესის ინიციალიზაცია (Seed პროცესი) 12

ინფორმაციის დასწავლის პროცესში არსებული პრობლემები 14

შეზღუდული მოცულობის პრობლემა 14

გვერდებს შორის ციკლების არსებობის პრობლემა 15

გვერდებს შორის ალიასების არსებობის პრობლემა..15

დასწავლის პროცესის პერიოდულობა 16

დასწავლის პროცესის შედეგი 17

შესწავლილი ინფორმაციის ანალიზი და ინდექსაცია 18

ინფორმაციის ანალიზი...18

ჰეშირების ფუნქციის შერჩევა...19

ჰეშირების ფუნქციის გაუმჯობესება..22

ინდექსირების პროცესის შედეგი...24

დასწავლული ვებ გვერდების რანჟირების პროცესი...24

რანჟირების ალგირთმი „PageRank" 25

რელაქსაციის ალგორითმი 26

ვებ გვერდების რანჟირება 26

რანჟირების პროცესის შეჯამება და სამომავლო გეგმები 28

შესრულებული სამუშაოების შეჯამება და მიღებული შედეგის სტატისტიკური ანალიზი 28

უნიკალური გვერდების რაოდენობა..29

უნიკალური სიტყვების რაოდენობა...29

15 ყველაზე ხშირად გამოყენებადი სიტყვა 29

15 ყველაზე ნაკლებად გამოყენებადი სიტყვა...30

სამომავლოდ ჩასატარებელი გეგმები 30

დასკვნა...31

გამოყენებულ ლიტერატურათა ნუსხა...32

შესავალი

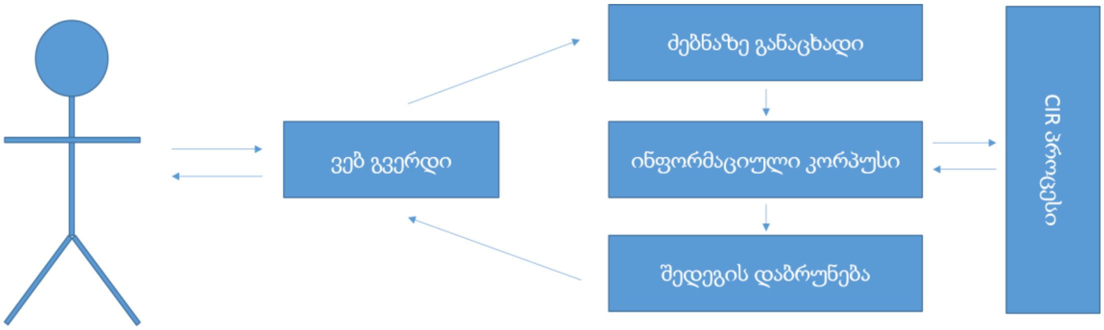
ინტერნეტ სამყარო შედგება მილიონობით სასარგებლო ინფორმაციისაგან. უამრავი მომხმარებელი, ყოველდღიურად იყენებს ძებნის ძრავებს, რათა მოიძიონ მათთვის სასურველი ინფორმაცია ინტერნეტში.

ძებნის ძრავის მუშაობის პრინციპი მდგომარეობს შემდეგში: იგი აბრუნებს რელევანტური ლინკების დალაგებულ მიმდევრობას, რომლებიც შეესაბამებიან დაინტერესებული პირის მიერ ძებნაზე გაკეთებულ განაცხადში არსებულ სიტყვას ან ფრაზას.

მსოფლიოში არსებული ძებნის ძრავები, პერმანენტულად ახორციელებენ ვებ სივრცის დასწავლის პროცესს, რათა მომხმარებლებს მიაწოდონ მუდმივად განახლებული ინფორმაცია. დღევანდელობაში არსებული ვებ სივრცის გათვალისწინებით, ცხადია თუ რა დიდ სირთულეებთან არის დაკავშირებული ძებნის პროცესის განხორციელება. თუმცა, საგულისხმოა ის ფაქტიც რომ კომპიუტერული ტექნოლოგიების სისწრაფე და მოცულობითი ტევადობა დღითიდღე იზრდება და ძებნის პროცესიც სულ უფრო და უფრო მეტ სირთულეებს უკავშირდება. მაგალითად, საძიებო სისტემების ბაზარზე არსებული ერთ ერთი წამყვანი კომპანია “Google” აცხადებს, რომ 2014 წელს მას შესწავლილი და დაინდექსებული ჰქონდა 45 მილიარდზე მეტი ვებ გვერდი. თუმცა ვებ გვერდის ერთჯერადად შესწავლა და დაინდექსება არ არის საკმარისი, საჭიროა თითოეული ვებ გვერდის გარკვეული სიხშირით შესწავლა, რათა მუდმივად განახლებული ინფორმაცია შევთავაზოთ ჩვენს მომხმარებელს. სწორედ ვებ გვერდების დამუშავების პერმანენტულობა წარმოადგენს ერთ–ერთ კომერციულ საიდუმლოებას ძებნის ძრავებზე მომუშავე კომპანიებისთვის.

ინფორმაციის მოძიება

როგორც უკვე აღვნიშნეთ, ინტერნეტში ინფორმაციის ძებნისას ძირითადი ობიექტი არის დაინტერესებული პირი, რომელიც ანხორციელებს ძებნის პროცესს ჩვენი საძიებო სისტემის დახმარებით.



სურ. 1

სურათზე მოცემულია ძებნის პროცესის ამსახველი მარტივი პროცედურა, რომლის დროსაც არსებობს დაინტერესებული პირი, რომელიც აკეთებს ძებნის განაცხადს საძიებო სიტყვის ან ფრაზის მეშვეობით. საძიებო სისტემა ანხორციელებს შემოსული განაცხადის ძებნას ინფორმაციის კორპუსში და უზრუნველყოფს ნაპოვნი მისამართების დალაგებული მიმდევრობის სწრაფად დაბრუნებას დაინტერესებული პირისთვის. თუმცა, ამ მარტივი პროცესის უკან უფრო მძლავრი და დროში გაწერილი პროცესი მიმდინარეობს, რომელიც პასუხისმგებელია ინფორმაციული კორპუსის ფორმირებაზე. ჩვენ მას CIR(ინფორმაციული კორპუსის ფორმირების) პროცესს ვუწოდებთ. CIR პროცესის აბრევიატურა შედგება პროცესის განცორციელებაში მონაწილე მოდულების დასახელებებისაგან და მდგომარეობს შემდეგში Crawling (ინფორმაციის დასწავლის პროცესი), Indexing (დასწავლული ინფორმაციის ინდექსაციის პროცესი) და Ranking (ტექსტურ კორპუსში არსებული ინფორმაციის რანჟირება)

აღსანიშნავია ის ფაქტი, რომ ინფორმაციული კორპუსის ფორმირების პროცესში მონაწილე ყველა მოდული ფუნქციონალურად დამოუკიდებელია და

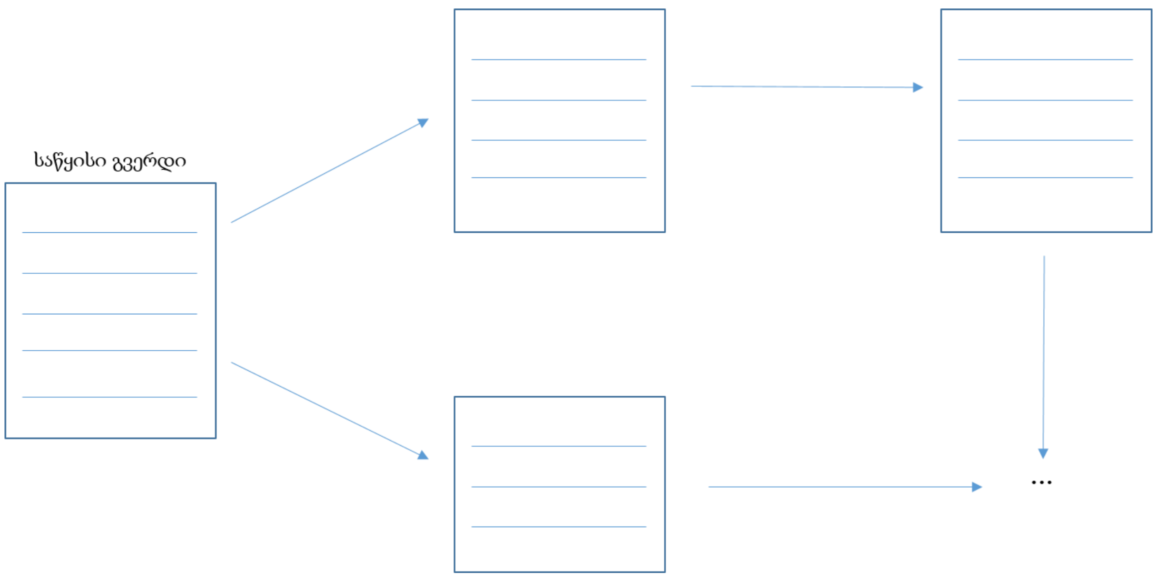
შესაძლებელია მოხდეს მათი გაშვება სხვადასხვა სერვერებზე “Hadoop” ტექნოლოგიის გამოყენებით, რაც კიდევ უფრო მძლავრს და დროის თვალსაზრისით მომხიბვლელს ხდის მოცემულ აპლიკაციას.

ვებ გვერდებიდან ინფორმაციის დასწავლა

ვებ გვერდებიდან ინფორმაციის დასწავლის (Crawling) პროცესი, როგორც უკვე აღვნიშნეთ მოდულურ აპლიკაციას წარმოადგენს, მისი როლი ინფორმაციული კორპუსის ფორმირების პროცესში უმთავრესია და სწორედ ამ მოდულის გაშვებით ხდება მთელი პროცესის წარმართვა.

დასწავლის პროცესის ინიციალიზაცია (Seed პროცესი)

როგორც ყველა აპლიკაციას, ინფორმაციის დასწავლის პროცესსაც სჭირდება შემავალი პარამეტრები რათა მოხდეს საწყისი მონაცემების ინიციალიზაცია და აპლიკაციამ შეძლოს პროცესის წარმართვა. საწყის მონაცემად შეგვიძლია ავიღოთ კონკრეტული ვებ გვერდის მისამართი რომელზეც გვსურს დასწავლის პროცესის შესრულება, მაგალითად https://tsu.ge/ ინფორმაციის დასწავლის პროცესის რობოტი განახორციელებს შესვლას ინიციალიზირებულ მისამართზე და მოახდენს კონკრეტული ვებ გვერდის კონტენტის ჩამოტვირთვას, შემდეგ მოხდება ჩამოტვირთული ვებ გვერდის ანალიზი, კონკრეტულად არსებობს თუ არა ამ გვერდიდან სხვა გვერდებზე გამავალი ლინკები და ასეთის არსებობის შემთხვევაში რობოტი რეკურსიულად მოახდენს მსგავსი პროცესის ჩატარებას როგორიც განხორციელდა ინიზიალიზირებული ვებ გვერდის შემთხვევაში.



სურ. 2

მოცემულ სურათზე ნათლად არის ასახული დასწავლის პროცესის ინიციალიზაცია და მისი შემდგომი გაგრძელება უკვე დასწავლულ გვერდებზე აღმოჩენილი ლიკნებისთვის, თუმცა ისმის კითხვა, თუ სადამდე შეიძლება განხორციელდეს რეკურსიული პროცესი, რადგან როგორც მოგეხსენებათ ინტერნეტ სივრცეში ვებ გვერდების საკმაოდ დიდი რაოდენობაა მოცემული და შეიძლება ჩვენი პროგრამა მისი შეზღუდული რესურსების გამო უბრალოდ გაჩერდეს და ვერ შეძლოს პროცესის დასრულება. ამ პრობლემას უფრო დეტალურად შემდეგ თავებში ჩავუღრმავდებით და შემოგთავაზებთ მისი გადაწყვეტის ჩვენეულ გზას. მანამდე კი ორი სიტყვით მოგახსენებთ თუ რა მოცულობის მონაცემებთან შეიძლება გვქონდეს საქმე დასწავლის პროცესის განცხორციელებისას. ცხადია სრული ვებ სივრცის მოცულობის გაზომვა შეუძლებელია, ამიტომ ჩვენს მიერ შემუშავებული დასწავლის რობოტი გაშვებულ იქნა თბილისის სახელმწიფო უნივერსიტეტის ვებ გვერდზე და საიტზე არსებული გვერდების ზომა საშუალო მოცულობის გვერდისთვის დაახლოვებით უდრის 50-60 კილობაიტს, აქედან გამომდინარე შეიძლება დაახლოვებით ვივარაუდოთ თუ რა სიდიდის მონაცემებითან შეიძლება გვქონდეს საქმე რაც საკმაოდ დიდ რესურსს მოითხოვს.

ინფორმაციის დასწავლის პროცესში არსებული პრობლემები

ინფორმაციის დასწავლის პროცესში, როგორც წინა თავში აღვნიშნეთ ერთ–ერთი პრობლემა შეიძლება გამოწვეული იყოს ვებ სივრცის უზარმაზარი მოცულობით, რამაც შეიძლება გამოიწვიოს შეზღუდული მოცულობის მქონე რობოტისთვის დასწავლის პროცესის შეწყვეტა და წარუმატებლად დასრულება. თუმცა ამასთან ერთად შეიძლება სხვა პრობლემებიც შეგვხვდეს. გავეცნოთ თითოეულს ცალცალკე და შევეცდებით წარმოვადგინოთ მათი გადაჭრის ჩვენეული ხედვა.

შეზღუდული მოცულობის პრობლემა

ამ პრობლემის გადასაჭრელად ჩვენს მიერ შემუშავებულ იქნა სამი მეთოდი: პირველი მდგომარეობს დასწავლის პროცესში განხორციელებული რეკურსიული ფუნქციისათვის n ზომის სიღრმის შემოღება და მისი მაქსიმალური ზღვარის მიღწევის შემთხვევაში პროცესის შეწყვეტა, მეორე მეთოდის მუშაობის პრინციპი დამოკიდებულია იმაზე თუ რა მონაცემების სერვერზე გვაქვს გაშვებული დასწავლის რობოტი, ანუ შესაძლებელია შეზღუდვა დავადოთ ჩვენი არსებული სერვერის მონაცემების შესაბამისად რათა არ მოხდეს მისი გადავსება და საბოლოოდ შეწყვეტა და ბოლოს მესამე მეთოდის იდეა მდგომარეობს დასწავლის პროცესის გაშვების კონკრეტულად ერთ რომელიმე ვებ გვერდზე, სწორედ ეს მეთოდი იქნა გამოყენებული ჩვენს მიერ წარმოდგენილი ძებნის ძრავის შემუშავების პროცესში და გაშვებულ იქნა თბილისის სახელმწიფო უნივერსიტეტის ვებ გვერდზე.

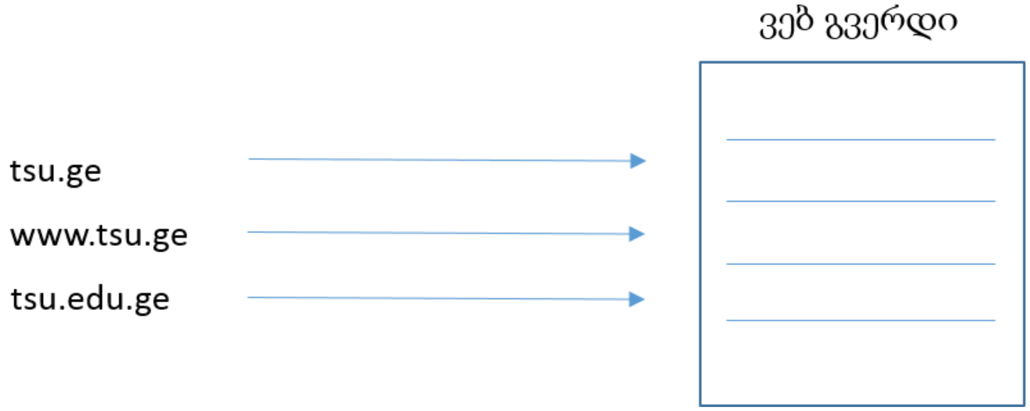
გვერდებს შორის ციკლების არსებობის პრობლემა

არანაკლებ მნიშვნელოვან პრობლემად შეიძლება ჩაითვალოს გვერდებს შორის ციკლების არსებობის პრობლემა. როგორც უკვე აღვნიშნეთ დასწავლის პროცესი მიმდინარეობს რეკურსიულად, თითოეულ გვერდზე არსებული ლინკებისთვის. აქედან გამომდინარე სრულიად შესაძლებელია რომ ნებისმიერ ორ გვერდზე არსებული ლინკები ერთმანეთზე მიუთითებდნენ, რამაც შეიძლება გამოიწვიოს დასწავლის პროცესის უსასრულოდ გაგრძელება ერთი და იგივე გვერდებისთვის და შესაბამისად მისი ჩაციკვლა. ამ პრობლემის გადასაჭრელად ჩვენს მიერ შემუშავებული მეთოლოგიის პრინციპი მდგომარეობს შემდეგში, რეკურსიის თითოეულ ბიჯზე ვებ გვერდიდან ამოღებულ მისამართებს ვაჯგუფებთ ერთად და ვინახავთ სიაში, რომელშიც არსებულ ლინკებზე დასწავლის პროცედურა ჯერ არ განხორციელებულა, დავარქვათ ამ სიას პირობითად „ToCrawl". მას შემდეგ რაც დასრულდება კონკრეტულ მისამართზე დასწავლის პროცესი განხორციელდება, მოხდება მისი გადატანა მეორე სიაში "AlreadyCrwaled" რომელშიც ვინახავთ ვებ გვერდების მისამართებს, რომელზეც დასწავლის პროცესი უკვე წარმატებით დასრულდა. შესაბამისად გამოირიცხება ციკლების არსებობის პრობლემა და ასევე დაზღვეულები ვიქნებით რომ არ მოხდეს ერთი და იგივე ვებ გვერდზე დასწავლის პროცესის რამოდენიმეჯერ შესრულება, რაც თავისთავად თავიდან აგვაცილებს ზედმეტი დროის და რესურსის ხარჯს.

გვერდებს შორის ალიასების არსებობის პრობლემა

ბოლოს შევეხოთ შედარებით ნაკლებად მნიშვნელოვან პრობლემას, რომელიც შეიძლება იყოს გამოწვეული ვებ გვერდებს შორის არსებული ალიასებით. ყველას მოგვეხსენება, რომ ინტერნეტში ხშირად გვხვდება გვერდები, რომლებზეც ერთი და იგივე ინფორმაციაა, მაგრამ მათზე სხვადასხვა მისამართი მიუთითებს. ამ შემთხვევაში პრობლემა შეიძლება იყოს ისევ და ისევ ზედმეტი დროის და რესურსის დახარჯვა და ასევე საბოლოოდ დაინტერესებული

პირისთვის გამოტანილ მისამართებში სხვადასხვა მისამართის გამოტანა, რომელიც რეალურად ერთი და იგივე მონაცემებზე მიუთითებს. ამან შეიძლება გამოიწვიოს მომხმარებლის უკმაყოფილება და ჩვენი საძიებო სისტემისადმი ავტორიტეტულობის დაკარგვა. პრობლემის აღმოსაფხვრელად შესაძლებელია გამოყენებულ იქნას კონკრეტულად ვებ გვერდების მონაცემების ანალიზი, რომელიც საკმაოდ დიდ ძალისხმევას მოითხოვს. ამ საკითხს ოდნავ მოგვიანებით კიდევ შევეხებით და ვეცდებით ნათლად წარმოვაჩინოთ მისი აქტუალობა და გადაჭრის აუცილებლობა.



სურ. 3

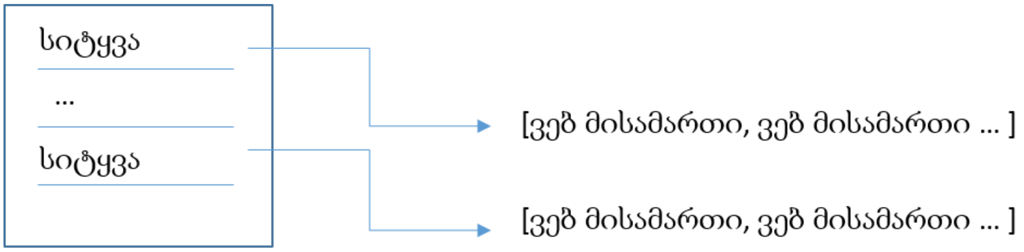
დასწავლის პროცესის პერიოდულობა

როგორც უკვე აღვნიშნეთ ინფორმაციის დასწავლის პროცესი არ წარმოადგენს ერთჯერად პროცესს, იმისათვის რომ ჩვენმა ძებნის ძრავმა შეძლოს მუდმივად განახლებადი ინფორმაციის მომხმარებლამდე მიწოდება, პერმანენტულად უნდა მოხდეს დასწავლის პროცესის გაშვება, თუმცა დასწავლის პროცესის არამიზნობრივმა და ხშირმა გაშვებამ დროის ზედმეტ ხარჯთან ერთად შეიძლება ჩვენი საძიებო სისტემაში არსებული ინფორმაციული კორპუსის გადავსება გამოიწვიოს. ამიტომ ჩვენი აზრით უნდა მოხდეს რობოტის შექმნა რომელიც გაეშვება უკვე დასწავლულ ვებ გვერდებზე და მოახდენს მათზე

განახლებების არსებობას, რის შემდეგაც უკვე მოხდება დასწავლის პროცესის ხელახალი გაშვება.

დასწავლის პროცესის შედეგი

მას შემდეგ, რაც მოხდება დასწავლის პროცესის დასრულება კონკრეტული ვებ მისამართისთვის, საშუალება გვექნება მოვახდინოთ ინფორმაციული კორპუსის ფორმირება, რომლის თავდაპირველი სტრუქტურაც მოწემულია სურათაზე



სურ. 4

როგორც სურათიდან ჩანს, ამ ეტაპისათვის ჩვენ უკვე გაგვაჩნია უნიკალური სიტყვა ფრაზების მიმდევრობა რომლებიც შეიცავენ ინფორმაციას თუ რომელ გვერდებზე ხდება ამ სიტყვა ფრაზების განმეორება, შესაბამისად უკვე გვაქვს საშუალება, რომ მომხმარებელს მივაწოდოთ მისთვის სასურველი სიტყვის ან ფრაზის შესაბამისი ვებ მისამართების მიმდევრობა. თუმცა, დასამუშავებელი ინფორმაციის მოწულობიდან გამომდინარე შეიძლება შევხვდეთ პრობლემას, რომელიც გამოხატული იქნება ინფორმაციის მოძიების სისწრაფეში. ამ პრობლემის ანალიზსა და მისი გადაჭრის გზებზე დაწვრილებით მოგახსენებთ შემდეგ თავში.

შესწავლილი ინფორმაციის ანალიზი და ინდექსაცია

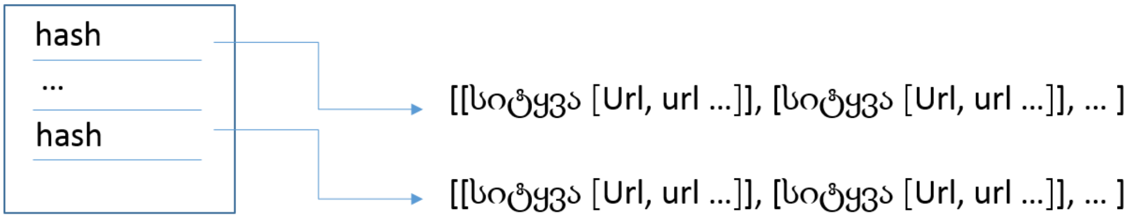
ბუნებაში არსებულ ნებისმიერი სახის ინფორმაციას სჭირდება ანალიზი და მისი სტრუქტურიზაცია, რადგან ნორმალიზაციის გარეშე არანაირი ინფორმაცია არ შეიძლება წარმოადგენდეს რაიმე სახის ფასეულობას.

ინფორმაციის ანალიზი

როგორც უკვე აღვნიშნეთ, ინფორმაციის დასწავლის პროცესის დასრულების შემდეგ, ჩვენ უკვე გვაქვს მომხმარებლისთვის საძიებო ვებ მისამართების მიწოდების საშუალება, თუმცა ამ ეტაპზე უკვე თავს იჩენს არსებული ინფორმაციის ნორმალიზაციისა და შენახვის აქტუალობა. ასევე ისეთი ალგორითმების შემუშავება, რომელიც უპირატესობას მიანიჭებს ჩვენს საძიებო სისტემას, სხვა სისტემებთან შედარებით. როგორც ვიცით, ალგორითმისთვის უპირატესობის ძირითად კომპონენტს შესრულების დრო წარმოადგენს, ამიტომ ჩვენც, ჩვენი საძიებო სისტემის ზუსტად ამ მიმართულებით გაუმჯობესებაზე ვიზრუნებთ.

ჩავატაროთ ანალიზი, თუ რამ შეიძლება შეზღუდოს ჩვენი ალგორითმის სისწრაფე ძიების პროცესში. თუ დავუკვირდებით, ჩვენი ძიების პროცესი დამყარებულია ინფორმაციული კორპუსის სრულ გადარჩევაზე, თუმა რეალურ ცხოვრებაში სულ სხვანაირად ხდება სიტყვის ძიება, მაგალითად ლექსიკონში სიტყვის ძიებისას ჩვენ არ ვახდენთ წიგნის სრულ გადარჩევას, არამედ ვფურცლავთ მხოლოდ იმ გვერდებს, რომელზე არსებულ სიტყვებსა და ჩვენს საძიებო სიტყვას საერთო პირველი ასო გააჩნია. განვახორციელოთ მსგავსი მიდგომა ჩვენი საძიებო სისტმისთვისაც, ამ შემთხვევაში საჭირო იქნება შემოვიღოთ სასრული რიცხვი b რომელიც წარმოადგენს უჯრების ზომას რომელშიც გადანაწილებული იქნება ჩვენი საძიებო სისტემის მიერ აღმოჩენილი და დასწავლული სიტყვები. ასევე დაგვჭირდება პროცედურა - ეგრედ წოდებული

ჰეშირების ფუნქცია რომელიც მოგვცემს უჯრის ზუსტ მისამართს, თუ სად შეიძლება მდებარეობდეს საძიებო სიტყვა.



სურ. 5

სურათზე მოცემულია ჩვენს მიერ შემოთავაზებული პროცესის სქემატური აღწერა, საჭიროა მხოლოდ ისეთი ჰეშირების ფუნქციის შერჩევა რომელიც საუკეთესო შედეგს მოგვცემს ჩვენი ძებნის ძრავის მიერ დასწავლული სიტყვებისათვის. სწორედ ასეთი ფუნქციის შერჩევა წარმოადგენს დიდ პრობლემას მსგავსი სირთულის ამოცანებისათვის.

ჰეშირების ფუნქციის შერჩევა

ჰეშირების ფუნქციის შერჩევის პროცესში, რომელმაც საკმაოდ დიდი დრო მოითხოვა, ჩვენ შევხვდით გარკვეული ტიპის პრობლემებს, რომლის განხილვასა და ანალიზს დავუთმობთ შემდეგ თავს.

პირველ რიგში როდესაც დავიწყეთ ჰეშირების ფუნქციის შერჩევა, უპირატესობა მივანიჭეთ წინა თავში შემოთავაზებულ ძებნის ალგორითმს, რომელიც ასახავს პროცესს თუ როგორ ხდება სიტყვების ძებნა რეალურ ცხოვრებაში. როგორც წინა თავში აღვნიშნეთ საჭიროა შემოვიღოთ ცვლადი b რომელშიც შენახული იქნება უჯრების რაოდენობა, თუ რამდენ უჯრაში უნდა გადავანაწილოთ სიტყვები და ცვლადი k რომელიც აღნიშნავს დასახარისხებელი სიტყვების ზომას. მოცემული განმარტებიდან გამომდინარე ლოგიკურად

შეგვიძლია ჩამოვაყალიბოთ რამოდენიმე მკაცრი განმარტება რომელსაც უნდა აკმაყოფილებდეს ჩვენს მიერ შერჩეული ჰეშირების ფუნქცია:

- მოცემული F(b, k, s) -> x ჰეშირების ფუნქცია, სადაც b არის უჯრების რაოდენობა და k არის შესანახი სიტყვების რაოდენობა, იძლევა x უჯრის ვალიდურ მისამართს s სიტყვისათის მაშინ და მხოლოდ მაშინ თუ სრულდება პირობა k>b
- მოცემული F(b, k, s) -> x ჰეშირების ფუნქცია ვალიდურია მაშინ და მხოლოდ მაშინ თუ მისი მნიშვნელობათა არე განსაზღვრულია 0, b-1 შუალედში.

მოცემული განმარტებებიდან გამომდინარე უკვე საშუალება გვაქვს მოვახდინოთ ჰეშირების ფუნქციის ფორმირება. ესე იგი ჩვენს ჰეშირების ფუნქციას უნდა გადაეცემოდეს სიტყვა და აბრუნებდეს ამ სიტვის შესაბამის უჯრის რიცხვით მნიშვნელობას.

$$< \text{String} > \to (Hash\ String)\ \to Number(0 \ldots, b-1)$$

იმისათვის რომ დაბრუნებული მნიშვნელობა ყოველთვის იყოს შესაბამის შუალედში, ვიყენებთ მოდულის ოპერაციას. ხოლო სტრინგის შესაბამისი რიცხვითი მნიშვნელობის ასაღებად ვიყენებთ პროგრამირების ენა პითონში არსებულ Ord ფუნქციას, აქედან გამომდინარე ჩვენს მიერ შემუშავებული ჰეშირების ფუნქციის ფსევდო კოდს ექნება შემდეგი სახე:

```
def has_string(keyword, bucket):
return ord(keyword[0])%buckets
```

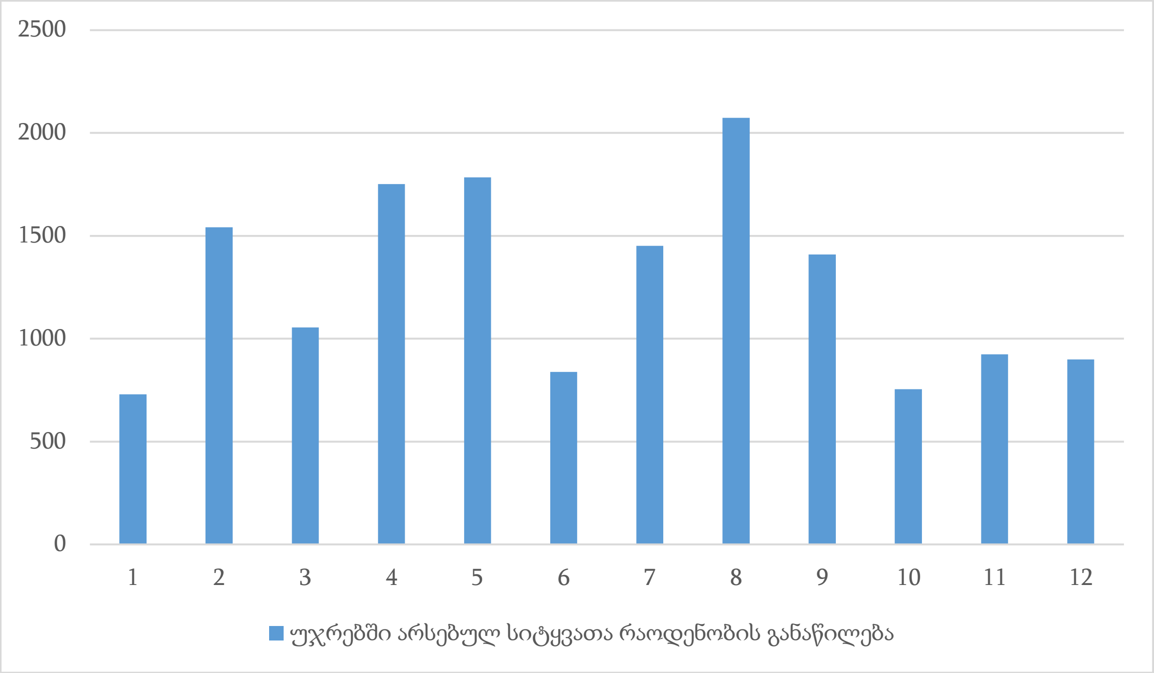
მოცემული ჰეშირების ფუნქციით, უკვე შეგვიძლია რეალური ინდექსირების პროცესის განხორციელება, თუმცა მანამდე გადავწყვიტეთ მისი მართებულობის შემოწმება, რადგან საუკეთესო ჰეშირების ფუნქციის შემთხვევაში

თითოეულ b რაოდენობის უჯრაში არსებული სიტყვების რაოდენობა დაახლოვებით თანაბრად უნდა იყოს განაწილებული.

არსებული ჰეშირების ფუნქციის ვალიდურობის დასამტკიცებლად, შევქმენით დამატებითი ფუნქცია რომელიც ჩვენს მიერ შერჩეული ტექსტისათვის დაგვიბრუნებს თითოეულ უჯრაში არსებული სიტყვების რაოდენობებს.

```
def test_hash_function(func, keys, size):
    results = [0] * size
    keys_used = []
    for w in keys:
        if w not in keys_used:
            hv = func(w,size)
            results[hv] +=1
            keys_used.append(w)
    return results
```

მოცემული ტესტირების ფუნქცია გავუშვით b=12 ზომის უჯრისთვის შემთხვევითად აღებულ ტექსტზე, რომელშიც მოცემული იყო დაახლოვებით 107533 სიტყვა და დაბრუნებული შედეგი არც თუ ისე სახარბიელო გამოდგა. დაბრუნებული შედეგის რიცხვითი გრაფიკი გამოსახულია 6 სურათზე. რაც წარმოადგენს არაოპტიმალურ შედეგს ჩვენი ძებნის ძრავისთვის ამიტომ საჭირო გახდა უფრო უკეთესი ჰეშირების ფუნქციის შექმნა რომელიც უკეთეს შედეგს მოგვცემდა.



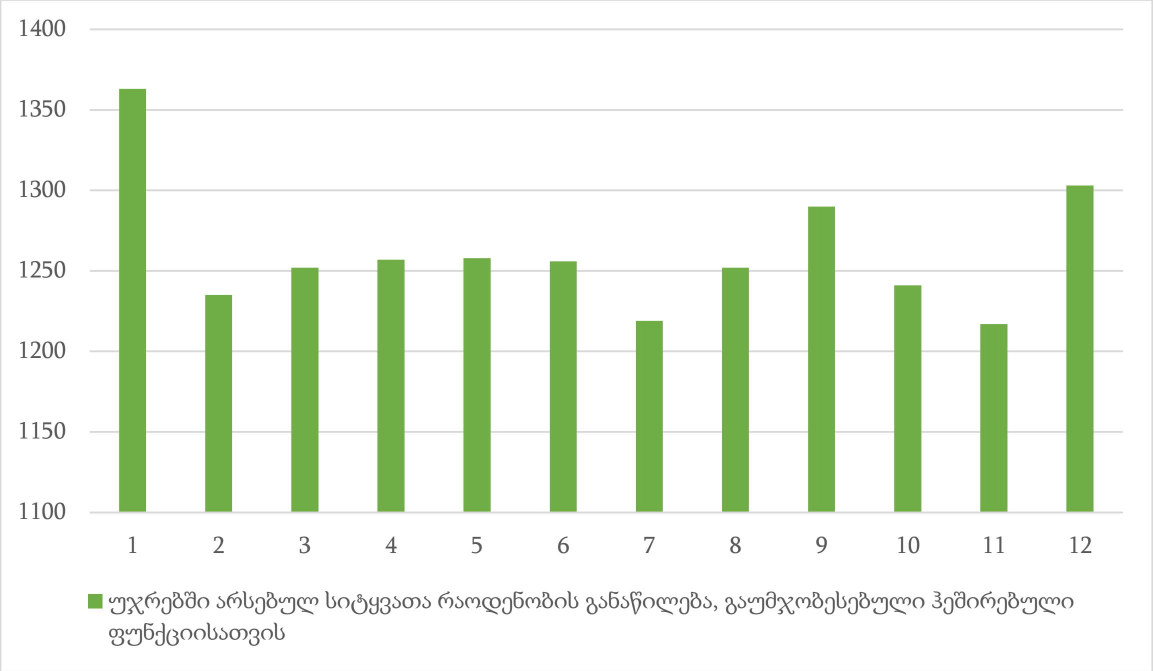
სურ. 6

ჰეშირების ფუნქციის გაუმჯობესება

წინა თავში ჩატარებული კვლევების საფუძველზე საჭირო გახდა უფრო საიმედო ჰეშირების ფუნქციის შერჩევა. ამისათვის გადავწყვიტეთ არა მარტო პირველი ასოს, არამედ სიტყვაში არსებული ყველა სიმბოლოს გამოყენება ჰეშირებული მნიშვნელობის ფორმირებაში, მისი ფსევდო კოდი მდგომარეობს შემდეგში:

```
def hash_string(keyword, buckets):
    h=0
    for c in keyword:
        h=(h+ord(c)) % buckets
    return h
```

ისევ და ისევ მოცემული ჰეშირების ფუნცქიის რეალურ რეჟიმში გაშვებამდე მოვახდინეთ მისი შემოწმება იგივე შემავალი პარამეტრებისთვის. რამაც წინა რეზულტატთან შედარებით საკმაოდ ოპტიმალური შედეგი მოგვცა.



სურ. 7

ამიტომ, მოვახდინეთ მისი გამოყენება ჩვენი არსებული ტექსტური კორპუსისათვის, რადგან თუ ინდექსირების პროცესის ჩატარებამდე ჩვენს კორპუსში ინფორმაციის ძებნის სირთულე იყო O(N), ჰეშირების ფუნქციის გამოყენებამ ინფორმაციულ კორპუსში ძებნის სირთულე დაიყვანა C+M სახემდე სადაც C ჰეშირების კონსტანტაა თუ რა დრო ჭირდება ჰეშირების ფუნქციის განხორციელებას და M კი თითოეულ უჯრაში არსებული ელემენტების რაოდენობა. უნდა აღინიშნოს ასევე რომ ზოგადად ინდექსაციის პროცესის შესულებამ ჩვენს ინფორმაციულ კორპუსზე საკმაოდ შედეგი მოიტანა ძებნის

განხორციელებაში, რაც დაახლოვებით 0,05 წამს უდრის, ათი სხვადასხვა სიტყვის ძიების დროს.

ინდექსირების პროცესის შედეგი

ინდექსირების პროცესის დასრულების შემდეგ ჩვენ უკვე გაგვაჩნია სრულფასოვნად ნორმალიზებული ინფორმაციული კორპუსი, რომელშიც სწრაფად შეგვიძლია განვახორციელოთ ძებნის პროცესი და მოძიებული შედეგი მივაწოდოთ დაინტერესებულ პირს, თუმცა გადასაწყვეტი რჩება ძებნის ძრავის ფორმირებაში მონაწილე კიდევ ერთი – რანჟირების პროცესის განხორციელება ჩვენი ინფორმაციული კორპუსის მონაცემებზე.

დასწავლული ვებ გვერდების რანჟირების პროცესი

ძებნის ძრავში ვებ მისამართების რანჟირების პროცესი საკმაოდ რთული და მნიშვნელოვანია. მოდულები რომლებიც ჩვენ წინა თავებში განვიხილეთ პასუხისმგებლები არიან ინფორმაციის მოძიებასა და მის სწრაფ გამოტანაზე, თუმცა დამეთანხმებით რომ ინფორმაციის სწრაფად გამოტანას აზრი არ ექნება თუ ჩვენს მიერ მომხმარებლისთვის შეთავაზებული ინფორმაცია არ იქნება ღირებული და მის ინტერესებზე მორგებული. საგულისხმოა ის ფაქტი რომ საძიებო კომპანიებისთვის სწორედ ამ მოდულში გამოყენებული ალგორითმები წარმოადგენენ კომერციულ საიდუმლოებას და ზოგიერთი მათგანი და კონკურენტი კომპანიის ჩაძირვის მიზეზიც კი გამხდარა. ასეთს წარმოადგენს კომპანია “Google”-ის მიერ შემუშავებული რანჟირების ალგორითმი „PageRank”, რომელმაც 1998 წელს უპირობო ლიდერის პოზიცია მოუტანა კომპანიას და იმ პერიოდში არსებული ყველაზე მძლავრი საძიებო კომპანია “AltaVista” გაკოტრების პირას მიიყვანა. სწორედ მისი მუშაობის პრინციპებზე და უპირატესობებზე ვისაუბრებთ ამ თავში.

რანჟირების ალგირთმი „PageRank”

როგორც უკვე აღვნიშნეთ, რანჟირების ალგორითმი “PageRank”-ი შემუშავებულია კომპანია “Google”-ის, კერძოდ კი მისი დამფუძნებლების ლარი ფეიჯისა და სერგეი ბრინის მიერ.

ალგორითმის ძირითადი იდეა დამყარებულია, რეალურ ცხოვრებაში ადამიანების პოპულარობის განმსაზღვრელ ფაქტორზე. ვიგულისხმოთ რომ რაც უფრო მეტი მეგობარი ყავს ადამიანს, მით უფრო პოპულარულია და ვცადოთ მსგავსი მოდელის წარმოდგენა ინტერნეტ სივრცისთვის. იმისათვის რომ განვსაზღვროთ „მეგობრობის“ ცნება ვირტუალურ სამყაროში, კერძოდ კი ინტერნეტ სივრცეზე, შეგვიძლია გამოვიყენოთ ვებ გვერდებზე არსებული მისამართები. კერძოდ, რაც უფრო მეტი მისამართი მიუთითებს რომელიმე გვერდს მით უფრო პოპულარულია, თუმცა უნდა გავითვალისწინოთ ისიც თუ რამდენად პოპულარული გვერდიდან ხდება სხვა გვერდზე მითითება, ამის გათვალისწინებით საშუალება გვეძლევა უკვე განვსაზღვროთ ვებ გვერდის პოპულარობის გამოსათვლელი ალგორითმი, რომლის მათემატიკური რეალიზაცია მდგომარეობს შემდეგში:

$$Popularity(P) = \sum_{f \in friends(P)} Popilarity(f)$$

როგორც სურათზე გამოსახული ფორმულიდან ჩანს, პროცესი მარტივია, თითოელი გვერდის ავტორიტეტულობას განსაზღვრავს, მასზე მიმთითებელი გვერდების ავტორიტეტულობის ჯამი. თუმცა პრობლემას საწყისი მონაცემების არსებობა წარმოადგენს. რადგან თავდაპირველად ჩვენ არ გაგვაჩნია ინფორმაცია არცერთი ვებ გვერდის ავტორიტეტულობის შესახებ. ამისათვის შემოღებულია ეგრედ წოდებული რელაქსაციის ალგორითმი.

რელაქსაციის ალგორითმი

რელაქსაციის ალგორითმის იდეა მარტივია და მდგომარეობს შემდეგში, საწყისი მონაცემებისთვის შეგვიძლია ვიგულისხმოთ რომ ყოველი გვერდის ავტორირეტულობა არის ერთის ტოლი, შემდეგ კი იტერაციული პროცესის დახმარებით მოვახდინოთ მათი გაუმჯობესება სანამ არ მივაღწევთ სასურველ შედეგს. სასურველი შედეგი კი წარმოადგენს ჩვენი ალგორითმის გაჩერების პირობას. რადგან ვებ გვერდის ავტორიტეტულობის დადგენისთვის გვჭირდება მასზე იტერაციული პროცესის რამოდენიმეჯერ შესრულება, ამიტომ ჩვენი ავტორიტეტულობის დათვლის ფორმულისათვის დაგვჭირდება დამატებითი პარამეტრი t, რომელიც წარმოადგენს P გვერდზე იტერაციული პროცესის ჩატარების რაოდენობას. აქედან გამომდინარე ჩვენი ავტორიტეტულობის განმსაზღვრელი ფუნქცია მიიღებს შემდეგ სახეს:

$$Popularity(t, P) = \sum_{f \in friends(P)} Popularity(t-1, f); \text{სადაც } t > 0;$$

ჩვენს მიერ ინიზიალიზაციისას ნაგულისხმევი პოპულარობიდან გამომდინარე t=0 მნიშვნელობისთვის, $Popularity(0, P)$ მიიღებს ერთის ტოლ მნიშვნელობას.

ვებ გვერდების რანჟირება

ვებ გვერდებისათვის შესაბამისი რიცხვითი რანკის მინიჭების იდეა წინა თავში განხილული გვერდის პოპულარობის განმსაზღვრელი ალგორითმის მსგავსია. აქედან გამომდინარე რანკის მინიჭების ფუნქციას ექნება შემდეგი სახე:

$$Rank(t, url) = \sum_{p \in inlinks(url)} Rank(t-1, p);$$

სადაც, inlinks არის p გვერდისთვის, შემავალი მისამართების სიმრავლე, თუმცა ამ შემთხვევაში საგულისხმო ფაქტი გამავალი მისამართების რაოდენობაც არის, ანუ კონკრეტული გვერდის რანკის გამოთვლისას ყურადღება უნდა მივაქციოთ თუ რამდენი გამავალ მისამართები გააჩნია თითოეულ გვერდს. და ფორმულა მიიღებს შემდეგ სახეს:

$$Rank(t, url) = \sum_{p \in inlinks(url)} Rank(t-1, p)/outliks[p]$$

თუმცა ამ შემთხვევაშიც ვაწყდებით პრობლემას, რომელიც გამოწვეულია ისეთი გვერდების არსებობით, რომელსაც შეიძლება საერთოდ არ ქონდეს შემავალი და გამავალი მისამართები, ამიტომ იმისათვის რომ მოვახდინოს მინიჭებული რანკების თანაბარი განაწილება შემოვიღოთ ცვლადი d, რომელიც წარმოადგენს ალბათობას იმისას თუ რამენად ხშირად შეიძლება ჩვენი რანკის განმსაზღვრელმა რობოტმა მიიღოს ახალი გვერდის რანკის მინიჭებაზე გადასვლის გადაწყვეტილება, იმ შემთხვევაში თუ შეხვდა ისეთი გვერდი რომელსაც არ გააჩნია შემავალი და გამავალი მისამართები. მის მნიშვნელობის მისანიჭებლად პრაქტიკაში მიღებულია რომ იყოს 1–ზე ნაკლები. ასევე იმისათვის რომ შევინარჩუნოთ რანკების მნიშვნელობათა არის მინიმალური ზღვარი მოგვიწევს რანკების ინიციალიზირებული მნიშვნელობის შეცვლაც რაც იქნება 1/N სადაც N არის ჩვენს ინფორმაციულ კორპუსში არსებული ვებ გვერდების რაოდენობა. მიღებული განსაზღვრებიდან გამომდინარე ჩვენს რანკის მიმნიჭებელ ფუნქციას ექნება შემდეგი სახე:

$$Rank(t, url) = d * \sum_{p \in inlinks(url)} \frac{Rank(t-1, p)}{outliks[p]} + \frac{1-d}{N}$$

რანჟირების პროცესის შეჯამება და სამომავლო გეგმები

როგორც უკვე აღვნიშნეთ, წინა თავში აღწერილი მეთოდი წარმოადგენს კომპანია “Google”-ის მიერ შემუშავებულ ერთ–ერთ ალგორითმს, რომელიც გამოყენებულია ვებ გვერდების რანჟირების პროცესში, თუმცა იგი წარმოადგენს ერთ მცირე პარამეტრს, რომლის გამოყენებაც ხდება ძებნის ძრავში. ასე რომ იგი არ წარმოადგენს ჩვენს მიერ შესრულებული ძებნის ძრავში არსებული რანჟირების პროცესის საბოლოო ვარიანტს. ასევე უნდა აღინიშნოს ისიც რომ მსოფლიოში არსებული უდიდესი კომპანიებიც კი ყოველდღიურად რამოდენიმე სხვადასხვა სახის რანჟირების ალგორითმს გვთავაზობენ, რათა მოხდეს ძებნის ძრავის მაქსიმალურად მორგება მომხმარებელთა მოთხოვნაზე. შესაბამისად ცხადია თუ რამდენად აქტუალურია ამ პრობლემის ირგვლივ მუშაობა და ჩვენს სამომავლო კვლევასაც სწორედ ეს წარმოადგენს, რათა მოხდეს რანჟირების პროცესის უფრო ღრმა დამუშავება და მისთვის კონკრეტული ალგორითმების შემუშავება რომლებსაც უპირატესობა ექნება არსებულ სხვა ძრავებთან მიმართებაში და მორგებული იქნება ქართული ენის და მისი მსგავსი მონათესავე ენებისათვის ძებნის პროცესში არსებულ სირთულეების გადაჭრაზე.

შესრულებული სამუშაოების შეჯამება და მიღებული შედეგის სტატისტიკური ანალიზი

შესრულებული სამუშაოს იდეა მდგომარეობს ქართულ ვებ სივრცეში არსებულ ძებნასთან დაკავშირებული პრობლემების მიმოხილვასა და მათი აღმოფხვრის გზების ძიებაში, რაც ჩვენი აზრით სტიმულის მიმცემი იქნება ქართულ ბაზარზე საძიებო სისტემების გაჩენისა და განვითარებისათვის. როგორც წინა თავებში აღვნიშნეთ განხორციელებული კვლევის საფუძველზე შეიქმნა სრულფასოვანი ძებნის ძრავი და მოხდა მისი გაშვება თბილისის სახელმწიფო უნივერსიტეტის ვებ გვერდზე, რამაც საშუალება მოგვცა შეგვედგინა ვებ გვერდზე არსებული ინფორმაციის სტატისტიკა.

უნიკალური გვერდების რაოდენობა

თბილისის სახელმწიფო უნივერსიტეტის საიტი ერთი შეხედვით პატარა შეიძლება მოგვეჩვენოს, თუმცა საკმაო რაოდენობით აღმოჩნდა მასზე არსებული გვერდების რაოდენობა რომელიც ჯამში შეადგენს 11465 უნიკალურ გვერდს.

უნიკალური სიტყვების რაოდენობა

ასევე სიტყვების რაოდენობაზეც შეიძლება აღინიშნოს, რომ არც თუ ისე მწირია და იგი შეადგენს 99770 სიტყვას. კავშირების, პირის ნაცვალსახელებისა და ზედსართავი სახელების გამოკლებით.

15 ყველაზე ხშირად გამოყენებადი სიტყვა

„სახელმწიფო (2353), უნივერსიტეტის (2304), თბილისის (2276), ჯავახიშვილის (2204), სახელობის (2169), აკადემიური (1555), საბჭოს (1547), საქართველოს (1425), ფაკულტეტის (1260), მეცნიერებათა (1240), იურიდიული (1235), მუხლის (1170), პირველი (1133), დამტკიცებული (1119), განათლების (1118)“

საინტერესო სახის სტატისტიკას წარმოადგენს ასევე, თუ რომელია ყველაზე პოპულარული ვებ გვერდეზე არსებული სიტყვებიდან. რომელმაც მეტნაკლებად მოსალოდნელი შედეგი მოგვცა, თუმცა სიტყვები მოცემულია სხვადასხვა ბრუნვაში და შესაძლოა არ იძლეოდეს ას პროცენტიან სიზუსტეს, უფრო მეტი სიზუსტისათვის საჭიროა, შეიქმას ალგორითმის შექმნა რომელიც ქართული სიტყვებისათვის განახორციელებს ფუძის ამოღების ალგორითმს და მოხდება სიტყვების საწყის ფორმაში წარმოდგენა. როგორც უკვე აღვნიშნეთ ამ ტიპის ალგორითმების შემუშავება წარმოადგენს ჩვენი ნაშრომის შემდეგ ეტაპს.

15 ყველაზე ნაკლებად გამოყენებადი სიტყვა

„რექტორები (1), რეზიდენტურა (1), ინსტიტუტები (1), კონკურსები (1), მოდელირებაში (1), დოკუმენტები (1), სიახლეები (1), ჰუმანიტართა (1), ბრძანებები (1), ოქმები (1), მისალმება (1), ავტორიზაცია (1), კულტურა (1), სპორტი (1), ბიუჯეტები (1)“

როგორც ჩანს ყველაზე ნაკლებად გამოყენებადი სიტყვა „რექტორები“ ყოფილა, მაგრამ როგორც აღვნიშნეთ ფუძის ამოღების ალგორითმის გამოყენებით შესაძლოა ეს სტატისტიკა შეიცვალოს.

სამომავლოდ ჩასატარებელი გეგმები

ნაშრომის პროცესში ჩატარებულმა კვლევებმა და მათმა პრაქტიკულმა რეალიზებამ კიდევ ერთხელ ცხადყო წამოყენებული პრობლემის აქტუალობა და საშუალება მოგვცა მისი შემდგომი განვითარების გეგმებზე დაგვეწყო ფიქრი, რაც საკმაოდ ფართო არეალს წარმოადგენს კომპიუტერული მეცნიერებების სფეროში. ჩვენს მიერ სამომავლოდ განსახორციელებულ გეგმებთან დაკავშირებით მცირედი საუბრები გვქონდა წინა თავებშიც, ამიტომ ამ თავში შევეცდებით მათ თავმოყრას.

პირველ რიგში უნდა გაგრძელდეს კვლევები დასწავლის პროცესის პერმანენტულად განახლების საკითხში, რათა ჩატარებული კვლევა ეყრდნობა მხოლოდ ერთხელ განხორციელებად დასწავლის პროცესს, თუმცა საჭიროა მისი უწყვეტ პროცესად წარმართვა, ამისათვის კი როგორც უკვე აღვნიშნეთ საჭიროა რობოტის შექმნა რომელიც უკვე დასწავლილი გვერდებისათვის განახლებების აღმოჩენის საშუალებას მოგვცემს.

ასევე ღრმა კვლევის საგანს წარმოადგენს, ძებნის ძრავის მორგებას ქართული ენის და მისი მონათესავე ენებისათვის, რათა მოხდეს მისი ოპტიმიზაცია და ამ კუთხით განვითარება. ამ კუთხით განსავითარებლად შესაძლებელია ალგორითმების შემუშავების პროცესში საჭირო გახდეს ამ დარგის

სპეციალისტების ჩართულობა, რაც კიდევ ერთხელ ხაზს უსვამს მის მასშტაბურობასა და განვითარების აუცილებლობაზე.

კიდევ ერთ მეტად მნიშვნელოვან კვლევის საგანს წარმოადგენს ძებნის ძრავის განვითარება არა მარტო სიტყვა ფრაზების, არამედ კონკრეტული წინადადების მნიშვნელობის გამოყენებით ასევე ქართული ენისთვის, ეს მიდგომა კომპიუტერული მეცნიერებების სფეროში ცნობილია როგორც „Data Meaning”.

დასკვნა

დასკვნის სახით, ნაშრომზე შეიძლება ითქვას რომ იგი წარმოადგენს ქართულ ენაზე მორგებულ ერთ–ერთ პირველ სრულფასოვნად რეალიზებულ ძებნის ძრავს, მასში მოყვანილი პრაქტიკული და თეორიული შედეგების რეზულტატები მნიშვნელოვანია ინტერნეტ სივრცეში ქართული ენის განვითარებისათვის რაც კიდევ ერთ წინ გადადგმულ ნაბიჯად შეიძლება ჩაითვალოს ამ სფეროს პოპულარიზაციისათვის. ასევე ნაშრომში მოყვანილი არაერთი მაგალითი და საბოლოო შედეგები ხაზს უსვამს მისი გამოყენების თვალსაზრისით პოპულარობაზე, რასაც დიდი იმედი გვაქვს რომ კიდევ უფრო მეტ პრაქტიკულ მნიშვნელობას შესძენს დაგეგმილი სამომავლო სამუშაოები.

გამოყენებულ ლიტერატურათა ნუსხა

1. https://www.udacity.com, Build a Search Engine & a social network, 2014
2. Ziyang Wang. 2006. Incremental Web Search: Tracking Changes in the Web. PhD thesis, New York University.
3. http://www.worldwidewebsize.com/
4. http://www.internetlivestats.com/
5. J. Cho and S. Roy Impact of search engines on page popularity. In Proc. 13th international World Wide Web Conference, 2004
6. E. Coffman, J. Z. Liu and R. R. Weber. Optimal robot scheduling for web search engines. Technical report, INRIA, 1997.
7. https://www.google.ge/
8. https://www.wikipedia.org/
9. M. Najork and J. Wiener. Hight-performance web crawling. SRC Research Report 173, Compaq System Research Center, 2001
10. A. Heydon and M. Najork. Mecator: a scalable, extensible web crawler. In Proc. 8th International World Wide Web Conference, 1999
11. R. Kosala and H Blockeel. Web mining research: a survey, In Proc 6th ACM SIGKDD Special Interest Group on Knowledge Discovery and Data Mining, 2000.