KADIR HAS UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ARAMA MOTORLARI MİMARİSİ, WEB SAYFALARININ İÇERİK SKORU VE GOOGLE PAGERANK FORMÜLÜNÜN İNCELENMESİ

MUHİTTİN IŞIK

Ocak, 2013

ARAMA MOTORLARI MİMARİSİ, WEB SAYFALARININ İÇERİK SKORU VE GOOGLE PAGERANK FORMÜLÜNÜN İNCELENMESİ

MUHİTTİN IŞIK

Enformasyon Teknolojileri Programı'nda Yüksek Lisans için gerekli kısmi şartların yerine getirilmesi amacıyla Fen Bilimleri Enstitüsü'ne teslim edilmiştir.

KADIR HAS UNIVERSITY

Ocak, 2013

KADIR HAS UNIVERSITY GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ARAMA MOTORLARI MİMARİSİ, WEB SAYFALARININ İÇERİK SKORU VE GOOGLE PAGERANK FORMÜLÜNÜN İNCELENMESİ

MUHİTTİN IŞIK

APPROVED BY:

Jüri Üyesi: Prof.Dr. Hasan DAĞ _____

Danışman: KHAS. Ü. Enformasyon Tek. Öğr. Üyesi

Jüri Üyesi: Doç. Dr. Mustafa BAĞRIYANIK _____

ITU. Elektrik Mühendisliği Öğr. Üyesi

Jüri Üyesi: Yrd. Doç. Dr. Öznur Yaşar DİNER _____

KHAS. Ü. Yönetim Bilişim Sistemleri Öğr. Üyesi

KABUL TARİHİ: 10/01/2013

Ben Muhittin IŞIK, bu Yüksek Lisans Tezinde sunulan çalışmanın şahsıma ait olduğunu ve başka çalışmalardan yaptığım alıntıların kaynaklarını kurallara uygun biçimde tez içerisinde belirttiğimi onaylıyorum.

…….....

Muhittin IŞIK

STRUCTURE OF SEARCH ENGINES, CONTENT SCORE OF WEB PAGES AND INVESTIGATION OF GOOGLE PAGERANK FORMULATION

ABSTRACT

There is no sufficient source either in the academic field or in the current technology market about search engines, though the importance of this area increases rapidly. Especially along with online shopping has become popular recently, the interest to this area enters in the process of a quick development. Each sector wants to run for putting their own web sites to the top, therefore, both the universities and private sectors in the field of information technology have started to make publishing about search engines and train their personnel. Research is released in sections for both producing a regular source and making a deep analyzing for the logic behind of search engines. Whereas the first section focuses on the structure of search engine, the second, third and fourth sections make a study of the reasoning that search engines focus on the process of putting the pages in order. While the second one includes a focus on calculating the content score, the third one, basically, Google Search engine, focuses on calculating the popularity score, which search engines use while putting the pages in order. In the fourth one, researches are made about the components of the formula which is used while calculating the popularity score. Lastly, the fifth one includes results of the research, and a focus on the ideas and interpretations about the future of search engines.

Keywords:

Structure of Search Engines, The Mathematics of Web Search, Content Score, Popularity Score, Google PageRank, Search Engines Optimization

ARAMA MOTORLARI MİMARİSİ, WEB SAYFALARININ İÇERİK SKORU VE GOOGLE PAGERANK FORMÜLÜNÜN İNCELENMESİ

ÖZET

Ülkemizde arama motorlarının önemi hızla artmasına rağmen, maalesef ki hem akademik ortamda hem de güncel teknoloji piyasasında bu alanla ilgili yeterli kaynak oluşturulamamıştır. Özellikle son dönemlerde internet üzerinden alış verişin yaygınlaşmaya başlamasıyla birlikte bu alana duyulan ilgi hızla bir gelişim süreci içine girmiştir. Artık her sektör, web ortamındaki aramalarda kendilerine ait web sayfalarını ilk sıralara koyma yarışına girmişlerdir. Bu yüzdendir ki gerek ülkemizdeki üniversiteler gerekse bilişim alanındaki özel sektörler bu alan ile ilgili yayınlar oluşturmaya ve bireyler yetiştirmeye başlamışlardır. Arama motorları alanında hem derli toplu bir kaynak oluşturmak hem de arama motorlarının derinlemesine çalışma mantığını incelemek için, araştırma bölümler şeklinde sunulmuştur. Birinci bölüm arama motorlarının mimarisi üzerine yoğunlaşırken ikinci, üçüncü ve dördüncü bölümler web sayfalarını sıralarken arama motorlarının hangi mantık üzerine odaklandığını incelemektedir. İkinci bölümde özellikle web sayfalarını sıralarken arama motorlarının kullandığı içerik skorunun hesaplanması üzerinde durulmuştur. Üçüncü bölümde temelde Google arama motoru olmak üzere arama motorlarının web sayfalarını sıralarken kullandığı popülarite skoruna odaklanılırken, Dördüncü bölümde popülarite skorunun hesaplanmasında kullanılan formülün bileşenleri üzerinde durulmuştur. Son olarak beşinci bölümde ise araştırmanın sonuçları ve arama motorlarının geleceğine dair fikirler ve çıkarsamalar üzerinde durulmuştur.

Anahtar Kelimeler:

Arama Motorlarının Mimarisi, Web Arama Motorlarının Matematiği, İçerik Skoru, Popülarite Skoru, Google PageRank, Arama Motorları Optimizasyonu.

ARAMA MOTORLARI MİMARİSİ, WEB SAYFALARININ İÇERİK SKORU VE GOOGLE PAGERANK FORMÜLÜNÜN İNCELENMESİ

ÖNSÖZ

İnsanoğlu tarih boyunca, yaşadığı zorlukların bir sonraki nesil tarafından da yaşanmaması için büyük bir çaba sarf etmiştir. Bu yüzdendir ki atalarından kendisine miras kalan bilgilere yeni bir şeyler daha ilave ederek bir adım daha öteye gidilmesini sağlamış ve bu bilgileri büyük bir inançla bir sonraki nesle aktarmıştır. Nesilden nesile aktarılan bu bilgi miktarı 19.yüzyılın ortalarına kadar düzenli bir artış göstermesine rağmen 19.yüzyılın sonları ve 20.yüzyılın başlarında, internetin gelişmesi ile birlikte, insanoğlunun şimdiye kadar biriktirdiği bilgi miktarının milyonlarca katına ulaşılmıştır. Son 15-20 yıl içinde çağlar boyunca biriktirilen bilgi miktarının milyonlarca katına ulaşılması tabii olarak bilginin erişimi ve yönetimi sorunlarını da beraberinde getirmiştir. Bu ani bilgi miktarının artışını sağlayan ve insanoğlunun gelmiş geçmiş en büyük kütüphanesi olan internetin diğer kütüphanelerden en büyük farkı ise, kimseye ait olmamasıdır. Yani bu devasa kütüphaneye erişim için ya da bilgi girişi için herhangi bir zorunluluğun ya da kimlik sorgulamasının bulunmamasıdır. Bu özelliğinden dolayıdır ki, günümüzün en büyük sorunlarından biri de internet kirliliğidir. Bu kirlilikten dolayı arzu edilen bilgiye ulaşmak tahmin edildiği gibi bir hayli zor olmaktadır. Bu sorunu aşabilmemizi sağlayacak en güvenli ve en pratik yöntem ise, oluşan bu devasa Web çöplüğünde çok iyi çalışan ayrıştırıcılar yani arama motorları geliştirmektir. Bu çalışmanın temel odak noktası da tamda bu ayrıştırıcılar üzerinedir. Yani devasa web çöplüğünde ayrıştırıcılık görevini üstlenen arama motorlarının, yapısı ve çalışma mantığı üzerine yoğunlaşılmıştır. Bu çalışmamda bana yol gösteren, desteğini esirgemeyen ve değerli fikirleri ile çalışmama yön veren ve katkı sağlayan değerli hocam Dr. Hasan DAĞ'a sonsuz teşekkürlerimi bir borç bilirim.

İstanbul 2013 Muhittin IŞIK

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

FORMÜLLER LİSTESİ

BÖLÜM I

1. Giriş

1.1. Arama Motorlarına Giriş

Günümüzde **arama motorları** denilince internet üzerinde bulunan bilgiyi elde etmek için kullandığımız araçlar akla gelir. Oysaki arama motorlarının hem günümüzdeki uygulamaları hem de çıkış noktaları bu tanımı yetersiz kılmaktadır. Çünkü gelinen teknolojiyle birlikte arama motorları birçok alanda kullanılmaya başlandı. Öyle ki kullandığımız kişisel bilgisayarlarda, sağlık sektöründe, eğitim sisteminde ve buna benzer birçok alanda tarama yaparken arama motorlarını bir fiil kullanmaktayız. Bu bağlamda düşünüldüğünde arama motorları bir bakıma çıkış noktalarına geri dönmektedir. Yani World Wide Web ile doruk noktasına ulaşan arama motorları, giderek daha spesifik alanlarda daha profesyonel gelişimler göstermektedir.

Arama motorlarını daha derinden anlayabilmek için arama motorlarının tarihini oluşturan Bilgiyi Geri Getirme (Information Retrival) ya da bir başka deyişle Bilgiyi Elde Etme alanına göz atmakta fayda var. 1960'lardan 1990'lara kadar bu alana liderlik edenlerden birisi olan Gerard Salton, 1968 de Bilgiyi Elde Etme'nin güzel bir tanımını yapmıştır. Şöyle ki,

Bilgiyi Elde Etme; Bilginin yapılandırılması, analiz edilmesi, organizasyonu, depolanması, aranması ve bilginin geri getirilmesi ile ilgilenen bilim dalıdır (Croft, Metzler ve Strohman, 2010: 1).

Günümüz teknolojisi her geçen gün yeni bir boyut kazansa da Salton tarafından bilgiyi elde etme üzerine yapılan bu tanım halen geçerliliğini korumakta, arama motorları ve Bilgiyi Elde Etme alanı için uygun ve eksiksiz bir tanım olarak görülmektedir. "Bilgi" terimi çok geniş bir kullanım alanına sahip olduğu bir gerçektir. Fakat Bilgiyi Elde Etme alanı her çeşit alan ile ilgili bilgiyi aramak ile uğraşır. Yani bir başka deyişle,

Bilgiyi Elde Etme; "Elimizdeki dokümanlardan spesifik bir bilgiyi arama veya elde etme sürecidir" denilebilir.

Bilgiyi Elde Etme alanının geçmişine baktığımızda temel olarak metin ve metin dokümanları üzerine odaklanmıştır. Romanlar, bilimsel araştırmalar, eğitim kitapları, mektuplar, mecmualar ve gazeteler bu dokümanlara verilebilecek birkaç örnektir. Saydığımız bu dokümanlardan bilgiyi elde etmek için kullandığımız basit bir yapı vardır. Bu yapı, yayının adı, yazarı, yayın tarihi ve yazının içeriği gibi birçok parçadan oluşur. İşte bu noktada Bilgiyi Elde Etme alanı belirli bir düzene göre sıralanmış kitaplar, dergiler, dosyalar vb. dokümanların daha hızlı ve daha pratik ulaşılması olarak karşımıza çıkmaktadır. Fakat gelişen teknoloji ile birlikte bu dokümanlara ulaşmak belirli bir mantığa göre sıralamanın ötesinde kendini bilgisayar dünyasına bırakmıştır. Böylece dokümanlarımız tozlu raflarından kurtulup, yeni sanal raflarına kavuşmuşlardır. Dokümanlarımızın ev sahipliğini üstlenmiş sanal raflarımız olan veri tabanları ise, bilgiyi elde etme alanı için çok büyük avantajlar sağlamıştır. Veri tabanlarına aktarılan dokümanlarımızın eski usul yapıları da (sayfanın başlığı, yazarı, yayın tarihi vb.) sanal raflarında özellikler, nitelikler (attributes), alanlar (fields) gibi yapı başlıkları altında kendilerine yer bulmuşlardır.

Gelişen teknoloji ile birlikte bir bilgiyi elde etmek için harcadığımız zaman, özellikle gelişen veri tabanları sayesinde, minimuma ulaşmıştır diyebiliriz. Her geçen gün çeşitli bilim alanları ile ortak çalışılarak geliştirilen modellemeler sayesinde daha hızlı ve kaliteli sonuçlar elde edilmektedir. Başlı başına bir bilim dalı olan ve önemi giderek artan Bilgiyi Elde Etme alanı, Veri Madenciliği ve Veri Ambarlama, Veri Tabanı ve Yönetimi, Bilgi Sistemleri ve Analizi, Linner Cebir, Olasılık ve İstatistik gibi birçok alanı da kanatları altına alarak ilerlemeye devam etmektedir.

1.2. Bilgiyi Elde Etme ve Arama Motorlarının Tarihçesi

Bilgiyi Elde Etme'nin tarihine baktığımızda, kâğıdın icadından öncelere dayandığını fark ediyoruz. Özellikle eski Romalıların ve Yunanlıların bilgilerini papirüslerin üzerine kaydetmeleri bizim için dikkate değer ipuçları sağlamaktadır. Eski Romalılar bazı papirüslerin üzerine etiketler iliştirdikleri söylenir. Bu etiketler ile, ilgili papirüsün içeriği hakkında kısa bir bilgi verildiği gibi, okuyucuları gereksiz yere ilgisiz dokümanları karıştırarak zaman kaybetmeleri önlenmesi amaçlanırmış. Aslında bu durumu ele aldığımızda özellikle Bilgiyi Elde Etme alanının tarihi hakkında çok güzel ipuçları verdiğini söyleyebiliriz. Çünkü mantık olarak ele aldığımızda bu etiketlerin arama motorlarının sonuçlarını kullanıcıya özet şeklinde sunmasına benzer olduğunu düşünebiliriz. Yine aynı şekilde milattan önce II.

yüzyılda Yunanlıların papirüs tomarlarını düzenlemek için içerik tabloları kullandıkları söylenir. Bu içerik tablolarının günümüz arama motorlarında kullanılan veri tabanlarındaki kayıtlara eşdeğer olduğunu düşünebiliriz. Fakat Bilgiyi Elde Etme'nin tarihi hakkında elde ettiğimiz bu bilgiler yazılı olmadığından, sadece yorumlar ve çıkarsamalar üzerinden bir tarih oluşturmaktayız.

Bilgiyi Elde Etme'nin tarihine bakmaya devam edecek olursak, ülkemizde bulunan ve dünyanın en büyük kütüphanesi olduğu söylenen Bergamalılar kütüphanesiyle karşılaşırız. Tarihte yer etmiş Bergama Kütüphanesinin en geniş papirüs tomarlarını topladığı bilinir. Gelin görün ki Mısır Hükümdarı Epiphane papirüsün hammaddesini Bergamalılardan kesmesinin ardından, Bergamalılar alternatif yollar ararlar ve parşömeni icat ederler. Zaten Parşömen (Parchment) kelimesi köken olarak da Bergama (Pergamum) kelimesinden gelmektedir. Bilindiği gibi parşömenler de çok ince hayvan derisinden yapılmaktadır ve papirüslere nazaran parşömenler daha zor rulo haline geldiklerinden kitap yaprakları gibi üst üste dikildiği, kullanımı daha kolay olduğundan zamanla papirüs rulolarının yerini aldığı bilinir. Nihayet parşömenlerin ardından kâğıdın icat edilmesiyle birlikte bilgiyi yazılı olarak kaydetme ve elde edilen dokümanları toplama anlamında hızlı bir gelişme sürecine girilmiştir (Langville ve Meyer, 2006).

İlerleyen yüzyıllarda bu hızlı süreç yazılı basının 1450'li yıllarda Johann Gutenberg tarafından yeniden icat edilmesiyle gelişimini kat be kat artırmıştır. 1700'lü yıllarda Amerika'da Benjamin Franklin teşvikiyle halk kütüphanelerinin kurulması bu durumu takip etmiştir. Halk kütüphanelerinin büyümesi ve halk tarafından ulaşılabilir olması, dokümanları arama konusuna olan ilgiyi artırmıştır. Sonraki yüzyıllarda da kaynakları belli bir hiyerarşiye göre sıralama artarak devam etmiştir. Gelişmeler ilerleye dursun 1940 ve 1950'li yıllarda dijital bilgisayarların icadıyla, bilgisayarlı arama sistemleri kendi amaçları doğrultusunda yavaşça ilerlemeye başlamıştır. Öyle ki İlk bilgisayarlı arama sistemlerinde ilgili kaynağı bulmak için özel bir sentaks kullanıldığı ve kullanıcının sorgulamasına ilişkin bilgi başlığını kullanarak da ilgili kitabı getirdiği bilinir. 1960'lı yıllardaki Cornell SMART sistemi ise ilk bilgisayarlı arama sistemlerine verilebilecek örnektir diyebiliriz (Langville ve Meyer, 2006).

1989'larda kaynakların depolanması, erişilmesi ve aranması Tim Berners-Lee tarafından World Wide Web'in icat edilmesiyle devrim yaşamıştır. Bu sistemde mevcut web sunucularının listesinin sistemde saklanmasıyla aranan bilgiye erişim sağlanıyordu. Tabi ki internetin yaygınlaşması ve web sunucularının artmasıyla bu liste takip edilemez hale gelmiştir. Ardından 1990'lı yıllarda Alan Emtage tarafından geliştirilen Archie ise, internet üzerinden bilginin aranmasında kullanılan ilk araç olarak bilgiyi elde etme tarihinde yerini almıştır. Archie, FTP sitelerindeki dosyaların listesini kullanıcıya sağlarken, site içeriğinin indekslenmesi işlemini gerçekleştiremiyordu. 1993 yılına gelindiğinde ise Matthew Gray, bilinen ilk web robotunu geliştirerek, Perl üzerinden "Wandex" indeksini oluşturmaya başladı. Gray'in temel amacı Wandex'i kullanarak internet büyüklüğünü belirlemekti ve bunda da başarılı oldu diyebiliriz (Dündar, 2009).

1994 yılında Crawler tabanlı tarama özelliği olan ilk ticari arama motoru WebCrawler Washington Üniversitesi'nde geliştirildi. Önceki sürümlerine nazaran tüm kelimeler üzerinde arama imkânı sağlıyordu. Bu özellik ise arama motorlarının en temel standardını oluşturuyordu. Ayrıca internet üzerinden kullanıma açılan ilk arama motoruydu. 1994 yılından sonra ise ardı ardına arama motorları ortaya çıkmaya başladı. Bunlardan en bilinenleri ise Magellan, Excite, Inktomi, Nothern Light, AltaVista ve Yahoo'dur. O vakitler genel anlamda bu arama motorları kelime bazlı ve konu dizinleri şeklinde arama yapıyorlardı (Dündar, 2009).

1998 yılına gelindiğinde ise bilgiyi elde etmede Link Analiz Sistemi kullanılarak arama motorlarının gelişiminde devrim yaşanmıştır. En başarılı arama motorları link analiz tekniklerini kullanarak ve web üzerinde bulunan link bilgilerini de elde ederek, arama sonuçlarının kalitesini arttırmaya başlamışlardır. Web aramaları hızlı bir gelişim gösterirken, web araştırmacıları da titiz bir şekilde Google ve AltaVista gibi arama motorlarını gıpta ederek kullanmaya ve incelemeye başlamıştırlar.

Mayıs 2004'te yapılan bir araştırmada web kullanıcılarının %37'si Google arama motorunu kullanırken, %27'si Yahoo'u kullandığı ortaya çıkmıştır (Langville ve Meyer, 2006). 2008 yılına gelindiğinde ise Hitbox'ın arama motorları üzerine dünya çapında yaptığı bir araştırma da, % 82,7 kullanım oranıyla Google'ın ezici bir farkla rakiplerini geride bıraktığı görülmüştür. Ardından Çin'den gelen rakibi Baidu yüzünden Google 2009 yılında kullanım oranı %78,4 düşerek etkisini az da olsa

yitirmeye başlamıştır (Dündar, 2009). Zamanla diğer rakiplerinin de yoğun çalışmasıyla birlikte bu oran ilerleyen yıllarda daha da düşmeye başlamıştır. Fakat Google günümüzde ilkelerini ve kurallarını sıkı sıkıya uygulayan ve gün geçtikçe geliştirdiği algoritmalar sayesinde, arama motorları arasında birinciliğini hala kimseye kaptırmamıştır.

1.3. Geleneksel Bilgi Elde Etme Modelleri

Bu bölümde Bilgiyi Elde Etme'nin iki temel bölümü olan Web Ortamında Bilgi Elde Etme (Web Information Retrival) ve Geleneksel Bilgi Elde Etme (Traditional Information Retrival) alt başlıklarının farkını özetlemeye çalışacağız.

"Web Ortamında Bilgiyi Elde Etme", internet ortamında birbirine linklenmiş dünyanın en büyük doküman kaynaklarını araştırırken, "Geleneksel Bilgi Elde Etme" daha küçük, daha kontrollü ve linklenmemiş içerik ile uğraşır. Araştırmamızın başında da bahsettiğimiz gibi geleneksel linklenmemiş içerik Web'in geçmişini oluşturur ve günümüzde de halen kullanılmakta ve artarak devam etmektedir. Yani bir kütüphanede yapılan kitap sorgulaması, bir iş yerinin kendine ait veri tabanında yaptığı arama sorgulamaları Geleneksel Bilgi Elde Etme kategorisinde değerlendirilir. Geleneksel Bilgi Elde Etme kategorisindeki uygulamalar daha statik, daha organizeli ve uzmanları tarafından daha kategorizeli oluşturulur. Geleneksel Bilgi Elde Etme'nin geçmişine baktığımızda dokümanlar dosyalar gibi fiziki ortamlarda barındırılırken, günümüzde bilgisayar ve web sayfaları gibi ortamlarda tutulmaktadır. Bu içerikler belirli yazılımlar tarafından sanal makinelerde tutularak sorgulamalara uyacak bir şekilde dizayn edilirler. Geleneksel Bilgi Elde Etme'de temel olarak üç temel arama tekniği kullanılmaktadır. Bunlar, Boolean Modeli, Vector Space Modeli ve Probabilistic Modelidir. Şimdi bu modelleme tekniklerini kısa bir şekilde aşağıda açıklamaya çalışacağız.

1.3.1. Boolean Model Arama Motorları

Boolean arama modeli bilinen ilk arama motoru modelidir ve günümüzde de özellikle kütüphanelerde halen kullanılmaktadır. Ayrıca exact-match retrival (tam eşleşme) ismiyle de anılmaktadır. İsminden de anlaşılacağı gibi sorgulanan kelimenin tam eşleşmesini bulduğunda sorgu sonucunu getirirken, eşleşmediğine herhangi bir şey ekrana getirmez. Boolean cebri, sıralama (ranking) tekniğine göre oldukça basittir. Boolean arama modeli sıralama derecesini kullanmamasının yanında

bütün dokümanları eşit derecede önemli görür. Dediğimiz gibi oldukça basit bir arama mantığına sahiptir. Doğru ve yanlış (True and False) gibi iki sonuç mantığına göre çalışır ve sorgulamayı bu sonuca göre sonlandırır. Ayrıca kelimeleri ararken operatörleri olan AND, OR ve NOT'ı kullanır. Örneğin, AND operatörü x ve y gibi iki kelimenin mantıksal sınamasını yaparken, her iki kelimenin de ilgili dokümanda eşleşmesi gerektiğini düşünür. Fakat OR operatöründe ilgili dokümanda x veya y kelimelerinden herhangi birinin eşleşmesi durumunda sorgulamayı sonlandırır.

Boolean arama modeli birçok arama motorunun temelini oluşturmuştur. Ayrıca bazı yönlerden de avantajlı sayılabilir. Birincisi çok kolay ve kullanıcılar için rahattır. Düz bir mantıkla çalışır. İkincisi sorgulama süreci oldukça kısadır ve paralel sorgulama yapabilir. Son olarak üçüncüsü de çok büyük boyutlu dokümanlarda bile rahatlıkla uygulanabilir olmasıdır.

Boolean arama modelinin bu avantajlarının yanında dezavantajları da vardır. Gerçi bu durum kullanıcının aktifliğine bağlı olsa da, günümüz kullanıcıları için pek de uygun değildir. Gelişmiş bir sıralama algoritmasına sahip olmadığından basit aramalarda bile elverişsiz sonuçlar üretebilir. Bu duruma birer örnek vererek açıklık getirelim.

Örneğin Murat kelimesini araştırdığımızı düşünelim. Dediğimiz gibi Boolean arama modeli gelişmiş bir arama modeli kullanmadığından bütün dokümanları eşit derecede görecek ve karşınıza Murat kelimesi geçen sayısız doküman getirecektir. Karşınıza IV. Murat, Murat 131 arabaları, Kara Murat veya Murat isimli sanatçıların gelmesi olasıdır. Aşağıdaki örneğe bir göz atalım,

Padişah AND Murat

Bu sorgulamada ise içerisinde Padişah ve Murat kelimelerini içeren bütün dokümanları karşınıza getirecektir. Boolean arama modeli insan beyni gibi algılamaya sahip olmadığından şöyle bir sonuçla da karşılaşmanız olasıdır.

"1990 yıllarına damgasını vurmuş Murat 131 arabaları kendi devrinde arabaların Padişahı olarak bilinirdi."

Normalde IV. Murat ile ilgili sonuçları görmeyi beklerken, böyle bir sonucun çıkması Boolean arama modelinde olasıdır. Böyle bir sonuç ile karşılaşmamız,

taranan dokümanların içeriğinde murat ve padişah kelimesinin birlikte geçtiğini gösterir. Bu gibi sonuçlarla karşılaşmamak için kullanıcıların NOT operatörünü kullanması tavsiye edilirdi. Örneğin,

Padişah AND Murat AND NOT (araba OR taşıt)

Bu tarz bir sorgu girilmesi birçok ilgisiz dokümanın silinmesini sağlayabilir. Tabi daha elverişli sonuçlar almak için de sorgu uzatılabilir. Örneğin,

Padişah AND Murat AND Osmanlı AND Biyografisi AND Doğumu AND Vefatı AND NOT (araba OR taşıt)

Şeklinde bir aramada yapılabilir. Fakat Boolean arama modelinde bu tarz bir sorgulamada çok fazla AND operatörü kullanmak bazen hiçbir sonuç döndürmeyebilir. Ayrıca Boolean arama modeli Web arama motorlarındaki kelime sıklığının önemini göz önüne almadığından, bir dokümanda 1000 adet Murat kelimesinin geçmesi bir anlam ifade etmemektedir. Boolean arama modelindeki bu tarz kısıtlamalardan dolayı araştırmacılar farklı arama modelleri geliştirmeye yönelmişlerdir. Buna verilebilecek ilk örnek ise Vector Space modeldir.

1.3.2. Vector Space Model Arama Motorları

Bir diğer arama modelimiz olan Vector Space arama modeli 1960'ların başında Gerard Salton tarafından geliştirilmiştir. Bu modelin avantajı hem basit olması hem de kelime önem derecesini (term weighting), sıralamayı (ranking) ve ilgililik dönütünü (relevance feedback) kullanarak bir çerçeve, başka bir deyişle bir tablo oluşturmasıdır. Yani Vector Space modeli tekst içeriklerini sayısal vektörlere ve matrislere dönüştürür, ardından matris analizini kullanarak bahsettiğimiz değerleri bulur.

Bu modelde dokümanlar ve sorgular "n" boyutlu vektör mesafesinin bir parçası olarak kabul edilir. Burada ki "n" indeks içerisindeki kelimelerin (kelime, kelime grupları, deyimler) sayısını belirtir. Buna göre bir D_i dokümanın indeks terimlerini bir vektör ile göstermek istersek;

$D_i = (d_{i1}, d_{i2}, d_{i3}, \ldots, d_{in})$,

Burada D_{ij}, i. dokümandaki j. teriminin önem derecesini (weigthing) verir. Buna göre elimizdeki n sayıda dokümanın kelimelerinin önem derecesini sunan bir matris oluşturmak istersek;

	$Terim_1$	$Terim_2$	$Terim_3$	. . .	$Terim_n$
Dok_1	d_{11}	d_{12}	d_{13}	. . .	d_{1n}
Dok_2	d_{21}	d_{22}	d_{23}	. . .	d_{2n}
Dok_3	d_{31}	d_{32}	d_{33}	. . .	d_{3n}
.	.	.	.	. . .	.
.	.	.	.	. . .	.
.	.	.	.	. . .	.
Dok_m	.	.	.	. . .	.

Buradaki her bir satır ilgili dokümanı temsil ederken, her bir sütun da ilgili dokümanda bulunan kelimelerin önem derecesini belirtir. Bu yapıyı biraz daha görselleştirmek istersek;

Elimizde 3 adet doküman bulunduğunu düşünelim.

Dok_1 = Bilgisayar Donanım Parçaları ve Bilgisayar Satışları

Dok_2 = Bilgisayar Donanım ve Yazılım Ürünleri

Dok_3 = Teknik Destek Bilgisayar Satışları Masaüstü Bilgisayarlar

Algıda kolaylık sağlaması açısından bu defa dokümanları sütun, kelimeleri satır olarak gösterelim.

Terimler	Dokümanlar		
	Dok_1	Dok_2	Dok_3
Bilgisayar	2	1	1
Destek	0	0	1
Donanım	1	1	0

Masaüstü	0	0	1
Parça	1	0	0
Satış	1	0	1
Ürün	0	1	0
Teknik	0	0	1
Yazılım	0	1	0

Şekilde gösterildiği gibi 3 adet dokümanın basit bir vektör sunumu verilmiştir. Şekildeki satırlar terimleri temsil ederken, sütunlar dokümanları temsil etmektedir. Terim önem derecesi de basit bir şekilde ilgili dokumanda terimin kaç defa tekrarlandığını belirtir. Örneğin Dok_1'in vektörel gösterimi (2, 0, 1, 0, 1, 1, 0, 0, 0) şeklinde olacaktır. Sorgu vektörü de aynı mantıkla oluşturulur. Örneğin, S = ($s_1, s_1, s_1, \ldots, s_x$) şeklinde gösterilir. Buradaki s_j, J. Terimin önem derecesini, başka bir deyişle ağırlığını verecektir. Örneğin "Bilgisayar Donanımı" şeklinde bir sorgu girdiğimizde, vektörümüz (1, 0, 1, 0, 0, 0, 0, 0, 0) olacaktır. Görüldüğü gibi gayet basit bir mantıkla çalışmaktadır. Yani elimizdeki sıralama ile sorgu sıralamasının benzerliğini karşılaştırıyoruz. Daha açık bir ifade ile benzerlik ölçümünü (similarity measure) kullanıyoruz. Buna göre sorgu vektörü ile en iyi eşleşen doküman sıralamada önceliği alacaktır. Birden fazla benzerlik ölçümü, bu değeri ölçmek için kullanılabilir Fakat kosinüs korelâsyonu (**cosine correlation**) bunlar içinde en başarılı olanıdır. Kosinüs korelâsyonu sorgu vektörü ile doküman vektörleri arasındaki açının kosinüsünü ölçer. Bu ölçümde kosinüs korelâsyonunun diğer benzerlik ölçümlerine göre tercih edilmesinin herhangi bir teorik sebebi yoktur fakat sorgu kalitesi bakımından kaliteli sonuçlar üretmektedir (Croft, Metzler ve Strohman, 2010). Bu konuya açıklık getirmek için öncelikle kosinüs korelâsyonunun formülünü verip ardından basit bir örnek yapalım.

$$\cos ine(D_i, S) = \frac{\sum_{j=1}^{t} D_{ij} . S_j}{\sqrt{\sum_{j=1}^{t} D_{ij}^2 . \sum_{j=1}^{t} S_j^2}}$$

(1.1)

Elimizde iki adet doküman olduğunu ve her dokümanda da 3 adet kelime olduğunu düşünelim. Her bir dokümandaki kelimelerin ağırlıkları sırasıyla, D_1= (0.3, 0.5, 0.8)

ve D_2= (0.4, 0.6, 0.9) olduğunu, sorgu vektörünün de S_1= (0.2, 0.7, 0.1) olduğunu düşünürsek;

$$\text{Cosine } (D_1, S) = \frac{(0.3 \times 0.2) + (0.5 \times 0.7) + (0.8 \times 0.1)}{\sqrt{(0.3^2 + 0.5^2 + 0.8^2)(0.2^2 + 0.7^2 + 0.1^2)}} = \frac{0.49}{\sqrt{0.5292}} = 0.7274$$

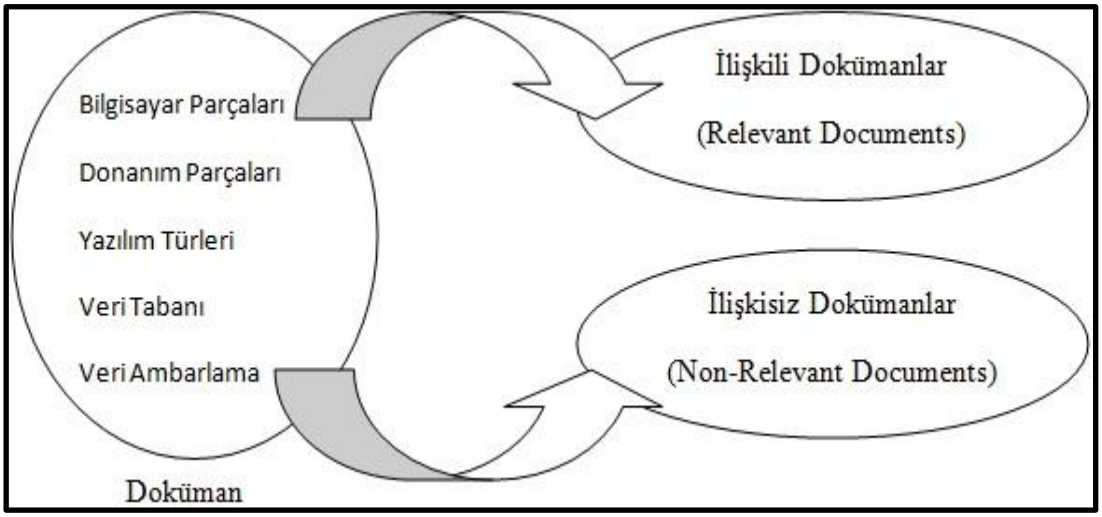
$$\text{Cosine } (D_2, S) = \frac{(0.4 \times 0.2) + (0.6 \times 0.7) + (0.9 \times 0.1)}{\sqrt{(0.4^2 + 0.6^2 + 0.9^2)(0.2^2 + 0.7^2 + 0.1^2)}} = \frac{0.59}{\sqrt{0.7182}} = 0.8474$$

Sonuçtan da görüldüğü gibi D_2 dokümanımız D_1 dokümanına göre daha yüksek bir skora sahiptir. Böylece D_2 vektörümüzün sorgu vektörü ile daha iyi eşleştiği sonucuna varıyoruz. Görüldüğü gibi Vector Space modeli terim önemi mantığını kullandığından Boolean modeline göre daha iyi sonuçlar üretmektedir.

1.3.3. Olasılık Modeli Arama Motorları

Olasılık (**Probabilistic**) modeli daha çok kullanıcıların bir alana mahsus aramalarını tahmin etmek için kullanılır. Yani varsayımlar üzerinde daha net sonuçlar üretmek için tercih edilir. Oysaki Boolean ve Vector Space modelleri eşleşmelerde daha örtük varsayımlarda bulunur. Olasılık modeli tekrarlı (reqursively) bir mantıkla işler. Başlangıç parametresiyle tahminlerde bulunur ve ardından bulduğu bu başlangıç tahminlerinden yeniden (iteratively) tahminler üretir, ta ki ilişki sıralamasını sonlandırana kadar. Ne yazık ki olasılık modeli ile programlama yapmak oldukça güçtür. Bu yüzdendir ki birçok araştırmacı kendini kısıtlanmış olarak düşündüğünden olasılık modeli ile uğraşmaktan kaçınır. Bizler daha çok kompleks insan davranışlarını formülize ettiğimizden dolayı Bilgiyi Elde Etme alanında epey bir zorluk yaşamaktayız. Bu bağlamda Bilgiyi Elde Etme modellerinin geçerliliği, teorik olmaktan çok deneysel olmak zorunda kalmıştır. Bu teorik yaklaşımlardan biri de olasılık modelidir. Olasılık modelinde bir dokümanın sorgu ile ilişkisi doğal olarak diğer dokümanlardan bağımsızdır. Fakat her bir dokümanın sorgu ile ilişki olasılığı birçok olasılık metotlarına göre farklılık göstermektedir. Bu yüzden basit olasılık modelini ele alıp, daha ince ayrıntıların, bu araştırmanın kapsamının dışına çıktığından detaylara girmeyeceğiz.

Bu model de Veri Ambarlama alanında sıklıkla kullanılan sınıflandırma (Classification) tekniği kullanılmaktadır. Yani dokümanlar iki bölüme ayrılır, ilişkili dokümanlar (the relevant documents) ve ilişkisiz dokümanlar (the non-relevant document). Verilen yeni bir doküman, bu mantıkla arama motoru tarafından ilişkili ya da ilişkisiz alanlarına ayrıştırılır (Croft, Metzler ve Strohman, 2010). Bunu yaparken de Bayes sınıflandırma tekniğini kullanır. Şöyle ki, elimizdeki bir dokümana "D", ilişkili olması durumuna A ve ilişkisi olmaması durumuna da B dersek, P(A/D) > P(B/D) ise ilişkilidir denir ve burada P(A/D) koşullu olasılık ve verilen dokümanın ilişki olasılığını gösterir.



Şekil 1.1 Bir Dokümanı İlişkili ya da ilişkisiz Olarak Sınıflandırma (Croft, Metzler ve Strohman, 2010)

Peki, P(A/D)'yi nasıl hesaplayacağız? P(A/D)'yi, P(D/A)'yı hesaplayarak bulabiliriz. Şöyle ki, ilişki setine bakarak hesaplamalar yapılır. Farz edelim ilişki setindeki birkaç spesifik kelime hakkında bilgimiz var. Mesela "Bilgisayar" kelimesinin ilişki setindeki olasılığı 0,05 ve Donanım kelimesinin ilişki setindeki olasılığı 0,07. Eğer yeni doküman "Bilgisayar" ve "Donanım" kelimelerini içeriyorsa, gözlemlenen olasılık kelimelerin değerlerinin kombinasyonu olacaktır. Yani 0,05 x 0,07 = 0,0035 olacaktır. Bayes kuralından bilindiği gibi formülümüz;

$$P(A/D) = \frac{P(D/A)\,x\,(P(A)}{P(D)} \qquad \textbf{(1.2)}$$

Böylece ilişki seti kararımızı belirleyebiliriz. Eğer P(D/A) x P(A) > P(D/B) x P(B) ise verilen dokümanı "ilişkili" setinde gösterebiliriz.

1.3.4. Meta Model Arama Motorları

Temel anlamda gördüğümüz temel üç arama motoru modelinin dışında bir de meta model arama motorları (Meta-Search Engines) vardır. Fakat bu arama motoru temel anlamda kendine has bir model içermemektedir. Meta model arama motorları daha önce bahsettiğimiz üç klasik arama motoru modelinin birleşiminden oluşur. Çalışma mantığına gelince, eğer bir arama motoru belirli bir alanda iyi ise bir diğeri bir başka alanda iyidir. Öyleyse bu üç arama motorundan sorguladığımız sorguya göre en iyi sonucu vereni alıp kullanıcıya sunabiliriz. Böylece kullanıcıların takdirini daha fazla alabiliriz. Meta model arama motorlarına örnek vermek gerekirse, www.copernic.com ve www.surfwax.com gibi arama motorları birçok bireysel arama motorlarının en iyi özelliklerini kullanarak sonuçlarını üretir. Meta model arama motorları bahsettiğimiz gibi sorguyu ilk önce birden fazla arama motoruna gönderir. Ardından gelen sonuçları değerlendirerek en iyi sonucu döndürecek şekilde uzun birleştirilmiş bir listede sunar (Langville ve Meyer, 2006). Ayrıca Meta model arama motorları belli alanlara özel (subject-spesific) arama motorlarını da barındırırlar. Böylece spesifik bir alanda sorgulama yapılmak istenirse daha sağlıklı sonuçlar üretilmesi sağlanır. Örneğin www.monster.com buna verilebilecek güzel bir uygulamadır.

Ayrıca bu anlattıklarımızın dışında da arama modelleri vardır. Örneğin Ranking Based on Language models, Complex Queries and Combining Evidence gibi faklı çalışma prensipleri ve algoritmaları olan arama modelleri de vardır. Fakat daha ilerisi araştırmamızın kapsamı dışına çıktığından bu modellerin anlatımına girilmeyecektir.

1.4. Web Ortamında Bilgi Elde Etme

Arama motorları bilgiyi elde etme yöntemlerinin çok büyük dokümanlar üzerindeki pratik uygulamalarıdır. 1989'da Tim Berners-Lee'nin World Wide Web'i Bilgiyi Elde Etme dünyasına kazandırdığından beri Web Ortamında Bilgiyi Elde Etme tamamen spesifik bir uğraş alanı olup Geleneksel Bilgiyi Elde Etme 'den ayrılmıştır. Her ne kadar web arama motorları temel olarak Geleneksel Bilgiyi Elde Etme modellerini kullansalar da birçok açıdan farklılıklar gösterir. Bu farklılıklara değinmeden önce birkaç konuya açıklık getirelim.

Günümüz arama mimarileri genel olarak iki temel ilke üzerine kuruludur. Bunlar "kalite" ve "Hız" ilkeleridir. Arama mimarilerinden de beklentimiz bu iki ilke

üzerinden sistemin gerekliliklerini ve amaçlarını olabildiğince karşılamasıdır. Bu iki temel ilkeyi açıklamamız beklenirse:

Kalite (Effectiveness-Quality): Olası bir sorgulama için, eldeki kaynaklardan olabildiğince bizim istediğimiz sonucu vermesidir.

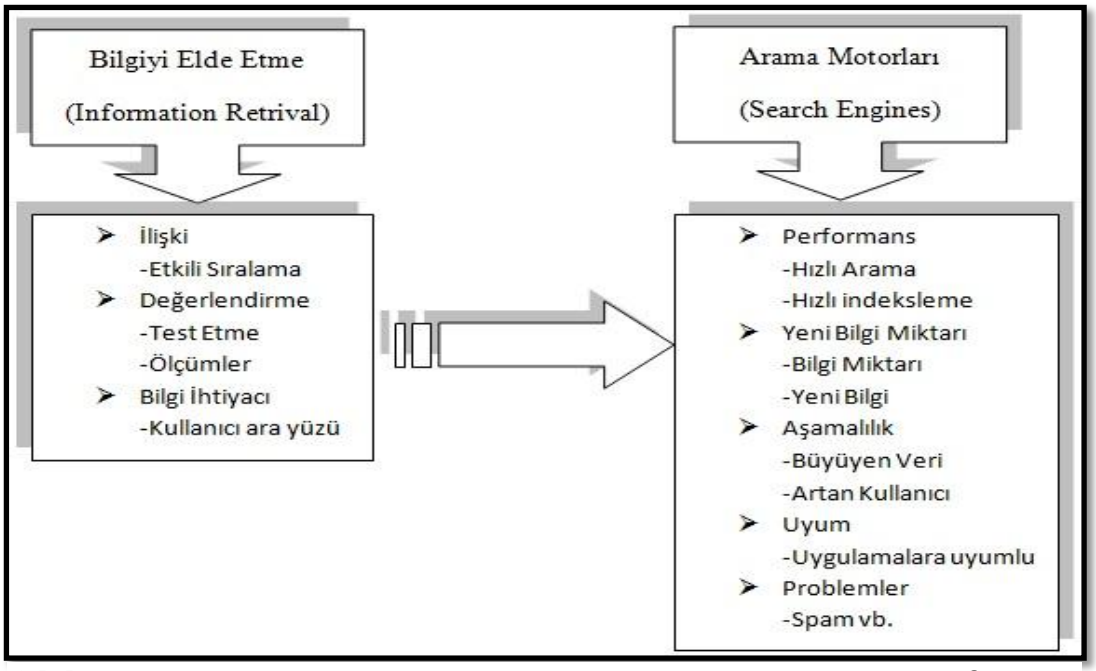
Hız (Efficiency-Speed): Sorgulama sürecinin olabildiğince kısa sürmesidir.

Ayrıca bunun dışında da arama mimarilerinden beklentilerimiz olabilir. Fakat bu beklentilerimizde bahsettiğimiz bu iki temel ilke ile her zaman bağlantılı olacaktır. İsteğimiz, yaptığımız bir sorgulamanın arama motoru tarafından olabildiğince hızlı ve istediğimiz sonucu vermesidir. Arama mimarileri de yapılan bir sorgulamanın mümkün mertebede kaliteli ve hızlı sonuç vermesi için indeksleme işlemlerini olabildiğince titiz yapması gerekir.

Bu kriterlere bağlı kalarak arama motorları, bilgiyi elde etme alanının da içinde yer alan birkaç durumun üstesinden gelmesi gerekir. Bunlar etkili bir "*sıralama algoritması*", "*değerlendirme*" ve "*kullanıcı ara yüzü*"dür. Bununla birlikte *performans* ölçüm değerlerimiz olan, "*cevaplama süresi* (response time)", *sorgulama miktarı* (query throughput), *indeksleme hızı* (indexing speed)'nın da etkili olması beklenir (Croft, Metzler ve Strohman, 2010). Buradaki cevaplama süresinden kastımız, sorgulamanın girildiği süre ile sonuçların gösterildiği süre arasında geçen zaman, sorgulama miktarından kastımız, verilen bir zamanda işlenen sorgu miktarı, indeksleme hızı ise, dokümanların sorgulamaya hazır olması için indeksleme bölümüne ne kadar sürede dönüştürüldüğüdür. Zaten bilindiği gibi mimarisi çok iyi dizayn edilmiş bir indeks bölümü, sorgulamanın hızını ve kalitesini etkileyecektir. Diğer bir önemli performans özelliği ise yeni bilgilerin ne kadar hızlı indeksleme bölümüne birleştirildiğidir. Var olan bilgiler ne kadar köklü, çok (Coverage) ve yeni (freshness) ise arama motorunun kalitesi de o derece artacaktır.

Arama motorlarında bir diğer önemli konu ise aşamalık (scalability) tır. Yani arama motorunun artan veri ve kullanıcı miktarına göre ihtiyaca cevap vermesi gerekir. Başka bir deyişle arama motorlarında birçok uygulamanın birçok görevi yerine getirebilir durumda olması gerekir. Ve ayrıca arama motorumuzun indeks yapısı, algoritması ve ara yüzü de birçok uygulamaya uyum (adaptable) göstermesi gerekir.

Son olarak da bir arama motorunun iyi bir spam belirleyicisi olması gerekir. Spamdan kastımız bazen istenmeyen mailler olurken arama motorları için indeksimiz ile alakasız bilgi diyebiliriz. Özellikle web arama motorlarının başlıca sorunu, spamlar ile mücadele etmektir. Çoğunlukla ticari amaçlı olan bu spamlar bazen tamamen sistemi çökertmeye yönelik olabilir. Son dönemlerde sıklıkla karşılaştığımız kelime spamları, şahsi web adreslerinin arama motorlarında üst seviyelere çıkmak için, içerikte alakasız veya gereksiz yere içeriği kelime bombardımanına tutmasıdır. Kullanıcılar ilgili sitelere girdiklerinde aradıkları konuyla ilgili sitede sadece ilgili konunun başlığını bulması ya da hiç karşılaşmaması (zemin rengi üzerine aynı renk yazı gizlemesi gibi) bu duruma örnek gösterilebilecek durumlardır. Arama motorlarının bu tarz spamlara yönelik pek şansı olmasa da yeni geliştirilen algoritmalar ve kullanıcılardan istenen şikâyet iletileri ya da en basit çözümle, ilgili siteye tıklandıktan sonra geçirilen süre (girdiğimiz sitenin boş veya alakasız olduğunu görüp kısa sürede siteden ayrılmamız) bu sorunun üstesinden gelmek için verilen mücadele yollarından bir kaçıdır. Tabi bu arada bu tarz spamların arama motorlarının kalitesini düşürdüğü için, arama motorlarının tepkisi de çok sert olmaktadır. Google gibi arama motorları bu tarz siteleri belirlediğinde bir daha kendi arama motorunda sıralamaya almamaktadır. Bu durumda doğal olarak ticari sitelerin gözünü korkuttuğundan bir nebze de olsa web site yöneticilerinin bu tarz yöntemlere başvurmalarını önlemektedir. Konuyu toparlamak açısından Bilgiyi Elde Etme ile Arama Motorları arasındaki ilişkiyi bir şekille belirtmek istersek.



Şekil 1.2 Bilgiyi Elde Etme ile Arama Motorları arasındaki ilişki (Croft, Metzler ve Strohman, 2010)

Web arama motorlarının birçok açıdan geleneksel arama motorlarından farklılık gösterdiğini söylemiştik. Çünkü web;

- Çok daha büyük,
- Dinamik,
- Link Yapılı,
- Kendi Kendine Organizelidir.

Hepimizin bildiği gibi web içeriğinin artması o kadar hızlı gelişmektedir ki, boyutunu ölçmek imkânsız hale gelmiştir. Şu an itibariyle içeriğin miktarı hakkında diyebileceğimiz tek şey büyük hem de çok büyük olduğudur. 1994 yılında World Wide Web solucanı 110 bin web sayfasını dizinlerken, 1997 yılında 100 milyon web sayfasını dizinlediği iddia ediliyordu. Bu sayı 2000 yılında ise 1 milyar üzerine çıktığı bilinir. Bununla birlikte arama motorlarındaki sorgulamalar da orantılı olarak artmıştır. Örneğin 1997 Kasım'da Alta Vista'da günlük 20 milyon sorgu girildiği bilinir (Sezgin, 2009). Bu bağlamda geleneksel arama motorlarının sorguladığı içerik miktarı ile web arama motorlarının sorguladığı içerik miktarı kıyaslanamayacak düzeydedir.

Web'in dinamik özelliğine gelince, geleneksel arama mimarilerinin etkilendiğinin kat be katı fazlası etkilenir. Geleneksel arama mimarilerinde eklenen birkaç dokümanın içeriği etkilemesi pek olası görülmez. Fakat Web'e geldiğimizde, Junghoo Cho ve Hector Garcia-Molina'nın 2000 de yaptığı bir araştırma da bütün web sayfalarının % 40'ı haftada bir, ".com" sitelerinin ise %23'ünün günlük değiştiğini göstermiştir (Langville ve Meyer, 2006). Düşünün örneğin ülkemizde Sağlık Bakanlığı kendi veri tabanına günlük en fazla ne kadar bilgi girebilir ki? Ama web dünyasına geldiğimizde ise sadece bireysel kullanıcıların kendi blog sayfalarına günlük hem yazı, hem fotoğraf hem de video eklediğini ve bunun dünya üzerinde bulunan 7 milyarın üzerindeki insanın sadece %1'nin bu işlemi günlük gerçekleştirdiğini düşünürsek, korkunç rakamlara ulaşacağımız olasıdır.

Web arama motorlarının işini zorlaştıran bir diğer durum ise linkli yapısıdır. Bir web sayfasından bir başka web sayfasına gidişimizi kolaylaştıran linkler, arama motorlarının kalitesi için çok önemli bir faktördür. Çünkü araştırılan konu ile ilgili bir web sayfası, yine kendi üzerinden araştırılan konu ile ilgili bir başka sayfaya link vermektedir. Bu durumda web sayfaların kategorizeleştirilmesini ve arama

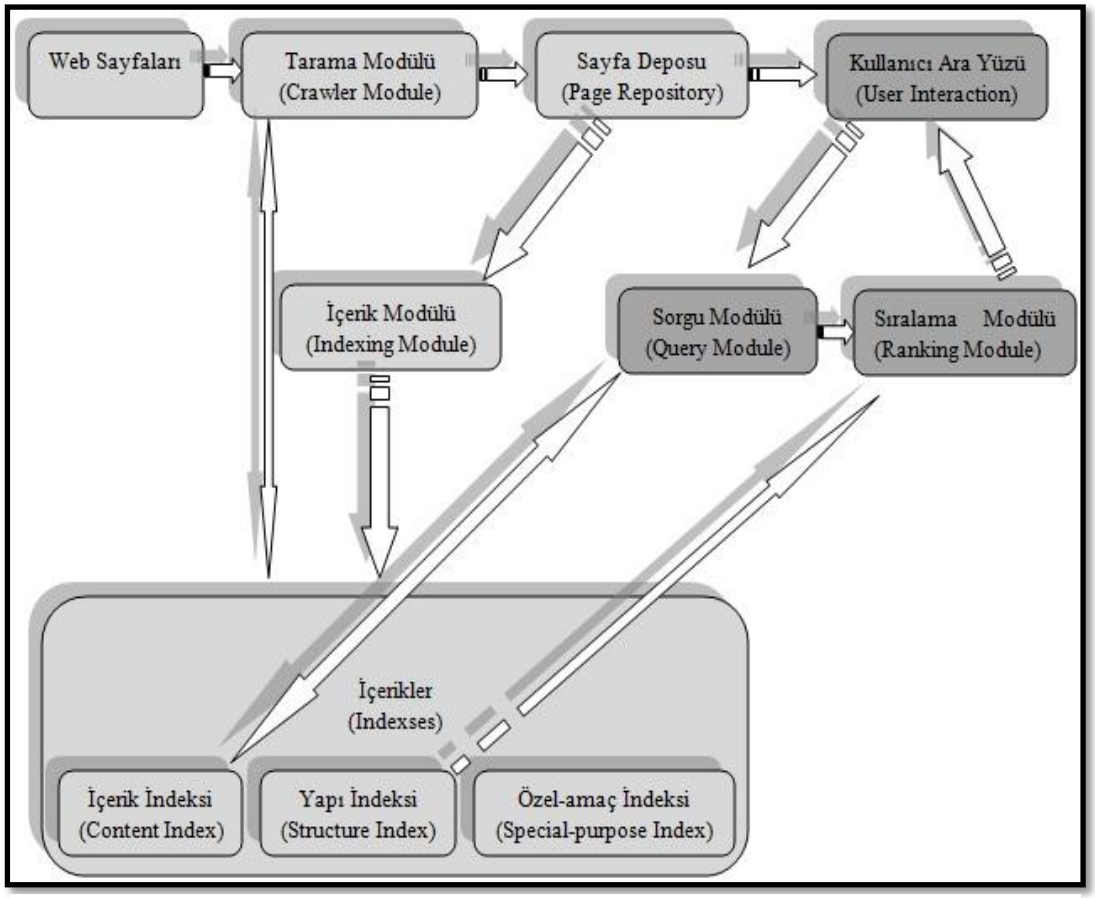
mimarilerinde kullanılan sıralama algoritmasını sağlamada büyük katkı sağlamaktadır. Fakat link üzerinden oluşturulan sıralama algoritmaları da yine spam sorunları ile karşı karşıya kalmıştır. Link spamları, arama motorlarının algoritmasını çözümleyip, oluşturulan mantığa göre kendi web sayfalarına link aktarmaktadırlar. Bu durum ise ister istemez web arama motorlarının işini bir hayli zora sokmaktadır.

Web arama motorlarının işlerini zorlaştıran sorunlardan biri de web'in içsel mekanizması, bir başka deyişle Web'in kendi kendine organizasyonudur. Geleneksel arama motorlarında eğitilmiş uzmanlar tarafından sınıflandırılan, oluşturulan, sorgulanan içerikler, web dünyasında tamamen sahipsizdir. Herkes istediği şekilde mail atabilir, içerik oluşturabilir, link verebilir düzeydedir. Bu durumda doğal olarak kaos ortamını yaratmaktadır. Özellikle günümüzde sıklıkla bahsettiğimiz *internet kirliliği* sorununun ortaya çıkmasına neden olmuştur. Çünkü web içerik ortamı için herhangi bir giriş izni, standart, içerik yasası, yapı veya format olmadığı için, herkes dilediği biçimde hareket edebilmektedir. Böylelikle düzensiz içerikler, alakasız veya kırık linkler, varmış gibi duran içi boş dosyalar, resimler, videolar vb. bir çöplük yığını gibi artmaktadır. Hal böyle olunca web arama motorlarının işi bir hayli zorlaşmaktadır tabi. Özellikle daha önceden de bahsettiğimiz spam içerikler de bu sorunun tuzu biberi olmaktadır.

1.5. Web Arama Sürecinin Temel Taşları

Bu bölümün son parçası olarak web ortamında bilgi elde etmeninin temel taşlarını kısaca ele alacağız. Web arama motorları günümüzde karşılaştığı problemlerden dolayı devamlı olarak algoritmalarını değiştirmektedirler. Önceleri içeriğe (content score) göre sıralanan web sayfaları, ardından popülerliğine (popularity score) göre sıralanmıştır. Fakat her iki yönteminde zamanla deforme olması bir başka deyişle spamlarla mücadele edememesi, bu her iki yöntemin birleştirilme fikrini doğurmuştur. Yani günümüzde arama motorlarının çoğu hem içerik skorunu hem de popülerlik skorunu kullanarak "*kapsamlı skoru* (overall score)" elde etmektedir ve bu sonuca göre web sayfalarını sıralamaktadırlar. Aşağıdaki bölümde bu skorların oluşmasını sağlayan ya da etkileyen temel taşların açıklaması verilecektir. Ardından gelen bölümde ise içerik skorunu oluşturan süreçler aşamalı olarak anlatılacaktır. Bu bölümden sonraki bölümlerde ise popülarite skorunu oluşturan *"Sayfa Değeri"*ni, yani günümüzde artık sıklıkla duyduğumuz *PageRank Değeri*'nin matematiksel hesaplama formülleri üzerinde durulacaktır.

Arama motorları mimarisi iki temel süreçten oluşur. Bunlar *içerik süreci* (indexing process) ve *sorgulama süreci* (query process) dir. İçerik bölümü aramanın yapılabilmesi için yapıyı inşa ederken, sorgulama süreci de sorgulama için bu yapıyı kullanır. Kullanıcı da sorgusuyla bu dokümanlardan sorgusuna en uygun sıralama listesini elde eder. Aşağıdaki şekilde görüldüğü gibi yapı iki bölüme ayrılmıştır. Birinci bölüm (açık renkli kutucuklar) sorgulamadan bağımsız (query-independent)



Şekil 1.3 Bir Arama Motorunun Bölümleri (Langville ve Meyer, 2006)

çalışırken ikinci bölüm (koyu renkli kutucuklar) kullanıcının girdiği değere göre, yani sorguya bağımlı bir şekilde çalışmaktadır.

Tarama Modülü (Crawling Module) : Bu bölüm öyle bir yazılım barındırır ki, bu yazılım web dokümanlarını tarar ve gerekli bilgileri toplayıp kategorize eder. Bu işi yaparken kendine çeşitli alanlarda çalışabilecek örümcekler (spiders) üretir. Tarama Modülü bir sonraki bölüme bilgileri aktarmak için bu verileri devamlı toplar. Bu veri deposunun içeriği tekst (text) ve data bilgilerinden (metadata) oluşur. Data bilgileri tekstin içeriğinden çok, ne tür bir doküman (resim, mail, pdf vb.) olduğu, dokümanın uzunluğu, dokumanın oluşturulma tarihi ve buna benzer bilgileri barındırır.

Sayfa Deposu (Page Repository) : Bu bölüm tarama bölümünden gelen malzemeleri karşılar. Yani örümceklerin taradığı web sayfaları bu bölümde depolanır ve indeksleme modülüne gidene kadar bu bölümde kalırlar. Fakat indeksleme bölümüne gitmeden önce bu bölümde bir dizi işlemlerden geçerlerler. Bu işlemlerin temel adımlarına daha sonra değineceğiz. Kısaca değinmek gerekirse, bu bölüme gelen dokümanlar aramada kullanılmak için kelimelere (index terms) ayrılır. Ayrıca belirli yönlerden de özellikleri (features) tanımlanır. Özellikten kastımız indeks terimlerinin deyim, insan isimleri, tarihler ve sayfada gecen linkleri belirtiyorsa, bu bilgilerin tutulmasıdır.

İndeks Modülü (Index Module) : İndeks modülü sayfa deposundan aldığı sıkıştırılmamış içeriği, yani web sayfalarını, belli indekslere gönderilmek üzere sıkıştırır. Bu bağlamda indeks modülünün aramada rahatlık sağlaması için hem zaman açısından hem de alan oluşturma açısından etkili olması gerekir. İndeks modülü indeksler bölümünü oluşturduğu için, bir düzen içinde çalışması gerekir. Yani indeks modülü indeks bölümünü oluştururken yeni gelen sayfaların veya var olan sayfaların güncellemelerini hesaba katarak indeks bölümünü oluşturması gerekir.

İndeksler (Indexes) : İndeks bölümü web sayfaları hakkında sıkıştırılmış bilgileri saklar. Birçok çeşit indeks türü vardır. Örneğin içeriği, terimleri, başlıkları vb. tutan "içerik indeksi (content index)", ki bu içerikler sıkıştırılmış formda dönüştürülmüş dosyalarda (inverted file) bulunur, ilgili web sayfasında geçen link bağlantılarını ve link yapılarını sıkıştırılmış biçimde tutan "yapı indeksi (structure index)" ve özel amaçlar için oluşturulmuş "özel amaçlı indeksler (special-pupose indexes)" bulunur. Özel amaçlardan kastımız, bazen sorgulamalarda faydalı olabilecek resim indeksleri, video indeksleri, pdf indeksleri gibi içeriklerdir.

Kullanıcı Ara Yüzü (User Interaction): Kullanıcı ara yüzü, arama motoru ile kullanıcı arasındaki bağlantıyı kuran bölümdür. Bu bölüm kullanıcının sorgusunu alıp dönüştürerek sorgulamanın gerçekleşmesini sağlar. Ayrıca bu bölüm sıralama listesinin gerçekleşmesinden sonra kullanıcıya dökümün sunulmasını tedarik eder. Kullanıcı ara yüzü girilen sorgunun daha kaliteli gerçekleşmesi için, girilen sorguyu arıtma teknikleri (kırpma, silme, birleştirme gibi) kullanarak bir sonraki bölüme aktarır.

Sorgulama Modülü (Query Module): Sorgulama modülü kullanıcıdan aldığı girdiyi, arama motorunun anlayacağı dile çevirerek sorgunun cevaplanmasını sağlar. Bunu yaparken çeşitli içerik indekslerine başvurur. Mesela içerik indeksine başvurur ve dönüştürülmüş dosyalardan sorgu ile ilgili web sayfalarını bulur. İlgili web sayfaları bulunduktan sonra, bu web sayfalarını sırlama modülüne göndererek sırlamanın gerçekleştirilmesini sağlar.

Sıralama Modülü (Ranking Module) : Arama motorlarının en can alıcı bölümü olan sıralama modülü, sorgulama modülünden kendisine gelen ilgili web sayfalarını en ilgiliden en ilgisize doğru sıralayarak, kullanıcı ara yüzüne gönderir. Bu görevi yerine getirirken olabildiğince etkili yani kaliteli ve hızlı olması gerekir. Bir arama motorunun hızlı ve kaliteli olması için, ilk başta iyi bir indeksleme tekniği ve iyi bir sıralama algoritması kullanması gerekir. Sıralama modülü de bunu gerçekleştirmek için bilgiyi elde etme modellerini kullanarak aldığı web sayfalarını sıralar. Bu sıralamayı yaparken daha önce de bahsettiğimiz, içerik skoru ve popülarite skorunu kullanır. Bu iki skorun birleşiminden oluşturduğu kapsamlı skorun sonucunu da bir liste halinde kullanıcının rahatlıkla algılayabileceği bir yapıya dönüştürür.

1.6. Web Temel Taşlarının Bileşenleri

1.6.1. Tarama Modülü (Crawling Module) :

Şekil 1.4 Web Temel Taşlarının Bileşenleri

Tarayıcı (Crawler) : Tarayıcılar arama motorları için içerik toplayan bileşenlerdir. Başlıca görevi dokümanları belirleyip, taramak olan tarayıcılar çeşitli türlerde olabilirler. Bu türlerden en bilineni web tarayıcısı (web crawler) dır. Web tarayıcılarına, web robotu, web örümcekleri gibi isimlerde verilmektedir. Bir web tarayıcısı web sayfalarını gezerken karşılaştığı linkleri takip ederek sayısız sayfaya ulaşabilir. Ulaştığı sayfalar mimaride zaten bulunuyorsa güncellemelerini tararken, yeni karşılaştığı web saylarını ise ayrıntılı tarayarak, mimariye gönderir. Tarayıcılar belli amaçlara yönelik programlanabilirler. Yani sadece belli siteleri taramak için

görevlendirilmiş tarayıcılar bulunurken, sadece belli konuları takip etmek için programlanmış tarayıcılar da bulunmaktadır.

Tedarikçiler (Feeds) : Bu mekanizma devamlı değişim içinde bulunan dokümanların takibi için kullanılır. Örneğin haber bültenleri devamlı bir değişim içinde olduğundan, yeni gelen haberin sisteme aktarılması için tedarikçiler kullanılır. RSS'ler bunlara verilebilecek en güzel örnektir. RSS'ler web tedarikçilerinin standartlarını kullanarak haberlere, bloglara, videolara ulaşır. RSS tedarikçilerinin temeli XML üzerinedir. Yani XML kullanılarak oluşturulur.

Dönüştürücü (Concersion) : Tarayıcılar tarafından taranan içerikler belirli formatlarda gelir. Bu formatlar HTML, XML, Word, Excell, Pdf gibi format türlerinde olabilirler. Fakat bunların arama mimarisine alınabilmesi için belirli formlara dönüştürülmesi gerekir. Elde edilen bu içerikler dönüştürücü sayesinde tekstsel içeriklere dönüştürülür. Özellikle Word ve Pdf gibi içerikler taramaya uygun olabilmesi için dönüştürücüler tarafından arama mimarisinin diline dönüştürülür.

Doküman Veri Deposu (Document Data Store) : Doküman veri deposunda toplanan içerik hakkında bilgiler depolanır. Bu bilgiler data bilgileri olduğu gibi, link bilgileri, çapa (anchor) bilgileri de olabilir. Yani bu bölümde elde edilen içeriğin, resim, mail, pdf gibi yapılardan hangisi olduğu belirlenir ve bu bilgi bu bölümde saklanır. Bu bilgiler daha sonra sorgulama anında faydalı görülürse kullanılırlar.

1.6.2. Sayfa Deposu (Page Repository) :

Ayrıştırıcı (Parser) : Ayrıştırıcılar gelen dokümanın içeriklerini bölümlere ayırır. Arama kalitesi için çok önemli bir görev üstlenen ayrıştırıcılar, gelen içeriğin başlıklarını, alt başlıklarını, linkleri vb. belirler ve sorgulamaya hazır hale getirir. Ayrıca indeks içindeki tekst sembollerini (tokens) de belirlenen mantığa göre ayrıştırır. Bu sembollerden kastımız büyük harf, küçük harf, virgül, tire, tırnak gibi işaretlerdir. Mesela aramada "Bilgisayar" kelimesi ile "bilgisayar" kelimesini aynı kelime gibi mi algılayacak? Ya da "dünyanın" kelimesi ile "Dünya'nın" kelimesindeki tırnak işaretini nasıl değerlendirecek? Bu gibi durumlar arama motorları için çok büyük önem arz etmektedir. Özellikle kelime tabanlı arama mantığında olan mimariler birleşik kelimeleri, deyimleri, "-" işareti ile birleştirilmiş kelimeleri neye göre sınıflandıracak? İşte ayrıştırıcılar bu nokta da dilin yapısına göre programlanmaktadırlar.

Ayrıca dokümanlar HTML, XML gibi formatlarda gelince, ayrıştırıcının işi bir nebze kolaylaşmaktadır. Çünkü belirli etiketler arasına alınan içerikler, indeksi parçalamada kolaylıklar sağlamaktadır. Mesela <h1> Bilgisayarların Tarihçesi<h1> gibi H1 etiketi arasına alınan içeriğin ana başlık olduğu rahatlıkla belirlenir.

Silici (Stopping) : Bu bölümde arama motorlarının sıralamasını etkilemeyen kelimelerin atılması sağlanır. Örneğin "ve", "bir", "tek", "çok", "için" gibi kelimeler sıralama listesini pek etkilemezler. Bu gibi kelimelerin atılması arama süresinin daha hızlı gerçekleşmesini sağlar. Bu atılacak kelimeler listesi pek uzun değildir. Kendi dilimiz için düşündüğümüzde belki 70 ya da 80'ni geçmeyecektir. Fakat bazen bu silme işlemi bazı aramalara engel olabilir. Mesela hepimizin bildiği "Bir bir biri biri birine, bakar bakar bakar dururum" parçasını aramak epey zor olacaktır. Ya da İbrahim Tatlıses'in "Tek tek" parçasını aramak güç olabilir. Bu ve buna benzer durumlardan dolayı bazı arama motorları bu işlemi es geçmektedir.

Kök Bulucu (Stemmer) : Kök bulucular arama motorlarının daha hızlı işlem yapması için, kelimelerin eklerini atarak kök kısmını bulurlar. Örneğin "göz", "gözlük", "gözlükçü" gibi kelimelerin hepsini bir kategori altında toplar. Yani kelimenin kökü olan "göz" kelimesini alır. Bu tür bir işlem bazı kelimeler için uygun olsa da örneğimizde olduğu gibi bazen arama kalitesini düşürebilir. Gözlükçü ararken göz ile ilgili bir listenin kaşımıza çıkması gerçekten sinir bozucu bir durumdur. Bu işlem bazı diller için tamamen fiyaskoyla sonuçlanabilir. Özellikle Arapça gibi kelimeye gelen eklerin birçok anlam değişikliğine yol açtığı diller için uygun değildir. Keza Çince gibi diller için de pek uygun olmayabilir. Çünkü zaten kelimeler çoğunlukla tek heceden oluşmaktadır.

Link Analiz (Link Analysis) : Link analiz ile web sayfalarında geçen link ve çapa (anchor) tekstler ayrıştırıcı eşliğinde belirlenir. Daha sonra bu bölümler daha önce bahsettiğimiz doküman veri deposuna kaydedilir. Bu link bilgileri daha sonra popülarite skorunun belirlenmesinde kullanılır. Çapa tekstleri de web sayfasının kendi içindeki hareketlerini gösterir.

Bilgi Çekme (Information Extraction) : Bu bölümde ise elde edilen içerik terimlerinin özellikleri belirlenir. Özellikten kastımız terimin isim, sıfat, fiil vb. mi olup olmadığına karar verilir ve bu bilgiler daha sonra kullanılır. Bu işlem ile özel isimleri, yer isimleri, şirket isimleri gibi önem arz eden bilgiler elde edilmiş olur.

Sınıflayıcı (Classifier) : Sınıflayıcı ile elde edilen dokümanın ne ile ilgili olduğu belirlenir. Genellikle konulara göre (Haber, müzik, spor vb.) sınıflandırma yapılır. Ayrıca bu işlem ile dokümanın spam olup olmadığı, reklamdan ibaret olup olmadığı da belirlenmiş olur.

1.6.3. İndeks Modülü (Index Module) :

Doküman İstatistikleri (Document Statistics) : Bu bölüm elimizdeki dokümanlar hakkında istatistikî bilgileri barındırır. Bu bilgiler kelimeler hakkında, kelime özellikleri hakkında ya da dokümanın tamamı hakkında olabilir. Mesela bir kelimenin dokümandaki frekansı veya aramalarda bir dokümana kaç defa başvurulduğu, dokümanın boyutu, dokümanın konusu gibi hem içerik skorunda kullanılabilecek hem de popülarite skorunda kullanılabilecek bilgiler bu bölümde yer alır.

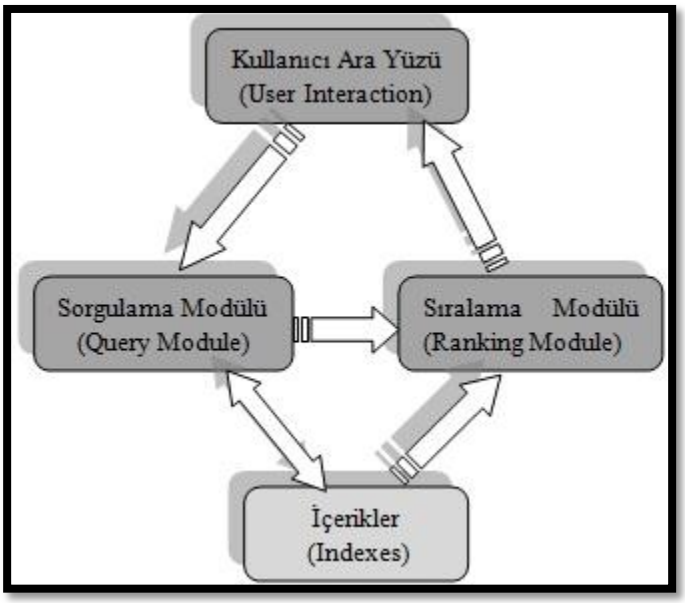
Önem Derecesi-Ağırlık (Weighting) : Bu bölümde ise doküman da geçen kelimelerin ağırlıkları hesaplanır ve bu sonuç doküman istatistiklerine kaydedilir. Kelimelerin ağırlıkları hesaplanırken daha önce bahsettiğimiz bilgiyi elde etme yöntemleri kullanılır. Elde edilen sonuçlarda daha sonra sıralama algoritmasında kullanılır. Kelime ağırlıklarını belirlemede en çok kullanılan yöntem *tf.idf* yöntemidir. Tf (the term frequency), bir dokümandaki kelimelerin frekansları belirtirken, idf (inverse document frequency) ise bütün dokümanlardaki kelime frekanslarını belirtir. Böylece kelimelerin ağırlıklarını bulmak için logN/n formülü kullanılır. Buradaki N, arama motorundaki bütün dokümanları temsil ederken, n sadece belli terimleri içeren dokümanları temsil eder (Croft, Metzler ve Strohman, 2010).

Çevirme-Dönüştürme (Inversion) : Bu bölümde yeni bir doküman geldiğinde veya var olan bir sayfanın güncellemesi geldiğinde, çevirmeleri yani dönüşümleri sağlar. Tarayıcıdan (crawler) gelen dokümanı alıp, indeks düzenine sokar. Bu işi yaparken hızlı ve kaliteli bir sorgu için epey dikkatli bir ayarlama yapması gerekir.

İndeks Dağıtımı (Index Distribution) : Bu bölümde ise arama mimarisinde bulunan dokümanlar sadece bir bilgisayara sığdırılamayacağından dağıtılır. Bunu yaparken ağ üzerinden birden fazla bilgisayara ulaşır ve kaydederken de sorgularken de bu şekilde çalışır. Hem indeksleme işleminin hem de sorgulama işleminin hızlı

gerçekleşmesi için paralel programlama teknikleri kullanılır. Böylece iş paylaşımı sağlanırken, çok daha hızlı ve etkili sorgulamalar sağlanır.

1.6.4. Kullanıcı Ara Yüzü ve Sorgulama Modülü (Query Module):



Şekil 1.5 Sorgulama Modülünün Çalışma Mantığı

Sorgu Girdisi (Query Input) : Sorgu girdisi, sorgulamayı gerçekleştirmek için ayrıştırıcıyı (parser) ve ara yüzü kullanılır. Alınan sorgu ayrıştırıcı sayesinde parçalanır ve sorgu diline dönüştürülür. Alınan değerler arama mimarisindeki dokümanlar ile eşleştirilmesi yapılır. Birçok arama motoru basit sorgulama dili ile çalışır. Yani alınan sorguyu tek tek terim olarak ya da tırnak içine alınmış deyimler olarak değerlendirir ve bilgiyi elde etme modellerini kullanarak sonuca ulaşır. Bu bakımdan bir konu araştırırken aranan konu ile ilgili temel terimlerin girilmesi, ayrıntılı, açıklamalı ifadelerin girilmesinden çok daha sağlıklı sonuçlar verebilir.

Sorgunun Dönüştürülmesi (Query Transformation) : Bu bölümde alınan sorguyu daha önce bahsettiğimiz tekst işaretlemeleri (Tokenizing), silici (stopping), kök bulucu (stemming) gibi işlemlerden geçirerek indeks mimarisindeki yapıya dönüştürülmesi işleminin yapıldığı bölümdür.

Sonuçların Yansıtılması (Results Output) : Bu bölümde ise toplanılan sonuçlar kullanıcın rahatlıkla kullanabileceği listeler halinde sunulması aşamasıdır. Bunu yaparken elde ettiği dokümanları kırpar (snippets) yani özetleyerek gösterir ve önemli gördüğü kelimeleri kalın puntolar ile gösterir.

1.6.5. Sıralama Modülü (Ranking Module)

Skorlama (Scoring) : Bu bölümde sıralama algoritması kullanılarak, dokümanların skorları belirlenir. Bu işlemi yaparken daha önce bahsettiğimiz bilgiyi elde etme modellerini kullanılır. Farklı mimariler kullanan arama motorları bulunsa da temel olarak anlattığımız modeller üzerine kurulu yapılardır.

Performans Optimizasyonu (Performance Optimization) : Bu bölümde sıralama algoritması üzerinde çalışmalar yapılır. Bu çalışmalarla sorgu süresinin azaltılması ve sorgu çıktılarının artırılması, kaliteli olması amaçlanır.

Dağıtım (Distribution) : Bu bölümde içerik dağıtım işlemini yaptığımız gibi, sıralama işlemini de belirli bölümlere dağıtırız. Ardından bu bölümlerden gelen sıralama değerlerini birleştirip tek sonuca ulaşırız. Ayrıca eğer daha önce yapılan sorgulamalar var ise yerel hafızaya bakıp var olan sonuçlar üzerinden çözümler üretilebiliriz. Böylece arama motorunun zamandan kazanması sağlayabiliriz.

BÖLÜM II

2. Tarama, İndeksleme ve Sorgulama Süreçleri

Arama motorları mimarisinin temel adımları olan tarama, indeksleme ve sorgulama süreçleri bu bölümde ayrıntılı olarak anlatılacaktır. Bu bölümün asıl amacı, her arama mimarisinde farklılık gösteren içerik skorunun temel yapıda nasıl hesaplandığının incelenmesidir. Bu bölümün ardından gelecek bölümde ise, popülarite skorunun belirlenmesinde kullanılan PageRank hesaplanması incelenecektir.

2.1. Tarama Süreci (Crawling Process) :

Tarayıcı modülü, küçük yazılım programları ile talimatlandırılmış örümcekler veya robotlar içeren arama mimarisinin ilk bölümüdür. Bu örümcek ve robotlara verilen talimatlar web sayfalarının nasıl taranması gerektiği bilgilerini içerir. Tarayıcı modülü bu örümcek ve robotlara bir URL uzantısı verir. Örümcekler de aldıkları bu uzantıdan başlayarak sayfaları ziyaret etmeye başlar ve sayfalar üzerinde bulunan linkleri de hafızalarına alarak yeni sayfalara erişirler. Bu işlemi tarayıcı modülünün verdiği talimatlar doğrultusunda maksimum sayfa sayısına ulaşıncaya kadar devam ederler.

Tarayıcılar için birkaç önemli husus bulunur. Bunlardan birincisi, her tarayıcının her sayfayı taraması gerektiği gibi bir zorunluluk yoktur. Bazı arama motorları sadece belirli konular çerçevesinde tarama işlemini gerçekleştirirler. Örneğin sadece .net uzantılarını taraması, sadece resim barındıran web sayfalarını taraması ya da sadece kişisel web sayfalarını taraması gibi belirli bir alana yönelik çalışabilirler.

Tarayıcılar için bir önemli hususta, web sayfalarını hangi sıklıkla tarayacaklarıdır. Web dünyasının dinamik olmasından dolayı, web sayfaları devamlı güncellenmektedir. Bu durum tarayıcıların işini epey zorlaştırmaktadır. Bu yüzden tarayıcıların işi tam bir sirkülasyon işidir. Yani sonsuz bir döngü içinde çalışıyorlar diyebiliriz. Şöyle ki yeni ve güncellenmiş sayfaları getiren örümcekler her defasında verilen yeni bir URL uzantısı ile yeniden işe başlarlar. Bir bakıma işçi arıların polenleri toplayıp, kovana getirip yeniden polen toplamak için kovandan uçması gibi bir şeydir. Bu benzetmeden dolayı olsa gerek ki Google web robotlarına "web arısı" ismini vermektedir. Tarayıcıların web sayfalarını devamlı kontrol etmesi yerine

geliştirilen algoritmalar sayesinde zamanla oluşan, web sayfalarının birim zamanda gerçekleşen güncelleme sayısını belirleyerek, bazı web sitelerini daha sık tararken bazı web sitelerini haftada bir ya da ayda bir taramaktadırlar. Özellikle haber siteleri saatlik, günlük taranırken; kişisel web sayfaları haftada bir ya da ayda bir taranmaktadır. Ayrıca bazı arama motorları bu tarama işini demokratik yöntemlerle yapsa da, bazı arama motorları sitelerin popülaritesine göre yapmaktadırlar. Tarama işleminde yaşanan bu sıkıntıdan dolayı günümüzde çok farklı yöntemler geliştirilmiştir. Bazen site sahipleri güncellenen sayfalarını kendileri manüel olarak arama motorlarına bildirir. Bazen de sayfalarının güncellenme sıklığını, güncellenme zamanını yada güncellenen sayfalarını geliştirilen programlar sayesinde (robot.txt) arama motorlarına kendileri bildirirler. Programlar ile yapılan güncellemelerde tarayıcılar web sayfanıza geldiğinde eğer bir önceki tarama işleminden bu yana bir güncelleme yoksa sayfanızı taramayıp bir sonraki web sayfasına geçerler. Böylece hem tarayıcıların zamandan kazanması sağlanırken, hem de tarayıcıların web sayfanızı tararken trafiğinizi engellemesinden kurtulmuş olursunuz. Çünkü tarayıcılar web sayfanızı tararken bulunduğunuz portun bant genişliğini işgal ederler, bu durumdan dolayı da kullanıcılarınız sitenize erişmekte sorun yaşarlar. Bu durum da doğal olarak web site sahiplerini fena halde kızdırır.

2.1.1. Tarayıcı Politikaları:

Web arama motorları önceleri bir taraftan kullanıcının girdiği değerlere göre web sayfalarını tararken bir taraftan da ilgili web sayfalarının listesini oluşturmak için kullanıcıyı bekletirlerdi (Dündar, 2009). Zamanla web devasa boyutlara ulaşınca bu işleme alternatif yöntemler geliştirdiler. Arama motorları bu sorunun üstesinden gelmek için taradıkları web sayfalarını kendi veri tabanlarına kaydetmeye başladılar. Veri tabanlarına kaydettikleri bu web sayfalarını çeşitli yöntemlere göre indeksleyip, web sayfaları ile ilgili istatistikî bilgileri de hesaplayıp, sorgulamanın daha hızlı gerçekleştirilmesini sağladılar. Böylece kullanıcının girdiği sorgu daha önceden hesaplanmış değerlere göre eşleştirilerek daha hızlı ve daha etkin sonuçlar üretmeye başladılar.

Arama motorları, kullandıkları web robotları ile öncelikle ziyaret ettikleri web sayfalarının ana sayfalarından başlayarak siteyi kopyalamaya başlarlar. Ardından sitenin diğer sayfalarını da kopyalayıp, sitenin haritasını çıkarırlar. Bu işlemleri yaparken bir taraftan da web sayfası üzerinde bulunan linkleri de hafızaya alarak,

ziyaret edilecek bir sonraki web sayfasını belirlerler. Daha önce de belirttiğimiz gibi tarayıcılar web sayfalarında gezinirken, ilgili web sayfasını epey meşgul ederler. Bunu önlemek için ise kurallar oluşturulmuştur. Şimdi bu kurallara kısaca bir göz atalım.

Politeness Policy: Dediğimiz gibi tarayıcılar ilgili web sayfasını tararken eğer taranan site çok güçlü değil ise gerçek kullanıcılara hizmet vermekte aksaklıklar yaşar. Bunu önlemek için ise taramaya başlamadan önce ilgili web sunucuya tarama isteği göndermesi gerekir. İlgili web sunucuda bu isteği alıp taramanın başlayıp başlamayacağına karar verir. Bu cevap verme süresi saniyeler bazen dakikalar sürdüğü için tarayıcılar (crawlers) bekletilmek durumunda kalır. Tarayıcılar da bu bekletilme süresinden nefret ettiklerinden paralel programlama tekniğinde kullanılan thread'leri (aynı görev için programlanmış programcıklar) kullanırlar. Böylece bir thread beklerken bir diğeri başka bir web sayfasını tarar. Ayrıca bu ilkenin güzel bir uygulaması daha vardır. Şöyle ki bu kural gelen tarayıcıya birim zamanda ne kadar sayfa tarayacağını söyler. Misal bir web sunucusunun bir tarayıcıya izin verdiğini düşünelim. Ve tarayıcı saniyede 100 sayfa tarıyor fakat bu durumun trafiği aksattığını düşünelim. Bu ilke ile web sunucusu tarayıcıya web sunucusundan saniyede en fazla 10 sayfa tarayabilirsin diye uyarı verir. Böylece tarayıcıların trafiği aksatması önlenir. Bu durumu yüzlerce arama motoru üzerinden düşünürsek eğer, bazı önemli sitelerin devamlı web tarayıcılarına çalışması gerekirdi ki, bu durum da web sayfalarının anlamını yitirmesine sebep olurdu.

Re-visit Policy: Web sayfalarındaki değişikliklerin hangi aralıklarda yapıldığını bildirir.

Selection Policy: Web tarayıcıların hangi sayfaları tarayıp, hangi sayfaları tarayamayacağı belirtir.

Parallelization Policy: Yukarda bahsettiğimiz tarayıcıların paralel çalıma mantığına göre, her thread taradığı web sayfalarını diğer bütün thread'lere bildirir. Böylece taranmış sayfaların tekrar taranması önlenmiş olur.

2.1.2. Bilgilendirme Dosyası

Tanımlamaların ardından web masterların tarayıcıları bilgilendirmek için kullandıkları robot.txt mantığına bakalım. Web masterlar robot.txt ile tarayıcılara

tarama zamanını ve tarama frekansını bildirirler. Bunu yaparken de robot.txt'yi site adreslerinin kök dizinin de sunarlar. Örneğin, http://www.khas.edu.tr/robots.txt şeklinde tarayıcılara sunarlar. Daha iyi anlamak için Robot.txt'nin yapısına bir göz atalım.

User-agent: *

Allow:

Yukarıdaki içerikte "User-agent" bölümü arama motorlarının tarayıcılarını belirtmektedir. "*" işareti ile de bütün tarayıcılar kastedilmektedir. Bir alt satırdaki "Allow" ise bütün web tarayıcılarına izin verildiği belirtilir. Daha açıklayıcı olması için somut bir örnek verelim.

User-agent: MsnBot

Allow:

User-agent: Googlebot

Disallow:

Yukarıdaki örnekte de web site yöneticisi sitesinin içeriğine MsnBot'un taramasına izin verirken, GoogleBot'un sitesini taramasına izin vermemektedir. Bir sitenin bütün içeriği sunucularda tutulduğu için, bazen sitenin bazı kısımlarının gizli kalması ve hem tarayıcılar hem de kullanıcılar tarafından ulaşılmaması gerekir. Eğer sitenin bazı sayfalarının tarayıcılar tarafından taranması istenmiyorsa bu durumda robot.txt kullanılabilir. Bu duruma da bir örnekle açıklık getirelim.

User-agent: *

Disallow: /ozel/

Disallow: /gizli/

Disallow: /diger/

Allow: /diger/genel/

Yukarıdaki örnekte de bütün web tarayıcılarının "/ozel/", "/gizli/", "/diger/" klasörlerine erişimi yasaklanıyor. "Allow" bölümünde ise sadece "/diger/genel/" alt klasörüne erişim izni veriliyor.

2.1.3. Site Haritası

Hem arama motorlarının hem de site yöneticilerinin işini kolaylaştırmak için bir dizi işlemden birisi de site haritalarıdır. Site haritaları sayesinde arama motorları ile web site yöneticileri arasında iletişim sağlanmaktadır. Böylece hem arama motorlarının siteleri tararken yarattıkları trafik önlenmekte hem de site yöneticilerinin istediği gibi bir tarama sağlanmaktadır. Robot.txt'de olduğu gibi site haritası da genellikle kök dizininde tutulurlar. Örneğin, http://www.khas.edu.tr/sitemap/sitemap.xml ile tarayıcıya sunulurlar. Bu bölümde site haritasına bir örnek verip tarayıcılar konusunu kapatalım.

```
<? Xml version= "1.0" encoding= "UTF-8" ?>
< urlset xmlns="http://www.khas.edu.tr/schemas/sitemap/0.9">
        <url>
                <loc>http:// www.khas.edu.tr/</loc>
                <lastmod>2012-06-06</lastmod>
                <changefreq>daily</changefreq>
                <priority>0.9</priority>
        </url>
        <url>
                <loc>http:// www.khas.edu.tr /akademisyenler</loc>
                <lastmod>2012-01-01</lastmod>
                <changefreq>monthly</changefreq>
                <priority>0.6</priority>
        </url>
        <url>
                <loc>http:// www.khas.edu.tr /duyurular</loc>
                <changefreq>hourly</changefreq>
                <priority>0.8</priority>
        </url>

</urlset>
```

Görüldüğü gibi site haritası çeşitli etiketler içermektedir. Bunların başında "loc" etiketi gelmektedir. "Loc" etiketi ile URL uzantısı verilir. "changefreq" etiketi ile de ilgili bölümün hangi sıklıkla değiştirildiği belirtilir. Bu sıklık değişkenleri "never", "allways", "hourly", "daily", "weekly", "monthly" gibi değerler olabilir. "priority" etiketine geldiğimizde ise tarayıcıya bu sayfanın ne kadar değerli olduğunu belirtiniz. Yani "http://www.khas.edu.tr/" bölümünün değeri 0.9 iken "http://www.khas.edu.tr/akademisyenler" bölümünün değeri 0.6 olması, tarayıcıya "http://www.khas.edu.tr/" sayfasının bir diğer sayfa olan "http://www.khas.edu.tr/akademisyenler" sayfasından daha önemli olduğunu belirtir.

Ayrıca günümüzde siteler sayfalarını HTML (Hypertext Markup Language) standardına göre hazırlamaktadırlar. Fakat HTML standardında kesin bir düzenin olmaması aynı kodlamanın farklı tarayıcılarda farklı yorumlanmasına yol açmaktadır. Bu durum HTML sayfalarının tarayıcılar tarafından işlenmesini zorlaştırmaktadır. Bu işlemin daha etkin olmasını sağlamak amacıyla XML'in (Extended Markup Language) standardı geliştirilmiştir (Dündar, 2009). XML'in en önemli özelliği olan ağaç yapısı tarayıcıların işini epey kolaylaştırmıştır. Bu yüzden tarayıcılar HTML düzensizliğinden arındırılmış bir yapıya ulaşmak için belirli programlar sayesinde HTML belgelerini XML'e dönüştürmektedirler. Böylece daha etkin bir indeksleme gerçekleştirmektedirler.

2.2. İndeksleme Süreci (Indexing Process)

Örümcekler ve web robotları tarafından indeksleme bölümüne getirilen web sayfaları bu bölümde bir dizi işleme tabi tutulurlar. Elde edilen web sayfaları sırasıyla daha önce bahsettiğimiz ayırıcı, silici, kök bulucu, sınıflayıcı gibi işlemlere tabi tutularak indeks bölümüne aktarılırlar. Bu işlemleri anlatmadan önce birkaç durumu izah etmekte fayda var.

Web sitelerinin indeks bölümüne aktarılması için tablolar kullanılır. Bu tablolar alabildiğine büyük boyutlarda olabilirler. Arama motorları da bu büyük tablolarda kolaylıkla işlem yapabilmek için, arama motorları ile iç içe olan bir diğer bilim alanı olan Veri Tabanı Sistemleri'nden yararlanırlar. Özellikle ilişkisel veri tabanları (Relation Database) arama motoru indeksleri için çok büyük rahatlıklar sağlar. Zaten bilgi depolamak ve sorgulamak amacıyla oluşturulan ilişkisel veri tabanları, arama motorları için web sayfalarını depolamak ve çeşitli uygulamalar geliştirmek için gayet uygundur. Birçok veri tabanı da ağda bulunan sunucularda çalıştıkları için, dokümanlara ağ üzerinden ulaşmakta bir o kadar kolay olur. Ayrıca veri tabanı sistemleri birçok uygulama aracı içerdiği için, toplanan verileri yönetmek de daha kolay olmaktadır.

Arama motorlarının mimari yapısı birbirinden farklı olsa da genellikle hepsi aynı mantık üzerine kuruludur. Bu bağlamda anlatacaklarımız temel düzeyde arama motorlarının uyguladığı işlemler olup, günümüzdeki en büyük arama motorları mimarisinde de kullanılmaktadır. İndeksleme işlemlerine başlamadan çoğu arama motoru tarafından oluşturulmuş Büyük Tablo'dan (Big Table) bahsetmek indeksleme

tarihi açısından yararlı olacaktır. Büyük Tablo web sayfalarını depolamakta kullanılan dağıtık bir veri tabanı sistemi idi. Günümüzde kullanılan ilişkisel veri tabanlarına göre birçok zorluğu bulunmaktaydı. Eski sistemlerde Büyük Tablo gerçekten de çok büyük bir boyuta sahipti. Öyle ki boyutu bazı arama motorlarında petbayte'ları bulurdu. Her veri tabanında bir adet bulunurdu fakat Büyük Tablo, tablocuklar dediğimiz yüzlerce parçadan oluşurdu. Tablocuklardan kastımız her bir dokümana ait olan satırlardı. Yani Büyük Tablo'yu yüzlerce satır ve sütundan oluşmuş devasa bir tablo olarak düşünebilirsiniz. Her satırın kolonlarında ilgili web sayfası ile ilgili bilgiler bulunurdu. Büyük tablonun olumsuz tarafı ise, kaydedilen dosyaların değiştirilememesiydi. Neyse ki günümüzde ilişkisel veri tabanları kullanılmaktadır ve güncellemeler rahatlıkla yapılabilmektedir. Ayrıca birçok karmaşık işlemde ilişkisel veri tabanlarında rahatlıkla yapılabilmektedir. İlişkisel veri tabanlarında her bilgi ayrı tablolarda kaydedilir ve her tablonun kendisi ile ilgili başka tablolar ile bağlantısı vardır. Zaten adını da buradan almaktadır. Yani sitenin istatistikî bilgileri bir tabloda yer alırken, terim içerikleri ve ya link bilgileri bir başka tabloda yer alır.

İşte elde ettiğimiz web sayfaları ile ilgili bilgiler bu tablolara kaydedilir. Şunu bilmekte fayda var. İndeksleme bölümü hemen hemen bir sözlük mantığında işler. Yani kelimeler kronolojik bir mantık çerçevesinde depolanır. Sözlükten farkı aranan kelimeye geldiğinizde, o kelimenin bulunduğu dokümanları gösterir. Ayrıca aranan kelime ile ilgili, kelimenin dokümandaki yeri, sayısı, kelimenin ağırlığı ve bulunduğu konum gibi bilgileri de içerir. Bu tablo yapısını birazdan göstereceğiz fakat şimdi sırasıyla elde edilen web sayfalarına uygulanan işlemlerden bahsedelim.

Kopya İçeriklerin Belirlenmesi (Detecting Duplicates) : Yapılan araştırmalar gösteriyor ki web sitelerinin yaklaşık %30'u diğer %70'lik bölümü oluşturan web sitelerinin kopyasıdır (Croft, Metzler ve Strohman, 2010). Bu durum arama motorlarının boşa kürek çekmesine sebep olduğundan, bu siteler arama motorları tarafından belirlenmeye çalışılır. Çünkü aynı içeriklerin bulunması hem veri tabanında yer kaybına yol açtığından hem de arama kalitesini düşürdüğünden arama motorları tarafından istenmeyen bir durumdur. Genellikle kopya içerikli siteler belirlendikten sonra belirli işlemlere tabi tutulurlar. Ya veri tabanından atılırlar ya da sıralama işlemlerinde grup olarak değerlendirilirler. Kopya içerik belirlemek basit bir

işlemdir. ASCII kodlama düzenine göre aynı harf sıralamasını belirli bir eşik değerinin üstünde bulunduranlar, kopya içerik barındırdığı kabul edilir.

Kirliliğin Önlenmesi (Removing Noise) : Bazı site içerikleri o kadar fazla reklam içerir ki, sitenin büyük bir kısmı reklamdan oluşur. Bu reklam içeriklerinin veri tabanlarına kaydedilmesi arama motorlarının kalitesini düşürür. Bu yüzden arama motorları sınıflandırma tekniklerini kullanarak, web sayfalarının kendine özgü içeriğini hem kelimelerden hem de XML yapısından belirlerler. Bu içerikle alakası olmayan bölümler veri tabanına kaydedilmeden silinirler.

İşaretlemelerin Belirlenmesi (Tokenizing) : Bazı kelimeler ayrıştırıcı tarafından parçalandığında bir anlam ifade etmezler. Bunun için bazı kelime yapılarının kontrol edilmesi gerekir. Örneğin bazı kelimeler tırnak, nokta, tire gibi işaretler içerebilirler. Genellikle Türkçe dil yapısında pek bulunmasalar da, özellikle teknolojiden kaynaklı farklı dillerden bazı kelimeler Türkçeye girmiştir. Örneğin X-ray, N-95, I.B.M, T.C gibi yapılar arama motorları için önem arz ederler. Bunların belirlenmesi için de hem dilin kendine özgü yapısı hem de bazı bilim alanlarında bulunan birleşik kelimeler belirlenerek bunların parçalanması önlenir.

Silme İşlemi (Stopping) : Daha önce bahsettiğimiz gibi silme işlemi, arama motorlarının sıralamasını etkilemeyen ve sürekli kullanılan “bir”, “ve”, “çok” gibi kelimelerin hafızaya alınmamasıdır.

Kök Bulma İşlemi (Stemming) : Kök bulma işlemi de daha önce bahsettiğimiz gibi aynı köke sahip kelimelerin belirlenip, tabloda kendisine ayrılmış bölüme alınmasıdır. Yani bir dokümanda geçen “göz”, “gözüm”, “gözlerin” vb. türevlerinin sayısı belirlenirken kelimenin kökü olan “göz” kelimesi alınarak hesaplanmasıdır.

Deyimler ve Birleşik Kelimelerin Belirlenmesi: Arama motorlarının sorgu içeriklerini parçaladıklarından bahsettik. Fakat bazı kelime grupları vardır ki genellikle birlikte bulunurlar. Daha önce de söylediğimiz gibi indeks yapısı, bir sözlüğün yapısına benzer. Mesela sözlükten “zil” kelimesine baktınız. Sözlük zil kelimesinin tanımını vermesinin ardından bu kelimenin diğer kelimelerle birleşiminden oluşan kelime gruplarını da verir. Çünkü bazen iki kelime birleştiğinde kendi anlamlarını yitirip bambaşka bir anlama bürünebilirler. Örneğimize dönecek olursak, sözlük zil kelimesinin tanımının ardından “zil zurna” sarhoş olmak gibi bir

kelime grubunun da açıklamasını verecektir. Görüldüğü gibi "zil" ve "zurna" kelimesi birbirinden alakasız olsa da birleşerek kendi anlamlarını yitirip bambaşka bir anlama bürünmüşlerdir. Arama motorlarının bu durumu hesaba katmasıyla hem arama hızını artıracak hem de arama kalitesini artıracaktır. Bu durum sadece deyimlerle bitmemektedir. Birleşik isim guruplarını da düşünebiliriz. Örneğin, "Orta Doğu", "Amerika Birleşik Devletleri", "Kuzey Kore", "İnsan Hakları", "Güney Doğu" gibi duyduğumuzda hemen aklımızda bir anlam ifade ettiği kelime gruplarını da birleşik olarak ele almak ve doküman içinde geçen frekanslarını hesaplamak, daha kaliteli bir arama sonucu kullanıcıya göstermemizi sağlayacaktır.

Doküman Yapısı (Document Structure) : Daha öncede belirttiğimiz gibi dokuman yapıları sıralama algoritmasında önemli bir yer tutar. Mesela dokümanın yazarı, yayın tarihi gibi bilgiler önemlidir. Ayrıca dokümanın HTML yapısı da bize birçok ipucu verir. Sıralama algoritmasını hesaplarken kelimelerin dokümanda bulunduğu konum ve yapısı büyük önem arz eder. Kelimenin dokümanda başlık kısmında bulunması ile gövde kısmında bulunması, doküman içindeki değerini bize gösterir. Ayrıca kelimenin kalın puntolarla yazılması ya da kullanılan yazı türünden farklı bir yazı karakterinin kullanılması (mesela *italik* olarak yazılması) ya da kelimenin alt çizgi içermesi, bize bu kelimenin diğer kelimelerden daha önemli olduğunu vurgular. İşte arama motorları da sıralama algoritmasını oluştururken bu durumları göz önüne alarak hesaplamalarını gerçekleştirirler.

Link Analiz (Link Analysis) : Arama motorları web sayfalarında geçen linklere büyük önem verirler. Çünkü bir site başka bir siteye link veriyorsa, "anlattığımız konu ile ilgili ek ve ya ayrıntılı bilgileri link verdiğim sayfadan da araştırabilirsiniz" demektedir. Bu durum da arama motorlarının kalitesini artırmak için kullandıkları konu sınıflaması (Topical Classification) için önemli bir değerdir. Çünkü ilgili siteleri sınıflayarak daha kaliteli sonuçlar üretilebilmektedir. Bu duruma bir örnek verelim.

Daha fazla bilgi için < a href= "http:// www.osym.gov.tr" > tıklayınız
Görülüğü gibi başka bir siteye link vermek için HTML'in "<a>…" etiketini kullanmaktayız. Ayrıca bu durum şunu da ifade etmektedir. Anlattığımız konunun ayrıntılı açıklamasını ya da doğruluğunu bu siteden de öğrenebilirsiniz demektir. Bu durum da sizin link verdiğiniz sitenin önemli bir site olduğunu gösterir. Popülarite

skorunun hesaplanmasında bu durumu ayrıntılı bir şekilde ele alacağız. Burada bilmemiz gereken, indeksleme bölümünde linklerin ayıklandığı ve tablolara kaydedildiğidir.

Arama motorları indeksleme bölümünde yukarıda anlattığımız işlemlerden çok daha fazlasını yapmaktadır. Fakat yapılan işlemler o kadar fazla ki araştırmamızın dışına çıktığından yapılan diğer işlemlere girilmeyecektir. Fakat bizim için en can alıcı bölüm olan terim indekslemeye bölümüne göz atmakta yarar vardır.

2.2.1. Terim İndeksleme

İndeksleme bölümüne gelen dokümanlar çeşitli işlemlerden geçtikten sonra sıkıştırılmış bir formda tablolarda saklanır. Dönüştürülmüş bu dosyalarda (inverted files) bizim için önemli olan *sayfa belirleyicisi* tablosuna bir göz atalım. Terim indeksleme tablosunun bir sözlüğü andırdığını tekrar hatırlatalım. Normalinde kelimeler A'dan Z'ye doğru sıralanır ve tablodaki her kelimenin karşısında kelimenin hangi dokümanda bulunduğu, hangi konumda olduğu ve kaç defa geçtiği ile ilgili bilgiler verilir. Aşağıdaki örnekte kelimeler tabloya tahminen yerleştirilmiştir. Bir gerçekliği yoktur. Zaten her arama motorunun içerdiği kelime sayısı farklı olduğundan bu sıra her arama motorunda farklılık göstermektedir.

-Terim 1 (abajur) – 5, 127, 367, 1356

.

-Terim 59 (bilgisayar) - 96, 198, 3598, 23568

.

-Terim 286 (donanım) - 96, 252, 1265, 3598, 56894

.

-Terim 5268 (klavye) – 255, 986, 1256, 5987, 26548

.

-Terim n (zurna) – 782, 1598, 2658, 13256

Buna göre terim 1 (abajur) numarası 5, 127, 367 ve 1356 olan web sayfalarında bulunmaktadır. Dönüştürülmüş listeye göre terim 59 (bilgisayar) ve terim 286 (donanım), numarası 96 ve 3598 olan web sayfalarında beraber bulunmaktadır. Elimizdeki bu liste dönüştürülmüş dosyaların en basit halidir diyebiliriz. Şöyle ki terim 59 (bilgisayar), numarası 96, 198, 3598, 23568 olan web sayfalarında

geçmektedir. Fakat terim 59'u barındıran web sayfaları arasındaki değer farkını nasıl belirleyeceğiz? Başka bir deyişle terim 59 bir sitede başlık konumundayken bir diğer site de içerik konumunda olabilir. Fakat biz şunu biliyoruz ki eğer bir kelime başlıkta yer alıyorsa o sayfanın değeri bizim için daha önemlidir. Çünkü büyük ihtimalle kelimeyi başlığında barındıran bir web sayfası, içerik bölümünde de başlıkla ilgili bir konu anlatıyor demektir. Ayrıca bu listeye bakarak kelimelerin dokümandaki yapısını da belirleyemeyiz. Yani aradığımız kelime dokümanda büyük fontlarla mı yazılmış yoksa kalın puntoyla mı yazılmış yoksa kelime italik mi yazılmış olduğunu bu listeye bakarak belirleyemeyiz. Bu gibi durumların dışında bir de kelimenin ilgili sayfada kaç defa geçtiğini bilmiyoruz. Fakat bildiğimiz şu ki bir dokümanda bir kelime ne kadar çok geçerse o dokumanın aradığımız kelime ile ilgisi bir o kadar fazladır. Peki, bu çıkmazdan kurtulmak için ne yapmamız gerekir?

Arama motorlarının gelişimine baktığımızda ilkel anlamda bu mantığa göre çalışmaktaydı. Fakat bu tarz sorgulama sonuçları web dünyasının gelişmesi ile birlikte anlamını yitirmeye başladı. Arama motorunun ara yüzünde girdiğimiz kelimelerin geçtiği web sayfalarının görüntülenmesi, web sayfalarını ayıklama işini biz kullanıcılara bıraktığını gösterir. Tabi ki böyle bir arama motoru ile de kimse çalışmak istemeyecektir. Bu yüzden arama motorları tablo yapılarını vektör mantığında biraz daha geliştirerek, daha gerçekçi sonuçlar üretmeye başladılar. Gelin bizde anlattığımız bu yapıyı biraz daha ilerleterek konunun daha iyi anlaşılmasını sağlayalım.

-Terim 1 (abajur) – 5 [1, 0, 3], 127 [0, 0, 4], 367 [1, 1, 10], 1356 [1, 0, 8]

.

-Terim 59 (bilgisayar) –96 [1, 1, 25], 198 [0, 0, 5], 3598 [1, 1, 21], 23568 [1, 0, 3]

.

-Terim 286 (donanım) – 96 [1, 0, 13], 252 [1, 0, 5], 1265 [0, 0, 2], 3598 [1, 1, 24]

.

-Terim 5268 (klavye) – 255 [0, 0, 2], 986 [1, 1, 11], 1256 [1, 0, 25], 5987 [0, 0, 3]

.

-Terim n (zurna) – 782 [1, 1, 13], 1598 [1, 0, 9], 2658 [1, 1, 2]

Buna göre terim 1 (abajur), numarası 5 olan web sayfasında geçmektedir. Ayrıca 5 [1, 0, 3] bölümündeki "1", kelimenin başlık kısmında geçtiğini, "0", kelimenin meta etiketinde (meta etiketi ilgili web sayfasını tanımlayan kelimelerdir. HTML kodlama kısmının HEAD etiketleri arasında <meta name= "keywords" content= "Bilgisayar, donanım, dizüstü bilgisayar, ekran kartı" /> şeklinde yer alır.) yer almadığı, "3" ise dokümanda kelimenin 3 defa geçtiğini göstermektedir. Aynı şekilde terim 1 (abajur) numarası 127 olan web sayfasında geçtiği ve 127 [0, 0, 4] bölümünde ise "0" kelimenin başlık kısmında bulunmadığı, bir sonraki "0" kelimenin meta kısmında geçmediği, "4" ise kelimenin dokümanda 4 defa geçtiğini belirtmektedir. Görüldüğü gibi sayfa tanımlamasında üç boyutlu vektör kullanılmıştır. Arama motorları sayfaları sıralarken bu üç boyutlu vektörden yola çıkarak hesaplamaktadır. Tahmin ettiğiniz gibi arama motorları sayfaları daha ince ayrıntılarla değerlendirmek isterse vektör sayısını özelliklere göre artırabilir.

2.3. Sorgulama Süreci (Query Process)

Son olarak göreceğimiz sorgulama süreci, birinci bölümde de belirttiğimiz gibi kullanıcının sorgusuna bağımlıdır. Oysaki tarama süreci ve indeksleme süreci kullanıcıdan bağımsız (query-independent) gerçekleşmekteydi. Yani sorgulama süreci kullanıcının sorguyu girmesi ile başlayan ve milisaniyelerde dönütlerin oluşturulduğu, kullanıcı girdisine bağımlı olarak gerçekleşen bir süreçtir. Birçok arama motoru sorgulama sürecinin olabildiğince hızlı gerçekleşmesi için önceden birçok hesaplama yaparlar. Özellikle içerik indeksi (content index) ve yapı indeksinde (structure index) yapılan hesaplamalar, sorgulama sürecinin olabildiğince kısa sürmesini sağlar.

Sorgulama sürecinin tarihsel gelişimi her ne kadar Google'dan önce de sürekli ilerleme gösterse de, Google ile birlikte büyük bir sıçrayış göstermiştir. Şubat 2003'te Google her gün yaklaşık olarak 250 milyon aramaya cevap verdiği söylenir. Yine aynı yılda Overture 167 milyon ve Inktomi 80 milyon aramaya cevap verdiği belirtilmiştir (Langville ve Meyer, 2006).

Sorgulama sürecinin tarihsel gelişimini bir tarafa bırakarak, arama motorlarının bu işlemi nasıl gerçekleştirdiğine bir göz atalım. Dediğimiz gibi her arama motorunun kendine özel hesaplamaları vardır. Burada anlatacaklarımız temel düzeyde olup, daha

çetrefilli hesaplamalarda yapılmaktadır. Fakat anlatacaklarımızın dışında yapılacak her işlem genellikle vereceğimiz bilgiler temelinde yapılmaktadır.

2.3.1. İçerik Skorunun Hesaplanması:

Sorgulama sürecinin daha hızlı anlaşılması açısından indeks bölümünde oluşturduğumuz sıralamayı kullanalım.

-Terim 1 (abajur) – 5 [1, 0 , 3], 127 [0, 0, 4], 367 [1, 1, 10], 1356 [1, 0, 8]

.

-Terim 59 (bilgisayar) –96 [1, 1, 25], 198 [0, 0, 5], 3598 [1, 1, 21], 23568 [1, 0, 3]

.

-Terim 286 (donanım) – 96 [1, 0, 13], 252 [1, 0, 5], 1265 [0, 0, 2], 3598 [1, 1, 24]

.

-Terim 5268 (klavye) – 255 [0, 0, 2], 986 [1, 1, 11], 1256 [1, 0, 25], 5987 [0, 0, 3]

.

-Terim n (zurna) – 782 [1, 1, 13], 1598 [1, 0, 9], 2658 [1, 1, 2]

Oluşturulan bu tablo içeriğine göre kullanıcı "*Bilgisayar Donanımı*" şeklinde bir sorgulama girmiş olsun. Buna göre sorgu modülü ilk öce "bilgisayar" kelimesinin bulunduğu terim satırını bulacaktır. Böylece terim 59'a ulaşacaktır ve bu terimin numarası 96, 198, 3598 ve 23568 olan web sayfalarında geçtiğini belirleyecektir. Yine aynı şekilde sorgu modülü "donanım" kelimesini tarayıp terim 286'ya ulaşacaktır. Terim 286 da numarası 96, 252, 1265 ve 3598 olan web sayfalarında bulunduğunu belirleyecektir. Ardından *Boolean AND* mantığıyla iki terimin birlikte geçtiği web sayfalarını bulacak ve buna göre Bilgisayar ve Donanım kelimelerinin geçtiği 96 ve 3598 numaralı web sayfalarını belirleyecektir. Çoğu arama motoru işlemlerini burada durdurur ve oluşturduğu listeyi kullanıcıya sunar. Oluşturduğu listede genellikle daha önceden içerik ve yapı indekslerine göre yaptığı sıralamadır. Fakat böyle bir dönüt pek sağlıklı olmayacaktır. Çünkü bu iki kelimeyi barındıran binlerce web sayfası vardır. Böylesi bir sonuçta, bütün yük kullanıcıya bindirilip, çıkan sonuçları tek tek değerlendirmesi istenecektir. Fakat farz edelim ki biz özellikle bilgisayar donanımını anlatan bir web sayfasını arıyoruz ve arama motorundan daha spesifik bir sonuç bekliyoruz. Gelin böyle bir sonuç üreten daha sıkı bir sıralama mantığını inceleyelim. Fakat incelemeye geçmeden önce şunu bir

daha belirtmekte fayda var. Biz burada içerik skorunu belirlemeye çalışıyoruz. Ayrıca popülarite skorunun da hesaplanması gerekir ve en son olarak bu iki skorun birleşiminden oluşan kapsamlı skor (overall score) hesaplanıp gerçek bir sıralama elde edilmesi gerekir. Çoğu arama motoru önceleri sadece içerik skoruna göre sıralama yaparken içerik spamlarından dolayı popülarite skoruna geçtiler. Fakat popülarite skorunun da spamları türeyince ikisinin birleşiminden oluşan kapsamlı skorun elde edilmesinin daha mantıklı olacağına kanaat getirdiler. Fakat günümüzde hala çoğu arama motoru popülarite skorunu kullanmaya devam etmektedir. Konuyu daha fazla dağıtmadan hesaplamalarımıza geri dönelim.

Hesaplamamızı aynı şekilde "Bilgisayar Donanımı" üzerinde devam edelim. Çözümü daha yakından görmek için kelimelerin bulunduğu iki satıra bir göz atalım.

-Terim 59 (bilgisayar) –96 [1, 1, 25], 198 [0, 0, 5], 3598 [1, 1, 21], 23568 [1, 0, 3]

.

-Terim 286 (donanım) – 96 [1, 0, 13], 252 [1, 0, 5], 1265 [0, 0, 2], 3598 [1, 1, 24]

Buna göre iki terimin beraber geçtiği web sayfaları {96, 3598} dir. Bu sayfaların üç boyutlu vektörden oluştuğunu biliyoruz. Daha önce de belirttiğimiz gibi birinci vektör boyutu kelimenin başlıkta geçip geçmediği, ikinci vektör boyutu kelimenin meta etiketinde bulunup bulunmadığı ve üçüncü vektör boyutu ise kelimenin ilgili dokümanda kaç defa geçtiğiydi. Bu üç boyutu da göz önüne alarak şöyle bir hesaplama yapıyoruz.

Bilgisayar ve Donanım Kelimelerini beraber bulunduran;

Sayfa 96 için içerik skoru = (1 + 1 + 25) x (1 + 0 + 13) = 27 x 14 = 378,

Sayfa 3598 için içerik skoru = (1 + 1 + 21) x (1 + 1 + 24) = 23 x 26 = 598.

Görüldüğü gibi ilgili kelimelerin vektörlerinin değerleri toplamının çarpılmasının ardından sayfa 3598'in sayfa 96'dan daha büyük bir skora sahip olduğunu görüyoruz. Böylece sayfa 3598 sıralamada daha öncelikli sıraya oturacaktır. Fakat unutulmamalıdır ki eklenecek her içerik özellik vektörü bu sıralamayı değiştirebilir. Biz burada sadece üç özellik vektörü içeren bir hesaplama yaptık. Ayrıca kelimenin içerikte faklı bir yazı karakteri ile geçmesi ya da kelimenin kalın puntolarla geçmesi gibi özellikler de hesaplamalara eklenebilir. Bunun dışında her özellik için çarpım sabiti de eklenebilir. Yani kelimenin başlıkta geçmesinin içerikte geçmesinden daha

anlamlı buluyorsak, bu vektör boyutunu diğer vektör boyutlarından daha yüksek bir çarpım sabiti ile çarpıp hesaplamalarımızı geliştirebiliriz. Bu düşüncemize göre farz edelim başlıkta geçmesini 10 sabiti ile çarparken, meta bölümünden geçmesini 5 sabiti ile çarpıp, kelimenin dokümanda bulunma miktarını ise 1 ile çarpalım. Buna göre sonuç;

Sayfa 96 için içerik skoru = (10 x 1 + 5 x 1 + 1 x 25) x (10 x 1 + 5 x 0 + 1 x 13)
= 40 x 23
= 920,
Sayfa 3598 için içerik skoru = (10 x 1 + 5 x 1 + 1 x 21) x (10 x 1 + 5 x 1 + 1 x 24)
= 36 x 39
= 1404.

Her ne kadar yapılan bu işlem sıralama sonucunu değiştirmese de faklı durumlarda bu sıralama sonucu değişebilir. Ayrıca göze çarpan bir durum daha var. O durum da şudur. Yaptığımız eklemeler sonucunda sayfa 3598'in içerik skoru sayfa 96'nın içerik skorundan bir önceki duruma nazaran daha önemli konuma geldi. Çünkü birinci durumda iki sayfa arasındaki değer farkı 220 iken ikinci durumda iki sayfa arasındaki fark 484 tür. Bunun çarpımdan kaynaklandığı düşünülebilir. Fakat bir durum daha vardır. Sayfa 96'nın içerik skoru hesaplanırken çarpımın ikinci bölümünde geçen (donanım kelimesinin hesaplanması) ikinci vektör (10 x 1 + 5 x 0 + 1 x 13) "0" ile çarpılmıştır. Yani sayfa 96'da "donanım" kelimesi meta bölümünde geçmemektedir. Oysaki sayfa 3598 de "donanım" kelimesi meta bölümünde geçmekte ve "1" ile çarpılmıştır. Bu durumda sayfa 3598'nin hak ettiği değeri aldığı görülür ve buna göre sıralamada iki sayfa arasındaki farkın daha da büyümesi gerektiğini gösterir.

İçerik skoru sorguya bağımlı olarak (guery-dependent), yalnızca dönüştürülmüş dosyalardan ve içerik indeksinden hesaplanabilir. Fakat popülarite skoru yalnızca yapı indeksi ile hesaplanır ki genellikle sorgudan bağımsızdır (query-independent) (Langville ve Meyer, 2006).

Görüldüğü gibi yaptığımız bu hesaplamalar daha çok aradığımız kelimelerin dokumanda geçip geçmediği ve kaç defa geçtiği ile ilgiliydi. Fakat kelimelerin birbirine göre olan konumlarını değerlendirmedik. Bu durum aradığımız "Bilgisayar Donanımı" konusunu anlatan web sayfalarına ulaşmamız için kaliteli sonuçlar değildir. Çünkü bilgisayar ve donanım kelimesini hem başlığında hem meta bölümünde hem de içerik bölümünde bulunduran bilgisayar donanım parçalarını

satan binlerce web sayfası vardır. Aradığımız sonuçlara ulaşmak için hesaplamalarımızı bir adım daha ileriye götürerek, kelimelerin bir birine göre konumlarını inceleyelim.

Biz sorgumuzda özellikle "Bilgisayar Donanımını" kelimelerinin birlikte geçtiği web sitelerini arıyoruz demiştik. Arama motorları da bu durumu göz önüne alarak, aranan kelimelerin dokümanda bulunduğu konumu belirten ayrı bir tablo tutarlar. Buna göre dokumanda bulunan her kelimenin kaçıncı sırada bulunduğu belirlenir ve hesaplamalar yapılırken kelimelerin bulunduğu konumlar değerlendirilir. Bu tabloya genellikle *Kelime Konumu* (Positions) adı verilir. Konunun anlaşılmasında zaman kazanmak için yine verdiğimiz örneğe geri dönelim. Aşağıdaki tablo terimlerin web sayfalarında geçtiği konumu belirtmektedir. Bir önceki tablodan farklıdır.

-Terim 1 (abajur) – 5 [1, 5, 15], 127 [10, 20], 367 [12, 19, 86, 159], 1356 [19, 50, 98]

.

-Terim 59 (bilgisayar) –96 [1, 21, 75], 198 [95], 3598 [11, 71, 121, 247], 23568 [50, 83]

.

-Terim 286 (donanım) – 96 [2, 30, 89], 252 [10, 50, 65, 59], 1265 [40, 78], 3598 [19, 81, 94]

.

-Terim 5268 (klavye) – 255 [20, 60, 72], 986 [11], 1256 [9, 50], 5987 [10, 70, 98, 159, 586]

.

-Terim n (zurna) – 782 [13, 48, 97], 1598 [18, 120, 915], 2658 [41, 81, 268]

Konum tablosuna baktığımızda terim 1 (abajur) olan kelimenin 5 numaralı sayfada 1. sırada, 5. sırada ve 15. sırada geçiyormuş. Yine aynı şekilde terim 5268 (klavye) olan kelimenin ise, 225 numaralı sayfada 20., 60. ve 72. sırada geçtiğini görüyoruz. Aradığımız "bilgisayar" ve "donanım" kelimelerine gelirsek;

-Terim 59 (bilgisayar) –96 [1, 21, 75], 198 [95], 3598 [11, 71, 121, 247], 23568 [50, 83]

-Terim 286 (donanım) – 96 [2, 30, 89], 252 [10, 50, 65, 59], 1265 [40, 78], 3598 [19, 81, 94]

Konum tablosundan görüldüğü gibi “bilgisayar” ve “donanım” kelimeleri sayfa 96 ve sayfa 3598’de birlikte geçiyor. Sayfa 96’yı incelediğimizde görüyoruz ki, “bilgisayar” kelimesi dokümanda 1.sırada yer alıyor. Yine aynı şekilde “donanım kelimesi sayfa 96’da 2. Sırada yer alıyor. Bu da demek oluyor ki bu iki kelime sayfa 96’da bir defa arka arkaya geliyor. Sayfa 96 için diğer iki konuma baktığımızda ise kelimeler birbiri ardına gelmiyor. Sayfa 3598’e baktığımızda ise “bilgisayar” ve “donanım” kelimesi hiçbir yerde arka arkaya gelmiyor. Örneğin sayfa 3598’de “bilgisayar” kelimesi 11.sıradayken, “donanım” kelimesi 19.sırada yer alıyor.

Bu hesaplamaları birden fazla kelime içinde yapabiliriz. Sadece kelimelerin birbirine olan uzaklıklarını bilmemiz yeterlidir. Mesela “İnternet Ortamında Yazarlık Dilleri” diye bir sorgu girdik. Buna göre kelimelerin sırasını birbirilerine göre belirleyebiliriz. Örneğin kelime toplamına “n” dersek, “İnternet” kelimesi “n-3” olacaktır ve aynı şekilde “yazarlık” kelimesi “n-1” olacaktır. Buna göre kelimelerin dokümanlarda bulunduğu konumlar karşılaştırılıp, daha kaliteli sorgu sonuçları üretilebiliriz.

Kelimelerin doküman içindeki konumlarının hesaplaması birçok spam dokümanlarını belirlememize olanak verir. Öyle ki site sahipleri şahsi web sitelerini sıralama da üst konumlara yerleştirmek için belirlenen zemin rengi üzerine aynı renk kelime gizleyerek, web sayfalarını kendi alanları ile ilgili kelime bombardımanına maruz bırakırlar. Genellikle kötü niyetli SOE danışmanlarının başvurduğu bu yöntem, kelime konumu yöntemleri ile kolay bir şekilde belirlenmektedir. Çünkü bu kelime bombardımanları genellikle bir mantık gözetmeksizin kelimeleri ardı ardına sıralarlar. Örneğin “donanım” kelimesini 1., 4., 5., 9. gibi sıraya rastgele koyduklarından, kelime konumlandırıcı tarafından rahatlıkla fark edilirler. Daha önce de söylediğimiz gibi, çoğu arama motoru böyle bir spamla karşılaştığında ilgili siteyi arama motorlarından tamamen kaldırarak ceza verir. Özellikle Google’ın bu anlamda hiç affı yoktur. “Banlanma” dediğimiz yasaklanma yöntemi ile Google belirlediği ilgili siteyi arama motorundan tamamen kaldırır. Bu durumda özellikle kariyer sahibi olan sitelerin gözünü korkuttuğundan, olabildiğince bu yöntemlere başvurmaktan çekinirler.

Kelime spamları ile ilgili kısa bir bilgilendirmeden sonra konumuza geri dönersek eğer bu yöntem ile elde ettiğimiz değerleri aynı şekilde başlık ve diğer kısımlara

uygulayabilirsek daha kaliteli sonuçlar üretebiliriz. Örneğimizden kısa bir bölüm alarak;

-İçerik Terim 59 (bilgisayar) –96 [1, 21, 75], 198 [95], 3598 [11, 71, 121, 247], 23568 [50, 83]

-Başlık Terim 59 (bilgisayar) –96 [1, 4], 198 [3], 3598 [2], 23568 [3]

-İçerik Terim 286 (donanım) – 96 [2, 30, 89], 252 [10, 50, 65, 59], 1265 [40, 78], 3598 [19, 81, 94]

-Başlık Terim 286 (donanım) – 96 [2, 5], 252 [0], 1265 [3], 3598 [5]

Görüldüğü gibi "bilgisayar" kelimesi sayfa 96'da başlık bölümde 1. ve 4. konumda geçmektedir. Yine aynı şekilde "donanım" kelimesi sayfa 96'da başlık bölümünde 2. ve 5. konumda geçmektedir. Bu da demek oluyor ki sayfa 96'nın başlığında "bilgisayar" ve "donanım" kelimeleri ikişer defa arka arkaya gelmektedir. Bu durum da bizim için aradığımız "Bilgisayar Donanımı" konusu için önemli bir ipucudur. Oysaki sayfa 3598'de "bilgisayar" kelimesi başlık kısmında 2. sıradayken, "donanım" kelimesi 5. sırada yer almaktadır. Böyle bir sonuçta tahminen sayfa 96'nın bilgisayar donanımından bahsederken, sayfa 3598'in bilgisayar ve donanım kelimesi içeren başka bir konudan bahsettiğini gösterir. Bu sonuçta bize aradığımız konu itibari ile sayfa 96'nın daha önemli bir konumda olduğunu gösterir. Terim konumlarını belirlemeden önceki sıralamada sayfa 3598 daha ön sıralamaya otururken bu işlemden sonra sıralamada büyük ihtimalle sayfa 96'nın gerisinde bir yerlerde konumlandırılacaktır.

Yaptığımız kelime konumu hesaplamasında ilgili kelimelerin kaç defa ardı ardına geldiği bu durumda önem kazanır. Yani bir sayfada ilgili kelimelerin bir defa arka arkaya gelmesi ile birden fazla arka arkaya gelmesi önem arz eder. Kelime konumu ile elde edilen bu hesaplama sonuçları da daha önce yaptığımız içerik skoruna eklenerek daha kaliteli sonuçlar üretilir. İçerik skoruna eklenecek kelime konum skoru her arama motoru için farklı anlam ifade eder. Demek istediğimiz içerik skoruna eklenecek her ek hesaplama, her arama motoru için değişmektedir. Her ne kadar teorik hesaplamalar daha kaliteli sonuçlar üreteceği düşünülse de deneyimsel sonuçların değerlendirmesi daha verimli sonuçlar vermektedir. Çünkü tahmin etmeye çalıştığımız insan beynidir. Yani kullanıcının girdiği kelimelerden neyi bulmak istediği kesin bir şekilde anlaşılmaz. Çünkü girilen kelimelerin anlamı kişiden kişiye

değişmektedir. Bu bağlamda içerik skorunu hesaplarken yapacağımız her ince hesaplama bazen istenilen sonuçları üretmeyebilir. Bazen basit bir Boolean mantığı kullanıcının isteğine cevap verebilir. Yani içerik skorunu hesaplama da bir eşik değeri vardır. Bu eşik değerini çok fazla aşıp ince hesaplamalara girildiğinde sonuçlar istenilenden uzaklaşabilir.

Arama motorları da bu bağlamda ürettikleri içerik skoru algoritmalarının etkili bir şekilde karşılık vermemesinden dolayı, popülarite skorunu kullanmaya başladılar. Çünkü sitelerin popülaritesi insan isteklerinin nereye doğru yoğunlaştığının bir belirtisi olarak karşımıza çıkmıştır. Arama motorları zamanla insan çoğunluğunun girilen sorgularda bulmak istedikleri arama sonuçlarını istatistiklere aktararak, daha doğal sonuçlar üretmeye başladılar. Örneğin sorgulamada "memurlar" kelimesini girdiğimizde, içerik skoruna göre memurlar ile ilgili en fazla bilgiye ve içeriğe sahip olan siteler gelmesi gerekirken, insanların en fazla girdiği siteler olan www.memurlar.net gibi siteler gelmektedir. Yani arama motorları zamanla insanların tek kelime de olsa neyi kastetmek istediklerini bulmaya çalıştı. Bu yüzden gerçekçi ve ya kaliteli bir arama motoru sıralaması için sadece içerik skoru yetmez bunun için insanların çoğunlukla görmek istedikleri sıralama listesini oluşturmak gerekir. Bunun için de popülarite skorunun hesaplanması gerekir. Bu bölümden sonraki bölümlerde de özellikle bu konu üzerine yoğunlaşacağız.

BÖLÜM III

3. Web Sayfalarını Popülariteye Göre Sıralama

Bir önceki bölümde arama motorlarının web sayfalarını sıralarken kullandıkları içerik skoruna değinmiştik. Bu bölümde ise günümüz arama motorlarının web sayfalarını sıralarken daha çok önem verdikleri popülarite skorunun nasıl hesaplandığını göreceğiz.

1998 yılına gelindiğinde web sayfalarını sıralamada kullanılan yöntemler için yeni bir çağ başladı. Bu yeni çağda web sayfalarını sıralarken içerik skorundan çok web sayfalarının karizmasına bakılmaya başlandı. Bir başka değişle web sayfalarının bilirkişiliğine bakılmaya başlandı. Karizma ve bilirkişiliğinden kastımız eğer bir konu hakkında herkes sizi gerçek, doğru, tarafsız, yetkili, bilgili, popüler, açıklayıcı gibi sıfatlarla nitelendiriyor ve gösteriyor ise siz ilgili bu alanda başvurulacak ilk kişilerdensinizdir diyebiliriz. Daha önceleri bu sıfatları arama motorları mimarisi belirlerken, yeni sistemde bu sıfatları belirlemede web dünyasının kullanıcıları belirlemeye başladı. Çünkü arama motorları bir web sitesine bu sıfatları yüklerken teorik hesaplamalarına göre belirliyordu. Oysaki yeni sistemde bu sıfatları insanların kendisi belirliyor. Böylece bu sıfatların bir web sitesine atfedilmesinde daha gerçekçi sonuçlar üretilmeye başlandı.

Bu yeni yöntemlere göre eğer siz web sitenizde ilgilendiğiniz bir konu ile ilgili bir başka siteye link veriyorsanız, link verdiğiniz sitenin önemli bir kaynak ve bilirkişi olduğunu belirtiyorsunuz demektir. Yine aynı şekilde eğer bir başka site sizin sitenize link veriyorsa sizi kaynak olarak gösteriyordur ki bu durum sizin değerinizi gösterir. Fakat şunu belirtmekte yarar var, sizi kaynak olarak kim gösteriyor? Bir başka deyişle atalarımızın dediği gibi "Yalancının şahidi şıracı" mı? Demek istediğimiz sizi kaynak olarak gösteren kişilerin sizin değerinizi artırabilmesi için öncelikle kendilerinin de değerli olması gerekir. Misal milliyet.com'un ya da bbc.com'un sizin sitenizi kaynak olarak göstermesi sıradan 100 sitenin kaynak göstermesinden çok daha değerli olabilir.

Link analiz sistemiyle ilk çalışanlardan birisi olan Jon Kleinberg, 1998 yılında HITS (Hypertext Induced Topic Search) diye adlandırdığı proje üzerinde bir dizi çalışma yapmıştır. Kleinberg IBM'de çalıştığı yıllarda arama motorları içerik skoruna göre

web sayfalarını sıralarken, Web'in link yapısını kullanarak geliştirdiği algoritma sayesinde, arama motorlarının sonuçlarını çok daha kaliteli bir noktaya taşımıştır. Yine aynı yılda Standford Üniversitesi'nde bilgisayar biliminde doktorasını yapan Sergey Brin ve Larry Page de benzer bir proje üzerinde çalışma yapmışlardır. "PageRank" olarak adlandırdıkları bu proje, Kleinberg gibi Web'in link yapısını kullanıyordu. Kleinberg tarafından geliştirilen ve ilk versiyonlarında sorgu bağımlı olan HITS algoritması, her bir web sayfasının popülarite skorunu bulmak için iç ve dış linklerin her ikisini de kullanıyordu. PageRank algoritması ise sorgudan bağımsız olarak geliştirilen baskın link analiz sistemini kullanıyordu. Sergey Brin ve Larry Page tarafından geliştirilen PageRank Google gibi devasa bir şirketin kurulumu ile hak ettiği değere kavuşurken, Jon Kleinberg tarafından geliştirilen HITS algoritması geç de olsa arama motoru olan Teoma (günümüzde www.ask.com tarafından kullanılıyor) tarafından ilerleyen yıllarda geliştirilerek kullanılmıştır.

Araştırmamızın kalan bölümleri popülarite hesaplamasında kullanılan "PageRank" sistemi üzerine yoğunlaşmıştır. Özellikle Sergey Brin ve Larry Page tarafından geliştirilen ve Google arama motoru tarafından kullanılan PageRank sisteminin matematiksel yapısına değinilmiştir.

3.1. Google PageRank Matematiği

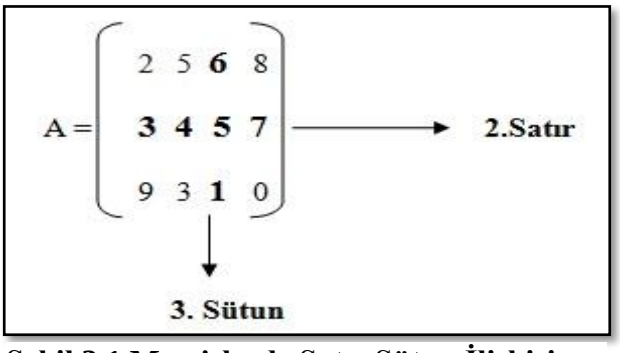
1995 yılında Standford Üniversitesi Bilgisayar Bilimi Bölümü'nde Google'ın kurucuları olan Lary Page ve Sergey Brin tanışırlar. Aynı grup içerisinde çalışmaya başlayan iki kafadar, birçok konu üzerinde tartışırlar. Lary Page bitirme projesi olan ve adına başlangıçta BackRub dediği Web'in link yapısını elde eden çalışmasıyla uğraşırken, bu çalışma Brin'in ilgisini çeker ve birlikte çalışmaya karar verirler (Wills, 2007). Çalışmaları devam ederken zamanla oluşturdukları bu yapının bir nevi arama motoru olduğunu fark ederler. Bu yüzden BackRub diye adlandırdıkları çalışmalarını Google (aslı googol olan 10^{100} sayısı) olarak değiştirirler. Birçok arama motoru şirketi tarafından arama mimarisine uygulanması zor olduğu düşünülerek dikkate alınmayan bu çalışmaya, iki kafadarın kendi kurdukları Web arama motoru şirketi olan Google ile hayat bulur.

İçeriğe göre sıralamayı belirleyen arama motoru şirketleri, zamanla Lary Page ve Sergey Brin tarafından oluşturulan PageRank algoritmasının çok daha iyi sonuçlar ürettiğinin farkına varırlar. Adını Lary Page'den alan PageRank, içerikten çok

Web'in link yapısının web sayfalarını sıralamada önemli olduğunu vurgular. Böylece içerik skoru ile popülarite skorunun birleştirilerek kapsamlı bir skorun elde edilmesiyle üretilen sonuçların, çok daha kaliteli sonuçlar üreteceği görülür. Kapsamlı skorun elde edilmesinde birçok faktör rol oynasa da en önemli kısmı PageRank hesaplamasıdır. Sonuç olarak bu başarılı çalışmanın ardından birçok araştırma, PageRank hesaplamasının geliştirilmesi üzerine yoğunlaşmıştır.

PageRank'in matematiksel çözümlemesini ayrıntılı kavrayabilmek için minimum miktarda lineer Cebir bilgimizin olması gerekir. Bu bağlamda araştırmamızda ihtiyaç duyacağımız kavramların ya da açıklamaların anlaşılması için birkaç hatırlatma yapmakta yarar var.

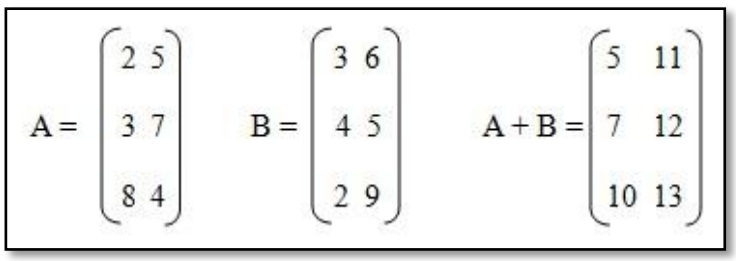
3.2. PageRank Hesaplamasında Temel Lineer Cebir İşlemleri



Şekil 3.1 Matrislerde Satır Sütun İlişkisi

m x n tane sayının, m satır ve n sütuna yerleştirilmesi ile oluşturulan tabloya matris denir. m satır n sütundan oluşan bir matrise, m x n tipinde bir matris ve a_{ij} sayılarına da matrisin öğeleri denir. i indisi i. satırda olduğunu ve j indisi ise j. sütunda olduğunu gösterir. Bu bağlamda a_{ij} nesnesi matrisin i. satır ile j. sütunun kesiştiği yerdir. Buna göre a_{23} öğesi, 2. satır ile 3.sütunun kesiştiği nokta olan 5 sayısıdır. Ayrıca A matrisi 3 satır ve 4 sütundan oluşan bir matristir.

İki matrisi toplarken yaptığımız işlem aynı satır ve aynı sütundaki sayıların üst üste gelip toplanması işlemidir. Şöyle ki,

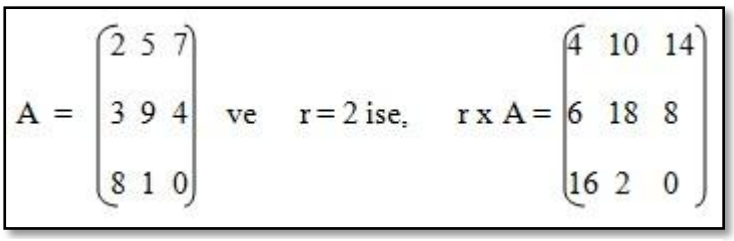


Şekil 3.2 Matrislerde Toplama İşlemi

İki matrisin toplanması için aynı tipten matrisler olduğuna dikkat etmemiz gerekir. Ayrıca aynı tipten matrisler arasında

matris toplamının, birleşme ve değişme özelliğinin olduğu bilinir.

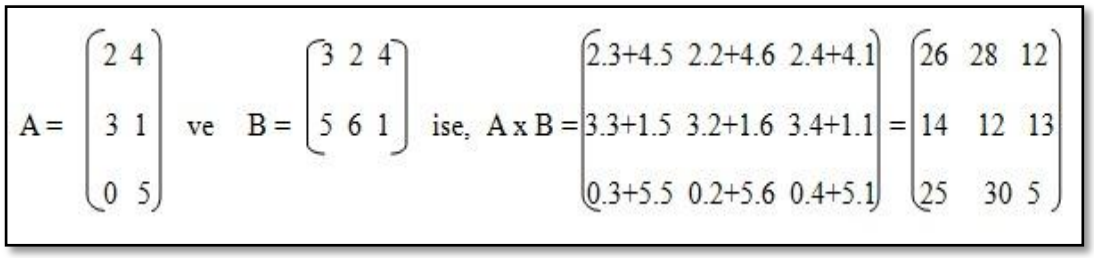
Bir matrisin "r" gibi herhangi bir sayı ile çarpılmasında ise, ilgili sayı matrisin bütün elemanları ile çarpılarak sonuca ulaşılır. Şöyle ki,



Şekil 3.3 Bir Matrisin Bir Sabit ile Çarpımı

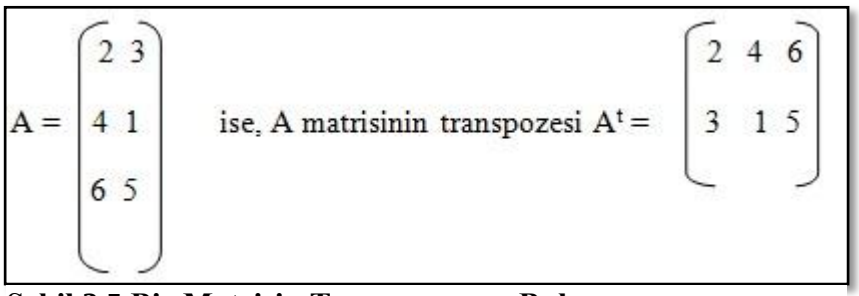
Buna göre A bir matris ve r, k birer reel sayı olsun. (r + k)A = rA + kA olur ve r(A + B) = rA + rB dir. Ayrıca (rk) A = r (kA) olur.

A ve B gibi iki matrisi çarpabilmek için ise, öncelikle A matrisinin sütun sayısının B matrisinin satır sayısına eşit olması gerekir. Ardından A matrisinin ilgili satırı B matrisinin ilgili sütunu ile çarpılıp toplanarak sonuca ulaşılır. Buna göre;



Şekil 3.4 Matrislerde Çarpma İşlemi

Bir A matrisinin satırları ile sütunlarının yer değiştirmesi ile elde edilen yeni matrise, A matrisinin transpozesi denir A^t ile gösterilir. Buna göre m x n tipindeki bir matrisin transpozesi n x m tipinde bir matristir. Şöyle ki;



Şekil 3.5 Bir Matrisin Transpozunun Bulunması

Buna göre A matrisinin transpozesinin transpozesi $(A^t)^t = A$ yine kendisidir. Ayrıca

A ve B gibi iki matrisin çarpımının transpozesi $(A \times B)^t = A^t \times B^t$ dir ve toplamının transpozesi de aynı şekilde $(A + B)^t = A^t + B^t$ dir.

T: V → V lineer dönüşümünde, x ϵ V olan sıfırdan farklı bir x vektörü için T(x) = λx eşitliğini sağlayan bir λ sayısı varsa, λ sayısına T dönüşümünün **öz değeri**, x vektörüne de λ öz değerine karşılık gelen **öz vektörü** denir (Çetin ve Orhun, 1998: 192).

Bu durumu bir örnek ile özetlemek gerekirse,

$A = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}$ gibi bir matrisin öz değer ve öz vektörünü bulmak için,

$$\text{Det}\left[\mathbf{A} - \boldsymbol{\lambda}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right] = \mathbf{0}$$

$$\text{Det}\left[\begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right] = \mathbf{0}$$

$$\text{Det}\begin{bmatrix} 1-\lambda & 0 \\ 2 & 3-\lambda \end{bmatrix} = -\lambda^2 + 4\lambda - 3 = \mathbf{0} \text{ olup, } \lambda = 3 \text{ ve } \lambda = 1 \text{ olur.}$$

Bulunan öz değerlerden λ=3 için öz vektörler bulunmak istenirse,

$$\begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = 3\begin{bmatrix} a \\ b \end{bmatrix} \text{ olup, } a = 0 \text{ ve } b = 1 \text{ bulunur.}$$

$$\text{Buna göre, } \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = 3\begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ olduğu görülür.}$$

Genelde olmasa da yukarıdaki örnekte olduğu gibi özelde, bir A matrisinin transpozu olan A^t matrisi de aynı öz değerlere sahiptir. Fakat A matrisinin transpozunun öz değerlerine karşılık gelen öz vektörler aynı olmayabilir. Yukarıdaki A matrisinin transpozu olan A^t matrisinin değeri 3 olan öz değeri için öz vektörü,

$$\begin{pmatrix}1 & 2\\ 0 & 3\end{pmatrix}\begin{pmatrix}1\\ 1\end{pmatrix} = 3\begin{pmatrix}1\\ 1\end{pmatrix}$$ olduğu görülür.

Bir A matrisinin öz değerine karşılık gelen öz vektörler birden fazla olabilir. Bu öz vektörler aynı öz değere karşılık gelmesine rağmen birbirileri arasında herhangi bir ilişki bulunmaz.

Bu bilgilerin ardından PageRank hesaplamasında sıklıkla bahsedeceğimiz matris türüne bir göz atmamızda yarar var. Kolon stokastik matris diye adlandırdığımız yapı bütün değerleri sıfırdan büyük veya sıfıra eşit olan ve ayrıca her bir kolonun değerleri toplamı 1 olan matris türüdür. Eğer bütün matris değerleri sıfırdan büyük ise bu matris türüne pozitif matris diyoruz.

$$A=\begin{pmatrix}1/2 & 0 & 1/3\\ 0 & 1 & 1/3\\ 1/2 & 0 & 1/3\end{pmatrix} \quad \text{ve} \quad A^t=\begin{pmatrix}1/2 & 0 & 1/2\\ 0 & 1 & 0\\ 1/3 & 1/3 & 1/3\end{pmatrix}$$

Şekil 3.6 Bir Matrisin Transpozu

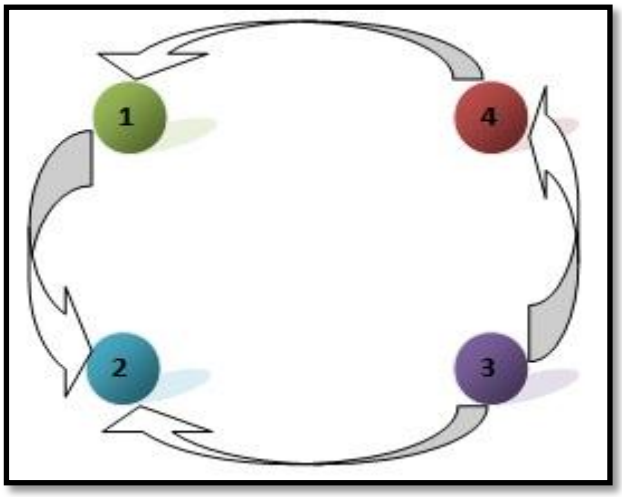
Görüldüğü gibi A matrisi girdileri pozitif olan kolon-stokastik bir matristir ama A matrisinin transpozu olan A^t matrisi kolon-stokastik değildir. Fakat A^t satırları toplamı 1 olan satır-stokastik ve öz değeri 1 olan bir matristir. Daha önce değindiğimiz gibi A ve A^t matrislerinin öz değerleri aynı olduğundan, A matrisinin de öz değeri 1 olur. Buna göre,

- Herhangi bir kolon-stokastik matrisin öz değeri 1'dir.
- Eğer A matrisi kolon-stokastik matris ise, öz değeri 1'e karşılık gelen herhangi bir öz vektörün değerleri ya yalnızca pozitif ya da yalnızca negatif değerlere sahiptir.
- Eğer A matrisi pozitif kolon-stokastik matris ise, öz değeri 1'e karşılık gelen tek bir öz vektörü vardır ve yalnızca pozitif değerleri olup, değerleri toplamı 1'dir.

3.3. PageRank Yapısında Yönlü Graflar

Graflar düğümler veya noktalardan oluşan bir yapının birbirileri arasındaki ilişkileri yansıtan nesnelerdir. Web dünyasında çoğu kez birden fazla noktanın kendi aralarında oluşturduğu ağları temsil etmekte kullanılır.

Şekilde görüldüğü gibi her bir noktaya düğüm diyoruz ve her bir düğüm bir web sayfasını temsil eder. Düğümlerden çıkan her bir ok ise linkleri temsil eder. Düğümlerden ve oklardan oluşan bu yapıya ise graf diyoruz. Çeşitli graf türleri



Şekil 3.7 Dört Düğümlü Yönlendirilmiş Graf

vardır. Fakat biz bu araştırmada özellikle yönlendirilmiş web grafları üzerinde duracağız. İnternet grafları çok büyük olmasına rağmen bütün grafların sonlu noktalardan oluştuğu düşünülerek hesaplamalar yapılabilir.

Buna göre eğer bir "i" düğümünden "j" düğümüne ya da "j" düğümünden "i" düğümüne bir ok varsa böyle bir yönlendirilmiş grafın düğümlerine bitişiktir ya da birleşiktir deriz. Ayrıca "i" ve "j" noktalarına da son noktalardır deriz. Bu durumda eğer bir ok "i" düğümünden çıkıp "j" düğümüne gidiyorsa, "i" noktasına "kuyruk" ve "j" noktasına da "baş" kısmı deriz. Bu bağlamda şeklimize baktığımızda düğüm 1 ve düğüm 2 birleşiktir çünkü 1.düğümden 2.düğüme bir ok çıkmıştır. Fakat düğüm 1'den düğüm 3'e bir ok çıkmadığından birleşiktir diyemeyiz. Bir düğüme gelen ok sayısı o web sayfasına gelen linkleri belirtir ki biz bu linklere "*iç linkler*" deriz. Aynı şekilde bir düğümden çıkan ok sayısı da o web sayfasından diğer web sayfalarına giden linkleri belirtir ki biz bu linklere de "*dış linkler*" deriz.

Şeklimiz incelendiğinde, 1.düğümden 2.düğüme bir ok çıkmıştır. Buna göre 1 numaralı web sayfası 2 numaralı web sayfasına link vermiştir deriz. Benzer şekilde 3.düğüm 2. ve 4. düğüme, 4.düğümse 1.düğüme link vermiştir. Fakat 2 numaralı web sayfası hiçbir düğüme link vermemiştir. Keza 2.düğüm 1. ve 3. düğümden link almıştır. Buna göre 2.düğümün 2 adet iç linki olduğu söylenir. Benzer şekilde

3.düğüm hiçbir web sayfasından link almamıştır. Fakat 3. düğüm 2. ve 4. düğümlere link vermiştir. Buna göre 3.düğümün 2 adet dış linki olduğu söylenir.

Sadece şeklimize bakılarak grafımız hakkında birkaç yorum yapabiliriz. Şöyle ki eğer bir web sayfası bir başka web sayfası tarafından öneriliyorsa bir başka değişle bir başka web sayfasından link alıyorsa, önerilen web sayfamızın önemli olduğunu anlarız. Bu bağlamda grafımızdaki 2 numaralı web sayfası 2 adet link aldığına göre diğerlerine kıyasla daha önemli olduğunu söyleyebiliriz. Tabi ki bu söylemlerimiz sadece graftan gördüğümüz kadarıdır. Biz biliyoruz ki bir web sayfasının önemi sadece aldığı link sayısı ile belirlenmez. Bir sayfanın sıralamadaki yerini etkileyen birçok faktör bulunur. Bu faktörlerin bir bölümünü önceki bölümlerde değinmiştik. Diğer faktörlere ise ilerleyen bölümlerde değineceğiz.

3.4. PageRank Hesaplamasına Kısa Bir Bakış

PageRank algoritması Google arama mimarisinin odak noktasıdır. Bu algoritma bir web sayfasının önemini belirlediği için, herhangi bir sorgulamada ilgili web sayfasının sonuçlar listesindeki konumunu belirleyecektir. PageRank algoritmasının mantığını bir kez daha tekrarlamamız gerekirse, "bir web sayfası ne kadar önemli ise, diğer web sayfaları tarafından o kadar link alır". Bu bakımdan PageRank algoritmasını bir seçim mantığı şeklinde düşünülebilirsiniz. Tek fark ise herkesin birden fazla kişiye oy kullanabilme hakkının olmasıdır. Siz, web sayfanızı en iyi temsil ettiğini, açıkladığını, yorumladığını düşündüğünüz web sayfalarına bağlantı veriyorsunuz. Ardından bu bağlantılar hesaplanıp, ilgili alan ve ya kelime taramasında hangi web sayfası en fazla tavsiye edilmiş ise, sıralama listesinde en üst konuma ilgili web sayfasının geçmesi sağlanır.

Bu oylama da bir noktaya daha değinmekte yarar var. Bizim ülkemizde tartışıldığı gibi çoğu ülkede de tartışılan "Üniversite mezunu bir kişi ile bir çobanın oyu denk midir?" Her ne kadar gerçek hayatta denk olarak görülüp oylama sonuçları hesaplansa da, sanal ortamda sınıfsal farklılıklar vardır. Bu farklılık oranı 1 ile 10 arasında değişen PageRank Puanıdır. Tabi ki bu sıralama da bir değer yani bir sınıf atlatmak kolay değildir. Mesela PageRank puanı 10 olan siteler parmakla sayılacak kadardır. Daha alt konumdakiler ise çoğu kez kurum ve ya kuruluşlar olmaktadır. Oylama da geçen bu sınıfsal farklılıklar tamamen ilgili web sayfasının bilirkişiliği ile belli olmaktadır. Örneğin "SBS Sınavı" şeklinde bir sorgulama girdiğimizde

www.meb.gov.tr'nin en ön sıraya gelmesi, herkes tarafından bilirkişi sayılmasındandır. Yani herkes tarafından link almasından dolayıdır. Fakat bazen konumlar değişebilir. Çünkü seçimi yapanlar insanlardır. Örneğin "KPSS Sonuçları" şeklinde bir sorgulama girdiğimizde, doğallığında karşımıza www.osym.gov.tr'yi beklerken www.memurlar.net'in gelmesine şaşmamalı. Her ne kadar sonuçlar osym.gov.tr'den yayınlansa da insanlar bu habere memurlar.net'ten ulaşmaktadır. Çünkü memurlar.net kamu personelleri ile ilgili en çok haberi barındırdığından (varsayalım) ve bundan dolayı en fazla bağlantı aldığından sıralama da osym.gov.tr'yi geçmesi doğal karşılanmalıdır. Tabi bu söylediklerimiz dinamik bir yapı üstünedir. Bu liste devamlı değişmektedir. Sonuç olarak özetlemek gerekirse, daha önce bahsettiğimiz gibi herkesin oyu bir değildir. Yani sitenize www.milliyet.com.tr'nin veya www.bbc.com'un link vermesi belki de 100 adet sıradan web sayfasının link vermesinden kat be kat avantajlı olabilir. Gelelim şimdi PageRank hesaplamasına.

Lary Page ve Sergey Brin PageRank hesaplamasını basit bir toplam hesaplaması ile özetlemişlerdir. Şöyle ki,

$$r(P_i) = \sum_{P_j \epsilon B(Pi)} \frac{r(P_j)}{|P_j|}$$

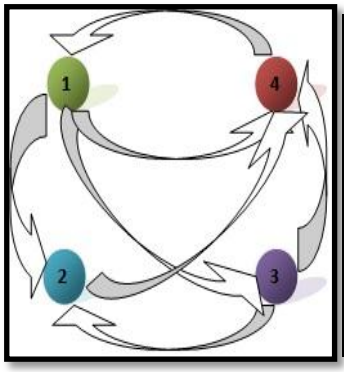
(3.1)

Formülde görüldüğü gibi P_i sayfasının PageRank'i $r(P_i)$ ile gösterilmek üzere, bütün sayfalardan P_i sayfasına gelen PageRank değerlerinin toplamıdır. $B_{(Pi)}$, P_i sayfasına gelen linkleri temsil etmek üzere, $|(P_j)|$ ise P_j sayfasından çıkan dış linklerin sayısını temsil eder. $r(P_i)$'de benzer şekilde formüldeki $r(P_j)$'nin $|(P_j)|$ ile gösterdiğimiz sayfalar tarafından derecelendirildiği iç linklerin PageRank değeridir. Bu durumda P_i sayfasına gelen iç linklerin değeri yani P_i sayfasına link veren sayfaların PageRank değeri olan $r(P_j)$'yi nasıl hesaplayacağız? Bu durumun üstesinden gelmek için Brin ve Page iteratif bir yöntem kullanmışlardır. Buna göre başlangıçta bütün web sayfalarının PageRank değerini ("n" Google indeksindeki sayfaların toplamını belirtmek üzere 1/n olarak) eşit almışlardır. Buna göre formülümüz uygulandığında indeksimizde bulunan her P_i sayfası için $r(P_i)$ değeri, her defasında $r(P_j)$'nin bir

önceki iterasyon değeri kullanılmıştır. Buna göre her "k +1" iterasyonda $r_{k+1}(P_i)$ değeri,

$$r_k(P_i) = \sum_{P_j \in B(Pi)} \frac{r_k(P_j)}{P_j} \quad \textbf{(3.2)}$$

şeklinde olur. Bu işlem başlangıçta her bir P_i sayfası için $r_0(P_i) = 1/n$ ile başlatılmıştır. Değerler ilerledikçe PageRank değeri nihai sabit değerlere ulaşacaktır. Bu hesaplamayı daha somut görmek için örnek bir graf üzerinde uygulayalım.



İter. 0	İter. 1	İter. 2	2.İter.Göre Sıralama
$r_0(P_1) = 1/4$	$r_1(P_1) = 1/4$	$r_2(P_1) = 11/24$	1
$r_0(P_2) = 1/4$	$r_1(P_2) = 5/24$	$r_2(P_2) = 3/24$	3
$r_0(P_3) = 1/4$	$r_1(P_3) = 1/12$	$r_2(P_3) = 1/12$	4
$r_0(P_4) = 1/4$	$r_1(P_4) = 11/24$	$r_2(P_4) = 8/24$	2

Şekil 3.8 Dört Düğümlü Graf **Tablo 3.1 İterasyonlara Göre Puan Dağılımı**

Görüldüğü gibi 2. iterasyon sonunda "1" numaralı web sayfası listenin en üst konumuna, 3 numaralı web sayfası ise listenin sonuna yerleşmiştir. Her ne kadar 3 numaralı web sayfası listenin 1.sinden link alsa da "1" numaralı web sayfası PageRank'ini parçalayarak dağıttığı için pek değerli olmamıştır. Benzer şekilde 4 numaralı web sayfası diğer bütün web sayfalarından link alsa da topladığı bu puanları, dışarıya verdiği tek link ile 1 numaralı web sayfasını yükseltmiş hatta listenin başına geçirmiştir.

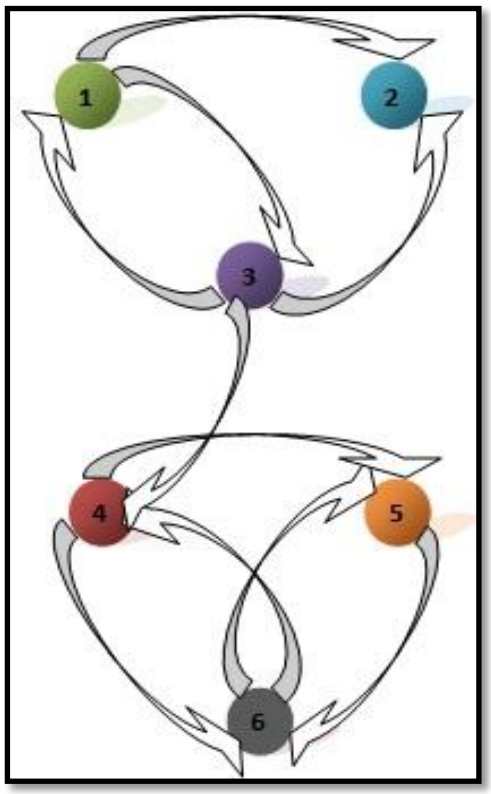
3.5. PageRank Hesaplamasında Matris Modeli

PageRank hesaplamaları birer matris problemi olarak görülebilir. PageRank hesaplamasını yukarıda verdiğimiz 1. ve 2. Epsilon formülleri ile nasıl hesaplayabileceğimizi gördük. Fakat matrisleri kullanarak her bir iterasyonda ilgili PageRank vektörünü bularak daha rahat ve anlaşılır bir şekilde hesaplayabiliriz. Ayrıca birçok işlemi de matrisler üzerinde kolaylıkla uygulayabiliriz. PageRank hesaplamasını yapmak için elimizdeki grafı matris yapısına dönüştürmek yeterlidir.

Bu işlemi somut olarak görebilmek için örnek bir graf üzerinde uygulama yapalım.

Grafımızı matrise aktarırken aşağıdaki kuralı uyguluyoruz.

$$H_{ij} := \begin{cases} 1/N_i & \text{Eğer } P_i \text{ sayfası } P_j \text{ sayfasına link vermiş ise} \\ 0 & \text{Eğer } P_i \text{ sayfası herhangi bir } P_j \text{ sayfasına link vermemiş ise} \end{cases}$$



Şekil 3.9 Altı Düğümlü Graf

Şeklimizde görüldüğü gibi yönlendirilmiş grafımız sadece 6 sayfadan oluşan Web'in çok küçük bir parçasını temsil etmektedir. Yönlendirilmiş grafımızı incelediğimizde 1 numaralı web sayfası 2 ve 3 numaralı web sayalarına link verdiğini görüyoruz. Buna göre 1 numaralı web sayfası toplandığı PageRank puanını bu iki sayfaya paylaştırıyor demektir. Her bir sayfaya "1/2" oranında pay düşecektir. Benzer şekilde 2 numaralı web sayfasına baktığımızda ise, bu web sayfası diğer hiçbir web sayfasına link vermediğini görüyoruz. Sırasıyla 3 numaralı web sayfası PageRank puanını 1, 2 ve 4 numaralı web sayfalarına "1/3" oranında, 4 numaralı web sayfası 5 ve 6 numaralı web sayfasına "1/2" oranında, 5 numaralı web sayfası 6 numaralı web sayfasına "1/1" oranında ve son olarak ta 6 numaralı web sayfası PageRank puanını 4 ve 5 numaralı web sayfalarına "1/2" oranında paylaştırıyor. Keza matrisimizde her bir P_i sayfasının link vermediği web sayfalarına "0" yerleştiriyoruz. Yani herhangi bir pay oranı vermiyoruz. Buna göre matrisimiz aşağıdaki şekilde olur.

$$H = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}$$

Buna göre her H_{ij}, P_i sayfasından P_j sayfasına yönlendirilen bağlantıyı belirtir. N_i ise her P_i sayfasından çıkan dış linklerin toplam sayısını gösterir. Yani her bir satır P_i sayfasından çıkan dış linkleri belirtirken, her bir sütun ise P_i sayfasına gelen iç linkleri belirtir.

Elde ettiğimiz değerlere göre sayfaların PageRank puanını iteratif olarak hesaplamak istersek, şöyle yazabiliriz.

$$r(P)^T_{(k+1)} = r(P)^T_{(k)}\, H, \qquad k = 0, 1, 2, \ldots$$

(3.3)

İlerleyen bölümlerde "π" ile göstereceğimiz PageRank vektörü,

$$\pi^T_{(k+1)} = \pi^T_{(k)}\, H, \qquad k = 0, 1, 2, \ldots$$

(3.4)

3.6. PageRank Yapısında "Random Walker"

PageRank algoritmasındaki yapıları ve problemleri anlayabilmek için Random Walker modelini bilmekte yarar var. Çünkü Random Walker bir sayfaya ne kadar çok uğruyor ise o sayfa o kadar link almış demektir ve bu da o sayfanın o kadar değerli olduğunu gösterir. Adından da anlaşılacağı gibi Random Walker (veya bir başka değişle sörfçü) rastgele bir sayfa seçerek hareketine başlar ve ardından seçtiği bu sayfada bulunan dış linkleri kullanarak bir başka web sayfasına geçer. Yaptığı bu hareket her geçtiği yeni web sayfası için tekrarlanır. Fakat bu harekette dikkat etmemiz gereken bir nokta var. Eğer Random Walker web sayfalarını dış linklere göre seçiyor ise, bir web sayfası ne kadar dışarıdan link (iç link) alırsa, Random Walker tarafından ziyaret edilmesi o kadar fazladır demektir. Fark edilmesi gereken bir başka durumda bu seçilme olasılığının bir önceki web sayfalarına bağlı olmamasıdır (Wills, 2007). Yani Random Walker'ın "a" sayfasından "b" sayfasına geçtiğini düşünürsek, Random Walker'ın bir sonraki adımda hangi sayfayı seçeceğini "a" sayfası etkileyemez. Böyle bir durumda Random Walker'ın bir sonraki adımını etkileyecek olan mevcut konumda bulunduğu "b" sayfası belirleyecektir. Buna göre Random Walker bir sonraki adımda "b" sayfasının bağlantı verdiği web sayfalarından birine geçecektir. Bu bağlamda Random Walker'ın bir sonraki adımda hangi sayfayı seçeceği ihtimalini "b" sayfasının PageRank puanının dağılımı belirleyecektir.

3.7. PageRank Hesaplamasında Kör Düğüm Sorunu

PageRank yapısında sayfalar arası geçişlerin nasıl sağlandığını gördük. Fakat şekildeki yönlendirilmiş grafa dikkatlice bakılacak olursa bu geçişlerde birkaç problemin yaşanacağı göze çarpacaktır. Çünkü Random Walker sayfalar arası

geçişleri dış linkler sayesinde gerçekleştirdiğini söylemiştik. Örneğin şekildeki grafımızda 2 numaralı düğüm kör düğümdür (dangling node). Çünkü 2 numaralı düğümde herhangi bir dış link yoktur. Yani herhangi bir web sayfasına bağlantı vermemiştir. Zaten matrisimizde de 2.satırın tamamı 0'lardan oluşmaktadır. Peki, başka bir düğüme geçme olasılığının sıfır olduğu Random Walker böyle bir düğüme geldiğinde yoluna nasıl devam edecektir? Bu düğümler çoğu kez resim, pdf, word gibi dosyalar olduğundan devasa web grafının da çoğunluğu böyle düğümlerden oluşmaktadır. Böyle bir durumda Random Walker ya bu sörf işlemini durduracaktır ya da yeni baştan başlayacaktır. Böyle bir çözümde Random Walker'ın performansını düşüreceğinden pek cazip bir yöntem değildir. Brin ve Page bu sorunu aşmak için PageRank algoritmasında şöyle bir yönteme başvurmuşlardır.

Eğer bir web sayfası diğer hiçbir web sayfasına bağlantı vermemiş ise, elde ettiği PageRank skoru, diğer bütün web sayfalarına eşit oranda paylaştırılır. "n" boyutlu bir matris için, bütün değerleri sıfır olan satırın her bir değeri 1/n ile değiştirilir. Bu işleme göre matris formülümüz şu şekilde olur.

$$\mathbf{S = H + (1/n)\, de^T}$$

(3.5)

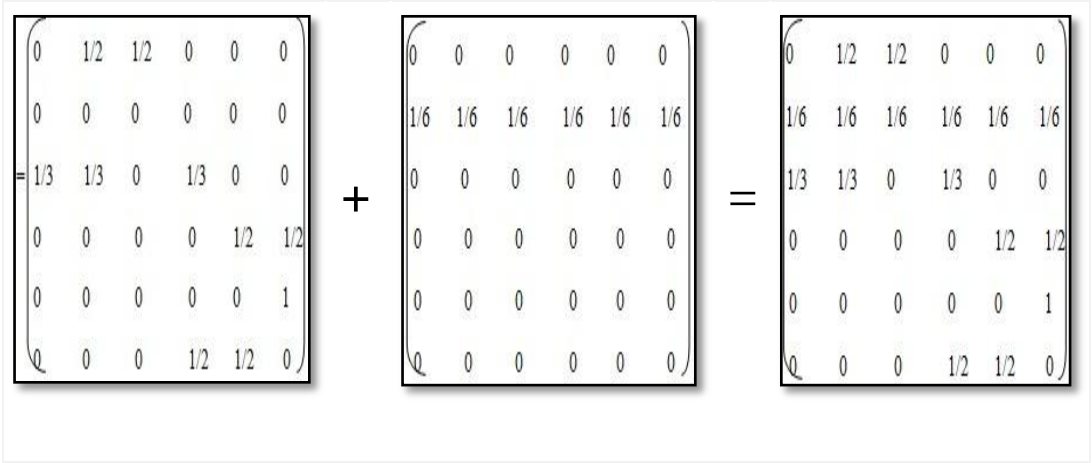
Formülümüzdeki "e", değerleri "1" olan kolon vektörü ve "d" ise H matrisindeki değerleri sıfır olan satırı tanımlayan kolon vektörüdür. Buna göre d'yi bulmak için,

$$d_i := \begin{cases} 1 & \text{Eğer } N_i=0 \text{ (Dış linkler sayısı sıfır ise)} \\ 0 & \text{Değilse} \end{cases} \quad ,i = 1, 2, \ldots, n.$$

Bu formülü şekilde verdiğimiz grafa uygularsak d_2 kolon vektörü yani sıfırlardan oluşan ikinci satırımız "1" değerini almak üzere,

$$\mathbf{S = H + (1/n)\, de^T = H + (1/6)} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \mathbf{(1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1) =}$$

Buna göre H matrisimizi de yerine koyarsak,



Sonucunu elde ederiz. Yani S'yi elde edebilmek için satır değerleri toplamı 1 olan satır stokastik bir matris kullandık. Bu sorunu çözmemize rağmen grafımıza baktığımızda halen birkaç sorunumuzun olduğunu görebiliriz. Şimdi bir sonraki probleme bir göz atalım.

3.8. PageRank Hesaplamasında Kör Alt Graflar Sorunu

PageRank algoritmasında kör düğümlerin yani dışarıya bağlantı vermeyen düğümlerin nasıl çözümlendiğini gördük. Fakat şekildeki grafımızı bölümlere ayırarak düşünürsek, Random Walker'ın yine bir çıkmaza girdiğini görebiliriz. Şekildeki grafımızda Random Walker'ın 3 numaralı düğümden 4 numaralı düğüme geçtiğini düşünürsek, Random Walker 4, 5 ve 6.düğümden oluşan bu alt grafta devamlı dönüp duracaktır. Çünkü bu üç düğüm birbirilerinin dışında dışarıya hiçbir bağlantı vermemektedir. Böyle bir durumda Random Walker alt grafta dönüp duracak ve bir üst grafa geçemeyecektir. Bu durumu aşmamız için matrisimizi indirgenemez matrise çevirerek, Random Walker'ın alt graflar arasında sonsuz bir döngüye girmesini önleyebiliriz.

"Teleportation" diye adlandırdığımız bu yöntem ile PageRank algoritmamızı indirgenemez duruma çevirip Random Walker'ın sayfalar arası geçişini küçük bir olasılıkla da olsa sağlayabiliriz. Bu yöntemi matematiksel olarak ifade etmemiz gerekirse,

$$G = \alpha S + (1-\alpha)(1/n)ee^{T} \quad \textbf{(3.6)}$$

Aynı şekilde “e” birlerden oluşan kolon vektörü, “α” ise “0” ile “1” arasında değişen ve genellikle 0.85 olarak alınan bir yumuşatma veya güç kırma (kör alt grafların gücünü kırma, hafifletme) faktörüdür (Teleportation olasılık faktörü). Bu değerin neden 0.85 olarak alındığı ilerleyen bölümlerde açıklayacağız. Bu formülü örnek grafımıza uygularsak,

$$G = 0.85 \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + (1-0.85)(1/6) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} (1\ 1\ 1\ 1\ 1\ 1) =$$

$$G = 17/20 \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + 1/40 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$G = \begin{pmatrix} 0 & 17/40 & 17/40 & 0 & 0 & 0 \\ 17/120 & 17/120 & 17/120 & 17/120 & 17/120 & 17/120 \\ 17/60 & 17/60 & 0 & 17/60 & 0 & 0 \\ 0 & 0 & 0 & 0 & 17/40 & 17/40 \\ 0 & 0 & 0 & 0 & 0 & 17/20 \\ 0 & 0 & 0 & 17/40 & 17/40 & 0 \end{pmatrix} + \begin{pmatrix} 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \end{pmatrix}$$

$$G = \begin{pmatrix} 1/40 & 9/20 & 9/20 & 1/40 & 1/40 & 1/40 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 37/120 & 37/120 & 1/40 & 37/120 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 9/20 & 9/20 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 7/8 \\ 1/40 & 1/40 & 1/40 & 9/20 & 9/20 & 1/40 \end{pmatrix}$$

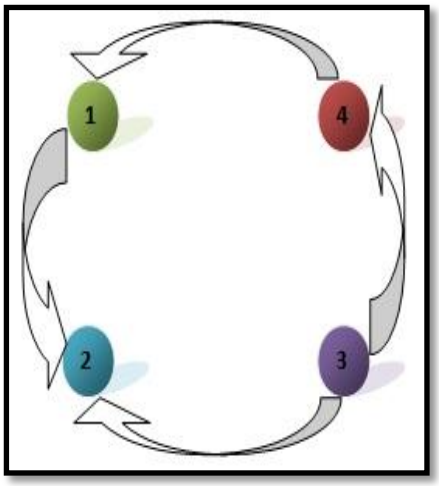
Böylece G matrisimizi elde ederek indirgenemez bir matris elde ettik. Buna göre formülümüze $(1-\alpha)(1/n)ee^T$ ekleyerek Random Walker'ın bütün sayfalar arasındaki geçişini 0.15 eşit oran olasılığıyla garantileyip, var olan olasılık değerlerine ekledik.

3.9. PageRank Vektörünün Hesaplanması

PageRank yapısındaki sorunların çözümünün ardından sıra PageRank vektörünün hesaplanmasına geldi. Bulacağımız PageRank vektörüne göre sayfaların önem sırası ortaya çıkmış olacaktır. Öncelikle birkaç bilgiyi gözden geçirelim ardından PageRank vektörünün hesaplanmasına geçelim.

Verilen bir graftaki düğümler arasında Random Walker eğer "i" noktasından "j" noktasına geçebiliyor ise, "i" noktasından "j" noktasına bir yol (path) vardır deriz ve böyle bir grafa da "*bağlı graf*" (connected graph) deriz. Eğer grafımızda Random Walker herhangi bir "i" noktasından herhangi bir "j" noktasına geçebiliyor ise böyle graflara da "sıkı bağlı graf" (strongly connected graph) deriz. Üzerinde işlem yaptığımız graf ilk önce bağlı graf olmasına rağmen (düğüm 1'den 2'ye geçiş varken, düğüm 2'den herhangi bir düğüme geçiş yok) yaptığımız işlemler sonucunda sıkı bağlı bir grafa (kör düğüm ve alt kör düğümlerin çözümünün ardından) çevirdik. Böylece Random Walker grafımızdaki herhangi iki düğüm arasında geçiş yapabilir düzeye gelmiş oldu. Bu durumu matrisler açısından düşünürsek eğer, pozitif bir "k" değeri için, ("k" defa matrisimizin çarpımı, yani A matrisi için A^k .) $B = I + A + A^2 + A^3 + \ldots + A^k$ değeri pozitif ise matrisimiz "*sıkı bağlı*" diyebiliriz. Buradaki birim matrisin eklenmesi düğümlerin kendilerine olan döngüleridir. Buna göre eğer "k" defada "i" noktasından "j" noktasına bir yol var ise, "i" noktasından "j" noktasına geçebiliriz demektir. Eğer B matrisimizin bütün değerleri pozitif olur ise herhangi bir

"i" noktasından "j" noktasına geçebiliriz demektir ve bu da grafımızın "sıkı bağlı" olduğunu gösterir.



Örneğin daha önce incelediğimiz yandaki şeklimize bir göz atalım. Her "i" noktasından her "j" noktasına geçişe "1" dersek matrisimiz;

$$A=\begin{pmatrix}0&1&0&0\\0&0&0&0\\0&1&0&1\\1&0&0&0\end{pmatrix}$$

olsun. Buna göre matrisimiz bağlı bir graftır fakat sıkı bağlı bir graf değildir. A matrisimize göre B matrisimizi hesaplarsak,

Şekil 3.10 Dört Düğümlü Graf

$$B=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}+\begin{pmatrix}0&1&0&0\\0&0&0&0\\0&1&0&1\\1&0&0&0\end{pmatrix}+\begin{pmatrix}0&0&0&0\\0&0&0&0\\1&0&0&0\\0&1&0&0\end{pmatrix}+\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&0&0\end{pmatrix}=\begin{pmatrix}1&1&0&0\\0&1&0&0\\1&2&1&1\\1&1&0&1\end{pmatrix}$$

Görüldüğü gibi B matrisimizin bütün değerleri sıfırdan büyük olmadığından pozitif değildir. Yani her bir "i" düğümünden "j" düğümüne bir yol olmadığından B matrisimiz sıkı bağlı bir matris değildir.

G matrisimizin PageRank vektörünü bulmadan önce G matrisimizin işlenmemiş hali H matrisine bir göz atarak oluşacak sıralama hakkında tahminler yürütebiliriz. Ardından PageRank vektörünü bularak yürüttüğümüz tahminlerle eşleşip eşleşmediğine değinebiliriz. Şekildeki grafımızın yapısından da görüldüğü gibi 1.,2., ve 3.düğümün kendi arasında, 4.,5., ve 6.düğümü de kendi arasında değerlendirebiliriz. 1, 2 ve 3.düğümden oluşan üst grafımıza baktığımızda 2 numaralı düğümün 1. ve 3. düğümden link aldığı ve dışarıya herhangi bir link vermediğini görüyoruz. Buna göre 2 numaralı düğümün diğer iki sayfadan da link aldığına göre diğer 1. ve 3. düğümden daha değerli olduğunu düşünebiliriz. 1 ve 3 numaralı düğümlere geldiğimizde ise 1 topladığı PageRank puanının 1/2'sini 3 numaralı düğüme verirken, 3 numaralı düğüm topladığı PageRank puanının 1/3'ünü 1 numaralı düğüme vermektedir. Ayrıca 1 ve 3 numaralı düğümler sadece birbirilerinden link aldıklarına göre 3 numaralı sayfanın PageRank değeri 1 numaralı

sayfaya göre daha yüksek olacaktır. Sonuç olarak PageRank sıralaması bu üç sayfa için 2 > 3 > 1 olacaktır.

Grafımızın alt grafını oluşturan 4, 5 ve 6.düğümlere geldiğimizde ise kabataslak baktığımızda 6.düğümün 4 numaralı düğümün PageRank puanının 1/2'sini alırken 5 numaralı düğümün PageRank puanının tamamını aldığını görüyoruz. Buna göre 6 numaralı düğüm diğer iki düğümden daha değerlidir. 4 ve 5 numaralı düğümleri kendi arasında kıyasladığımızda 4 numaralı düğüm 3 ve 6 numaralı düğümlerden link alırken, 5 numaralı düğüm 4 ve 6 numaralı düğümlerden link almaktadır. Her iki düğümde 6 numaralı düğümden link aldığına göre geriye 3 ve 4 numaralı düğümlerden gelecek PageRank puanına bağlı kalmaktadır. Eğer 3 > 4 ise önem sıralamamız 6 > 4 > 5 olacaktır. Eğer 4 > 3 ise sıralamamız 6 > 5 > 4 olacaktır. Gelin PageRank vektörünü hesaplayarak yürüttüğümüz tahminlerin eşleşip eşleşmediğini kontrol edelim.

H matrisi üzerinden tahminler yaptıktan sonra üzerinde işlem yaptığımız G matrisimize bir göz atalım. G matrisimizin bütün değerleri 0'dan büyük olduğundan her "i" noktasından "j" noktasına bir yol vardır. Buna göre G matrisimizdeki düğümleri önem sırasına göre sıralamak için daha önce de yaptığımız gibi her bir düğüme eşit oranda bir başlangıç değeri verelim. Altı adet düğümümüz olduğuna göre her bir sayfaya başlangıçta 1/6 düşecektir. Bu orandan başlayarak her bir iterasyonda değerler değişecek ve nihai sonuçlara ulaşılacaktır. Daha önce de bölüm "2.3" de bahsettiğimiz gibi bir sonraki iterasyonda ilgili sayfaya gelen iç linklerin değerleri hesaplanacak ve sayfaların yeni değerleri belirlenecektir. Yaptığımız bu işlem G matrisimizi "π^T" (daha önceleri $r(P_i)^T$ ile gösterdiğimiz PageRank vektörü) ile çarpmak ile aynıdır. Buna göre 1.adımda önem vektörümüz (importance vector);

$$\pi^T_1 = \pi^T G, \tag{3.7}$$

olacaktır. Bir sonraki adımda önem vektörümüz $\pi^T_2 = (\pi^T G)G$ olacaktır. Nihai sonuçlara ulaştığımızda önem vektörümüzün değişmediğini göreceğiz. Buna göre 1.adımımızı gerçekleştirmek istersek,

π^T =	(1/6 1/6 1/6 1/6 1/6 1/6)	G =	1/40 9/20 9/20 1/40 1/40 1/40 1/6 1/6 1/6 1/6 1/6 1/6 37/120 37/120 1/40 37/120 1/40 1/40 1/40 1/40 1/40 1/40 9/20 9/20 1/40 1/40 1/40 1/40 1/40 7/8 1/40 1/40 1/40 9/20 9/20 1/40
$\pi^T_1 = \pi^T$ G = (0,095833333 0,166666667 0,119444444 0,166666667 0,190277778 0,261111111)			

Olduğunu görürüz. Buna göre bir sonraki adımda vektörümüz $\pi^T_2 = (\pi^T_1)$G olur. Bu iterasyonu devam ettirirsek nihai "π^T_*" vektörünü elde ederiz. Böylece PageRank vektörümüzü elde ederiz. Aşağıda nihai "π^T_*" vektörü gösterilmektedir.

π^T_2	0,082453704	0,12318287	0,089340278	0,193425926	0,230416667	0,281180556
π^T_3	0,067763985	0,102806809	0,077493731	0,187265721	0,244158661	0,320511092
π^T_4	0,061520855	0,090320549	0,068363992	0,197738069	0,255369444	0,326687092
π^T_5	0,057165209	0,083311572	0,063941774	0,196007223	0,260676104	0,338898118
π^T_6	0,054919309	0,079214523	0,061097686	0,198951009	0,264137242	0,341680231
π^T_7	0,053533069	0,076873775	0,059562764	0,198747167	0,265990334	0,345292892
π^T_8	0,052766568	0,075518122	0,058642006	0,199516047	0,267107476	0,346449781
π^T_9	0,052313636	0,074739427	0,058124192	0,199554793	0,267733878	0,347534075
π^T_{10}	0,052056607	0,074289902	0,057821381	0,199758589	0,268100854	0,347972668
π^T_{11}	0,051907127	0,074031185	0,05764846	0,199795511	0,268310187	0,348307529
π^T_{12}	0,051821482	0,073882011	0,05754828	0,199852182	0,268431543	0,348464502
π^T_{13}	0,051771964	0,073796094	0,057490748	0,199869378	0,268501209	0,348570607
π^T_{14}	0,051743492	0,073746577	0,057457531	0,199886	0,26854144	0,34862496
π^T_{15}	0,051727066	0,07371805	0,057438416	0,199892673	0,26856459	0,348659206
π^T_{16}	0,051717608	0,073701611	0,057427393	0,199897771	0,268577939	0,348677678
π^T_{17}	0,051712156	0,07369214	0,057421045	0,199900169	0,268585627	0,348688862
π^T_{18}	0,051709016	0,073686682	0,057417386	0,199901782	0,268590058	0,348695075
π^T_{19}	0,051707206	0,073683538	0,057415278	0,199902613	0,268592611	0,348698754
π^T_{20}	0,051706163	0,073681726	0,057414064	0,199903134	0,268594082	0,348700831
π^T_{21}	0,051705563	0,073680682	0,057413364	0,199903416	0,26859493	0,348702046
π^T_{22}	0,051705216	0,073680081	0,057412961	0,199903586	0,268595418	0,348702738
π^T_{23}	0,051705017	0,073679734	0,057412728	0,199903681	0,268595699	0,348703141
π^T_{24}	0,051704902	0,073679534	0,057412595	0,199903737	0,268595861	0,348703371
π^T_{25}	0,051704836	0,073679419	0,057412517	0,199903768	0,268595955	0,348703504
π^T_{26}	0,051704798	0,073679353	0,057412473	0,199903787	0,268596009	0,348703581
π^T_{27}	0,051704776	0,073679315	0,057412447	0,199903798	0,26859604	0,348703625

π^T_{28}	0,051704763	0,073679293	0,057412433	0,199903804	0,268596058	0,348703651
π^T_{29}	0,051704756	0,07367928	0,057412424	0,199903807	0,268596068	0,348703665
π^T_{30}	0,051704751	0,073679273	0,057412419	0,199903809	0,268596074	0,348703674
π^T_{31}	0,051704749	0,073679268	0,057412416	0,19990381	0,268596077	0,348703679
π^T_{32}	0,051704748	0,073679266	0,057412415	0,199903811	0,268596079	0,348703681
π^T_{33}	0,051704747	0,073679265	0,057412414	0,199903811	0,26859608	0,348703683
π^T_{34}	0,051704746	0,073679264	0,057412413	0,199903812	0,268596081	0,348703684
π^T_{35}	0,051704746	0,073679263	0,057412413	0,199903812	0,268596081	0,348703684
π^T_{36}	0,051704746	0,073679263	0,057412413	0,199903812	0,268596082	0,348703685
π^T_{37}	0,051704746	0,073679263	0,057412413	0,199903812	0,268596082	0,348703685
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
π^T_*	0,051704746	0,073679263	0,057412413	0,199903812	0,268596082	0,348703685

Tablo 3.2 Artan İterasyonlara Göre PageRank Vektörü

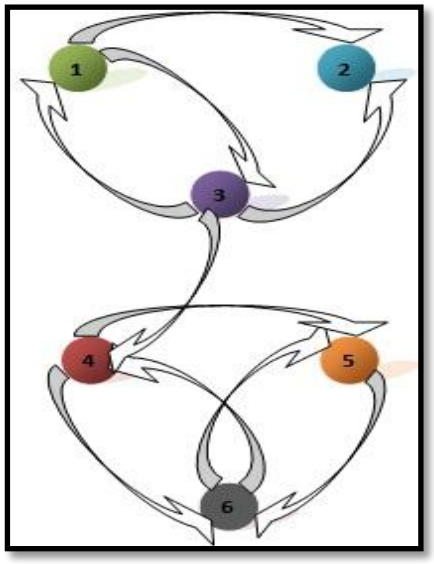
Görüldüğü gibi 36.iterasyonda değerler sabitlenmiştir. Buna göre π^T_* vektörümüz,

π^T_* = 0,051704746 0,073679263 0,057412413 0,199903812 0,268596082 0,348703685

olarak karşımıza çıkmıştır. Elde ettiğimiz π^T_* vektörüne göre üst graftaki düğümler arası sıralama 2 > 3 > 1 çıkmıştır. Yani daha önce yaptığımız tahmin doğru çıkmıştır. Alt grafta ise sıralamamız 6 > 5 > 4 çıkmıştır. Böylece yaptığımız tahmindeki 4 > 3 koşulu doğru çıkmıştır. Elde ettiğimiz π^T_* vektörüne göre 6 düğüm arasındaki sıralama ise 6 > 5 > 4 > 2 > 3 > 1 çıkmıştır. Çıkan sonuca göre en değerli sayfamız 6 numaralı düğüm olurken, en değersiz sayfamız 1 numaralı düğüm olmuştur. Buna göre çıkan sonucu yorumlayacak olursak, Random Walker'ımız % 5.170 olasılıkla 1 numaralı sayfayı ziyaret ederken % 34.870 olasılıkla 6 numaralı sayfayı ziyaret edecektir.

3.10. PageRank Hesaplanmasında Markov Zincirlerinin Yeri

Stokastik süreçlerin özel bir çeşidi olan Markov Zincirleri özellikle arama motorları başta olmak üzere birçok mühendislik alanında kullanılmaktadır. Markov zincirlerinin genel esprisi bir sonraki gerçekleşecek durumun sadece mevcut durumdan etkilenmesidir. Yani gerçekleşecek durum mevcut durumdan önce gerçekleşen durumlardan etkilenmez.



Şekil 3.11 Altı Düğümlü Graf

Bu durumu şekildeki temsili web grafımızı ele alıp değerlendirerek markov zincirlerinin arama motorları için önemini gösterelim. Varsayalım ki Random Walker'ımız 3.düğümden 4.düğüme geçti. Bu durumda 3.düğümden 4.düğüme geçen Random Walker'ımızın bir sonraki adımda 5. ve ya 6.düğüme geçme olasılığını 3. düğüm etkilemez. Çünkü bir sonraki geçiş olasılığını 4 nolu düğümde bulunan PageRank puanının paylaştırılması belirler ve bu olasılığı 3 nolu düğüm değiştiremez. Yani 4 nolu düğüme gelen Random Walker'ın nerden geldiği 5. ve 6.düğümü bağlamaz. Bu düğümler için önemli olan Random Walker'ın 4.düğümde bulunmasıdır.

3.10.1. Markov Zincirlerinde Graf Teorisi

Markov zincirleri aslında graf teorisinin birer uygulamasıdır. Graf teorisi de daha önce bahsettiğimiz gibi durumların düğümler (nodes) ile gösterildiği ve durumlar arası geçişinde oklar (edges) ile ifade edildiği graflar bütünüdür. Buna göre markov zincirleri bir durumun belirli istatistiksel değerlere göre değişip değişmeyeceğini belirler. Fakat mevcut değişimin gerçekleşmesi geçmiş durumlardan bağımsızdır. Yani mevcut durumu sadece bir önceki durum belirlerken, gelecek durumları da sadece mevcut durum belirler. Bu bağlamda "t" zamanındaki bir durum yalnız ondan bir önceki (t-1) zamanındaki durumuna bağlı olup, daha önceki zamanlardaki durumlara bağlı değil ise, böyle bir sürece Zamana Bağlı Markov Süreci denir ("Stokastik Matematiksel Modeller ve Süreçler", 2012). Markov modellerinin istatistiksel olma özelliğinden dolayı gerçekleşecek her bir stokastik olayın gerçekleşme olasılığını aşağıdaki formül ile gösterebiliriz.

$$P(x_{t+1} = x_{t+1} \mid x_t = x_t, x_{t-1} = x_{t-1}, \ldots, x_0 = x_0) = P(x_{t+1} = x_{t+1} \mid x_t = x_t) \quad \textbf{(3.8)}$$

Bir Markov zincirinde, belli bir t anında yapı *i* durumunda iken, onu takip eden dönemde *j.* durumda bulunma ihtimaline geçiş olasılığı denmektedir ve zamana bağlı olarak şöyle ifade edilebilir.

$P_{ij}^{t,t+1} = P(\pi(t+1) = j, \pi(t)=i)$

(3.9)

Buna göre bir markov zincirinde bir t zamanındaki herhangi bir *i* durumundan t+1 zamanındaki bir *j* durumunda şartlı bulunma olasılığı,

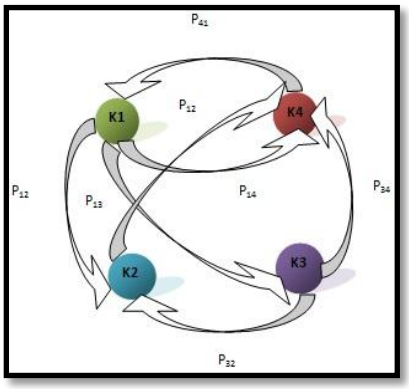
$$P_{ij} = \frac{P(i\ den\ j'\ ye\ geçme\ süreci)}{P(i\ durumundaki\ süreç)} \quad \text{olup}, P = P_{ij}, \quad i = 1,2,\ldots N\ ,\ j = 1,2,\ldots N$$

(3.10)

kare matrisi ile gösterilebilir. Belirttiğimiz formüle göre olasılıklı geçiş matrisi de aşağıdaki gibi olur

	Mevcut Durum	Gelecek Durum				
		1	2	3	. .	N
	1	P_{ii}	.	.	. .	P_{iN}
	2	.	.	.	. .	.
P =	3	.	.	.	. .	.
	.	.	.	.	. .	.
	.	.	.	.	. .	.
	N	P_{iN}	.	.	. .	P_{NN}

N=0 mevcut durumu yansıtmak üzere, verilen bir durum için sürecin alabileceği bütün durumlar her bir satırda belirtilmektedir. Buna göre P geçiş matrisinde "i" satır numarasını belirtmek üzere, π_i geçiş matrisinin satır vektörü olur. π_i satır vektörü bir olasılık vektörünü belirttiğine göre, π_3 üçüncü satırı temsil eder. Buna göre P geçiş matrisi π_i olasılık vektörlerinden oluşur ve $\pi_{i'}$'nin her bir elemanı bir durumdan bir başka duruma geçiş olasılığını verir. Bu her bir geçiş olasılığını "K" ile gösterirsek π_i sonlu sayıda geçiş ihtimallerinden oluşur. Bu sistemi web graflarımız üzerinden göstermek istersek,



K_1, K_2, K_3, K_4 düğümlerinden oluşan muhtemel durumlara sahip N=4 durumlu bir sistem düşünelim. Buna göre P_{ij} gösterimlerinin her biri ilgili durumun meydana gelme olasılığını tanımlar. Örneğin P_{32} elemanı K_3 ile başlayan bir durumun K_2 durumuna geçmesi ihtimalini verir.

Şekil 3.12 Dört Düğümlü Bir Graf

3.10.2. Web Grafların Markov Zinciri ile Formülize Edilmesi

Mevcut Durum (n=0)	Gelecek Durum				
	K_1	K_2	K_3	K_4	(K_j)
K_1	0	1/3	1/3	1/3	
K_2	0	0	0	1	
K_3	0	1/2	0	1/2	
K_4	1	0	0	0	
(K_i)					

Tablomuz Random Walker'ın halen bulunduğu K_i düğümünden bir sonraki adımda K_j düğümüne geçme ihtimalini göstermektedir. Tablodaki bilgileri geçiş matrisi ile göstermek istersek,

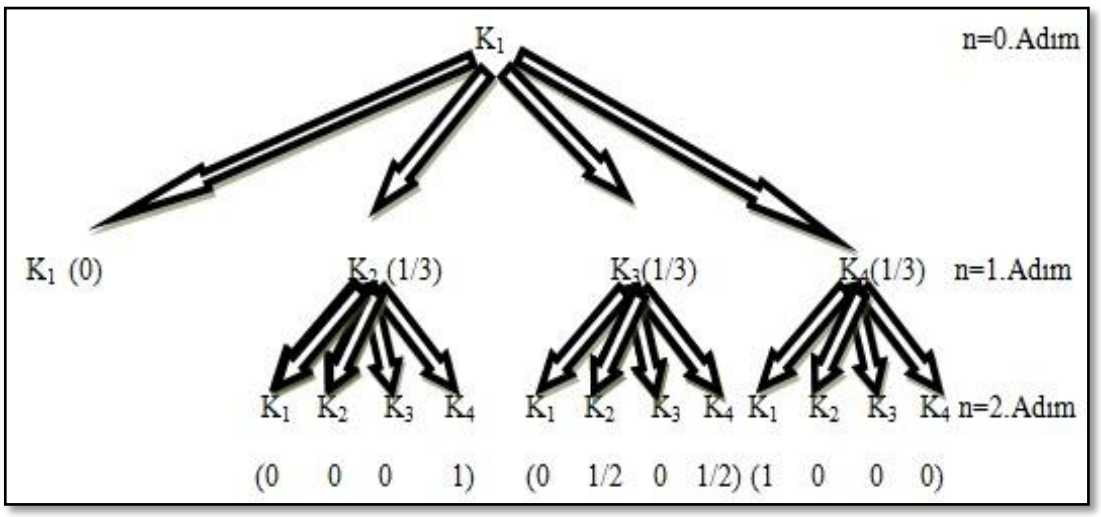
Buna göre geçiş matrisimiz

$$P = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \\ 0 & 1/2 & 0 & 1/2 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

- $0 \le P_{ij} \le 1$
- $\sum_{j=1}^{4} P_{ij} = 1 \qquad (i = 1,2,3,4)$

Şartlarını gerçekleştirir.

Buna göre Random Walker'ımızın K_1 numaralı düğümden K_2 numaralı düğüme 2 adım sonra geçme ihtimalini hesaplayalım. Aşağıdaki şekil Random Walker'ın 1 numaralı düğümden 2 numaralı düğüme 2 adımda geçişini göstermektedir. Buna göre Random Walker'ımızın 1 numaralı düğümden 2 numaralı düğüme geçişi şartlı olasılıklarının toplamı olarak bulunur. Şöyle ki,



Şekil 3.13 Random Walker'ın İki Adım Sonraki Durumu

$K_1 \rightarrow K_1 \rightarrow K_2$ $K_1 \rightarrow K_2 \rightarrow K_2$ $K_1 \rightarrow K_3 \rightarrow K_2$ $K_1 \rightarrow K_4 \rightarrow K_2$

(0) (1/3 x 0 = 0) (1/3 x 1/2 = 1/6) (1/3 x 0 =0)

Şartlı ihtimallerin toplamı=0 + 0 + 1/6 + 0 = 1/6 olur. Böylece K_1 numaralı düğümde bulunan Random Walker'ın 2 adım sonra K_2 numaralı düğüme geçme ihtimali 1/6 olur. Buna göre K_1 numaralı düğümde bulunan Random Walker'ın 1 adım sonra K_2, K_3, ve K_4 bulunma olasılıkları şu formülle özetlenebilir.

$$\pi_i^{n+1} = \pi_i^n \text{ x } P \qquad \textbf{(3.11)}$$

Formüle gör $\pi = (p_1, p_2, p_3)$ olasılık vektörünü ifade eder. Böylece n=2 için,

$$\pi_1^2 = \pi_1^1 \text{ x } P = (0 \;\; 1/3 \;\; 1/3 \;\; 1/3) \text{ x } \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \\ 0 & 1/2 & 0 & 1/2 \\ 1 & 0 & 0 & 0 \end{pmatrix} = (0 \;\; 1/6 \;\; 0 \;\; 1/6)$$

Buna göre K_1 numaralı düğümde bulunan Random Walker'ın 2 adım sonra;

K_2'de bulunma olasılığı = 1/6,

K_3'de bulunma olasılığı = 0,

K_4'de bulunma olasılığı = 1/6 olur.

Random Walker'ın 2 adım sonraki bütün geçiş ihtimallerini belirtmek istersek P matrisinin karesini almamız yeterli olacaktır.

$$P^2 = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \\ 0 & 1/2 & 0 & 1/2 \\ 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \\ 0 & 1/2 & 0 & 1/2 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1/6 & 0 & 1/6 \\ 1 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 \end{pmatrix}$$

Görüldüğü gibi Random Walker'ın 2 adım sonraki bütün geçiş ihtimallerinde bazı düğümler arası geçiş ihtimali sıfırdır. Devasa web grafında bu durum Random Walker'ın hareketini kısıtladığından daha önce bahsettiğimiz önlemler alınarak giderilir. Kör düğüm ve kör alt graflar diye adlandırdığımız bu durumlar Random Walker'ın özgür hareket etmesini engeller.

Web arama motorları için zamana bağlı markov zincirleri görüldüğü gibi biçilmiş kaftandır. Özellikle Random Walker'ın web grafları üzerindeki hareketini anlamak için markov zincirlerini bilmekte yarar vardır. Bir sonraki bölümde aşama aşama elde ettiğimiz Google PageRank formülünün detaylarına girilecektir. Böylece formülde yer alan değerlerin hassasiyeti ölçülecektir.

BÖLÜM IV

4. PageRank Modelindeki Parametreler

1998'de Sergey BRIN ve Lary PAGE PageRank modelini sunarken markov zincirlerinden bir haberdiler. İlginç olan yayınladıkları hiçbir yayında bir defa bile markov zincirlerinden bahsetmemeleriydi. Her ne kadar sonradan PageRank modelinin orijininin markov zincirlerine dayandığını fark etseler de, markov zinciri araştırmacıları PageRank modelinin markov zincirlerinin birer uygulaması olduğunu çakmış ve hemen bu alanda markov zincirlerinin PageRank modeli üzerindeki etkisini araştırmak ve geliştirmek için çalışmalarını başlatmışlardır.

PageRank modelindeki parametreleri incelemeden önce adım adım geliştirdiğimiz matrislerimizi ve matrislerimizdeki parametreleri hatırlayalım.

H → boşlukların (0'lar) çok olduğu, satır alt stokastik bir matristir.

S → boşluklu, stokastik, indirgenebilen bir matris.

G → Google Matrisi diye adlandırdığımız tamamen boşluksuz, stokastik, ilkel matris.

E → $\{(1/n)ee^T\}$ tamamen boşluksuz, Teleportation olasılık matrisi.

n → H, S, G, E matrisindeki toplam sayfa sayısı.

α → "0" ile "1" arasında değişen ve genellikle 0.85 olarak alınan bir hafifletme faktörüdür.

π^T → PageRank vektörü diye adlandırdığımız G'nin sabit satır vektörüdür.

4.1. PageRank Formül Yapısında "H" Matrisinin İncelenmesi

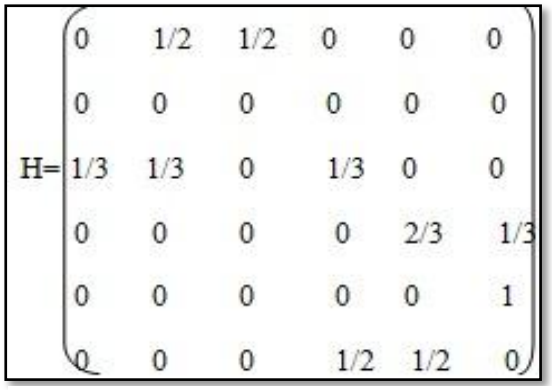
$$H = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ Q & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}$$

Şekil 4.1 "H" Matrisinin Yalın Hali

Sayfaların PageRank puanını iteratif olarak hesaplarken kullandığımız $\pi^{T}_{(k+1)} = \pi^{T}_{(k)}$ H formülümüzde dikkat etmemiz gereken birkaç nokta vardır. Bunlar,

- N büyüklüğündeki H matrisimizin her bir iterasyonunda çarpımının alınması gerekir.
- H matrisimiz çok fazla boşluklu yapıya sahiptir. Yani sıfır değerlerinin oranı çok fazladır.
- Eşitlikteki iteratif süreç basit sabit lineer süreçtir. Yani güç metodunun (power method) "H" matrisine uygulanmasıdır.
- "H" matrisi ilk bakışta markov zincirleri için stokastik geçişli olasılık matrisidir. Matrisimiz baktığımızda kör düğümlerin dışındaki bütün satırların satır stokastik olduğu görülür. Bu bağlamda "H" matrisine alt stokastik matristir diyebiliriz.

"H" matrisine baktığımızda her bir sayfa PageRank puanını dışarıya verdiği linklere eşit oranda paylaştırmaktadır. Peki, bu durum sıralama için ne kadar sağlıklıdır? Bir web sayfasının PageRank puanını link verdiği web sayfalarına eşit oranda paylaştırmak, bütün web sayfalarını eşit olarak görmek demektir. Örneğin 4.düğümümüz PageRank puanını bağlantı verdiği 5. ve 6.düğümlere eşit oranda paylaştırmıştır. Peki, 4.düğüm 5.düğümü daha çok önemsiyor ise ne olacak? İkinci bölümde tartıştığımız web sayfalarının içerik skorlarına göre web sayfaların benzerlikleri çok büyük değerde ise ne yapılabilir? Buna göre her bir web sayfasının dışarıya verdiği bağlantıların ilgili web sayfası ile benzerliği hesaplanarak, PageRank puanının paylaştırılma oranı değiştirilebilir. Misal "H" matrisimizi buna göre uyarladığımızı düşünelim. Şöyle ki,

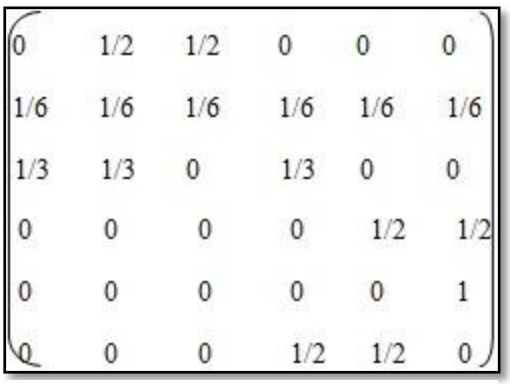


Şekil 4.2 "H" Matrisinin PageRank Puanının Paylaştırılmış Hali

Görüldüğü gibi "H" matrisimizdeki 4 numaralı düğüm PageRank puanının 2/3'nü 5 numaralı web sayfasına verirken, 1/3'nü 6 numaralı web sayfasına vermiştir. Bu oran 4 numaralı web sayfasının bağlantı verdiği web sayfaları ile benzerliğinden bulunmuştur. Tabi ki bir web sayfasının PageRank puanının paylaştırılma oranı bundan ibaret olmayabilir. Bağlantı verdiği web sayfalarının içerikteki kullanma sıklığına da bakabilir. Örneğin 4 numaralı web sayfası içerdiği konu üzerine kaynak olarak 10 defa 5 numaralı web sayfasını gösterirken 1 defa 6 numaralı web sayfasını göstermesi durumunda 4 numaralı web sayfasının PageRank puanını bu her iki web sayfasına da eşit oranda dağıtmak ne kadar adaletli olacaktır?

Ayrıca 4 numaralı web sayfasının yayın hayatına yeni başlamış bir web sayfasına link verdiğini düşünürsek, köklü web sayfaları ile aynı oranda pay alması ne kadar mantıklı olacaktır. Bu ve buna benzer sorunlardan dolayı birçok Random Walker modeli üretilmiştir. Çeşitli durumları göz önüne alarak sayfaları seçen "Akıllı Random Walker'lar" günümüzde giderek artmaktadır. Buna göre PageRank paylaşımını birçok kıstasa göre belirleyen webmaster'lar giderek daha spesifik matrisler üretmeye başlamışlardır.

4.2. PageRank Formül Yapısında "S" Matrisinin İncelenmesi



"H" matrisinde bulunan kör düğümlerin giderilmesi için "0" olan satır değerlerini $\mathbf{(1/n)e^T}$ ile yenileyerek, "H" matrisini stokastik bir yapıya dönüştürdük. Böylece Random Walker herhangi bir sayfada asılı kalmayacak ve hareketine devam edecektir.

Şekil 4.3 "S" Matrisi

S = H + (1/n)de^T Formülünü incelediğimizde, “d” ye kör düğüm vektörü diyebiliriz. Buna göre dış linklerin sayısı “0” ise, vektörün girdi değeri “1”, değil ise “0” değerini alacağını belirtmiştik. Yaptığımız bu *stokastik işlem* aslında markov zincirleri için geçişli olasılık matrisini işaret eder. “S” matrisimiz bu hali ile halen arzulanan yapıya dönüşmediği için, bir diğer işlem olan *ilkel işlemi* devreye soktuk. Çünkü Random Walker’ımız grafımızdaki alt kör graflar dediğimiz yapılara düşüyor ve sonsuz bir döngüye giriyordu. Bu yapıları göz önüne alarak uyguladığımız ilkel işlem ile matrisimizi indirgenemez ve periyodik (düzenli aralıklar ile tekrarlanmayan) olmayan bir yapıya dönüşmesini sağladık. “S” matrisimizi ilkel matrise çevirmek için “0” ile “1” arasında değişen ve genellikle 0.85 olarak alınan bir yumuşatma (kör alt grafların gücünü kırma, hafifletme) faktörü olan α’yı kullandık ve bu işlem ile “G” matrisimizi elde ettik. Yaptığımız bu adım ile basit bir güç iterasyonunu (power iteration) kullanarak eşsiz PageRank vektörünün elde edilmesini sağladık.

4.3. PageRank Formül Yapısında “G” Matrisinin İncelenmesi

1/40	9/20	9/20	1/40	1/40	1/40
1/6	1/6	1/6	1/6	1/6	1/6
37/120	37/120	1/40	37/120	1/40	1/40
1/40	1/40	1/40	1/40	9/20	9/20
1/40	1/40	1/40	1/40	1/40	7/8
1/40	1/40	1/40	9/20	9/20	1/40

Şekil 4.4 Sonuçta Elde Ettiğimiz "G" Matrisi

Yaptığımız ilkel işlem ile “G” matrisimizi ilkel bir matrise dönüştürdük. Bu haliyle “G” matrisimizi *Google Matrisi* diye adlandırabiliriz. Google Matrisi üzerine uyguladığımız güç metodu ile de PageRank vektörünü elde edip, sayfalar arası önem sıralamasını bulduğumuzu bir önceki bölümde bahsetmiştik. Şimdi ise “S” matrisi üzerinde yaptığımız bu ilkel işlem ile oluşan “G” matrisimizin bu işlem sonucunda kazandığı özelliklere bir bakalım.

- “G” matrisimiz düzenli aralıklar ile tekrarlanmayan yani aperyodik bir matristir.
- “G” matrisimiz indirgenemez. Yani Random Walker’ın uğradığı herhangi bir sayfadan diğer bütün sayfalara geçiş sağlanabilir.

- "G" matrisimiz pozitif stokastik bir matristir. Yani her satırdaki değerler toplamı 1'e tekabül ederken, bütün değerler 0'dan büyüktür.
- "G" matrisimiz tamamen boşluksuz bir yapıya sahiptir. Yani bütün matris değerlerimiz 0'dan farklıdır.
- "G" matrisimiz ilkeldir. Çünkü "k" değerleri için $G^k > 0$ dır. Bu sayede eşsiz pozitif bir π^T vektörü elde edilir.
- "G" matrisimiz istenilen yapı özelliklerine ulaşması için "H" matrisinin iki defa modifiye ettik. Bu haliyle "G" matrisimiz "H" matrisimizin yapay şeklidir.

Yaptığımız işlem adımlarından sonra "G" matrisi $G = \alpha S + (1-\alpha)(1/n)ee^T$ olarak karşımıza çıktı. Özet olarak PageRank metodunu ise,

$$\pi^T_{(k+1)} = \pi^T_{(k)} G, \qquad \textbf{(4.1)}$$

şeklinde özetledik. Bir başka değişle formülümüz, güç metodunun (power method) "G" matrisine uygulanmasıdır diyebiliriz.

4.4. PageRank Formül Yapısında "α" Faktörü

Bir önceki bölümde "α" faktörünün "0" ile "1" arasında değişen ve genellikle 0.85 olarak alınan bir yumuşatma (kör alt grafların gücünü kırma, hafifletme) faktörü olduğunu belirtmiştik. Bu işlem sayesinde Random Walker'ımızın kör alt graflarda asılı kalmasını önlemiş oluyorduk. Peki "α" faktörünün Brin ve Page tarafından 0.85 alınmasının sebebi nedir? Neden 0.6 ya da 0.9 değil de 0.85 olarak alınmıştır. Bunu belirleyebilmek için bir önceki bölümde hesapladığımız PageRank vektörümüzü bu defa "α" değerlerinin 0.7 ve 0.95 olması durumunda nasıl bir değişiklik yaptığını inceleyelim. Öncelikle "S" ve "G" matrislerimizin formüllerini hatırlayalım.

$S = H + (1/n)de^T$

$G = \alpha S + (1-\alpha)(1/n)ee^T$

"G" matrisimizin değerini hesaplamak için kullanacağımız "S" matrisinde bir değişiklik olmadığına göre, bir önceki bölümde oluşturduğumuz "S" matrisini aynı şekilde kullanabiliriz. Fakat bu defa "G" matrisimizin değerini hesaplamak için "α" değerini 0.85 değil de 0.7 olarak alalım. Buna göre G matrisimiz,

G= 0.7

$$\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + (1\text{-}0.7)\,(1/6) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} (1\ 1\ 1\ 1\ 1\ 1) =$$

G=7/10

$$\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + 1/20 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

G=

$$\begin{pmatrix} 0 & 7/20 & 7/20 & 0 & 0 & 0 \\ 7/60 & 7/60 & 7/60 & 7/60 & 7/60 & 7/60 \\ 7/30 & 7/30 & 0 & 7/30 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7/20 & 7/20 \\ 0 & 0 & 0 & 0 & 0 & 7/10 \\ 0 & 0 & 0 & 7/20 & 7/20 & 0 \end{pmatrix} + \begin{pmatrix} 1/20 & 1/20 & 1/20 & 1/20 & 1/20 & 1/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 1/20 & 1/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 1/20 & 1/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 1/20 & 1/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 1/20 & 1/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 1/20 & 1/20 \end{pmatrix}$$

$$G=\begin{pmatrix} 1/20 & 8/20 & 8/20 & 1/20 & 1/20 & 1/20 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 17/60 & 17/60 & 1/20 & 17/60 & 1/20 & 1/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 8/20 & 8/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 1/20 & 15/20 \\ 1/20 & 1/20 & 1/20 & 8/20 & 8/20 & 1/20 \end{pmatrix}$$

olarak buluruz. Buna göre elde ettiğimiz "G" matrisine güç metodunu (power method) uygulayarak, PageRank vektörümüzü elde edebiliriz. Formülümüzü tekrarlarsak,

$\pi^{T}_{(k+1)} = \pi^{T}_{(k)} G$ idi. Buna göre,

$\pi^{T}_{1} =$ (1/6 1/6 1/6 1/6 1/6 1/6)

$$G=\begin{pmatrix} 1/20 & 8/20 & 8/20 & 1/20 & 1/20 & 1/20 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 17/60 & 17/60 & 1/20 & 17/60 & 1/20 & 1/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 8/20 & 8/20 \\ 1/20 & 1/20 & 1/20 & 1/20 & 1/20 & 15/20 \\ 1/20 & 1/20 & 1/20 & 8/20 & 8/20 & 1/20 \end{pmatrix}$$

Bu iterasyonu devam ettirerek nihai "π^{T}_{*}" vektörünü elde edelim.

π^{T}_{1}	0,108333333	0,166666667	0,127777778	0,166666667	0,186111111	0,244444444
π^{T}_{2}	0,099259259	0,137175926	0,107361111	0,184814815	0,213333333	0,258055556
π^{T}_{3}	0,091054784	0,125795525	0,100744599	0,181374228	0,221008488	0,280022377
π^{T}_{4}	0,088183218	0,120052392	0,096545319	0,186191049	0,226164956	0,282863066
π^{T}_{5}	0,086533353	0,11739748	0,094870239	0,185535427	0,228175053	0,287488449

π^T_6	0,085832762	0,116119435	0,093983046	0,186453719	0,229254729	0,288356309
π^T_7	0,085476645	0,115518112	0,093588734	0,186401353	0,229730777	0,289284379
π^T_8	0,085314484	0,11523131	0,093393939	0,186564017	0,229967119	0,289529131
π^T_9	0,085235572	0,115095641	0,093303722	0,186570768	0,230076255	0,289718042
π^T_{10}	0,085198693	0,115031144	0,093260275	0,186600008	0,230128908	0,289780972
π^T_{11}	0,085181031	0,115000574	0,093239843	0,186604371	0,230153643	0,289820539
π^T_{12}	0,085172697	0,114986058	0,093230094	0,186609885	0,230165452	0,289835814
π^T_{13}	0,085168729	0,114979173	0,093225484	0,186611263	0,230171035	0,289844316
π^T_{14}	0,08516685	0,114975905	0,093223292	0,18661236	0,23017369	0,289847903
π^T_{15}	0,085165957	0,114974354	0,093222253	0,186612723	0,230174948	0,289849765
π^T_{16}	0,085165534	0,114973619	0,09322176	0,186612951	0,230175545	0,289850591
π^T_{17}	0,085165333	0,11497327	0,093221526	0,18661304	0,230175829	0,289851004
π^T_{18}	0,085165237	0,114973104	0,093221415	0,186613089	0,230175963	0,289851192
π^T_{19}	0,085165192	0,114973025	0,093221362	0,186613109	0,230176027	0,289851284
π^T_{20}	0,085165171	0,114972988	0,093221337	0,18661312	0,230176057	0,289851327
π^T_{21}	0,085165161	0,11497297	0,093221325	0,186613125	0,230176072	0,289851347
π^T_{22}	0,085165156	0,114972962	0,093221319	0,186613127	0,230176079	0,289851357
π^T_{23}	0,085165153	0,114972958	0,093221317	0,186613128	0,230176082	0,289851362
π^T_{24}	0,085165152	0,114972956	0,093221315	0,186613129	0,230176083	0,289851364
π^T_{25}	0,085165152	0,114972955	0,093221315	0,186613129	0,230176084	0,289851365
π^T_{26}	0,085165152	0,114972955	0,093221315	0,186613129	0,230176084	0,289851365
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
π^T_*	**0,085165152**	**0,114972955**	**0,093221315**	**0,186613129**	**0,230176084**	**0,289851365**

Tablo 4.1 "α" Değeri 0,7'ye Göre Değişen PageRank Vektörü

Görüldüğü gibi iterasyon değerlerimiz 26. iterasyonda tekrara düşmektedir. Oysaki "α" değerimiz 0.85 iken iterasyon değerimiz 37 idi. Buna göre "α" değerimiz düştükçe PageRank vektörümüze daha kısa sürede ulaşabiliyoruz. Ayrıca sayfalar arası değer sıralamamıza baktığımızda herhangi bir değişiklik görülmüyor. Yani α = 0.85 değeri ile aynı önem sıralamasına ulaştığımız görülüyor. Aynı şekilde sayfa önem sıralamamız 6 > 5 > 4 > 2 > 3 > 1 şeklinde bulduk. Fakat sonuçlarda göze çarpan bir nokta daha var. "α" değerimiz 0.85 iken Random Walker'ımız % 5.170 olasılıkla 1 numaralı sayfayı ziyaret ederken, "α" değerimiz 0.7 olduğunda Random Walker'ımız % 8.516 olasılıkla 1 numaralı sayfayı ziyaret etmektedir. Buna göre "α" değerimiz küçüldükçe önem sıralamamız değişmemesine rağmen, önem değeri düşük olan sayfalarımıza Random Walker'ın uğrama olasılığı artmaktadır. Aynı şekilde Random Walker'ımız "α" değeri 0,85 iken % 34.870 olasılıkla 6 numaralı sayfayı ziyaret ederken, "α" değerimiz 0.7 olduğunda Random Walker'ımız % 28.985

olasılıkla 6 numaralı sayfayı ziyaret etmektedir. Buna göre şöyle bir sonuç çıkarabiliriz. "α" değerimiz düştükçe sayfalar arası önem hassasiyet azalmaktadır. Yani "α" değerimiz düştükçe "α" değerindeki küçük değişimlerde PageRank'imizin hassasiyeti düşmektedir. Bu sonucu daha iyi görebilmek için şimdi de "α" değerimizi 0.95 alarak değişmeleri gözlemleyelim.

G= 0.95

$$\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + (1-0.95)(1/6) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} (1\ 1\ 1\ 1\ 1\ 1) =$$

G=19/20

$$\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + 1/120 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

G=

$$\begin{pmatrix} 0 & 19/40 & 19/40 & 0 & 0 & 0 \\ 19/120 & 19/120 & 19/120 & 19/120 & 19/120 & 19/120 \\ 19/60 & 19/60 & 0 & 19/60 & 0 & 0 \\ 0 & 0 & 0 & 0 & 19/40 & 19/40 \\ 0 & 0 & 0 & 0 & 0 & 19/20 \\ 0 & 0 & 0 & 19/40 & 19/40 & 0 \end{pmatrix} + \begin{pmatrix} 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 1/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 1/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 1/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 1/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 1/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 1/120 \end{pmatrix}$$

$$G=\begin{pmatrix} 1/120 & 58/120 & 58/120 & 1/120 & 1/20 & 1/120 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 39/120 & 39/120 & 1/120 & 39/120 & 1/120 & 1/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 58/120 & 58/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 115/120 \\ 1/120 & 1/120 & 1/120 & 58/120 & 58/120 & 1/120 \end{pmatrix}$$

olarak buluruz. Elde ettiğimiz "G" matrisine güç metodunu (power method) uygulayarak, PageRank vektörümüzü elde edelim.

$\pi^T_1 = (1/6 \;\; 1/6 \;\; 1/6 \;\; 1/6 \;\; 1/6 \;\; 1/6)$

$$G=\begin{pmatrix} 1/120 & 58/120 & 58/120 & 1/120 & 1/20 & 1/120 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 39/120 & 39/120 & 1/120 & 39/120 & 1/120 & 1/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 58/120 & 58/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 115/120 \\ 1/120 & 1/120 & 1/120 & 58/120 & 58/120 & 1/120 \end{pmatrix}$$

İterasyonu devam ettirerek nihai "π^T_*" vektörünü elde edelim.

π^T_1	0,0875	0,166666667	0,113888889	0,166666667	0,193055556	0,272222222
π^T_2	0,070787037	0,112349537	0,076284722	0,200092593	0,243194444	0,297291667
π^T_3	0,050278839	0,083902681	0,059745853	0,19149238	0,262379533	0,352200714
π^T_4	0,040537445	0,064419893	0,045500373	0,207832784	0,279872144	0,361837362
π^T_5	0,032941601	0,052196887	0,037788436	0,204814348	0,289126469	0,383132259
π^T_6	0,028564179	0,044211439	0,032245101	0,210552002	0,295872479	0,388554801
π^T_7	0,025544427	0,039112411	0,028901463	0,210107957	0,299909209	0,396424533
π^T_8	0,023678262	0,035811864	0,026659734	0,211979915	0,302629065	0,39924116
π^T_9	0,022445794	0,033692969	0,025250719	0,212085345	0,304333556	0,402191616
π^T_{10}	0,021664115	0,032325867	0,024329806	0,212705132	0,30544961	0,40352547
π^T_{11}	0,021156034	0,031446488	0,02374205	0,212830633	0,306161132	0,404663663

π^T_{12}	0,020830677	0,030879793	0,023361477	0,213045916	0,306622151	0,405259986
π^T_{13}	0,020620435	0,030515006	0,023117205	0,213118928	0,306917938	0,405710488
π^T_{14}	0,020485324	0,030280031	0,022959583	0,213197806	0,307108849	0,405968408
π^T_{15}	0,020398206	0,030128735	0,022858201	0,2132332	0,307231623	0,406150036
π^T_{16}	0,020342147	0,030031294	0,022792864	0,213263413	0,307310753	0,406259528
π^T_{17}	0,020306029	0,029968548	0,022750808	0,213279305	0,307361686	0,406333625
π^T_{18}	0,020282776	0,02992814	0,022723717	0,213291248	0,307394495	0,406379624
π^T_{19}	0,020267799	0,029902118	0,022706274	0,213298121	0,30741562	0,406410069
π^T_{20}	0,020258155	0,02988536	0,02269504	0,213302938	0,307429225	0,406429281
π^T_{21}	0,020251945	0,029874568	0,022687806	0,213305853	0,307437986	0,406441841
π^T_{22}	0,020247945	0,029867619	0,022683147	0,21330782	0,307443628	0,406449841
π^T_{23}	0,02024537	0,029863144	0,022680147	0,213309044	0,307447262	0,406455034
π^T_{24}	0,020243711	0,029860261	0,022678215	0,213309852	0,307449602	0,406458359
π^T_{25}	0,020242643	0,029858405	0,022676971	0,213310363	0,307451108	0,406460509
π^T_{26}	0,020241955	0,02985721	0,02267617	0,213310697	0,307452079	0,40646189
π^T_{27}	0,020241512	0,029856441	0,022675654	0,21331091	0,307452704	0,406462781
π^T_{28}	0,020241227	0,029855945	0,022675321	0,213311048	0,307453106	0,406463354
π^T_{29}	0,020241043	0,029855626	0,022675107	0,213311136	0,307453365	0,406463723
π^T_{30}	0,020240925	0,02985542	0,02267497	0,213311193	0,307453532	0,406463961
π^T_{31}	0,020240849	0,029855288	0,022674881	0,21331123	0,307453639	0,406464114
π^T_{32}	0,020240799	0,029855203	0,022674824	0,213311253	0,307453709	0,406464212
π^T_{33}	0,020240768	0,029855148	0,022674787	0,213311269	0,307453753	0,406464276
π^T_{34}	0,020240748	0,029855112	0,022674763	0,213311279	0,307453782	0,406464317
π^T_{35}	0,020240734	0,02985509	0,022674748	0,213311285	0,3074538	0,406464343
π^T_{36}	0,020240726	0,029855075	0,022674738	0,213311289	0,307453812	0,40646436
π^T_{37}	0,020240721	0,029855065	0,022674732	0,213311292	0,30745382	0,406464371
π^T_{38}	0,020240717	0,029855059	0,022674728	0,213311293	0,307453825	0,406464378
π^T_{39}	0,020240715	0,029855055	0,022674725	0,213311294	0,307453828	0,406464382
π^T_{40}	0,020240713	0,029855053	0,022674723	0,213311295	0,30745383	0,406464385
π^T_{41}	0,020240712	0,029855051	0,022674722	0,213311295	0,307453832	0,406464387
π^T_{42}	0,020240712	0,02985505	0,022674722	0,213311296	0,307453832	0,406464388
π^T_{43}	0,020240711	0,02985505	0,022674721	0,213311296	0,307453833	0,406464389
π^T_{44}	0,020240711	0,029855049	0,022674721	0,213311296	0,307453833	0,40646439
π^T_{45}	0,020240711	0,029855049	0,022674721	0,213311296	0,307453833	0,40646439
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
π^T_*	**0,020240711**	**0,029855049**	**0,022674721**	**0,213311296**	**0,307453833**	**0,40646439**

Tablo 4.2 "α" Değeri 0,95'e Göre Değişen PageRank Vektörü

Görüldüğü gibi "α" değerimiz 0.95 olduğunda iterasyon değerimiz 45'te sabitlenmiştir. Buna göre "α" değerimiz arttıkça PageRank vektörümüze daha uzun sürede ulaşabiliyoruz. Elde ettiğimiz PageRank vektörüne baktığımız da ise yine

önem sıralamasının değişmediğini görüyoruz. "α" değeri 0.95 için önem sıralamamız yine 6 > 5 > 4 > 2 > 3 > 1 şeklinde oldu. Fakat PageRank vektörümüzün değerlerine baktığımızda hassasiyetin bir hayli arttığını görüyoruz. Sonuçlara göre "α" değerimiz 0.95 iken Random Walker'ımız % 2.024 olasılıkla 1 numaralı sayfayı ziyaret ederken, % 40.646 olasılıkla 6 numaralı sayfayı ziyaret etmektedir. Tüm değerler göz önüne alındığında 4, 5 ve 6 numaralı düğümlerden oluşan alt grafımız, yüzdelik önem payının yaklaşık % 92'sini elde tutmaktadır. Bu sonuca göre Random Walker'ımız devamlı olarak bu alt grafta dönüp duracağı açıktır. Yani Random Walker'ın 1, 2 ve 3 numaralı düğümlerden oluşan diğer alt grafımıza uğrama olasılığı bir hayli düşüktür. Bir başka değişle "α" değerindeki ufak oynamalar hassasiyeti olabildiğine artıracaktır. Buna göre "α" değeri 1'e yaklaştıkça PageRank değeri olabildiğine hassaslaşacaktır. Bu durumu fark eden Brin ve Page yaptıkları istatistiksel sonuçların ardından "α" değerinin 0.85 değerinde eşik değerine ulaştığını görmüşlerdir. Böylece PageRank formülümüzdeki "α" değerinin 0.85 olarak alınmasını makul görmüşlerdir.

Özet olarak "α" değerindeki değişmelerin PageRank hassasiyetine olan etkisini sıralamak gerekirse,

- Küçük "α" değerleri için, "α" değerindeki küçük oynamalar PageRank hassasiyetini etkilemez.
- "α" değeri büyüdükçe, "α" değerindeki küçük oynamalar PageRank hassasiyetini giderek arttırır.
- "α" değeri 1'e yaklaştıkça "α" değerindeki küçük oynamalar PageRank hassasiyetini doruk noktasına ulaştırır.

PageRank yapısında arzu edilen, "α" değerinin 1'e yaklaşmasıdır. Fakat bu durum PageRank hassasiyetini çok fazla arttırdığından Google tarafından 0.85 olarak alınmıştır. Devasa web grafında 30 milyonu aşkın web sayfasını düşünürsek eğer bu eşik değerinin makul bir değer olduğu fark edilecektir.

4.5. PageRank Formül Yapısında Teleportation Matris "E"

Nihai PageRank formül yapısına ulaşmak için yaptığımız bir diğer modifiye işlemi de teleportation matris dediğimiz "E" üzerinde yaptığımız değişikliklerdi. Brin ve Page PageRank formül yapısındaki $(1/n)ee^{T,}$yi kullanmak yerine, $ev^{T,}$yi

kullanmayı önermişlerdir (Langville ve Meyer, 2006). Olasılık vektörü diye isimlendirdiğimiz $ev^T > 0$ ayrıca kişiselleştirme (personalization) ya da telepotation vektör olarak ta anılmaktadır. Çünkü v^T olasılık vektörünün bütün değerleri pozitiftir ve her bir düğümden diğer bütün düğümlere direkt bağlantı sağlar. Bu sayede "G" matrisimiz ilkel bir matrise dönüşürken, markov zincirleri için eşsiz sabit vektörü yani bir başka değişle PageRank vektörümüzün elde edilmesi sağlandı. Peki $(1/n)ee^T$ yerine ev^T'yi kullanmanın ne gibi avantajı olabilir? $(1/n)ee^T$'yi kullanırken temel felsefemiz Random Walker'ımızın bulunduğu sayfadan belirli olasılıkla diğer bütün sayfalara geçişini sağlamaktı. Bunu yaparken de her sayfaya eşit oranda pay veriyorduk. Bu bağlamda Random Walker'ın bulunduğu sayfadan başka bir sayfaya geçerken her sayfayı eşit görmek ne kadar mantıklı olacaktır? Bunun yerine formülümüzde ev^T'yi kullanarak Random Walker'ın bulunduğu sayfadan bir başka sayfaya geçerken kullanıcının ilgilendiği alanların göz önüne almasını sağlıyoruz. Böylece daha sağlıklı bir sıralamanın gerçekleştirilmesini sağlıyoruz. Tabi bu nokta da her bir kullanıcı için kişiselleştirme matrisinin hesaplanması büyük bir çaba gerektirir. Çünkü her bir kullanıcı için ilgilendiği alanlardaki sayfaların benzerlik oranlarının bilinmesi gerekir. Buna göre her bir kullanıcı sorgulaması için veya alanlar arası kombinasyon değeri çok yüksek bir sonucun birer birer PageRank değerinin üretilmesi gerekir ki genel anlamda bakıldığında imkânsız gibi görünebilir. Günümüzde bu işlem yapılmasa da bu belirleme için genellikle veri madenciliğinden yararlanılır. Var olan sayfalar arasında sınıflandırmalar yapılırken, benzerlikler de belirlenmeye çalışılır. Yapılan çalışmalarda PageRank vektörünün hesaplanmasında alanlara göre kategorileştirme yoluna gidilmesi düşünülmektedir. Buna göre sayfalar belli başlıklar altında toplanarak, sayfaların bulunduğu kategoriye göre kişiselleştirme matrisinde, paylaştırma oranları değiştirilir. Böylece haber sayfalarından haber sayfalarına fazla bir pay verilirken, spor, müzik, yemek gibi diğer saylara sıfıra yakın bir pay verilmiş olur. Böylece hem kör alt graflar sorunu çözülürken, hem de Random Walker'ın sayfalar arası geçiş sırasında alakasız sayfalara geçiş olasılığı azaltılmış ve kullanıcı isteğine göre bir sıralama elde edilmiş olur. Konumuza geri dönersek, yaptığımız bu değişiklik ile formülümüzdeki değişimi görmemizde yarar var. Buna göre formülümüz şu değişikliğe maruz kalacaktır.

$$\pi^T_{(k+1)} = \pi^T_{(k)} G \rightarrow = \pi^T_{(k)} G$$

$$\pi^T_{(k+1)} = \alpha\pi^T_{(k)}\, S + (1-\alpha)\, \pi^T_{(k)}(1/n)ee^T \quad \rightarrow \quad = \quad \alpha\pi^T_{(k)}\, S + (1-\alpha)\, \pi^T_{(k)}ev^T$$

$$\pi^T_{(k+1)} = \alpha\pi^T_{(k)}\, H + (\alpha\, \pi^T_{(k)}d + 1-\alpha)\, e^T/n \quad \rightarrow \quad = \quad \alpha\pi^T_{(k)}\, H + (\alpha\, \pi^T_{(k)}d + 1-\alpha)v^T$$

(4.2)

Buna göre teleportation matris diye adlandırdığımız "E" matrisimiz $(1/n)ee^T$ iken ev^T'ye dönüşmüştür. Sayfalar arasında yapılacak olan sınıflandırmaların ardından hesaplanan PageRank değerlerinin toplanması ile kapsamlı bir değer elde edilebilir. Buna göre ilgili sayfanın her bir kategoride elde ettiği puanlar toplanarak toplam PageRank puanı hesaplanabilir ya da sorgu bağımlı düşünülerek arama motoruna girilen sorguya göre, ilgili alandaki sayfaların sıralaması gerçekleştirilebilir. Bu durumu elimizdeki grafa uygulayarak teleportation matrisin PageRank'imize etkisini görebiliriz. Buna göre "α" değeri 0.85 iken, kişiselleştirme vektörümüzü (1/6 1/6 1/6 1/6 1/6 1/6) olarak almıştık. Peki, sorgulamamızın sınıflandırılmış bir "G" matrisinde olduğunu düşünürsek bu durumda sonuç nasıl değişecektir? Örneğin 1 ve 3 numaralı sayfalarımızın sorgumuzla ilgili sınıfın sayfaları olduğunu düşünelim. Buna göre bu iki sayfa teleportation matrisinde daha büyük değerler alırken diğer sayfalar 0'a yakın bir değer alacaktır. Diğer sayfaların hemen hemen 0'a yakın bir değer aldığına göre sonucu daha rahat görebilmemiz için bu sayfaların pay değerlerini "0" alıp, elimizdeki pay değerini 1 ve 3 numaralı sayfalar arasında paylaştırarak sonucu gözlemleyelim. Buna göre "G" matrisimiz;

$$G = \alpha\, S + (1-\alpha)ev^T,$$

(4.3)

olacaktır. "G" matrisimizi hesaplarsak eğer,

$$G = 0.85 \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + (1-0.85) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} ((1/2)(1\ 0\ 1\ 0\ 0\ 0)) =$$

$$G=17/20\begin{pmatrix}0 & 1/2 & 1/2 & 0 & 0 & 0\\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6\\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0\\ 0 & 0 & 0 & 0 & 1/2 & 1/2\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1/2 & 1/2 & 0\end{pmatrix} + 3/20\begin{pmatrix}1/2 & 0 & 1/2 & 0 & 0 & 0\\ 1/2 & 0 & 1/2 & 0 & 0 & 0\\ 1/2 & 0 & 1/2 & 0 & 0 & 0\\ 1/2 & 0 & 1/2 & 0 & 0 & 0\\ 1/2 & 0 & 1/2 & 0 & 0 & 0\\ 1/2 & 0 & 1/2 & 0 & 0 & 0\end{pmatrix}$$

$$G=\begin{pmatrix}0 & 17/40 & 17/40 & 0 & 0 & 0\\ 17/120 & 17/120 & 17/120 & 17/120 & 17/120 & 17/120\\ 17/60 & 17/60 & 0 & 17/60 & 0 & 0\\ 0 & 0 & 0 & 0 & 17/40 & 17/40\\ 0 & 0 & 0 & 0 & 0 & 17/20\\ 0 & 0 & 0 & 17/40 & 17/40 & 0\end{pmatrix} + \begin{pmatrix}3/40 & 0 & 3/40 & 0 & 0 & 0\\ 3/40 & 0 & 3/40 & 0 & 0 & 0\\ 3/40 & 0 & 3/40 & 0 & 0 & 0\\ 3/40 & 0 & 3/40 & 0 & 0 & 0\\ 3/40 & 0 & 3/40 & 0 & 0 & 0\\ 3/40 & 0 & 3/40 & 0 & 0 & 0\end{pmatrix}$$

$$G=\begin{pmatrix}3/40 & 17/40 & 20/40 & 0 & 0 & 0\\ 26/120 & 17/120 & 26/120 & 17/120 & 17/120 & 17/120\\ 43/120 & 17/60 & 3/40 & 17/60 & 0 & 0\\ 3/40 & 0 & 3/40 & 0 & 17/40 & 17/40\\ 3/40 & 0 & 3/40 & 0 & 0 & 17/20\\ 3/40 & 0 & 3/40 & 17/40 & 17/40 & 0\end{pmatrix}$$

olarak buluruz. Elde ettiğimiz "G" matrisine güç metodunu (power method) uygulayarak, PageRank vektörümüzü elde edelim.

$\pi^T_1 =$	(1/6 1/6 1/6 1/6 1/6 1/6)	$G=\begin{pmatrix} 3/40 & 17/40 & 20/40 & 0 & 0 & 0 \\ 26/120 & 17/120 & 26/120 & 17/120 & 17/120 & 17/120 \\ 43/120 & 17/60 & 3/40 & 17/60 & 0 & 0 \\ 3/40 & 0 & 3/40 & 0 & 17/40 & 17/40 \\ 3/40 & 0 & 3/40 & 0 & 0 & 17/20 \\ 3/40 & 0 & 3/40 & 17/40 & 17/40 & 0 \end{pmatrix}$

İterasyonu devam ettirerek nihai "π^T_*" vektörünü elde edelim.

π^T_1	0,145833333	0,141666667	0,169444444	0,141666667	0,165277778	0,236111111
π^T_2	0,143078704	0,13005787	0,157048611	0,168425926	0,180625	0,220763889
π^T_3	0,137921971	0,123730421	0,154233314	0,156746624	0,183830536	0,243537133
π^T_4	0,136227915	0,119844753	0,151145314	0,164731197	0,187649073	0,240401747
π^T_5	0,134802512	0,117699376	0,149874871	0,161973255	0,189159508	0,246490478
π^T_6	0,134138625	0,116429693	0,148965146	0,163897078	0,190271165	0,246298293
π^T_7	0,133700998	0,115709913	0,148503122	0,163377773	0,190827239	0,247880955
π^T_8	0,133468122	0,115291046	0,148215162	0,163817528	0,191177197	0,248030945
π^T_9	0,133327194	0,115051146	0,14805685	0,163740346	0,191368499	0,248455965
π^T_{10}	0,133248353	0,114912411	0,14796297	0,163842138	0,191482344	0,248551783
π^T_{11}	0,1332021	0,11483265	0,147909808	0,163836608	0,191546675	0,24867216
π^T_{12}	0,133175738	0,11478663	0,147878851	0,163861406	0,191584185	0,248713191
π^T_{13}	0,133160447	0,114760136	0,147861128	0,163863553	0,191605643	0,248749094
π^T_{14}	0,133151672	0,114744862	0,147850876	0,163870037	0,191618061	0,248764492
π^T_{15}	0,133146604	0,114736064	0,147844983	0,163871513	0,191625197	0,248775639
π^T_{16}	0,133143688	0,114730994	0,147841582	0,163873334	0,191629316	0,248781086
π^T_{17}	0,133142006	0,114728073	0,147839625	0,163873967	0,191631686	0,248784643
π^T_{18}	0,133141037	0,11472639	0,147838496	0,163874511	0,191633053	0,248786513
π^T_{19}	0,133140479	0,11472542	0,147837846	0,163874747	0,19163384	0,248787667
π^T_{20}	0,133140158	0,114724861	0,147837471	0,163874916	0,191634294	0,2487883
π^T_{21}	0,133139972	0,114724539	0,147837256	0,163875	0,191634555	0,248788678
π^T_{22}	0,133139865	0,114724354	0,147837131	0,163875054	0,191634706	0,24878889
π^T_{23}	0,133139804	0,114724247	0,14783706	0,163875082	0,191634793	0,248789015
π^T_{24}	0,133139769	0,114724185	0,147837018	0,1638751	0,191634843	0,248789085
π^T_{25}	0,133139748	0,11472415	0,147836995	0,163875109	0,191634872	0,248789127
π^T_{26}	0,133139736	0,114724129	0,147836981	0,163875115	0,191634888	0,24878915

π^{T}_{27}	0,13313973	0,114724117	0,147836973	0,163875118	0,191634898	0,248789164
π^{T}_{28}	0,133139726	0,114724111	0,147836968	0,16387512	0,191634903	0,248789172
π^{T}_{29}	0,133139723	0,114724107	0,147836966	0,163875121	0,191634906	0,248789176
π^{T}_{30}	0,133139722	0,114724105	0,147836964	0,163875122	0,191634908	0,248789179
π^{T}_{31}	0,133139721	0,114724103	0,147836963	0,163875122	0,191634909	0,24878918
π^{T}_{32}	0,133139721	0,114724102	0,147836963	0,163875123	0,19163491	0,248789181
π^{T}_{33}	0,133139721	0,114724102	0,147836963	0,163875123	0,19163491	0,248789182
π^{T}_{34}	0,133139721	0,114724102	0,147836962	0,163875123	0,191634911	0,248789182
π^{T}_{35}	0,13313972	0,114724102	0,147836962	0,163875123	0,191634911	0,248789182
π^{T}_{36}	0,13313972	0,114724102	0,147836962	0,163875123	0,191634911	0,248789182
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
π^{T}_{*}	0,13313972	0,114724102	0,147836962	0,163875123	0,191634911	0,248789182

Tablo 4.3 "α" Değeri 0,85 iken Paylaştırılmış Teleportation Matrisine Göre Değişen PageRank Vektörü

Görüldüğü gibi iterasyon değerlerimiz 36. iterasyonda tekrara düşmektedir. Oysaki iterasyon işlemimiz bir önceki teleportation vektörümüzde (1/6 1/6 1/6 1/6 1/6 1/6) 37. iterasyonda bitmişti. Buna göre teleportatiton vektörümüzde yapacağımız değişiklikler itersyon sayımızı etkileyecektir. Ayrıca sayfalar arası önem sıralamamıza baktığımızda da sıralamada değişiklikler olduğu görülmektedir. Yani (1/6 1/6 1/6 1/6 1/6 1/6) vektörü yerine (1/2 0 1/2 0 0 0) vektörünü kullandığımızda önem sıralamasının değiştiğini görüyoruz. Sonuca göre sayfa önem sıralamamız 6 > 5 > 4 > 3 > 1 > 2 şeklinde çıktı. Ayrıca çıkan sonuçlarda göze çarpan bir nokta daha vardır. Vektörümüz (1/6 1/6 1/6 1/6 1/6 1/6) iken Random Walker'ımız % 5.170 olasılıkla 1 numaralı sayfayı ziyaret ederken, vektörümüz (1/2 0 1/2 0 0 0) olduğunda Random Walker'ımız % 13.313 olasılıkla 1 numaralı sayfayı ziyaret etmektedir. Buna göre değişen vektör yapısı Random Walker'ın hareketini büyük oranda etkilemektedir. Aynı şekilde Random Walker'ımız % 34.870 olasılıkla 6 numaralı sayfayı ziyaret ederken, vektörümüz (1/2 0 1/2 0 0 0) olduğunda Random Walker'ımız % 24.878 olasılıkla 6 numaralı sayfayı ziyaret etmektedir. Buna göre iki değer arasında büyük bir oran farkı olduğu bariz bir şekilde görülmektedir. Teleportation vektörünün önem sıralamasına olan etkisini daha iyi anlamak için, "α" değerinde yapılacak olan değişikliklerde nasıl etkilendiğini görmekte yarar var. Buna göre teleportation matrisin etkini daha iyi kavrayabilmek için şimdi de "α" değerimizi 0.95 alarak değişmeleri gözlemleyelim.

G= 0.95

$$\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + (1\text{-}0.95) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} ((1/2)(1\ 0\ 1\ 0\ 0\ 0)) =$$

G=19/20

$$\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} + 1/20 \begin{pmatrix} 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \end{pmatrix}$$

G=

$$\begin{pmatrix} 0 & 19/40 & 19/40 & 0 & 0 & 0 \\ 19/120 & 19/120 & 19/120 & 19/120 & 19/120 & 19/120 \\ 19/60 & 19/60 & 0 & 19/60 & 0 & 0 \\ 0 & 0 & 0 & 0 & 19/40 & 19/40 \\ 0 & 0 & 0 & 0 & 0 & 19/20 \\ 0 & 0 & 0 & 19/40 & 19/40 & 0 \end{pmatrix} + \begin{pmatrix} 1/40 & 0 & 1/40 & 0 & 0 & 0 \\ 1/40 & 0 & 1/40 & 0 & 0 & 0 \\ 1/40 & 0 & 1/40 & 0 & 0 & 0 \\ 1/40 & 0 & 1/40 & 0 & 0 & 0 \\ 1/40 & 0 & 1/40 & 0 & 0 & 0 \\ 1/40 & 0 & 1/40 & 0 & 0 & 0 \end{pmatrix}$$

$$G=\begin{pmatrix} 1/40 & 19/40 & 20/40 & 0 & 0 & 0 \\ 22/120 & 19/120 & 22/120 & 19/120 & 19/120 & 19/120 \\ 41/120 & 19/60 & 1/40 & 19/60 & 0 & 0 \\ 1/40 & 0 & 1/40 & 0 & 19/40 & 19/40 \\ 1/40 & 0 & 1/40 & 0 & 0 & 19/20 \\ 1/40 & 0 & 1/40 & 19/40 & 19/40 & 0 \end{pmatrix}$$

olarak bulunur. Elde ettiğimiz "G" matrisine güç metodunu (power method) uygulayarak, PageRank vektörümüzü elde edelim.

$\pi^T_1 =$ (1/6 1/6 1/6 1/6 1/6 1/6)

$$G=\begin{pmatrix} 1/40 & 19/40 & 20/40 & 0 & 0 & 0 \\ 22/120 & 19/120 & 22/120 & 19/120 & 19/120 & 19/120 \\ 41/120 & 19/60 & 1/40 & 19/60 & 0 & 0 \\ 1/40 & 0 & 1/40 & 0 & 19/40 & 19/40 \\ 1/40 & 0 & 1/40 & 0 & 0 & 19/20 \\ 1/40 & 0 & 1/40 & 19/40 & 19/40 & 0 \end{pmatrix}$$

İterasyonu devam ettirerek nihai "π^T_*" vektörünü elde edelim.

π^T_1	0,104166667	0,158333333	0,130555556	0,158333333	0,184722222	0,263888889
π^T_2	0,091412037	0,115891204	0,099548611	0,191759259	0,225625	0,275763889
π^T_3	0,074873167	0,093293885	0,086770158	0,180861015	0,240422936	0,323778839
π^T_4	0,067248749	0,077813503	0,075336286	0,196043697	0,254475462	0,329082303
π^T_5	0,061176962	0,068120118	0,069263627	0,192491056	0,261755321	0,347192916
π^T_6	0,057719167	0,061778224	0,064844742	0,197635802	0,267135572	0,350886492
π^T_7	0,055315721	0,057732325	0,062198157	0,196986804	0,270329642	0,357437352
π^T_8	0,053837034	0,055112002	0,060415919	0,198619776	0,272492426	0,359522843
π^T_9	0,052857775	0,053430366	0,059298658	0,198631125	0,273843811	0,361938265

π^T_{10}	0,052237716	0,052345159	0,058567251	0,199158392	0,274730268	0,362961213
π^T_{11}	0,05183428	0,051647195	0,058100899	0,199240856	0,275294796	0,363881975
π^T_{12}	0,05157609	0,051197373	0,057798755	0,199420028	0,275660817	0,364346935
π^T_{13}	0,05140919	0,050907833	0,057604894	0,199473984	0,275895559	0,36470854
π^T_{14}	0,051301957	0,050721322	0,057479772	0,199538513	0,276047106	0,36491133
π^T_{15}	0,051232804	0,050601233	0,057399305	0,199565686	0,276144552	0,365056421
π^T_{16}	0,051188309	0,05052389	0,057347444	0,199590108	0,276207362	0,365142887
π^T_{17}	0,05115964	0,050474086	0,057314063	0,199602511	0,276247789	0,365201912
π^T_{18}	0,051141183	0,050442012	0,057292559	0,199612092	0,276273831	0,365238322
π^T_{19}	0,051129296	0,050421358	0,057278714	0,199617499	0,276290598	0,365262535
π^T_{20}	0,051121641	0,050408057	0,057269797	0,199621345	0,276301398	0,365277762
π^T_{21}	0,051116711	0,050399491	0,057264055	0,199623648	0,276308352	0,365287742
π^T_{22}	0,051113537	0,050393975	0,057260357	0,199625215	0,27631283	0,365294086
π^T_{23}	0,051111492	0,050390423	0,057257976	0,199626184	0,276315714	0,365298211
π^T_{24}	0,051110176	0,050388135	0,057256442	0,199626826	0,276317571	0,365300849
π^T_{25}	0,051109328	0,050386662	0,057255455	0,199627231	0,276318767	0,365302557
π^T_{26}	0,051108782	0,050385713	0,057254819	0,199627497	0,276319537	0,365303652
π^T_{27}	0,051108431	0,050385102	0,057254409	0,199627665	0,276320033	0,365304359
π^T_{28}	0,051108204	0,050384709	0,057254146	0,199627775	0,276320353	0,365304814
π^T_{29}	0,051108058	0,050384455	0,057253976	0,199627845	0,276320558	0,365305107
π^T_{30}	0,051107964	0,050384292	0,057253866	0,19962789	0,276320691	0,365305296
π^T_{31}	0,051107904	0,050384187	0,057253796	0,199627919	0,276320776	0,365305417
π^T_{32}	0,051107865	0,050384119	0,057253751	0,199627938	0,276320831	0,365305495
π^T_{33}	0,05110784	0,050384076	0,057253721	0,19962795	0,276320867	0,365305546
π^T_{34}	0,051107824	0,050384048	0,057253703	0,199627958	0,276320889	0,365305578
π^T_{35}	0,051107813	0,05038403	0,057253691	0,199627963	0,276320904	0,365305599
π^T_{36}	0,051107807	0,050384018	0,057253683	0,199627966	0,276320913	0,365305613
π^T_{37}	0,051107802	0,050384011	0,057253678	0,199627968	0,27632092	0,365305621
π^T_{38}	0,0511078	0,050384006	0,057253674	0,19962797	0,276320923	0,365305627
π^T_{39}	0,051107798	0,050384003	0,057253672	0,199627971	0,276320926	0,365305631
π^T_{40}	0,051107797	0,050384001	0,057253671	0,199627971	0,276320928	0,365305633
π^T_{41}	0,051107796	0,050383999	0,05725367	0,199627972	0,276320929	0,365305634
π^T_{42}	0,051107795	0,050383999	0,05725367	0,199627972	0,276320929	0,365305635
π^T_{43}	0,051107795	0,050383998	0,057253669	0,199627972	0,27632093	0,365305636
π^T_{44}	0,051107795	0,050383998	0,057253669	0,199627972	0,27632093	0,365305636
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
π^T_*	0,051107795	0,050383998	0,057253669	0,199627972	0,27632093	0,365305636

Tablo 4.4 "α" Değeri 0,95 iken Paylaştırılmış Teleportation Matrisine Göre Değişen PageRank Vektörü

Görüldüğü gibi vektörümüz (1/2 0 1/2 0 0 0) olduğunda iterasyon sayımız 44'te sabitlenmiştir. Elde ettiğimiz PageRank vektörüne baktığımız da ise önem sıralamasının 6 > 5 > 4 > 3 > 1 > 2 olduğunu görüyoruz. "α" değerimiz 0.95 olduğunda sıralamamız değişmese de göze çarpan bir başka nokta vardır. Sonuçlara göre Random Walker'ın sayfalar arasındaki geçiş olasılıkları küçük bir miktarda değişse de yaklaşık %84 olasılık ile yine 4, 5 ve 6 numaralı sayfalarımızdan oluşan alt grafımıza düşecektir. Buna göre "α" değerimiz artıkça teleportation matrisimiz PageRank vektörümüzü etkilemeyecektir. Yani "α" değeri 1'e yaklaştıkça teleportation matrisimiz önemini yitirecektir. Bu değerleri daha açık görebilmek için yaptığımız işlemlerin sonuçlarını tabloya dökelim ve ardından sonuçları tekrardan özetleyelim.

	"α" değeri	v^T (Personalization Vector)	G (Google Matrix)	π^T (P.R Vector)
Yapı 1	0.85	(1/6 1/6 1/6 1/6 1/6 1/6)	$\begin{pmatrix} 1/40 & 9/20 & 9/20 & 1/40 & 1/40 & 1/40 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 37/120 & 37/120 & 1/40 & 37/120 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 1/40 & 9/20 & 9/20 \\ 1/40 & 1/40 & 1/40 & 1/40 & 1/40 & 7/8 \\ 1/40 & 1/40 & 1/40 & 9/20 & 9/20 & 1/40 \end{pmatrix}$	1 → 0.051 2 → 0.073 3 → 0.057 4 → 0.199 5 → 0.268 6 → 0.348
Yapı 2	0.85	(1/2 0 1/2 0 0 0)	$\begin{pmatrix} 3/40 & 17/40 & 20/40 & 0 & 0 & 0 \\ 26/120 & 17/120 & 26/120 & 17/120 & 17/120 & 17/120 \\ 43/120 & 17/60 & 3/40 & 17/60 & 0 & 0 \\ 3/40 & 0 & 3/40 & 0 & 17/40 & 17/40 \\ 3/40 & 0 & 3/40 & 0 & 0 & 17/20 \\ 3/40 & 0 & 3/40 & 17/40 & 17/40 & 0 \end{pmatrix}$	1 → 0.133 2 → 0.114 3 → 0.147 4 → 0.163 5 → 0.191 6 → 0.248

Yapı 3	0.95	(1/6 1/6 1/6 1/6 1/6 1/6)	$\begin{pmatrix} 1/120 & 58/120 & 58/120 & 1/120 & 1/20 & 1/120 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 39/120 & 39/120 & 1/120 & 39/120 & 1/120 & 1/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 58/120 & 58/120 \\ 1/120 & 1/120 & 1/120 & 1/120 & 1/120 & 115/120 \\ 1/120 & 1/120 & 1/120 & 58/120 & 58/120 & 1/120 \end{pmatrix}$	1 → 0.020 2 → 0.029 3 → 0.022 4 → 0.213 5 → 0.307 6 → 0.406
Yapı 4	0.95	(1/2 0 1/2 0 0 0)	$\begin{pmatrix} 1/40 & 19/40 & 20/40 & 0 & 0 & 0 \\ 22/120 & 19/120 & 22/120 & 19/120 & 19/120 & 19/120 \\ 41/120 & 19/60 & 1/40 & 19/60 & 0 & 0 \\ 1/40 & 0 & 1/40 & 0 & 19/40 & 19/40 \\ 1/40 & 0 & 1/40 & 0 & 0 & 19/20 \\ 1/40 & 0 & 1/40 & 19/40 & 19/40 & 0 \end{pmatrix}$	1 → 0.051 2 → 0.050 3 → 0.057 4 → 0.199 5 → 0.276 6 → 0.365

Tablo 4.5 Değişen "α" Değeri ve Teleportation Matrisine Göre PageRank Vektörlerinin Kıyaslanması

Tablodan görüldüğü gibi, "α" değerimiz 0.85 iken vektörümüz (1/6 1/6 1/6 1/6 1/6 1/6)'dan (1/2 0 1/2 0 0 0)'ya dönüştürdüğümüzde teleportation matrisimizin PageRank vektörümüze etkisi çok fazla olup, sayfa önem sıralamamızı değiştirdiği gibi Random Walker'ın sayfalar arası geçiş olasılıklarını da çok büyük bir oranda değiştirmiştir. Fakat "α" değerimiz 0.95 iken vektörümüz (1/6 1/6 1/6 1/6 1/6 1/6)'dan (1/2 0 1/2 0 0 0)'ya dönüştürdüğümüzde teleportation matrisimizin PageRank vektörümüze etkisi çok az olup, Random Walker'ın sayfalar arası geçiş olasılığını da çok fazla etkileyememektedir.

4.6. Lineer Sistem Olarak PageRank Formülü

Örneğimizdeki küçük Google matrisinde formülümüzü kullanarak basit bir şekilde kesin sonuçlara ulaşabildik. Fakat matrisimizin 30 milyonu aşkın devasa web grafı olduğunu düşündüğümüzde Google matrisinin hesaplanması ve kesin sonuçlara ulaşılması çok fazla zaman ve hesaplama kaynakları gerektirecektir. Bu bağlamda güç metodu tahmini PageRank vektörünün hesaplanmasında kullanılan ilk iteratif

algoritmadır (Wills, 2007). Bu hesaplamada kullanılan diğer iteratif yöntemler olan klasik iteratif yöntem, Krylov yöntemi, dış değer (genelleştirme) yöntemi gibi birçok metot kullanılmaya çalışıldıysa da devasa web grafı için başarılı sonuçlara ulaşılamamıştır. Bu bağlamda bir matrisin baskın özvektörünü bulmak için kullanılan en eski ve en basit yöntem hala güç metodudur diyebiliriz.

4.7. Güç Metodu (Power Method)

Güç metodu bir matrisin özvektörüne karşılık gelen en büyük özdeğeri bulmanın basit bir yöntemidir. Daha önce de bahsettiğimiz gibi "G" matrisimizin her bir elemanı olan G_{ij}, 0 ile 1 arasında değiştiği ve "G" matrisimizin her bir satırının toplamı 1 olduğundan stokastik matristir. Bilindiği gibi $\lambda_1 = 1$ G'nin tekrarlanan bir özdeğeri değildir ve büyüklük açısından G'nin diğer özdeğerlerinden daha büyüktür. G'nin özdeğerlerinin sıralaması $\lambda_1 > \lambda_2 > \lambda_3 \geq \ldots \geq \lambda_n$ şeklindedir. Buna göre özsistem,

$\pi^T = \pi^T G,$

$\pi \geq 0,$

$\pi^T e = 1$ dir.

Formüldeki π'nin eşsiz çıktısına PageRank vektörü diyoruz. Buradaki $\lambda_1 = 1$ G'nin baskın özdeğeri olurken, buna karşılık gelen π'ye ise G'nin baskın sol özvektörü diyoruz. G'nin bir markov zinciri olduğunu düşündüğümüzde ise π, G'nin sabit dağıtık vektörü olur. Buna göre π'nin her i. elemanı i. düğümün PageRank skoru olur. Örneğin $\pi_i > \pi_j$ ise buna, i. düğümün PageRank skoru j. düğümün PageRank skorundan büyüktür deriz.

4.8. Lineer Sistem Olarak PageRank Problemi

Google matrisimizin temel tanımlamasına ele alırsak,

$\pi^T = \pi^T G$

$\pi^T = \alpha\pi^T S + (1-\alpha)\, \pi^T e v^T$

$\pi^T = \alpha\pi^T S + (1-\alpha)\, (\pi^T e)\, v^T$

$\pi^T = \alpha\pi^T S + (1-\alpha)\, v^T$ ($\pi^T e = 1$ olduğundan) olur.

Buna göre,

$\pi^T = \alpha\pi^T S + (1-\alpha)\, v^T$

$\pi^T - \alpha\pi^T S = (1-\alpha)\, v^T$

$\pi^T (I - \alpha S) = (1-\alpha)\, v^T$ olur. Böylece PageRank tanımlamamızı lineer sistem olarak belirtmek istersek,

$\pi^T (I - \alpha S) = (1-\alpha)\, v^T$ olur. Ayrıca $\| \alpha S \|_\infty = \alpha < 1$ olduğundan,

$\pi^T = (1-\alpha)\, v^T (I - \alpha S)^{-1}$

Formüldeki $(I - \alpha S)^{-1}$ güç metodunun iterasyonunda π'nin yaklaşık değerinin hesaplamada büyük rol oynar ve $(I - \alpha S)$ matrisi eşsiz bir matris değildir. Son olarak $(I - \alpha S)$ ve $(I - \alpha S)^{-1}$ ile ilgili birkaç önemli özelliği belirtelim. Buna göre,

- $(I - \alpha S)$ bir M-matristir.
- $(I - \alpha S)$ eşsiz bir matris değildir.
- $I - \alpha S$'nin satırları toplamı $1-\alpha$ dır.
- $\| I - \alpha S \|_\infty = \alpha + 1$ dir.
- $(I - \alpha S)$ bir M-matris olduğundan, $(I - \alpha S)^{-1} \geq 0$.
- $(I - \alpha S)^{-1}$ 'nin satırları toplamı $(1-\alpha)^{-1}$ olduğundan $\| (I - \alpha S)^{-1} \|_\infty = (1-\alpha)^{-1}$.
- $1 \leq (I - \alpha S)^{-1}_{ii} \leq 1/(1-\alpha)$ bütün $1 \leq i \leq n$ için (Wills, 2007).

BÖLÜM IV

5. Araştırma Sonuçları ve Arama Motorlarının Geleceği

5.1. Araştırma Sonuçları

Araştırmamızda arama motorlarının yapısını ve web sayfaları arasındaki sıralamayı oluşturmak için kullandıkları içerik ve popülarite skorunu nasıl elde ettiklerini inceledik. Sıralamayı oluşturmak için elde ettikleri içerik ve popülarite skorundan kapsamlı skoru elde ettiklerini belirtmiştik. Arama motorlarının kapsamlı skoru elde etmek için kullandıkları içerik ve popülarite skorunu hangi oranda kullandıkları, arama motorundan arama motoruna değişmekle birlikte kullanım amacına göre de değişmektedir. Gelişen teknoloji ile her iki skorunda dezavantajları olmakla birlikte çözülen algoritmalar yüzünden sayısız spam ile mücadele etmek zorunda oldukları bir gerçektir. Özellikle içerik skorunun hesaplanmasında sınırlı değerler olduğundan web site yöneticileri tarafından ince detaylara kadar çözülmüş durumdadır. Hal böyle olunca web sayfaları arasında sıralama yapmak epey bir zorlaşmaktadır. Bu durumdan ötürü her geçen gün arama motorları içerik sunumunda web site yöneticilerinden daha fazla ayrıntı ve düzen talep etmektedir. Öyle ki arama motorları web site yöneticilerinden kendi mimari düzenlerine uygun taleplerde bulunmakta ve bu talepleri yerine getiren web sitelerini sıralama yaparken ödüllendireceklerini söylemektedirler. Devasa bir çöplük yığınına dönüşmüş web ortamı için içerik düzeninin web site yöneticileri tarafından istenilen yönde revize edilmesi tabi olarak arama motorlarının işini kolaylaştırmaktadır. Günümüzde de özellikle arama motoru optimizasyonu (Search Engine Optimization- SEO) giderek popüler bir bilim alanı olma yolunda ilerlemektedir. Öyle ki artık bütün büyük site sahipleri bir SEO danışmanı ile çalışmak istemektedir. Günümüzün en büyük sorunlarından biri olan internet kirliliğinin giderilmesi arama motorları tarafından talep edilen bu içerik düzeni şartları sayesinde giderek pozitif yönde ilerleme göstermektedir. Aynı zamanda bilgiyi elde etme alanına da büyük bir katkı sağlamaktadır.

Araştırmamızın ikinci bölümünde de incelediğimiz gibi web site sahipleri site içeriklerinde ne kadar düzenli ve ayrıntılı çalışırlarsa, arama motorları tarafından o kadar fazla artı puan ile ödüllendirileceği anlaşılmaktadır. İçeriklerinde bulunan özgünlük, sayfa başlığı, yazı başlığı, konu başlıkları, meta anahtar kelimeleri,

eklenen resimlerin açıklamaları, boyutları ve yapısı, site haritasının oluşturulması, sitelerinde robot.txt'nin yer alması, stil dosyaları, kodlama uygunluğu, site içi navigasyon, görsellik, hız, kullanıcı etkileşimi, yeni teknolojilere uyumu, kullanıcı memnuniyeti, link yapısı, yönlendirme direktifleri, site benzerliği, tıklanma oranı, kullanıcıların sitede bulunma süreleri, kullanıcıların sitenizdeki konularla ilgili yorum sayısı, içerik anlatımında yapılan yazı süslemeleri, sosyal medya da yer alması gibi emek sarf edilmesi gereken yüzlerce özellik, içerik skorunun hesaplanmasında değerlendirmeye alınmaktadır. Tabi her şeyden önce sitelerin dürüstlüğü, yani içten ve samimi olması istenmektedir. Başka sitelerden çalınan, aşırılan her bir içerik, kendisi gibi kötü niyetli siteler ile iletişimi, yani karşılıklı çıkar doğrultusunda link alım ve satımları gibi negatif davranışlar da içerik skorunun değerlendirmesinde hesaba katılmaktadır.

Araştırmamızın üçüncü ve dördüncü bölümünde ayrıntılı bir şekilde incelediğimiz popülarite skoru, arama motorlarının özellikle Google'ın, bu hesaplamayı yaparken nasıl bir formül yapısı kullandığını aşama aşama belirtmiştik. Bütün arama motorları aynı formülizasyonu kullanmasa da temel mantığını bu yapı üzerinden inşa ettiklerini belirtmekte yarar var. Keza Google bile PageRank hesaplamasını her geçen gün değiştirmekte ve içerik skorunda olduğu gibi popülarite skorunda da karşılaşılan spamlarla mücadele etmek için yoğun bir çaba sarf etmektedir. Sadece PageRank metodu değil aynı zamanda diğer arama motorları tarafından kullanılan HITS, SALSA, Hybrid Ranking, Ranking Based on Traffic Flow gibi metodlarda da aynı sorunlar yaşanmaktadır. Her geçen gün üretilen yeni metotların web site yöneticileri, SEO danışmanları, matematik araştırmacıları, veri madenciliği uzmanları gibi arama motorları ile ilgilenenler tarafından çözülmesinin ardından spamlar türemektedir. Özellikle araştırmamızda da görüldüğü gibi PageRank değerinin büyük olduğu web sitelerinden link almanın sıralamada büyük farklılıklar yarattığının bilinmesinin ardından, büyük şirketler web sitelerini sıralamada ön sıralara çekebilmek için bu tarz sitelerden ücret karşılığında link almaya başlamışlardır. Gün geçtikçe arama motorlarının web sayfalarını değerlendirmede kullandığı algoritmayı bu tarz yöntemlerle arama motorlarını yanıltan web sitelerinin sayının artmasından dolayı, arama motorları bu tarz web sitelerini cezalandırma yoluna gitmiştir. Öyle ki popülarite skorunun oluşturulması gibi BadRank skoru dediğimiz kötü not değerlendirmesi de arama motorları tarafından hesaplanmaya başlanmıştır. Buna göre araştırmamızda incelendiğimiz PageRank skorunda link almak ne kadar önemli

ise BadRank skorunda da link vermek bir o kadar önemlidir. Arama motorları link verilen sayfaların BadRank skoruna bakarak sizin ne kadar dürüst olduğunuzu hesaplamaktadırlar. "Bana arkadaşını söyle, sana kim olduğunu söyleyeyim" deyiminde olduğu gibi arama motorları da link verdiğiniz siteleri ele alarak sizin bir profilinizi çıkarmaktadır.

Günümüzde hala büyük ölçüde geçerliliğini koruyan PageRank formülü gün geçtikçe yeni ihtiyaçlara göre revize edilmektedir. Dördüncü bölümde de ayrıntılı bir şekilde değerlendirilen PageRank formülümüzde günümüz veri madenciliğinde kullanılan yöntemler sayesinde büyük değişikliklere uğramıştır. Özellikle veri madenciliğinde kullanılan sınıflandırma teknikleri ile web dünyasında yer alan siteler kategorileştirilmekte ve PageRank puan dağılımında benzerlik oranlarından faydalanılarak paylaştırılma yoluna gidilmektedir. Bu sayede spam diye belirttiğimiz sahte linkler belirgin şekilde göze çarpmakta ve dürüst davranmayan web siteleri gerekli uyarıları almakta veya cezalandırılmaktadır.

Araştırmamızda göze çarpan bir diğer önemli hususta web sitelerinin PageRank puanlarını diğer sitelere dağıtırken nasıl bir yol izlediğidir. Özellikle çok yüksek PageRank puanına sahip web siteleri için hayati değer taşıyan bu durum, birçok çıkmazı da beraberinde getirdiğini araştırmamızda belirtmiştik. Örneğin PageRank puanını sadece belli başlı sitelere dağıtan PageRank puanı yüksek web sayfaları, link verdikleri sitelere sıralamada büyük bir katkı sağlamaktadır. Oysaki benzerlik oranı daha yüksek olan web siteleri bulunurken belli başlı sitelere PageRank puan paylaşımı yapmak tabi olarak adaletsiz bir ortam yaratmaktadır. Aynı şekilde PageRank puanı yüksek olup hiç bir siteye link vermeyen sitelerin PageRank puanının arama motorları tarafından sistemdeki her siteye eşit şekilde dağıtıldığını gördük. Bu durum da aynı şekilde adaletsiz bir ortam yaratmaktadır. Çünkü ilgili siteye benzerlik oranı yüksek olan sitelerin daha yüksek bir pay alması gerektiğinden bahsettik. Bu durumu yaratan arama motorları olabileceği gibi yüksek puana sahip web sayfaları da bilinçli bir şekilde rakiplerinin yüksek bir pay almaması için elindeki PageRank puanını çok fazla parçaya bölüp dağıtarak anlamsız bir hale getirebilir. Bu bağlamda özellikle algoritma üzerinde çok fazla bilgi birikimine sahip olan web site yöneticileri, belirli alanlarda aramaları tekelinde tutabilmektedir.

Yukarıda belirttiğimiz gibi arama motorları tarafından oluşturulan algoritmalarda birçok sorun oluşabilmektedir. Her geçen gün arama motorları ellerindeki algoritmaları spamlara karşı güçlendirmek için patlak veren noktaları tamir etme

yoluna gitmektedir. Tabi ki web sayfalarını sıralamada yeni yöntem ve teknikler kullanılabilir. Her ne kadar yeni yöntem ve teknikler çok zor gibi görünse de eldeki algoritmada oluşan çatlaklıkları gidermekte bir o kadar güç ve emek gerektiren bir iştir.

Bu bağlamda aşağıdaki bölümde arama motorlarının geleceğine dair bir kaç öngörüde bulunulmaktadır. Umudumuz şudur ki, yeni oluşturulacak algoritmaların her site sahibinin oluşturduğu fikir ve verdiği emek ile orantılı olsun.

5.2. Arama Motorlarının Geleceği

Her geçen gün artan veri miktarı ile boğuşan arama motorları, bir taraftan gün geçtikçe çeşitlenen web sitlerini daha iyi tarayıp, daha iyi bir sıralama oluşturmaya çalışırlarken, diğer taraftan gün geçtikçe bilinçlenen kullanıcıların ihtiyaçlarına daha iyi cevap vermeye çalışmaktadırlar. Fakat günümüz arama motorlarının, devasa boyutlara ulaşan web dünyası ve bilinçlenen kullanıcı profili ile başa çıkamayacağı herkes tarafından az çok tahmin edilen bir sondur. Bu bağlamda ilerleyen yıllarda çok farklı arama motorları ile karşılaşmamız kaçınılmazdır. Günümüz arama motorlarının belli bir doygunluğa ulaştığı düşünülürse bu alanda atılacak her girişimci adımın yenilecek pastadan olabildiğince fazla pay alması kuvvetle muhtemeldir. Araştırmamızda incelenen yöntem ve formüllerle, web siteleri arasında sıralama oluşturan arama motorlarının, kısa zamanda, gün geçtikçe artan bilinçli web kullanıcılarına alternatif arama motorları sunması gerekmektedir. Özellikle hayatımızda yer edinen bütün alanların, web dünyasında yeteri kadar alt kategoriler oluşturacak şekilde bilgi birikimine ulaşıldığı düşünülürse geleceğin arama motorları günümüzdeki gibi aranan sorguyla ilgili karma arama yerine, spesifik bir alana yönelik arama yapaya yoğunlaşacaklardır.

Günümüz arama motorlarının ömrünü doldurduğu düşüncesi sadece bilinçlenen kullanıcı ve artan bilgi miktarı ile alakalı değildir elbet. Ayrıca Arama motorlarının günümüz ihtiyaçlarını karşılayamamasının yanında, gelişen teknolojili ile birlikte artan spam sorunları ile başa çıkabilmesi de gerekmektedir. Araştırmamızda incelediğimiz Google PageRank formülü gibi diğer arama motorlarının da üç aşağı beş yukarı aynı mantıksal yapıyı barındırdığı bilinen bir gerçektir. Keza bu alanda ilerleyen web site yöneticileri gün geçtikçe PageRank mantığı hakkında (ülkemizde olmasa da) çok fazla bilgi sahibi olmaya başlayınca, üretilen spamlarla sıralamalar değiştirilebilmektedir. Her ne kadar arama motorları bu tarz yöntemlere başvuran

web sitelerin ağır bir şekilde cezalandıracağını bağıra bağıra belirtse de, milyonlarca web site yöneticilerini düşünürsek bu uyarılar pek de dikkate alınmamakla birlikte arama motorlarının bu alanda yapabileceği çok fazla çözüm de bulunamamaktadır. Bu durumun üstesinden gelmek için arama motorları her ne kadar spam belirleyici programcıklar üretseler de, en iyi değerlendirmenin kullanıcılar tarafından verileceğini bildirdiklerinden dolayı, kullanıcılardan şikâyet iletileri istemek durumunda kalmışlardır. Heyecanlı, hırslı ve sabırsız web kullanıcıları için bu iletiler zaman alacağından, kullanıcıdan yeterince dönüt alınamamaktadır. Bu durum da doğal olarak sonuçların sağlıklı değerlendirilememesine neden olmaktadır. Bu durumun üstesinden gelebilmek, kullanıcıdan istenilecek şikâyet iletileri şeklinde değil de, sıralamanın olabildiğince kullanıcının istek ve arzularına göre şekillendirip, kullanıcı bağımlı sorgulama yapmak spamlarla mücadelede etmekte daha büyük bir fayda sağlayacağı açıktır.

Aslına bakılırsa Google ve benzeri arama motorları PageRank değerlerine ek olarak farklı puanlama yöntemlerine başvursalar da, spamlar PageRank hesaplamasına tamamen hükmetmiş durumda değillerdir. Spamların önüne geçilmesinin en iyi yöntemlerinden birisi de, dediğimiz gibi değerlendirmeyi olabildiğince kullanıcılara bırakmaktır. Tabi bunu yaparken de kullanıcının hükmedebileceği arama motorları üretmek gerekir. Yani kullanıcının seçebileceği alternatif web siteleri arama motorları tarafından yapılan değerlendirmeler sonucunda değilde, kullanıcının profili düşünülerek oluşturulacak dökümler daha sağlıklı sonuçlar verecektir. Elde edilecek sonuçlar da istatistikî sonuçlara dökülerek sıralamanın gerçekleştirilmesi sağlanabilir. Böylece yapılan aramalarda devamlı sıralamayı tekelinde bulunduran web siteleri, kullanıcılardan gerekli cevabı alacaklardır.

Araştırmamızda da belirttiğimiz gibi arama motoruna girdiğimiz sorgu cümlesi belirli bir eşik değerinden sonra negatif yönde sonuçlar ürettiğini göstermiştik. Yani sorgu bölümüne girdiğimiz kelime sayısı belirli bir değere kadar çok iyi sonuçlar vermesine rağmen, eşik değerinin altında ve üstünde yapılan sorgulamalar da çok kötü sonuçlar verebilmektedir. Fakat yaptığımız bu eşik değerindeki sorgulamalar belirli sitelerin, daha önce dediğimiz gibi sıralamayı tekelinde tutması ile sonuçlanmaktadır. Bu durumdan dolayı da web dünyasında bulunan birçok faydalı ve içerikli site hak ettiği değeri alamamaktadır.

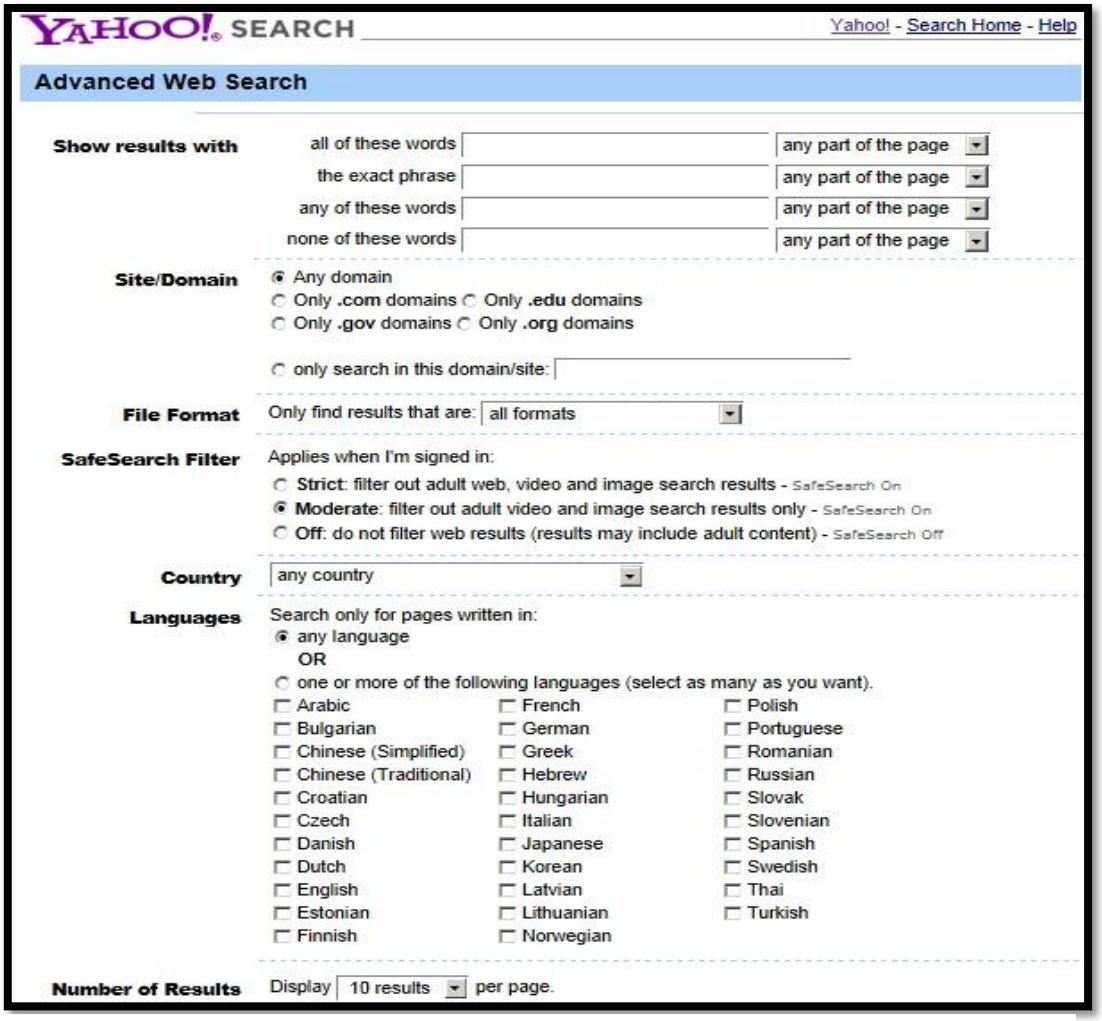
Günümüz kullanıcı profilinin gün geçtikçe ilerlemesinden dolayı, arama motorlarında yaşanan bu tatminsiz sonuçlar giderek artmaktadır. Ülkemizde ve dünyada hızla artmakta olan bu kullanıcı profillerinin, oluşturulacak alternatif arama motorlarına yönelmesi kaçınılmaz olacaktır. Bu bağlamda araştırmamızın sonunda arama motorlarının geleceğine dair bir kaç farklı yöntem ve teknik sunarak araştırmamızı sonlandırmak istiyorum.

5.2.1. Sorguya Özgü Arama Motorları

Araştırmamızın dördüncü bölümünde de bahsettiğimiz gibi kişiselleştirme faktörü, geleceğin arama motorlarına yön verecek gibi duruyor. PageRank formülümüzde yer alan v^T vektörü her bireye farklı kombinasyonlar sunarak, sıralama sonucunu bireye özgü olmasını sağlıyor. Araştırmamızda gördüğümüz gibi Google tek bir PageRank vektörünü (π^T) hesaplamak için epey bir çaba sarf ediyordu. Google gibi arama motorlarının tek bir PageRank vektörünü elde etmek için çok fazla zaman kaybedeceği düşünülürse, sorguya özgü PageRank vektörü uzak bir ihtimal gibi duruyor. Fakat farklı yöntem ve teknikler ile günümüz arama motorları yavaş yavaş sorguya özgü seçenekler sunmaya başladı. Eğer Google, Yahoo, Yandex, Bing ve benzeri arama motorları da devasa içeriklerini kullanıcıların isteklerine göre daha etkin şekilde cevap verebilecek bir algoritma geliştirebilirlerse, her kullanıcı kendi belirlediği sınırlar çerçevesinde sıralamayı değiştirebilecektir. Yani haber sitelerinde gezinmeyi seven bir kullanıcı, sorgulama yaptığında diğer alanlara yönelik web sayfaları sıralamaya alınmayacaktır (en basit düzeyde). Böylece kullanıcı belirlediği alanlarda sorgulama yaparak, ilgisiz sayfaların gelmesi önlenecektir. Keza daha kaliteli sonuçlar üretilirken hem kullanıcının istekleri karşılanacak hem de karma aramalarda yaşanan adaletsizlikler önlenmiş olacaktır.

Yapılan araştırmalarda kullanıcıların en fazla ilk yirmi sonucu gezdikleri sonraki sayfaları gezmektense sorguyu değiştirdikleri gözlenmiştir. Bu durum bize karma sorguda yapılan sorgulamaların her defasında ilk yirmi sonuçta dönüp durduğunu göstermektedir. Bu yüzden yeni yayın hayatına başlamış web sitelerinin sıralamada ön konumlara geçmek imkânsız bir hal almaktadır. Bu durum da doğal olarak web site sahiplerini üzen bir durumdur. Eğer kullanıcılar kendi istekleri doğrultusunda alanları belirleyip sorgulamayı başlatabilirler ise yaşanan bu adaletsizlikler de bir nebze önlenmiş olacaktır.

Bu yapının algılanmasına yönelik bir örnek vermek gerekirse;



Şekil 5.1 Kullanıcı Kontrollü Yahoo Arama Motoru Seçenekleri

Görüldüğü gibi kullanıcı sorgulamaya bağlamadan önce, sorgulamanın sınırlarını bir nebze de olsa çizebilmektedir. Böylece kullanıcı bulmak istediği bilgiye daha hızlı erişim sağlayabilecektir. Bu yapı aynı zamanda günümüzün en büyük problemlerinden biri olan internet kirliliği sorununa da bir nebze olsun çözmüş olacaktır. Yahoo'un sunduğu bu yarı kullanıcı bağımlı sorgu istenilen düzeyde olmasa da ileriye dönük arama motorlarının gelişimine büyük bir katkı sağlayacaktır. Fakat bu noktada dikkat edilmesi gereken, Yahoo bu tarz sorgulamaları yaparken yeni bir PageRank vektörü hesaplamadığıdır. Sadece kullanıcının istediği sayfaları alıp önceden hesaplanmış PageRank vektörüne göre sıralamaktadır. Fakat bizim burada bahsettiğimiz her defasında kullanıcının belirlediği değerlere göre PageRank'i hesaplarken puan dağılımında kullanıcının istediği alanlar arasında puanları dağıtarak yeni bir PageRank hesaplaması yapmasıdır. Yani v^T değerimizdeki dağılımlar eskisi gibi her sayfaya eşit oranda değil de, kullanıcının

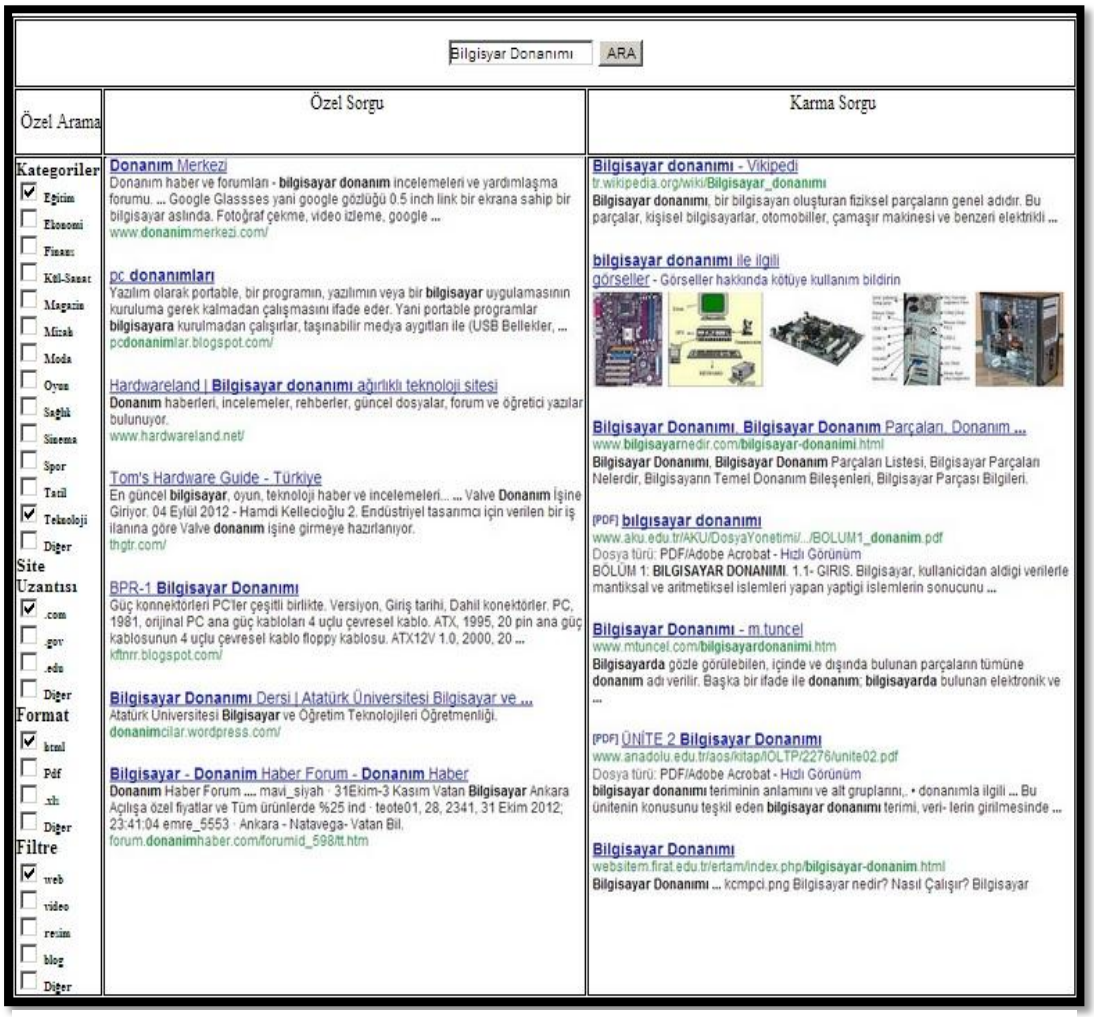
istediği alanlar çerçevesinde dağıtılmasıdır. Aynı şekilde Google kullanıcı ara yüzüne baktığımız da ise;

Şekil 5.2 Kullanıcı Kontrollü Google Arama Motoru Seçenekleri

Yine aynı şekilde Google ara yüzünün de Yahoo'dan daha az da olsa yarı kullanıcı bağımlı sorgulama gerçekleştirilebildiğini görebiliyoruz. Kullanıcı isterse üst kısımda bulunan görseller, oyun, haber, bloglar gibi belirli alanlara yönelik aramalar yapabilmektedirler. Yine yukarıda bahsettiğimiz gibi Google da bu hesaplamayı yaparken yeni bir PageRank değeri hesaplamamaktadır. Fakat biz konumuz itibariyle kullanıcının sorgulamada daha çok hükmedebileceği arama motorlarının ilerleyen yıllarda kullanıcılar tarafından tercih edileceğini düşünüyoruz. Yani kullanıcı sorgu ara yüzünde araştıracağı konu seçimini ilgili kutucuklardan işaretleyerek, PageRank vektörünü etkileyip sıralamayı değiştirebilmesinden bahsediyoruz. Yahoo ve Google'da görülen tam olarak kullanıcı kontrolünde olan konu odaklı arama değildir. Yani belli başlı bölümlere ayrılmış olmasına rağmen, sıralama oluşturulurken puan dağılımları kullanıcı isteğine göre olmamaktadır. Bahsettiğimiz yapı küçük sitelerde bulunan butonsal yapı ve açılır butonlar şeklindedir. Böylece kullanıcı arama yapacağı kelime öbeği ile ilgili alanları seçebilecek ve hatta kullanıcıya ilgili alanın alt kategorilerini bile seçme hakkı tanınacaktır. Böylece kullanıcının işaretlemiş olduğu alanlar göz önüne alınarak, araştırmamızda hesapladığımız gibi eşsiz bir PageRank vektörü hesaplanıp, sıralamanın istenilen şekilde ve düzeyde değiştirilebilmesi sağlanacaktır. Tabi bu yapının oluşturulabilmesi ve başarıya ulaşabilmesi için çok iyi bir sınıflandırma tekniğinin kullanılması gerekir. Ayrıca

araştırmamızda değindiğimiz gibi siteler kendi kategorilerini meta taglarında belirttikleri gibi, arama motorları da yaptıkları hesaplamalar doğrultusunda web sitelerini sınıflandırabildiklerini belirtmiştik.

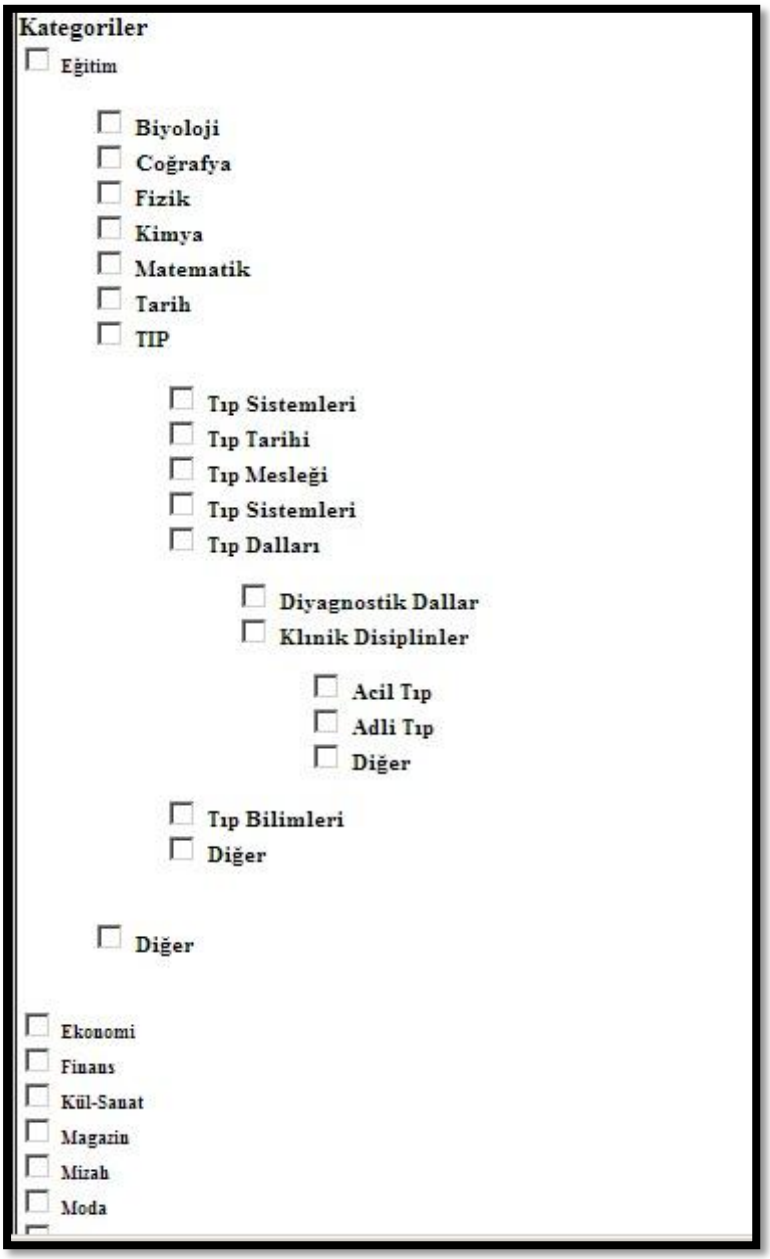
Bu duruma örnek teşkil edecek bir sorgulama sonucunu göstermek istersek;



Şekil 5.3 Kullanıcı Kontrollü Arama ve Karma Arama Sonuçlarının Karşılaştırılması

Örneğimizde görüldüğü gibi sorgu ara yüzünde "bilgisayar donanımı" şeklinde bir arama yaparken özel arama bölümünün kategori bölümünde "Eğitim ve Teknoloji", Site uzantısı bölümünde ".com", format türünde ".html", filtre bölümünde ise "web" kutucuğunu işaretleyerek belirlediğimiz değerlere göre özel arama yapılmasını istiyoruz. Sonuçlardan da görüldüğü gibi özel arama bölümünde sadece belirlediğimiz kriterlere göre puan dağılımı yapılarak PageRank puan sıralaması oluşturulurken, aramanın sağ bölümünde bulunan karma bölümde tamamen farklı bir sıralama görülmektedir. Bahsettiğimiz özel arama bölümünde her bir kategorinin alt bölümleri de olacak şekilde arama motorları tasarlanabilirse, yukarıda gördüğümüz

uygulamada olduğu gibi daha kaliteli sonuçlar elde etmiş olacağız. Aynı zamanda daha spesifik sonuçlara ulaşmamızda mümkün olacaktır. Bu duruma örnek olması için kategoriler bölümünü basit bir şekilde daha alt kategoriler şeklinde göstermek istersek;

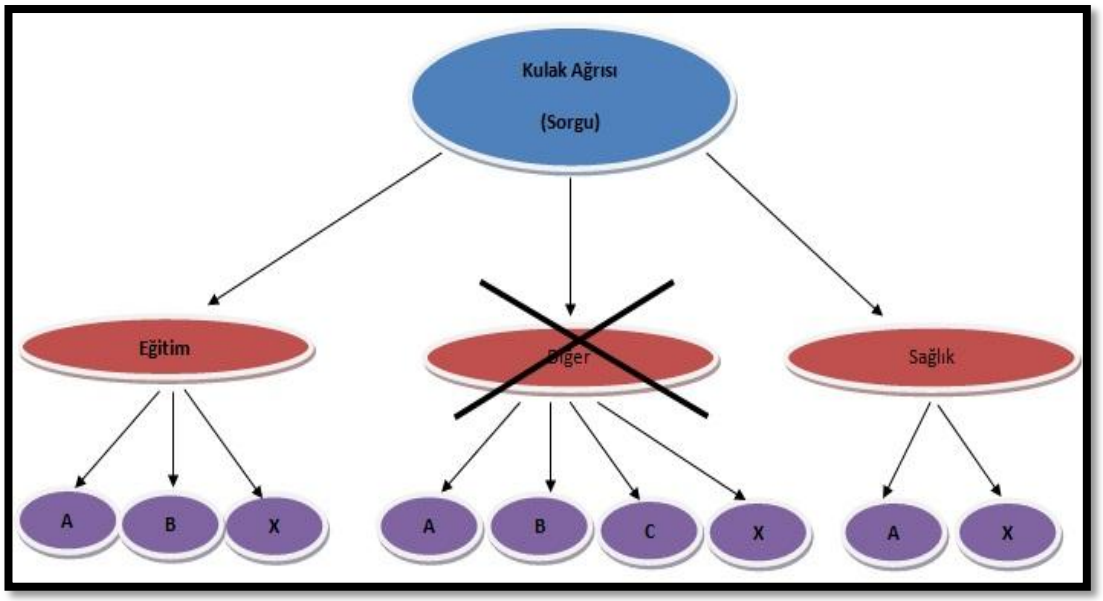


Şekil 5.4 Kullanıcı Kontrollü Arama Motorunda Örnek Kategori Gösterimi

Şeklinde daha alt kategorilere de ayırabiliriz. Tabi ki daha önce söylediğimiz gibi bu kategorilerin oluşturulması sıkı bir çalışma gerektirir. Bu konuda arama motorlarına en yakın dallardan biri olan veri madenciliği yöntemlerinden yararlanılarak web siteleri kategorileştirilebilir. Böylece bu tarz aramalarda devamlı ilk yirmi sırayı

tekelinde bulunduran sitelerin oluşturduğu adaletsiz durumda bir nebze de olsa azalacaktır.

Oluşturduğumuz bu yapı bir nevi veri madenciliğinde kullanılan sınıflandırma işlemlerine girer. Buna göre girilen sorgu karar ağaçlarında ilerlerken en son döngüde kalan web sayfaları arama motoru mimarisine yönlendirilerek sıralamaya konulması işlemiyle sayfalar sıralanır. Bu durumu daha açık görmek için yine görsel bir gösterim şekline bakalım.



Şekil 5.5 Kategoriye Göre Sorgunun Sınıflandırılması

Görüldüğü gibi sorgu başlamadan kullanıcının belirlediği alanlara göre tarama başlamakta ve böylece sıralamaya alakasız sayfaların gelmesi önlenmektedir. Ağaç yapılandırmasında kategori aşaması bitince, ardından bir sonraki filtreleme işlemleri olan site uzantısı, formatı, yapısı, dili gibi birçok özellik sırayla uygulanarak en son kalan web sayfaları sorgu bölümüne gönderilmektedir. Ardından sorgu bölümünde sıralanan web sayfaları kullanıcı ara yüzüne gönderilmektedir. Günümüz arama motorları kullanıcıdan alınacak değerlerden oluşacak PageRank vektör kombinasyonunun çok fazla olmasından kaynaklı bu işlemi şu an için gerçekleştirememektedirler.

5.2.2. Hiyerarşik Arama Motorları

Arama motorlarının kalitesi kullanıcılar tarafından anında hissedilir. Daha önce bahsettiğimiz gibi kullanıcıların birçoğu ilk yirmi sonuçtan sonra sıkılıp, sorgusunu

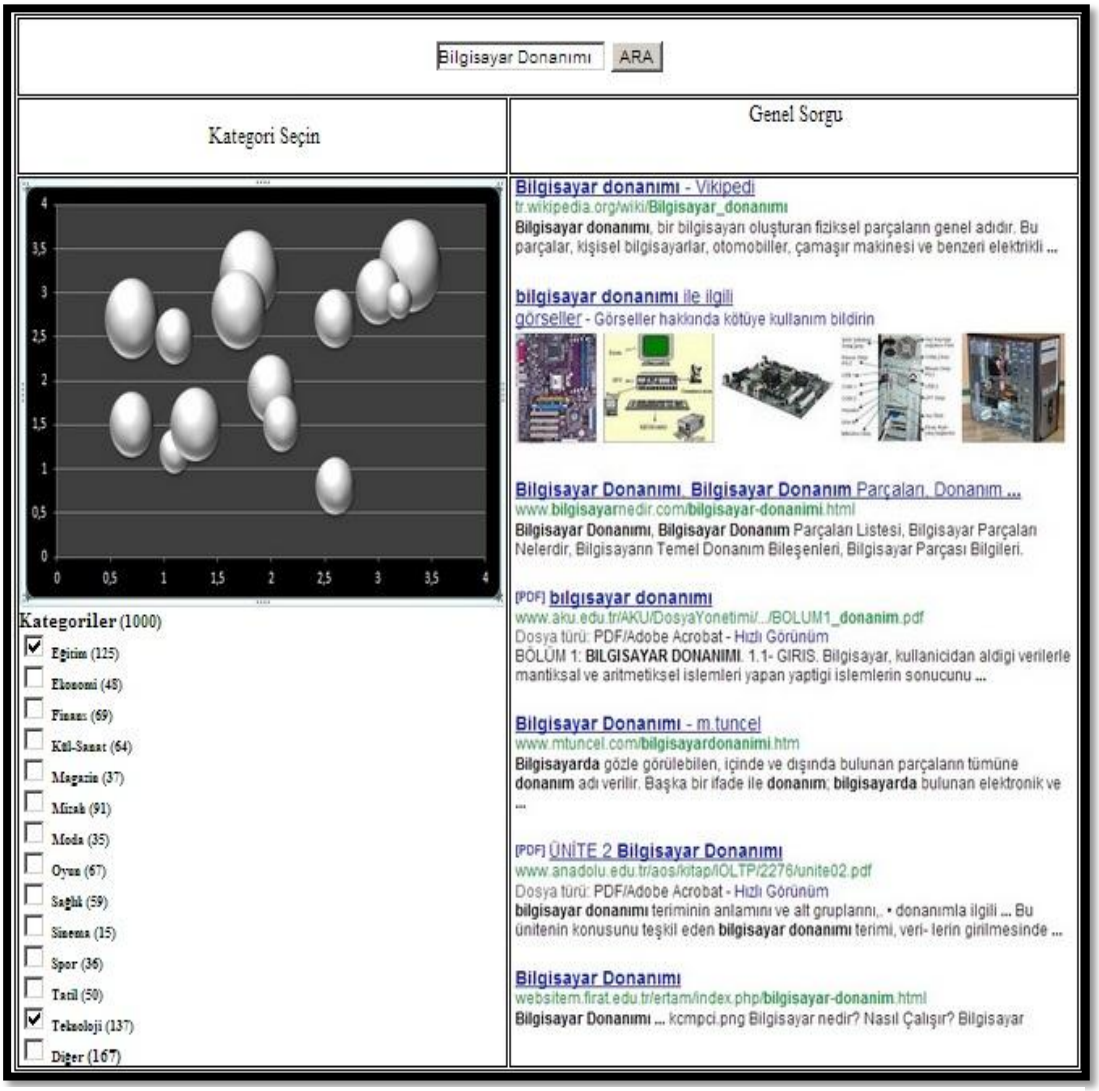
değiştirmektedirler. Arama motorları da bu durumu istatistikî olarak çözümlediklerinden olabildiğince sorguyla ilgili en yakın yirmi sonucu göstermeye çalışmaktadır. Her geçen gün bilinçlenen kullanıcı profili ile uyuşmayan karma sorgu yapısı, kendisine yeni bir kalıp aramaktadır. Düşünülen hiyerarşik arama motorları da bu sorunu giderecek gözüyle bakılmaktadır. Hiyerarşik arama motorlarında, kullanıcılar her gelen sorgu kümesinden (clustering) bir alt küme seçerek daha spesifik bir sorgulama sonucuna kavuşmaktadır. Her seçilen alt kümede ise arama motoru yeniden bir değerlendirme yapıp, yeni bir küme sunmaktadır. Kullanıcılar ise her defasında yeni bir alt küme seçip, kendilerini tatmin eden bir noktaya kadar ilerlemektedirler. Böylece kullanıcı bulmak istediği bilgiye daha hızlı ve daha kontrollü sahip olmaktadır. Bu durumu son dönemlerde web ortamında çıkan haritalara benzetebilirsiniz. Karşınıza ilk önce dünya haritası gelirken, tıkladığınız anda harita büyümekte ve belirlediğiniz noktadan daha fazla bilgi almaktasınız. Yani haritada yakınlaştırma butonuna her tıkladığınızda o noktaya ait daha ayrıntılı bilgi karşınıza geldiği gibi, istediğinizde uzaklaştırma butonu ile daha genel görünümde haritaya erişebiliyorsunuz. Hiyerarşik arama motorlarının da bu mantıkla çalışacağını düşünebilirsiniz. Buna göre aradığınız kelime ile ilgili sonuçların hangi alanda ne kadar sonuç bulunduğunu görerek seçip bir alt kategoriye geçiyorsunuz. Sonra bir sonraki alt kümede bulunan alt kümelerde aradığınız kelimenin ne kadar bulunduğuna bakıyorsunuz. Her atlamada arama motoru sıralamayı yenileyip, yeni bir sonuç karşınıza getiriyor. Aslında bu anlattıklarımızı "sorguya özgü" arama motorlarının tersi bir mantıkla çalıştığını düşünebilirsiniz. Sorguya özgü arama motorlarında kullanıcı, alanları sorgudan önce belirlerken, hiyerarşik arama motorlarında ise kullanıcı, sorgudan sonra alanları belirlemektedir. Kişiselleştirme vektörü aynı şekilde bu bölümde de karşımıza çıkmaktadır. Günümüzde bazı arama motorları bu durumu önceden hesaplanmış PageRank değerine göre hesaplarken, geleceğe dair düşüncemiz seçilen her kategoride PageRank vektörünün yeniden hesaplanmasıdır. Araştırmamızda anlattığımız gibi web siteleri arasındaki link alış verişi çok önemliydi. Her web sayfası elindeki PageRank puanını link verdiği sitelere dağıtıyordu. Buna göre bbc.com'un veya milliyet.com'un size verdiği puan ile sıradan yüz sitenin size verdiği puana eşdeğer olabileceğini belirtmiştik. Fakat Hiyerarşik arama motorlarında düşünülen kişiselleştirme (bu bölüm için alanlaştırma diyebiliriz) vektörünü kullanıcının her alt küme seçiminde değiştirilmesinden bahsediyoruz. Yani eğitim ile ilgili bir web sitenizin olduğunu ve bir de rakip bir

sitenizin olduğunu düşünün. Rakip siteye bbc.com link verirken, sizin sitenize OSYM link vermektedir. Buna göre ilk arama sıralamasında rakip site sizin önünüze geçerken, kullanıcının genel aramanın bir alt kümesi olan eğitim alanına tıkladığında, sizin siteniz OSYM'den link aldığı için sıralamada rakip sitenizin önüne geçmiş oluyorsunuz. Bu bölümdeki mantık, eğer kullanıcı eğitim ile ilgili bir araştıma yapıyor ise, arama motoru bu alanda en güvenilir ve ya en tanınmış ve ya PageRank puanı en yüksek olan kimdir diye bir soru soruyor. Cevaba göre OSYM'nin PageRank puan değeri bbc.com'dan daha yüksek olacağından, sizin siteniz sıralamada rakip sitenizin önüne geçiyor. Başka bir deyişle rakibiniz bir haber ile, bir durumu ile (ya da bir reklamı ile!) bbc.com'da geçiyor olabilir fakat bu durum rakip sitenizin sizden daha iyi eğitim içeriği sunduğu anlamına gelmez. Genel arama itibariyle rakip siteniz sizden daha tanınmış bir konumda olabilir, fakat kullanıcının eğitim ile ilgili arayacağı daha spesifik bir aramada sizden daha üst sıralarda bulunması mantıksız olacaktır. Tabi ki bbc.com'un link değeri bir alt kategoride de olacaktır fakat kendini tamamen eğitime adamış bir siteden ve ya eğitim alanında PageRank değeri çok yüksek olan OSYM'den daha yüksek olması beklenemez. Buna göre hiyerarşik arama motorlarında her alan kendi içinde birbirini değerlendirmektedir. Aslına bakarsanız web sayfaları da aynen insan topluluklarının birbirini yönetmesine benzer, her zümre kendi içinde seçim yaparken bir üst kurula geçtiğinde bağlı olduğu bir üst zümreye göre düşünülür. Yani iç içe geçen her bir halkada kendi içinde bağımsız ya da değerli fakat bir üst halka da kendisini çevreleyen halkaya bağlıdır.

Sorguya özgü arama motorlarında bahsettiğimiz gibi, Hiyerarşik arama motorlarında da düşündüğümüz mantık veri madenciliği alanında sıkça kullanılmaktadır. Zaten arama motorları ve veri madenciliği alanları iç içe geçmiş konumdadır. Veri madenciliği alanında kümeleme (clustering) yöntemleri dediğimiz yapılar hiyerarşik arama motorlarının alt yapısını oluşturmaktadır. Özellikle benzerlik ve uzaklık, hiyerarşik yöntemler, bölümlemeli yöntemler, yoğunluğa dayalı algoritmalar, Grid temelli algoritmalar ve hatta genetik algoritmalar bu yapı için denenebilir. Araştırmamızda da sıkça belirttiğimiz gibi, aramam motorları teorik incelemelerden çok deneysel incelemelerden gelişmektedir. Çünkü tahmin etmeye çalıştığımız insan beynidir. Teorik anlamda birçok anlam ifade eden bir algoritma, insanlar için bir anlam ifade etmez iken, çok basit ve ya farklı bir algoritma insanlar tarafından

rahatlıkla kabul edilebilir ve kullanılabilir. Arama motorlarının ara yüzünde girilen sorgu insanoğlu için çok farklı çağrışımlar uyandırırken, matematiksel olarak kullandığımız algoritma açısından hiçbir anlam ifade etmeyebilir. Bu bağlamda oluşturacağımız bütün algoritmalar için belirli kullanım ve kolaylık istatistikleri tutarak, her defasında değerlendirmeye tabi tutmak zorundayız. Arama motorları açısından insan davranışlarından elde ettiğimiz değerler oluşturduğumuz algoritmanın başarısına yansıyacaktır. Tabi ki bu süreç epey bir zaman, emek ve para gerektirebilir. Zaten günümüzde de büyük arama motorlarının farklı algoritmalar denemekten çekinmeleri bu sebeptendir.

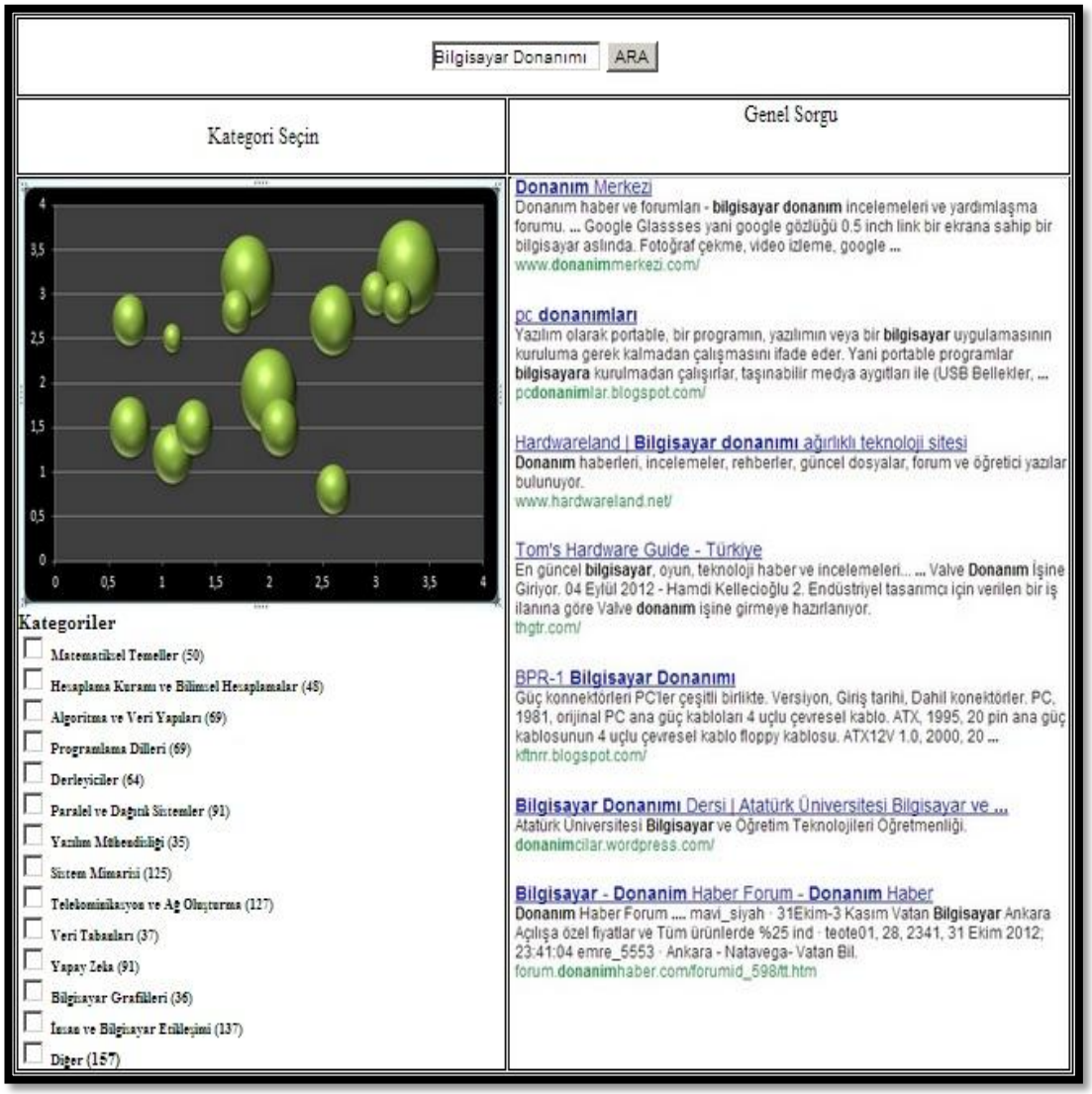
Bu anlattıklarımızın görsel bir sunumunu aktarmak istersek;



Şekil 5.6 Sorgu Sonucunun Kategorilere Bölünmesi

Görüldüğü gibi kullanıcı ilk aramada sağ taraftaki sonuçları görecektir. Sol bölümde ise kullanıcılar için ayrıntılı aramak istedikleri alan ile ilgili seçenekler

sunulmaktadır. Kullanıcı bu bölümde ayrıntılı arama yapmak istediği alan ile ilgili kutucukları işaretleyip ya da direkt grafikten istediği alana tıklayıp, daha ayrıntılı arama sonuçlarını görecektir. Bir sonraki aşamada ise görüntümüz şu şekilde olacağını düşünün.



Şekil 5.7 Sorgu Sonucunun Alt Kategorilere Bölünmesi

Görüldüğü gibi kullanıcı bir alt kategoriyi seçtiğinde sıralama değişecek ve sorgulama bir alt kümeye düşecektir. Aynı şekilde kullanıcı sorgulamayı bir adım öteye götürmek ister ise yine aynı şekilde sol bölümden ilgili alanların kutucuklarını işaretleyip yeniden sorgulamayı başlatabilecektir. Hiyerarşik arama motorları sayesinde kullanıcı karma aramada gezdiği ilk yirmi sayfadan sonra sorguyu değiştirmektense, ilgilendiği alan ile ilgili daha kaliteli bir ilk yirmi sayfa görecektir. Böylece sorgu kalitesi yükseldiği gibi, ilk yirmi sırayı tekelinde bulunduran siteler

etkisini yitirmiş olacaktır. Bu sayede web siteleri açısından pastadan kendi paylarına düşen oran bir nebze de olsa daha adaletli bir şekilde bölüştürülmüş olacaktır.

Yukarıda bahsettiğimiz mantık açısından benzer şekilde çalışan arama motorları dışında, farklı tür arama motorları da günümüzde giderek artmakta ve geleceğin arama motorları arasında yerini almaya başlamaktadır. Gelin gelecekte sıklıkla karşılaşacağımız diğer arama motorlarına da bir göz atalım.

5.2.3. Akıllı Arama Motorları

Geleceğin düşünülen bir diğer arama motoru ise akıllı arama motorlarıdır. Akıllı arama motorlarının tamamen kişiye özgü arama motorları olacağı düşünülmektedir. Buna göre arama motorları sizin özellikleriniz temelinde, arama geçmişinizi ve bilgisayarınızdaki çerezlerinizi (cookies) devamlı kaydederek sizin bir profilinizi çıkarmakta ve böylece sizin ilgilendiğiniz alanlara göre, zamanınız çoğunu geçirdiğiniz sayfalara göre istatistikler çıkarıp yapacağınız sorgulamalarda sıralamayı değiştirebileceklerdir. Daha önce bahsettiğimiz gibi arama motorları PageRank puan dağılımını da sizin profilinizi düşünerek paylaştıracaklardır. Ayrıca yeni yayın hayatına başlayan web sayfalarını, eğer sizin ilgilendiğiniz alan ise, size sunacak ve belki de sizin çok işinize yarayacak fakat PageRank değeri çok düşük olduğu için belki de hiç karşılaşmayacağınız web sayfalarını karşınıza getirecek ve sizi haberdar edecektir. Yani sizin oturup araştırmanız bulmanız gereken web sayfalarını akıllı arama motorları sizin yerinize bulacaktır. Bu bahsettiğimiz arama motorları sadece yeni web sayfalarını bulmakla sınırlı kalmayacaktır. Ayrıca ilgilendiğiniz alanlar, kişiler, haberler gibi devamlı takip etmeniz gereken bir akışı sizin yerinize takip edecek ve sizin bundan haberdar olmamızı sağlayacaktır. Çoğunlukla iş hayatında kullanılacağı düşünülen akıllı arama motorları, ilerleyerek her bireye hitap edecek düzeye geleceği düşünülmektedir. Hatta akıllı arama motorlarına "evcil hayvanım" şeklinde isimler şimdiden verilmeye başlanmıştır. Buna göre herkesin sanal evcil bir arama motoru olacak ve sizin yerinize bütün gündemi takip edecek, tarayacak ve sizi haberdar edecektir. Günümüzde bu takip işlerini genelde şirketler yapmaktadır. Bu şirketler sizin işiniz ile ilgili yasaları, haberleri, ilanları araştırıp size iletmektedir. Böylece sizlerde işinizin güncelliği, rakiplerinizin durumları gibi birçok durumdan haberdar olup belki de duruma göre gardınızı almaktasınız. İşte bahsettiğimiz sanal evcil hayvan dediğimiz akıllı arama motorları da bu görevi üstelenecek ve sizin gündemi takip etmenizi ve yakalamanızı sağlayacaklar. Hatta belli alanlarda bir kaç

tane sanal evcil hayvanımız olacak ve bize hizmet edeceklerdir. Bizler geleceğin akıllı arama motorlarını kullanırken, arama motorları da hızla gelişerek, telefon operatörleri gibi pastadan olabildiğince büyük pay almak için yarışacaklardır. Belki de düşünülen bu akıllı arama motorlarının sanal evcil hayvanları adet adet satışa sunulacak ve ihtiyacınız olan alanlar ile ilgili evcil hayvanları seçip, satın alacaksınız. Yani günümüz telefon operatörlerinde olduğu gibi evcil hayvanlarınızın özelliklerini sunulan paketlere göre belirleyip satın alacaksınız. Şimdiden geleceğin akıllı arama motorları sabırsızlıkla beklenmektedir.

5.2.4. Özel Amaç Arama Motorları

Son olarak değineceğimiz arama motorları ise özel amaç arama motorlarıdır. Arama motorlarının geleceğini düşündüğümüzde, özel amaç arama motorlarına yenik düşecekleri ve günümüzün dev arama motorlarının da yavaş yavaş parçalanarak özel amaç arama motorlarına dönüşecekleri şimdiden tartışılmaktadır. Özel amaç arama motorları, sadece belli alanlarda çalışma yapan arama motorlarıdır. Yani nasıl ki günümüzde karşılıklı sohbet programları denilince Msn veya Skype akla geliyorsa, ilerleyen yıllarda da belirli alanlara yönelik aramalarda belli başlı arama motorlarının isimleri anılmaya başlanacaktır. Bu tarz arama motorları günümüzde kullanılmakta ve giderek daha profesyonel hizmet vermeye başlamaktadırlar. Örneğin internet üzerinden kitap arayanlar için, aradığınız kitabı daha önceden veri tabanına kaydettiği kitap satışı ile ilgilenen web sitelerinden bularak karşınıza getirecektir. Hatta bulunduğunuz konuma göre size en yakın kitapçının yerini belirleyecek, iletişim bilgilerini getirecek ve hatta alacağınız kitap ile ilgili ayrıntılı bilgiler verecektir. Bir başka örnek ise sağlık alanından verebiliriz. Sadece eczanelerin bilgilerinin bulunacağı arama motorları olacaktır mesela. Herhangi bir aramada size en yakın eczaneyi bulacak hatta aradığınız ilacın ilgili eczanede bulunup bulunmadığını, fiyatını, iletişim adresini karşınıza getirecektir.

Özel amaçlı arama motorları günümüzde giderek artmaktadır. Öyle ki şimdiden bilgi toplama yarışına girilmiştir. Her alana ait bilgiler birileri tarafından depolanmaktadır. Bu mantık eskiden yaptığımız belli bir şeyin koleksiyonunu tutmaya benzer. Günümüzde de her alanda iş yapan büyük şirketler, alanları ile ilgili bilgileri yavaş yavaş depolamaktadır. İleride daha sivri alanlarda kullanacağımız özel amaç arama motorları da bu amaca hizmet edecektir. Elinde belirli bir alana yönelik en fazla bilgiyi toplayan arama motoru, en fazla kar eden arama motoru olacaktır.

Eczanelerden tutunda otobüslere, pastacılara, okullara, esnaflara, sinemalara, alış-veriş merkezlerine aklınıza gelebilecek bütün her şeyin bilgisini en fazla tutan özel amaç arama motoru, insanlar arasında o kadar tanınacak ve o kadar kullanılacaktır. Hatta değerli alanlara yönelik çalışan arama motorları arayacağınız bilgi başına ücret alacağı bile düşünülmektedir.

Günümüzde bu alanda dile getirilen birçok komplo teorisi de bulunmaktadır. Buna göre özel amaçlı arama motorlarında kullanılacak bilgilerin şimdiden birçok şirket tarafından satın alındığı söylenmektedir. Özellikle sağlık ve sigortacılık alanında bilgilerin satıldığı ve ya biriktirildiği dile getirilmektedir. Örneğin bu teoriye göre ileride sigortacılık alanında, özel amaçlı arama motorlarında kullanılacak bilgilerin şimdiden biriktirildiği ve ileride kullanılacağı şeklinde düşünceler dile getirilmektedir. Buna göre sağlık alanında hastanelerde yaptığınız bütün işlemler bu veri tabanlarına kaydedilmekte ve sizin bir profiliniz çıkarılmaktadır. Yani gelecekte bir sigorta şirketine gidip kendinizi olası bir kanser hastalığına karşı sigortalattırmak istediğinizi söylediğinizde, sizin hatta ailenizin bilgilerini özel amaçlı arama motorları sayesinde bulacak ve eğer kanser hastalıklarına karşı büyük bir yatkınlığınız var ise, sizi sigortalattıramayacaklarını söyleyecekleri düşünülmektedir. Fakat dediğimiz gibi bunlar şimdilik komplo teorilerinden ibaret. Umudumuz özel amaçlı arama motorlarının kötü niyetle kullanılmamasıdır.

Bu bahsettiğimiz komplo teorileri sadece sağlık alanından ibaret değildir elbet. Eğitim, eğlence, iş gibi birçok alanda da bilgilerimizin satıldığı ve büyük şirketler tarafından satın alındığı dile getirilenler arasındadır. Buna göre şirketler tarafından profiliniz çıkarılmakta ve size özel ürünler tasarlanacağı, sunulacağı ve ya satılacağı dile getirilmektedir. Özellikle sanal ortamda paylaştığımız resimler, videolar gibi, yaptığımız yorumlar gibi, attığımız twitler gibi aklınıza gelebilecek her şeyin bu alanla ilgilenen birileri tarafından depolandığı söylenmektedir. Öyle ki marketlerden aldığınız besinlerden bile yola çıkılarak bir profiliniz oluşturulmakta ve size özel teklifler ve ya fiyat indirimleri sunulacağı ve ya sunulduğu söylenmektedir. Yani aklınıza gelebilecek bütün şeylerin bilgisi aynen koleksiyoncular gibi o alanla ilgilenen birileri tarafından biriktirilmekte ve ya satın alınmaktadır. Ardından biriktirilen bilgiler ya ilgili veri tabanından ya da bütün web ortamından özel amaçlı arama motorlar sayesinde erişilmekte ve kullanılmaktadır.

Dediğimiz gibi özel amaçlı arama motorlarının veri tabanları şimdiden oluşturulmaya başlanmıştır. Özellikle iş, eğitim, sağlık, eğlence gibi alanlarda yoğunlukla kullanılacağı düşünülmektedir. Şimdiden gelecekte insanlar tarafından ilgi görecek alanlara yönelik bilgilerin biriktirilmesi, büyük kazançlar getireceği söylenmektedir. Zaten büyük düşünürler tarafından devamlı dile getirilen, "Geleceğin en fazla kar getireceği şey, bilgidir" demesi boşuna değildir.

KAYNAKÇA

Search Engine Optimization Starter Guide. (2010). Eylül 2012 tarihinde google.com: http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/tr//webmasters/docs/search-engine-optimization-starter-guide.pdf adresinden alındı

Stokastik Matematiksel Modeller ve Süreçler. (2012). Eylül 2012 tarihinde sakarya.edu.tr: 2012 http://web.sakarya.edu.tr/~kubat/?loc=download&file... adresinden alındı

The Mathematics of Web Search. (2012). September 2012 tarihinde math.cornell.edu: http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/index.html. adresinden alındı

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on search and development in information retrival* (s. 19-26). ACM.

Altıngövde, İ. S. (2009, July). Improving The Efficiency of Search Engines: Strategies For Focused Crawling, Searching, and Index Pruning. *Ph.D. Dissertation*. Ankara: Bilkent University, Dept. of Computer Engineering.

Aydemir, M. (2010). *Search Engine Optimization.* İstanbul: Kodlab Yayıncılık.

Brin , S., Page, L., Motwami, R., & Winogard, T. (1999). *The PageRank citation ranking: Bringing order to the Web.* Stanford University, Computer Science Department.

Brin, S., & Page, L. (1998, April). *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. Eylül 2012 tarihinde infolab.stanford.edu: http://infolab.stanford.edu/~backrub/google.html adresinden alındı

Budon, D. (2005, June). *The Basic of Search Engine Optimisation*. Eylül 2012 tarihinde simplyclicks.com: http://www.simplyclicks.com/Free-SEO-Book.pdf adresinden alındı

Croft, W. B., Metzler, D., & Strohman, T. (2010). *Search Engines Information Retrival in Pratice.* New Jersey: Pearson .

Çavdur, F. (2005). Arama Motorları Kullanıcı Oturumlarındaki Konu Değişikliklerinin Tespit ve Tahmin Yöntemleri. *Yayımlanmamış Yüksek Lisans Tezi*. Bursa: Uludağ Üniversitesi, Fen Bilimleri Enstitüsü.

Çetin, N., & Orhun, N. (1998). *Lineer Cebir.* Eskişehir: Anadolu Üniversitesi Yayınları.

Dündar, E. C. (2009). Arama Motorlarında Kullanılan Arama Robotu Mimarilerinin İncelenmesi ve Yeni Bir Yaklaşım Sunulması. *Yayımlanmamış Yüksek Lisans Tezi*. Edirne: Trakya Üniversitesi, Fen Bilimleri Enstitüsü.

Ekström, P.-A., & Andersson, E. (2004). Investigating Google's PageRank Algorithm. *Report in Scientific Computing*. Sweden: Uppsala University, Dept. of Scientific Computing.

Grappone, J., & Couzin, G. (2006). *Search Engine Optimization: An Hour a Day.* Indiana: Wiley Publishing.

Gülten, K. (2011). *Uzmanından SEO.* İstanbul: Dikeyeksen Yayıncılık.

Hill, B. (2005). *Google Search & Resque For Dummies.* Indiana: Wiley Publishing.

Humenberger, H. (2011, February). *How does Google come to a ranked list? - making visible the mathematics of modern society.* Eylül 2012 tarihinde teamat.oxfordjournals.org: http://teamat.oxfordjournals.org/content/early/2011/05/12/teamat.hrr007.full.pdf+html adresinden alındı

Langville, A. N., & Meyer, C. D. (2004, October 19). Eylül 2012 tarihinde The Use of The Linear Algebra by Web Search: http://meyer.math.ncsu.edu/Meyer/PS_Files/IMAGE.pdf adresinden alındı

Langville, A. N., & Meyer, C. D. (2004, October 20). *Deeper Inside PageRank.* Eylül 2012 tarihinde emba.uvm.edu: http://www.emba.uvm.edu/~tlakoba/AppliedUGMath/DeeperInsidePageRank.pdf adresinden alındı

Langville, A. N., & Meyer, C. D. (2006). *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton, New Jersey: Princeton University Press.

Odabaşıoğlu, C. (2009, Şubat). İnternet Arama Motorları Analizi. *Yayımlanmamış Yüksek Lisans Tezi*. İstanbul: Haliç Üniversitesi, Fen Bilimleri Enstitüsü .

Ridings, C. (2001). *PageRank Explained.* Eylül 2012 tarihinde netsavy.net: http://netsavy.net/Graphics/PageRank.pdf adresinden alındı

Ridings, C., & Shishigin, M. (2002). *PageRank Uncovered.* Elül 2012 tarihinde voelspriet2.nl: http://www.voelspriet2.nl/PageRank.pdf adresinden alındı

Sezgin, G. (2009, Haziran). Arama Motorlarının Davranışlarının Çözümlenmesi ve Web Sayfalarının Tasarım Aşamasında Yansıtılması. *Yayımlanmamış Yüksek Lisans Tezi*. İstanbul: Beykent Üniversitesi, Fen Bilimleri Enstitüsü.

Sission, D. (2004). *Google Secrets: How to get a top 10 Ranking.* Eylül 2012 tarihinde oocities.org: http://www.oocities.org/hiubwen/google_info.pdf adresinden alındı

Şeker, Ş. E. (2012). *Markov Model.* Eylül 2012 tarihinde bilgisyarkavramlari.com: http://www.bilgisayarkavramlari.com/2009/06/17/markof-modeli-markov-model/. adresinden alındı

Viney, D. (2008). *Get to the Top on Google.* London: Nicholas Brealey Publishing.

Wills, R. S. (2007). When Rank Trumps Precision: Using The Power Method to Compute Google's PageRank. *Ph.D. Dissertation*. Raleigh, North Carolina: North Carolina State University, Dept. of Mathematics.