T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

POPÜLARİTE TABANLI SIRALAMA ALGORİTMALARININ DOĞRULUK DEĞERLERİNİ ARTTIRMAYA YÖNELİK YÖNTEM GELİŞTİRİLMESİ

SİNEM KIRTAÇ

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. MEHMET SIDDIK AKTAŞ

İSTANBUL, 2014

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

POPÜLARİTE TABANLI SIRALAMA ALGORİTMALARININ DOĞRULUK DEĞERLERİNİ ARTTIRMAYA YÖNELİK YÖNTEM GELİŞTİRİLMESİ

Sinem KIRTAÇ tarafından hazırlanan tez çalışması 12.12.2014 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı' nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Dr. Mehmet Sıddık AKTAŞ

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Mehmet Sıddık AKTAŞ

Yıldız Teknik Üniversitesi _____

Doç. Dr. Banu DİRİ

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Arzucan ÖZGÜR

Boğaziçi Üniversitesi _____

Bu çalışma, Yıldız Teknik Üniversitesi BAP Projesi (Proje No: 2013-04-01-KAP03) ile desteklenmiştir.

ÖNSÖZ

Bu çalışmanın başından sonuna kadar yapmış olduğu önerileri ve katkıları için Sayın Yrd. Doç. Dr. Mehmet Sıddık AKTAŞ' a çok teşekkür ederim.

Bu çalışma süresince bana yardımcı olan proje arkadaşım Sayın Canan GİRGİN' e teşekkürü borç bilirim.

Hayatım boyunca desteklerini hiçbir zaman eksik etmeyen sevgili anneme, babama ve beni yüreklendiren ve çalışmalarım boyunca bana destek olan eşime çok teşekkür ederim.

Aralık, 2014

Sinem KIRTAÇ

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ viii

ŞEKİL LİSTESİ ix

ÇİZELGE LİSTESİ x

ÖZET xi

ABSTRACT xiii

BÖLÜM 1

GİRİŞ 1

1.1 Literatür Özeti 2
1.2 Tezin Amacı 4
1.3 Hipotez 5

BÖLÜM 2

GENEL BİLGİLER 7

2.1 Genel Bilgiler 7
2.1.1 Sıralama Algoritmalarının Ortaya Çıkma Nedenleri 15
2.1.2 Arama Motorlarının Karşılaştığı Zorluklar 16
2.1.3 Sıralama Algoritmalarındaki Eksiklikler 17
2.1.4 Motivasyon 17
2.2 Tez Kapsamında Kullanılan Teknolojiler 18
2.2.1 Apache Nutch 18
2.2.2 Apache Lucene ve Solr 18
2.2.3 Apache HBase 20

2.2.4 Apache Hadoop 20
2.2.5 MapReduce 23
2.2.6 Apache ZooKeeper 23
BÖLÜM 3
ARAŞTIRMA PROBLEMİNİN TANIMI 25
3.1 Araştırma Soruları 25
BÖLÜM 4
ÖNERİLEN ÇÖZÜM YÖNTEMİ 27
4.1 Klasik PageRank Algoritması 27
4.2 Önerdiğimiz İyileştirilmiş PageRank Sıralama Algoritması 28
4.3 Yöntemimizin Dayandığı Öngörüler 28
4.4 İyileştirilmiş PageRank Sıralama Yöntemi için izlediğimiz adımlar 30
4.5 Veri Kümesi 33
BÖLÜM 5
UYGULAMA DETAYLARI 36
5.1 Apache Nutch Üzerindeki Geliştirmelerimiz 38
5.2 Apache Lucene ve Solr 43
5.3 Apache HBase 43
5.4 Apache Hadoop 43
5.5 Apache ZooKeeper 43
BÖLÜM 6
YÖNTEMİN DEĞERLENDİRİLMESİ 44
6.1 Kullanıcı Değerlendirme Çalışması 45
6.2 Değerlendirme 48
BÖLÜM 7
SONUÇLAR VE GELECEKTE PLANLANAN ÇALIŞMALAR 50
7.1 Araştırma Sorularına Cevaplar 51
7.1.1 Dolaylı bağlantılar popülarite tabanlı algoritmaların geliştirilmesinde kullanılabilir mi? Dolaylı bağlantılar algoritma geliştirilmesinde yeterli midir? 51
7.1.2 Dolaylı bağlantılar kullanılarak doğruluk değerini artırmak için yöntem ne olabilir? 51
7.1.3 Web sayfasının içinde dolaylı bağlantı tespiti nasıl yapılabilir? 52

7.1.4 Web sayfasının içinden elde edilen dolaylı bağlantılarla geliştirilmiş sıralama algoritması klasik PageRank ya da TF-IDF algoritmalarıyla karşılaştırıldığında elde edilen sonuçlar nedir? 52
7.2 Gelecekte Yapılacak Çalışmalar 53
KAYNAKLAR 54
ÖZGEÇMİŞ 58

KISALTMA LİSTESİ

BİLGEM	Bilişim ve Bilgi Güvenliği İleri Teknolojiler Araştırma Merkezi
FTP	File Transfer Protocol
GFS	Google File System (Google Dosya Sistemi)
HDFS	Hadoop Distributed File System (Hadoop Dağıtık Dosya Sistemi)
idf	Ters Doküman Frekansı
Jughead	Jonzy' s Universal Gopher Hierarchy Excavation and Display
P	Precision-Tutturma
R	Recall-Bulma
tf	Terim Frekansı
TÜBİTAK	Türkiye Bilimsel ve Teknolojik Araştırma Kurumu
URL	Uniform Resource Locator (Standart Kaynak Bulucu)
Veronica	Very Easy Rodent-Oriented Net-wide Indeks to Computerized Archives

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1.1 Arama motoru çalışma şekli....9
Şekil 2.1.2 İçerik çekme süreci....11
Şekil 2.1.3 Arama motorlarında indeksleme....13
Şekil 2.1.4 Ters indeks örneği....14
Şekil 2.1.5 Arama motorlarında sorgulama işlemi....15
Şekil 4.4.1 Önerdiğimiz dolaylı bağlantıların tespit algoritması....31
Şekil 5.1.1 Çalışma ortamı....38
Şekil 6.1.1 Giriş sayfası....46
Şekil 6.1.2 Sorgu sayfası....47
Şekil 6.1.3 Sonuç sayfası....47
Şekil 6.1.4 Sonuç sayfası 2....48
Şekil 6.2.1 Farklı sıralama algoritmalarının tutturma ve bulma grafiği....49

ÇİZELGE LİSTESİ

Sayfa

Çizelge 4.4.1 Örnek metin-bağlantı eşleşmesi.....33
Çizelge 5.1.1 Webpage tablosu alanları 39

ÖZET

POPÜLARİTE TABANLI SIRALAMA ALGORİTMALARIN DOĞRULUK DEĞERLERİNİ ARTTIRMAYA YÖNELİK YÖNTEM GELİŞTİRİLMESİ

Sinem KIRTAÇ

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç. Dr. Mehmet Sıddık AKTAŞ

Web üzerinde erişilebilir doküman sayısı oldukça fazladır ve bu sayı katlanarak büyümektedir. Belirli konularda bilgi aramak için interneti kullanan insan sayısı giderek artmaktadır. Fakat Web devasa bir bilgi kaynağı olmakla beraber dikkatli kullanılmalıdır. Bilgi aramada kullanılan en önemli araç Web arama motorlarıdır. Web arama motorları performanslarını geliştirmek için bazı zor problemlerle karşı karşıyadır. Bir sorguya yanıt vermek için en uygun sonuçların bulunması çok önemli bir problemdir. Biz bu çalışmada popülarite tabanlı sıralama algoritmalarının bir örneği olan PageRank algoritmasının doğruluk değerini arttırmak için bir yöntem öneriyoruz.

Önerdiğimiz bu yöntemde Web sayfalarında bulunan dolaylı bağlantılardan yararlanıyoruz. Birçok Web sayfasında, başka Web sayfaları bahsedilirken gerçek bağlantı (hyper-link) verilmeden, sadece Web sayfalarının isimleri kullanılarak atıf yapılabilmektedir. Bu atıflar Web sayfalarına yöneltilmiş dolaylı bağlantılar olarak kabul edilebilir. Dolaylı bağlantılar, gerçek bağlantılar gibi Web sayfalarının popülaritesinin belirlenmesinde etkin olabilmektedir. Buna dayalı olarak geliştirdiğimiz yöntemde Web sayfaları arasındaki gerçek bağlantıların yanı sıra, dolaylı bağlantıları da dikkate alarak zenginleştirilmiş bir Web çizgesi oluşturuyoruz. Yöntemimizde, bağlantı tabanlı popülarite hesaplama algoritmalarının bir örneği olan PageRank algoritmasını, dolaylı bağlantıları da dikkate alacak şekilde geliştiriyoruz. Güncellediğimiz PageRank algoritmasını, zenginleştirilmiş Web çizgesi üzerinde koşturarak, yöntemimizin benzerlik tabanlı sıralama algoritmaları ve klasik PageRank algoritmasına göre daha iyi

sıralama sonuçları verip vermediğini araştırıyoruz. Son kullanıcılarla yaptığımız değerlendirme çalışması sonucunda, önerdiğimiz yöntem karşılaştırdığımız diğer iki yönteme göre tutturma ve bulma metrikleri açısından daha başarılı sıralama sonuçları vermiştir. Elde ettiğimiz sonuçlar dolaylı bağlantıların popülarite tabanlı sıralama algoritmalarının doğruluk yüzdelerini arttırdığını göstermektedir.

Anahtar Kelimeler: Popülarite tabanlı sıralama, dolaylı bağlantılar, PageRank, bağlantı tabanlı sıralama algoritmaları

ABSTRACT

DEVELOPING METHODOLOGY TO IMPROVE THE ACCURACY OF POPULARITY BASED RANKING ALGORITHMS VIA IMPLICIT LINKS

Sinem KIRTAÇ

Department of Computer Engineering

MSc. Thesis

Adviser: Asst. Prof. Mehmet Sıddık AKTAŞ

There exist high number of accessible documents on the Web and the number is growing exponentially. More and more people are turning to the Internet when searching for a particular information. However, the Web is tremendous resource and it must be used carefully and critically. Web Search engines are the primary tools for searching information. Web Search engines are faced with a number of difficult problems in enhancing the quality of their performance. It is a higly important problem to select best documents among the ones that match the query. We proposed a methodology to improve the accuracy of PageRank algorithm which is an example of popularity based ranking algorithms.

In this methodology, we used the implicit links that are available in Web pages. In many Web pages, other Web pages are only mentioned by name without hyperlinks. These names are accepted as implicit links which are linked to other Web pages. The implicit links can affect the calculation of Web pages' popularity as direct links. Based on this, we incorporate the implicit links into Web graph consisting of hyperlinks and create an enriched Web graph. In our methodology, we improve the PageRank algorithms which is an example of polularity based ranking algorithms by considering the implicit links. Hence, we perform the calculations of improved PageRank scores on the enriched Web graph and investigate whether the inclusion of implicit links provide

better ranking results than similarity based algorithms and plain PageRank algorithm. After user study, we show that improved PageRank algorithm gives better results in terms of precision and recall, compared with two other baseline algorithms. The results are promising and indicate that inclusion of implicit links improves the accuracy of pagerank computations.

Keywords: Popularity based ranking, implicit links, link-based ranking algorithms

BÖLÜM 1

GİRİŞ

Avrupa Nükleer Araştırma Merkezi olan CERN' de yazılım mühendisi olarak çalışan Tim Berners-Lee 1980’ li yıllarda World Wide Web' i keşfetmiştir [1]. O zamandan beri Web tartışmasız bir şekilde dünyanın tanıdığı en güçlü iletişim aracı haline gelmiştir. Dünya üzerinde insanların sadece %25' inin Web’ i kullanmasına rağmen, Web öğretme, öğrenme, satın alma, satma, bilgilendirme, bilgilendirilme, kabul etme, reddetme, paylaşma, katılma, karşılaşma ve kansere çözüm bulmaya kadar bütün problemleri çözme şeklimizi değiştirmiştir.

Web videolar, fotoğraflar ve etkileşimli içeriklerle dolu, birbirine bağlı Web sayfaları ile Web uygulamalarını içeren, gittikçe büyüyen bir dünyadır. Günümüz dünyasında her gün muazzam miktarda veri üretilmesi de bunu desteklemektedir. Birçok kullanıcının internet üzerinde sürekli olarak resim, video, doküman, e-posta ve benzeri veriyi paylaştığı düşünüldüğünde çok büyük rakamlar ortaya çıkacaktır. Google’ ın bir günde 20 Petabaytlık veriyi işlediği [2] ve her yıl bir önceki yıla oranla çok daha fazla veri ürettiği göz önünde bulundurulmalıdır. Çoklu kaynaklardan üretilen veri miktarı devasa boyutlara ulaşmıştır ve katlanarak büyümektedir.

Büyüyen veride doğru ve güvenilir bilgiye ulaşmak büyük önem arz etmektedir. Bilgi tek başına güçtür ve bu bilgiye en kısa sürede ulaşanlar teknoloji yarışında bir adım öne geçeceklerdir. Hemen her gün kullanıcılar internet üzerinde, bilgisayarlarında, e-posta kutularında, fotoğraf arşivlerinde arama yapmaktadırlar. Aradıkları bilgiye kolay ve hızlı şekilde ulaşmak için etkili araçlara duyulan gereksinim günümüzde arama motorları olarak adlandırılan teknolojileri ortaya çıkarmıştır.

1.1 Literatür Özeti

Şuanki arama motorları işlevsel olarak geleneksel bilgi erişim sistemlerine benzemektedirler. Dokümanlar içinde sözcükler için dizin yapısı oluşturmakta ve kullanıcı sorgularına sıralı doküman listesi ile cevap vermektedirler [3]. Arama motorları yapılan sorgulamalar için alakalı sonuç döndürmek isterler ve bunun içinde farklı algoritmalar kullanmaktadırlar [4], [5], [6], [7], [8] fakat bunlardan bazıları bunu gerçekten başarmıştır [9], [10].

Farklı bağlantı tabanlı yaklaşımlar, Web arama sonuçlarını iyileştirmek için önerilmiştir. HITS algoritması sayfa içeriğinin (authority değeri) ve diğer sayfalara olan bağlantıların değerini (hub değeri) tahmin etmek için sorgu zamanlı işleme gerçekleştirmektedir [7]. Bharat ve Henzinger bir sorgu konusuna özel dokümanların elde edilmesindeki doğruluk derecesini artırmak için HITS algoritmasını içerik analizi çalışmasıyla genişletmiştir [11]. Chakrabarti vd. [12] genel konular için kaynak listelerini derlemede HITS algoritmasından faydalanmışlardır. Page ve Brin tarafından geliştirilen PageRank algoritması, Web üzerindeki bütün sayfalar için öncelikli önem tahminlerini içeren sıralama vektörü hesaplamaktadır [13]. Bu vektör bir kere çevrimdışı olarak ve sorgudan bağımsız hesaplanır. Sorgu zamanında bu önem dereceleri sorgu sonuçlarını sıralamada kullanılır [5]. Bir sayfa için önceden PageRank değerinin hesaplanma maliyeti düşük olduğu için PageRank algoritması HITS algoritmasından verimlilik açısından daha avantajlıdır. Dahası, PageRank küçük bir alt küme yerine tüm Web çizgesi kullanılarak oluşturulur, dolayısıyla yerel bağlantı hatalarına karşı daha az duyarlıdır. Sonuç olarak bu çalışmada bağlantı tabanlı yaklaşımlardan PageRank seçilmiş ve PageRank tabanlı sıralama sonuçlarında doğruluk sonuçlarını iyileştirmek için araştırma yapılmıştır.

PageRank sonuçlarının kullanıcı ihtiyaçlarını karşılaması için birçok araştırmacı çeşitli stratejiler geliştirmiştir. Çoğu, PageRank ve belli bir konu arasındaki alakayı anlamaya çalışmıştır. Haveliwala [14] de, konuya karşı önyargı sorununu geliştirerek konuya duyarlı bir algoritma önermiştir. Bir sorgu işlemi olduğunda algoritma varolan çeşitli temel konular ile kullanıcının sorgusunun konusu arasındaki benzerliği ölçmekte ve kullanıcının sorgu konusu yerine varolan temel konular arasında en alakalı bulduğu

konuyu koymaktadır. Diğer bir çalışmada konuya karşı önyargı sorunu için ek çözümler önerilmiştir. Web sayfasının konusuna karar vermek için sayfada sık geçen kelimelerden yararlanılmıştır. Ek olarak, Wei Huang ve Bin Li [15] daki gibi daha eski sayfaların PageRank değerinin daha fazla olması gerektiğini savunmuşlardır. Decai Huang vd. [16] da PageRank' ın konu belirlerken yaşadığı sapma sorununu çözen yeni bir algoritma önermişlerdir. Bu algoritma iki Web sayfası arasındaki dosya vektör kosinüs ölçüsünü kullanarak bu Web sayfalarının konu korelasyon değerini hesaplamakta ve PageRank değerini bu benzerliğe göre hesaplamaktadır. Chia-Chen YenJih-Shih Hsu [17] çalışmasında birleşmiş PageRank algoritması önermiştir. Bu çalışmada sayfadan verilen bağlantıların benzerlikleri en sık terimler kümesi kullanılarak hesaplanmaktadır. Algoritma, daha sonra Pagerank ağırlıklarını hesaplanan benzerliklere göre güncellemektedir. Yukarıda bahsedilen yöntemler PageRank değerini artırmak için sayfaların konu benzerliğini ele alma noktasında birleşmektedirler. Fakat biz bu çalışmada konu benzerliğini ele almamaktayız. Bunun yerine sadece içeriği çekilen sayfalarda geçen ifadeleri, kısaltmaları, kısacası metin içeriğini kullanarak PageRank hesabını dolaylı olarak artırmaktayız.

Web sayfalarının dilinin ortaya çıkarılması, içeriğe göre özelleşmiş işlemlerin yapılabilmesini, farklı dildeki Web sayfalarının içeriğinin çekilmemesini sağlar. Bu çalışmada da sadece Türkçe sayfalar için PageRank geliştirmesi yapılacağı için dil tespit eklentisi kullanılmıştır. Dil tespitinde kullanılan en yaygın teknik, farklı dillerdeki eğitim metinlerinden elde edilmiş karakter n-gram modelleri ile dokümanların her dil için skorlanmasıdır. Dil tespiti için geliştirilmiş birçok kütüphane mevcuttur. Apache Tika projesi kapsamında geliştirilen dil tespit kütüphanesi örnek verilebilir [18]. Bir diğeri ise ücretli ürün olan "detectLanguage" kütüphanesidir [19]. TÜBİTAK BİLGEM Bilişim Teknolojileri Enstitüsü Arama Motoru Projesi kapsamında geliştirilen odaklanmış içerik çekme sisteminde, internet içeriklerinin dilinin bulunabilmesi için açık kaynak kodlu "lang-id" kütüphanesi [20] alınıp eklentilerle geliştirilmiştir. Temel olarak kullanılan bu kütüphanede istatiksel karakter n-gram dil modelleri kullanılmıştır [21]. Biz de çalışmamızda geliştirilen bu eklentiyi kullanmaktayız.

PageRank hesabını iyileştiren bir diğer yaklaşım aramaları kişiselleştiren yaklaşımdır. Bu yaklaşıma göre arama sonuçları kişinin geçmiş tecrübesine ve tercihlerine göre

hesaplanır. Carol L. Barry [9]' deki çalışmasında köprü (hyperlink) özellikleri kullanarak Web aramalarının kişiselleştirilmesini incelemiştir. Aktaş vd. [22], [23] tarafından, köprüler ile kullanıcı profillerinin eşleşmesi ölçüsünde bağlantılar ağırlıklandırılarak PageRank vektörleri kişiselleştirilmiştir. A. Sieg vd. [24] çalışmalarında Pagerank sonuçlarını iyileştirmede kişiselleşmiş ontolojileri kullanan yeni bir yöntem önermişlerdir. Bu çalışmada kullanıcıların mevcut bir alan ontolojisindeki kavramlara olan ilgisi skorlanarak ontolojik kullanıcı profilleri inşa edilmiştir. Kişiselleştirilmiş ontolojiler kullanıcının arama geçmişinden faydalanılarak uyarlamalı olarak inşa edilmektedir. Bir başka yaklaşım Google Kişiselleştirilmiş Arama uygulamasıdır. Bu uygulama kullanıcının arama geçmişine göre sonuçları yeniden sıralayarak o kullanıcıya has bir profil saklar ve bunu günceller [25]. Bizim çalışmamızda ise aramada kişiselleştirme uygulanmamaktadır. Biz, dolaylı bağlantı faktörü yaklaşımıyla PageRank hesabını değiştirerek iyileştirmekteyiz.

1.2 Tezin Amacı

Bu çalışmada PageRank algoritmasının geliştirilmesi hedeflenmektedir. PageRank algoritması Web' i bağlantılardan ve belgelerden oluşan bir çizge olarak modellemekte ve her sayfa için bir önem skoru hesaplamaktadır. Çok ve kaliteli bağlantı alan sayfaların önem skoru büyük olurken, az ve düşük kalitede bağlantı alan sayfaların önem skoru ise küçük olmaktadır. Hesaplanan bu önem skoru, sayfaların popülaritesini veya önem derecesini göstermektedir. Popülaritesi yüksek olan sayfaların kullanıcının daha fazla ilgisini çekeceği düşünülmekte ve bir sorgu ile eşleşen çok fazla sayıdaki belge arasından, önem skoru yüksek olanlar seçilmektedir. Dolayısıyla PageRank algoritmasının sayfaların popülaritesini doğru ve objektif bir şekilde hesaplaması çok önemlidir.

PageRank algoritması sayfalar için önem skoru hesaplarken sadece o sayfaya verilen açık http bağlantılarını kullanmaktadır. Fakat birçok durumda bir sayfaya metin içinden ismiyle atıf yapılmakta ama açık bağlantı verilmemektedir. Bu tür bağlantısız yapılan atıflar, hedef sayfanın popülaritesini ve bilinilirliğini kullanıcı nezdinde artırmaktadır. Dolayısıyla bu atıfların da hedef sayfanın PageRank skorunu yükseltmesini beklememiz gerekir. Bu tezde Web sayfaları içerisindeki metinlerden diğer sayfalara isimle yapılan

atıflar tespit edilecek ve bu atıflar hedef sayfalara yapılan birer dolaylı bağlantı olarak değerlendirilecektir. Açık http bağlantılarından oluşturulan Web çizgesi, dolaylı bağlantıların eklenmesiyle zenginleştirilecektir.

Her sayfa için PageRank skoru zenginleştirilmiş Web çizgesi üzerinden hesaplanacaktır. Bu çizge üzerinden yapılacak önem skoru hesaplaması, normal Web çizgesi üzerinden yapılan hesaplamalara göre daha doğru ve objektif olacaktır. Dolayısıyla kullanıcıya döndürülecek 10 belgenin seçimi yeni önem skorlarına göre daha iyi yapılacak ve kullanıcının ihtiyacını karşılayacak sayfalar daha büyük bir isabet oranıyla bulunacaktır. Bu hesaplama yöntemi özellikle Web sitesi ana sayfalarının önem skorlarının daha doğru ve objektif bir şekilde hesaplanmasında etkili olacaktır. Çünkü metin içinden isimle Web sitesi ana sayfalarına yapılan atıflar (dolaylı bağlantılar) http bağlantılarından çok daha fazladır. Tezin bir diğer katkısı ise farklı algoritmaların değerlendirilmesinde tam geri bildirim sağlayan bir yöntem geliştirilmesidir.

1.3 Hipotez

Yapacağımız bu çalışmada PageRank algoritmasını iyileştirmeyi hedeflemekteyiz. Bu algoritmaların iyileştirilmesi, açık http bağlantılarından oluşturulan Web çizgesinin, atıflardan oluşturulacak dolaylı bağlantılarla zenginleştirilmesi ile yapılacaktır.

Bu çalışmada sıklıkla bahsedilecek olan üç kavram bulunmaktadır. İlki gerçek bağlantıdır. Gerçek bağlantı; Web sayfalarında bulunan açık http bağlantılarıdır. Bir diğeri ise gerçek bağlantı metnidir. Bu da açık http bağlantılarının ifade edildiği metinlerdir. Örneğin bir Web sayfasında YTÜ bağlantı olarak verildiyse, YTÜ ifadesi açık http bağlantısının metnidir yani gerçek bağlantı metnidir. Üçüncü kavram ise Web sayfalarının başka sayfalar için bağlantı vermeden yaptıkları atıfları temsilen kullanılacak olan dolaylı bağlantılardır.

Birçok Web sayfasında diğer sayfalara metin içinden isimle atıflar yapılmakta, fakat o sayfalara açık http bağlantısı verilmemektedir. Bu tezde bu tür atıflar tespit edilecek ve hedef sayfalara dolaylı (gizli) bağlantılar olarak değerlendirilecektir. Atıflardan oluşturulacak dolaylı (gizli) bağlantılar, açık http bağlantılarından oluşturulan Web çizgesine dahil edilecektir. Web sayfaları için PageRank önem skoru hesaplama,

zenginleştirilmiş Web çizgesi üzerinde yapılacaktır. İyileştirilmiş PageRank algoritması birçok durumda daha doğru ve kaliteli sonuçlar verecektir. Özellikle Web sitesi ana sayfalarının önem skorları daha doğru ve objektif bir şekilde hesaplanacaktır. Bu da elektronik ticaret firmaları için çok önemlidir. Bu çalışma Nutch [26] açık kaynak kodlu arama motoru kullanılarak gerçekleştirilecektir. Yapmış olduğumuz literatür taramasında, atıflardan oluşturulan gizli bağlantılarla Web çizgesinin zenginleştirilmesi ve bunun çizge tabanlı Web algoritmalarında kullanılması ile ilgili yapılmış bir çalışma bulunamamıştır.

BÖLÜM 2

GENEL BİLGİLER

2.1 Genel Bilgiler

İnternet büyüdükçe, buna paralel olarak, arama motorları da artmaya başlamıştır ve daha yeni teknolojilerle daha başarılı sonuçlar elde etmeye uğraşılmıştır. Bu sebeple internetin doğuşundan bu yana birçok arama motoru geliştirilmiştir ve halen arama motorları konusunda yenilikler ve geliştirmeler hızla devam etmektedir [27].

İlk arama motoru 1990 yılında Alan Emtage tarafından Archie adıyla kurulmuştur [28]. Archie, bir üniversitedeki bazı birimlerde bulunan bilgisayarlar arası FTP (File Transfer Protocol) arşivlerini taramakta ve bu arşivlerdeki dosyalardan bir arama listesi oluşturmaktaydı. Arama yapan kişiler, dosya ismini tam olarak yazdıklarında FTP listelerinde bulunan sonuçlar görüntülenmekteydi.

1991 yılında Gopher protokolünün [29] geliştirilmesi, metin dosya içeriklerinin de indekslenmesini mümkün kılmıştır. Daha sonra ise bu protokolü kullanan Veronica ve Jughead arama motorları geliştirilmiştir. Veronica ve Jughead, Archie gibi dosya adlarında Gopher' da indeksli bir şekilde tutulan dosya isimlerinde arama yapabilmekteydi. Veronica (Very Easy Rodent-Oriented Net-wide Indeks to Computerized Archives) Gopher listelerindeki dosya isimlerini, Jughead (Jonzy's Universal Gopher Hierarchy Excavation and Display) ise Gopher sunucularındaki menü bilgilerini sunan bir araçtı.

1993 yazında halen tam anlamıyla bir arama motoru bulunmamaktaydı ancak 2 Eylül 1993 tarihinde Oscar Neirstrasz, periyodik olarak sayfaları kopyalayan ve standart bir format haline getiren bir perl kodu geliştirdi ve ilk arama motoru yayınlandı [30].

Haziran 1993' de Matthew Gray perl tabanlı world wide Web gezginini, bir arama robotu bileşenlerine sahip ilk arama motoru, üretti ve Wandex adı verilen bir indeks sistemi oluşturup kullandı [31]. Bu arama robotu internette dolaşarak, Wide Web Wanderer' ın veritabanına URL' leri ekliyordu.

Kasım 1993' de Web' in ikinci arama motoru olan AliWeb geliştirilmiş fakat bu arama motorunda herhangi bir arama robotu kullanılmamıştır. Web sayfalarının AliWeb' de aranabilir olması için sayfalar hakkında kısa bir açıklama ve anahtar kelimeler içeren bir index dosyası hazırlanması ve bunun da AliWeb' e bildirilmesi gerekmekteydi [32].

Aralık 1993' de, bir arama motorunun kullandığı üç temel özelliği de (tarama, indeksleme ve arama) içeren JumpStation arama motoru geliştirilmiştir. Jumpstation' da Web sayfalarını bulmak ve indeks oluşturmak amacıyla bir Web robotu ve sorguları çalıştırmak için bir Web form arayüzü kullanılmıştır [33].

Bu tarihe kadar yapılan tüm arama motorları kısıtlı bir içeriği indeksleyebilmekteydi. Tüm metin indeksleme işlemini yapan ilk arama motoru 1994' de geliştirilen WebCrawler' dır [34]. Bu arama motoru kullanıcıya Web sayfasındaki tüm kelimeler için arama imkânı sağlamaktaydı. O zamandan sonra bu özellik bütün büyük arama motorlarında kullanılmıştır.

Bu tarihten sonra da Altavista, Infoseek, Magellan, Excite, Yahoo ve Lycos gibi birçok arama motoru geliştirilmiştir. Bunlardan günümüzde en bilinenlerinden biri Yahoo, Stanford Üniversitesi öğrencileri Jerry Yang ve David Filo tarafından kurulmuştur [35]. O dönemlerde oldukça popüler olan Yahoo, büyük arama motorlarından biri olarak hizmet vermeye devam etmektedir.

1997' de ise Yandex arama motoru resmi olarak kullanılmaya başlandı. Rus dili morfolojisine dayanan Yandex arama motorunun 1997 sürümü kelimeler arasındaki uzaklığı dikkate almakta ve bir Web belgesinin arama sorgusuyla ilgi düzeyini belirlemek için karmaşık bir algoritma kullanmaktaydı [36].

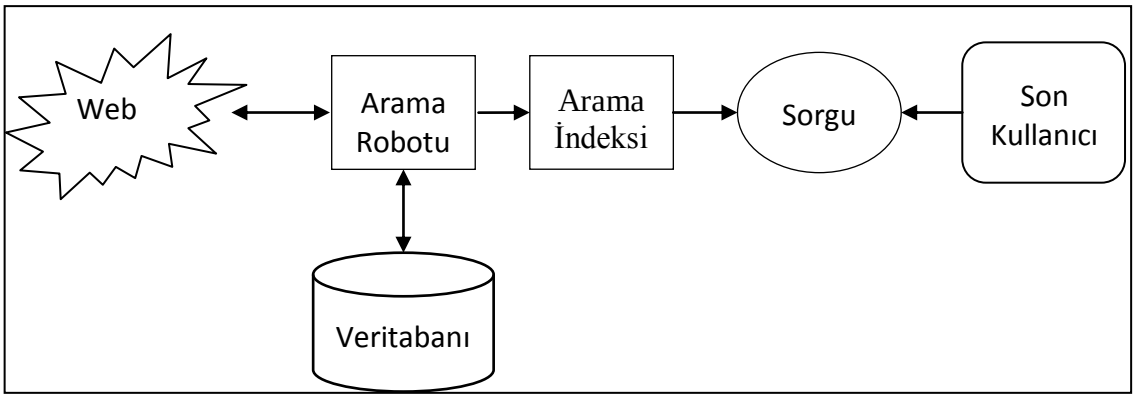
1999 yılında ise Larry Page ve Sergey Brin tarafından Google kurulmuştur. Google arama motoru internette her sayfayı değerlendiren PageRank yöntemini kullanarak birçok arama için daha iyi sonuçlar elde etmiştir [37]. Google arama motoru olmanın yanında mail, video gibi hizmetleri de vererek büyümesine devam etmiştir.

Google Çin hükümeti tarafından yasaklanınca, Baidu arama motoru 2000 yılında Robin Li ve Eric Xu tarafından kurulmuştur [38].

Bing arama motoru 2009 yılında Microsoft tarafından geliştirilmiş ve "Windows Live Search" yerine kullanılmaya başlanmıştır [39]. 40 farklı dilde hizmet vermektedir. Bing, arama sorgusu için verilen kelimeyi anahtar kelime olarak görmek dışında aramayı yönlendiren bir ipucu olarak görerek arama özelliklerine anlamsallığı da katmıştır.

İnternet üzerinde Türkçe içerik arttıkça, Google, Yandex ve Bing gibi arama motorları Türkçe için arama yeteneği edinmiştir. Bunun yanında Türkçe arama motorlarının sayısında da artış gözlenmiştir. Türkiye' de internet kullanımına 1993 yılında başlanmış ama Türkçe olarak arama yapılabilen ilk arama motoru 1994 yılında hizmet vermeye başlamıştır. Şu anda Türkçe aramanın yapılabildiği birçok arama motoru vardır. Arabul, Arama, Netbul, bu arama motorlarından bazılarıdır.

Arama motorları arayüzleri üzerinden sorgu alıp sonuçları kullanıcıya göstermektedirler. Bu işlem arka planda Şekil 2.1.1' deki gibi gerçekleşmektedir. Arama robotları otomatik olarak sunucuları tarayıp internet sayfalarının bilgilerini getirmekte, arama motoruna gelen veri veritabanında tutulmakta ve son kullanıcıların tutulan veri üzerinden arama yapabilmesi için veri indekslenmektedir.



Şekil 2.1.1 Arama motoru çalışma şekli

Bu teknolojiye sahip arama motoru üç kısımdan oluşmaktadır. Bunlar ”Arama Robotu”, ”Arama İndeksi” ve ”Arayüz” dür.

Arama robotu: Arama motorlarının bilgiye kolay ve hızlı ulaşmak için duydukları ihtiyaç doğrultusunda sayfaların taranması, bu sayfaların analiz edilmesi ve elde edilen bütün bilginin (sayfa başlığı, sayfa içeriği, sayfanın dili vb.) depolanması için ilk olarak İnternet ortamındaki sayfaların içeriğinin çekilmesi gerekmektedir. Bu işlemi yaparken de Apache Nutch gibi arama robotu (içerik çekme aracı) kullanılmaktadır. Bu araç belirli bir düzeni takip ederek istenilen tüm sayfaları ziyaret etmekte, bu esnada da ziyaret edilen sayfaların kopyasını almakta yani içeriğini çekmektedir. Bundaki amaç da kullanıcı herhangi bir sorgu yaptığında arama motorunun hızlı bir şekilde aranan bilgiyi kullanıcıya sunabilmesi için erişilebilir bilginin önceden depolanması ve arşivlenmesidir. İçerik çekme araçları (arama robotları), internet üzerindeki yazı, resim, video, müzik vb. her türlü bilginin kullanıcılar tarafından erişilebilir olmasında büyük rol almaktadır. Temelde internet sayfalarının içeriğinin çekilmesi için kullanılan araçlar birden fazla amaca dönük olarak da hizmet verebilmektedir. Örneğin, belirli bir konuda (örn. çocuk kitapları ve sağlık yayınları) internette yer alan kaynakların indeksinin oluşturulması, internetteki içeriğin arşivlenmesi ve böylelikle bu içeriklerin zaman içindeki değişimlerinin analizinin yapılması, içeriği çekilecek kaynakların sınıflandırılabilmesi ve isteğe bağlı sınıfların içeriğinin çekilmesi gibi işlemlerde de içerik çekme araçları kullanılmaktadır.

İçerik çekme işlemi bir çekirdek internet sayfası kümesinin belirlenmesi ile başlar. Bu küme araç tarafından ilk etapta ziyaret edilmesi istenen çekirdek sayfalardan oluşmalıdır. Bu başlangıç kümesinin belirlenmesi ve ziyaret kuyruğuna yerleştirilmesini takiben aşağıdaki adımlar kesintisiz tekrarlanarak içerik çekme işlemi gerçekleştirilir:

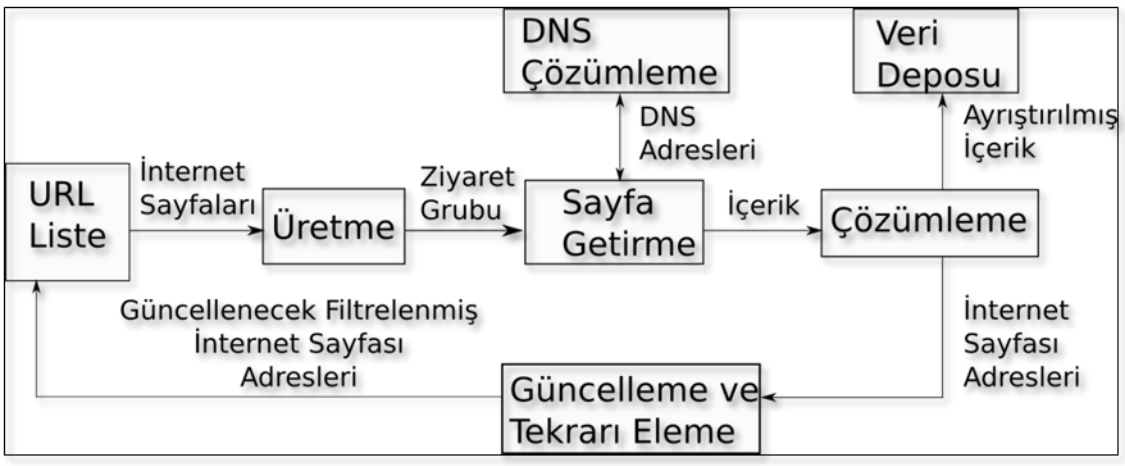
- Ziyaret kuyruğunda sıradaki internet sayfası ziyaret edilir.
- Ziyaret edilen sayfanın içeriği çekilir ve bu sayfanın bağlantılı olduğu başka internet sayfaları belirlenip ziyaret kuyruğuna eklenir.
- Ziyaret edilen sayfanın tekrar ziyaret edilebilmesi için bu sayfa yeniden ziyaret kuyruğuna yerleştirilir.

İçerik çekme işlemi birden fazla iş parçacığı (thread) tarafından yürütülebilir. Bu iş parçacıkları paralel olarak birden fazla internet sayfasının ziyaret edilmesini ve içeriğinin çekilmesini gerçekleştirebilir.

İçerik çekme işlemi genel hatlarıyla Şekil 2.1.2‘ de gösterilen mimari ile gerçekleştirilmektedir.

- İnternet sayfalarının yer aldığı URL listesi (URL frontier) sonraki içerik çekme adımlarında ziyaret edilecek sayfaları içermektedir.
- İnternet Ad Sunucusu (Domain Name Server - DNS) çözümleme modülü URL listesinde yer alan internet sayfalarını barındıran sunucuları belirlemektedir.
- Sayfa getirme modülü içeriğinin çekilmesine karar verilen sayfaların içeriğini ilgili sunuculardan çekmektedir.
- Çözümleme modülü içeriği çekilen internet sayfalarını çözümlemekte ve içerik ile bağlantıları tespit edip ayrıştırmaktadır.

Tekrarı eleme modülü ziyaret edilen sayfada tespit edilen bağlantıların URL listesinde hali hazırda olup olmadığını kontrol etmekte veya yakın zamanda içeriğinin çekilip çekilmediğini belirlemektedir.



Şekil 2.1.2 İçerik çekme süreci

Tek bir iş parçacığı tarafından, Şekil 2.1.2‘ de verilen mimari kullanılarak bir internet sayfasının içeriğinin çekilmesi sırasında yürütülen işlemler şu şekilde özetlenebilir. İş parçacığı, URL listesinden ziyaret sırası gelen bir internet sayfasının adresini alır ve DNS Çözümleme Modülü' nün yardımıyla sayfanın bulunduğu sunucuyu ve sayfanın IP

adresini belirler. İlgili sunucudan, genellikle HTTP protokolünü kullanan Sayfa Getirme Modülü ile bu internet sayfasının içeriğini çeker. Geçici bir alanda tutulan bu içerik, Çözümleme Modülü sayesinde yazılı içerik ve bağlantılar olarak ayrıştırılır. Ayrıştırılan içerik sonrasında başka uygulamalar tarafından kullanılabilmek için indekslenip depolanabilir. Ayrıştırılan bağlantıları ise Tekrarı Eleme Modülü bazı işlemlere tabi tutar ve URL listesine eklenmesine karar verilenleri bu listeye ekler. Tek bir iş parçacığı tarafından yürütülen içerik çekme işlemi URL listesindeki internet sayfaları için bu adımları kesintisiz olarak devam ettirir. Bununla beraber aynı iş parçacığı kontrol amaçlı bazı işlemleri de (örn. istatistikleri oluşturma, gerekli görülen hallerde işlemi sonlandırma ve işlem denetim noktaları belirleme) periyodik olarak devam ettirir. Birçok arama motoru içerik çekme işlemini birden fazla iş parçacığı tarafından dağıtık olarak gerçekleştirerek dengeli ve etkili bir işlem sürdürme politikası izler [40]. Bu çalışmada da birden fazla iş parçacığı ile dağıtık ortamda içerik çekme işlemi gerçekleştirilmiştir. Bu işlem için de Apache Nutch aracı kullanılmıştır.

Arama İndeksi: Arama robotlarından gelen bilgilerin düzenli bir şekilde kaydedilmesi, arama motoru indeksinin yöntemine göre; kullanıcıdan gelen arama sorgusuna cevap verilmesini sağlamaktadır.

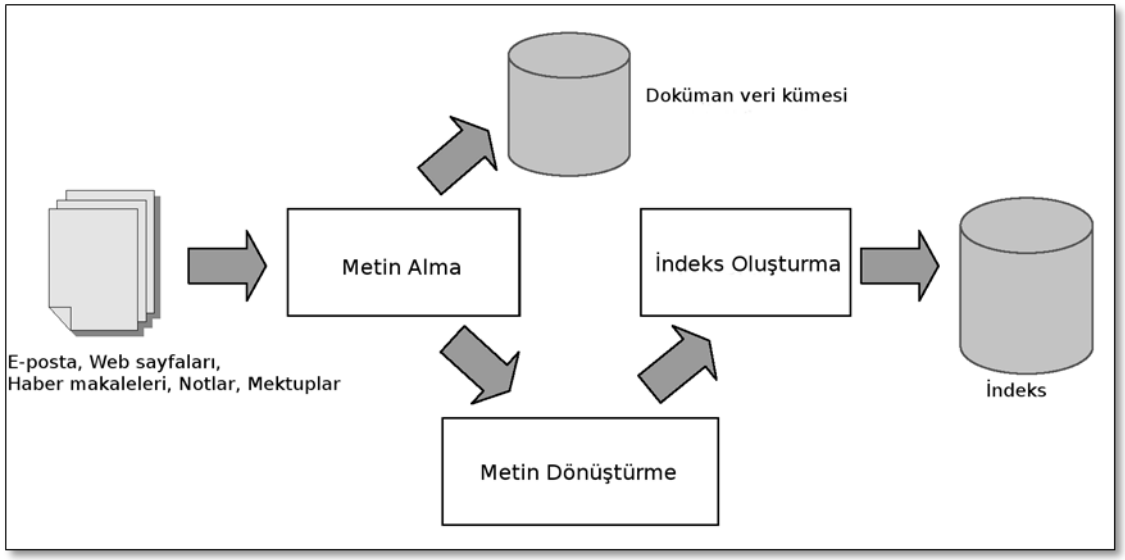
Arama, bilgiye erişim sistemlerinin sunduğu temel işlev olmakla birlikte, bu alan, metin gruplandırma (text clustering), metin sınıflandırma (text categorization), özetleme (summarization), bilgi çıkarımı (information extraction) ve çoklu ortam bilgisine erişim (multimedia information retrieval) gibi pek çok araştırma alanıyla da ilgilidir [41].

Bir bilgiye erişim sisteminin temel hedefi, kullanıcıya sorgusuyla ilgili mümkün olduğunca çok doküman sunulmasının yanı sıra, ilgili olmayan dokümanların da mümkün olduğunca az sunulmasıdır [42]. Bir arama motoru, bilgiye erişim tekniklerinin büyük ölçekli doküman kümelerine pratik bir uygulamasıdır. Bir arama motoru mimarisi tasarlanırken, kullanıcı sorgularına karşılık en ilgili dokümanların sunulmasının yanında, sorguların mümkün olduğunca hızlı bir şekilde işlenmesi de amaçlanır.

Bir arama motorunun sonuçlarının kalitesinin değerlendirilmesinde, sıklıkla iki temel ölçüt kullanılır: Tutturma (precision) ve bulma (recall) [43].

Tutturma, arama sonuçlarında sunulan dokümanlardan kullanıcı sorgusuyla ilgili olanların sayısının sunulan toplam doküman sayısına oranı $(|\dot{I} \cap S|/|S|)$ olarak tanımlanır. Bulma ise, kullanıcıya sunulan arama sonuçlarının, sistemde kullanıcı sorgusuyla ilgili belgeleri hangi oranda kapsadığına $(|\dot{I} \cap S|/|\dot{I}|)$ işaret eder. Buradaki 'İ' değeri kullanıcı sorgusu ile ilgili olan belge sayısını, 'S' ise toplam doküman sayısını göstermektedir.

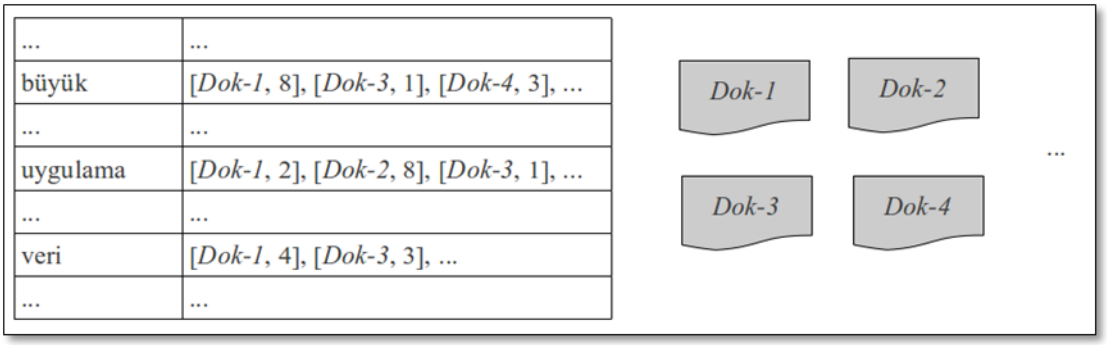
Bir arama motoru mimarisi, temelde iki önemli işlevi yerine getirir: İndeksleme işlemi ve sorgulama işlemi. İndeksleme, arama işleminin gerçekleştirilmesi için gerekli yapıların oluşturulmasını sağlar. İndeksleme aşamasının temel bileşenleri, metin alma, metin dönüştürme ve indeks oluşturmadır (Şekil 2.1.3) [43]. Metin alma bileşeni, dokümanların internet, kurumsal bir intranet, masaüstü ya da diğer bilgi kaynaklarından alınmasını içerebilir. Metin dönüştürme, dokümanlardan indeks terimlerinin ve özelliklerinin elde edilmesini sağlar. İndeks oluşturma bileşeni ise, dönüştürülen metni kullanarak hızlı aramayı sağlayacak indeks veya veri yapılarını oluşturur.



Şekil 2.1.3 Arama motorlarında indeksleme

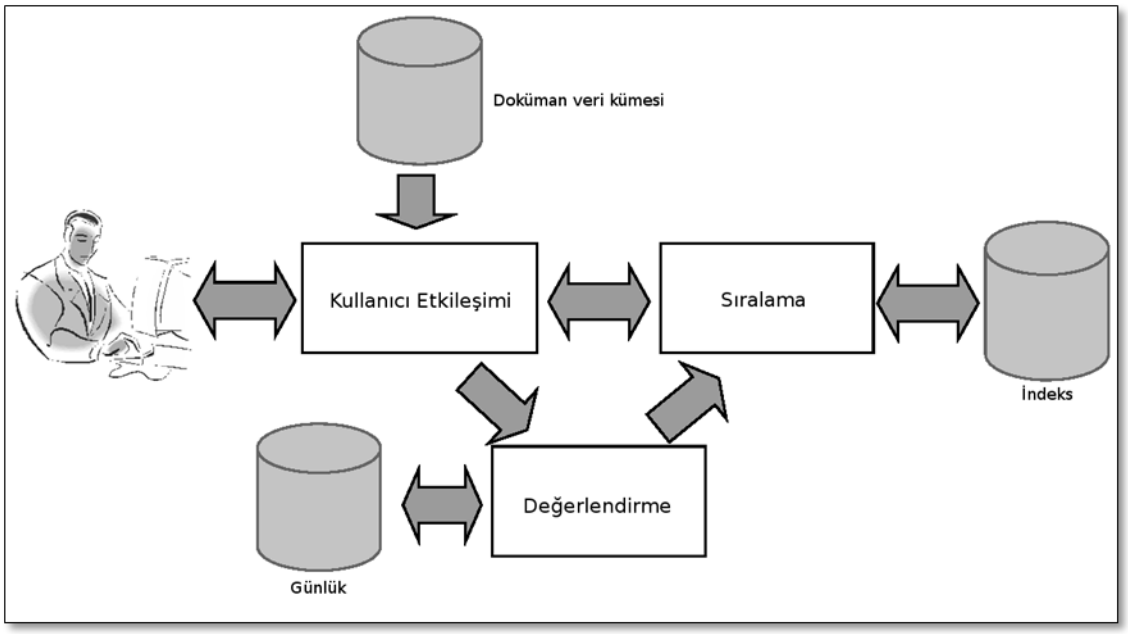
İndeks oluşturmada, en çok kullanılan yöntemlerden biri, ters indeks (inverted index) tutma yöntemidir. Bu yöntemde, kitaplardaki dizinlere benzer biçimde, her bir indeks terimi için, terimin geçtiği dokümanlar, konumlar gibi ayrıntılı özellikler tutulabilir [43]. Şekil 2.1.4' de terimlerin geçtiği dokümanların (örneğin Dok-1, Dok-2, Dok-3 gibi) ve

terim sıklıklarının tutulduğu basit bir ters indeks örneği verilmiştir. Örnekte "veri" kelimesi, Dok-1' de dört kere, Dok-3' te ise üç kere geçmektedir.



Şekil 2.1.4 Ters indeks örneği

Sorgulama işleminde, kullanıcı sorgusundaki terimler ve indeks yapıları kullanılarak, ilgili dokümanların sıralı bir listesinin üretilmesi amaçlanır. Bu aşamanın temel bileşenleri, kullanıcı etkileşimi, sıralama (ranking) ve değerlendirmedir (Şekil 2.1.5) [43]. Kullanıcı etkileşimi bileşeni, arama yapan kullanıcı ile arama motoru arasındaki arayüzü sağlar. Kullanıcı sorgusunun alınması ve indeks terimlerine dönüştürülmesinin yanı sıra, arama sonuçlarının kullanıcıya uygun bir şekilde (örneğin 2-3 satırlık kısa özetlerle birlikte) sunulmasından sorumludur. Sıralama bileşeni, dokümanların kullanıcı sorgusu ve kullanılan bilgiye erişim modeline göre (örneğin, Boole modeli, vektör uzayı modeli, olasılıksal modeller gibi [42]) puanlanarak en ilgiliden başlayarak sıralanmasını amaçlar. Değerlendirme bileşeni ise, arama motorunun başarımını ve performansını ölçmek için kullanıcılar ve sistemle ilgili günlük (log) tutar.



Şekil 2.1.5 Arama motorlarında sorgulama işlemi

Arama indeksi sürecinde de Apache ailesinden Solr aracı kullanılmıştır.

Arayüz: Arama motorları kullanıcılardan gelen sorguların, arama motoru indeksine iletilebilmesi ve gelen cevapların kullanıcılara gösterilebilmesi için arama motoru arayüzünü kullanırlar. Gelen cevapların sıralaması, içerik çekme ve indeksleme aşamalarında yapılan çalışmaların bir sonucudur.

Arama motoru arayüzü, kullanıcının verdiği anahtar kelimeleri indekse gönderirken özel komutları da kabul edebilir. Bunlar ve/veya, dâhil etme gibi mantıksal aramayı güçlendiren komutlar olabilmektedir.

2.1.1 Sıralama Algoritmalarının Ortaya Çıkma Nedenleri

- **Bilginin düzenlenmesi:** Arama motorları insanlar arasında varolan bilginin paylaşılması ve herkesin erişibilmesi için geliştirilmiştir. Daha sonra geliştirilen arama motorları ile çevrimiçi bilgiye erişim daha kolay hale gelmiştir. Bugün ise arama motorlarının amacı internette dağınık olarak bulunan bilgiyi organize etmek ve bulunmasını sağlamak olmuştur.
- **Arama yeteneği az olan kullanıcı:** Webdeki bilgi ve arama yeteneği az olan yeni kullanıcıların sayısı artmaktadır. Bu da arama motorlarının kişilerin bireysel ihtiyaçlarını karşılamasını gittikçe daha da zor hale getirmektedir.

- **Aramanın daha geniş içerikte yapılmak istenmesi:** Arama motorları ilk olarak metinde arama olarak geliştirilmiştir fakat internet verisinin artması ve çeşitlenmesi sonucu, resim, video gibi farklı içeriklerde arama yapabilme ihtiyacı ortaya çıkmıştır.
- **Millileşme:** Çin devletinde Google' ın yasaklanması sonucu kendi arama motorları için algoritma geliştirme ihtiyacı oluşmuştur. Başka ülkelerde de bu teknolojinin millileştirme fikri ile arama algoritmaları geliştirme çalışmaları yapılmaktadır.
- **Rekabet:** Birbiriyle rekabet halinde olan arama motorlarının kullanıcıya daha iyi sonuçlar verebilmek için geliştirdiği arama algoritmaları için günümüzde hala iyileştirme çalışmaları yapılmaktadır.

2.1.2 Arama Motorlarının Karşılaştığı Zorluklar

- **Veri büyüklüğü:** Arama motorları ne kadar popüler olsa da birçok eksiklikleri bulunmaktadır. Kullanıcı herhangi bir arama yaptığında dönen sonuç listesi çok fazla olabilir ve ilk sırada çıkan sayfalar kullanıcının ihtiyaçlarını karşılayamayabilir. Tipik bir kullanıcı arama sonuçlarının sadece ufak bir kısmı ile ilgilenir bu yüzden yararlı küçük bilgi kümesine ulaşabilmek için gereksiz bilgi yığınıyla uğraşmak istemeyecektir.
- **Verinin gürültülü olması:** Kullanıcının aradığı sorguya istediği cevapları alabilmesi için verinin gruplanması ve önceliklendirilmesi gerekmektedir. Birçok farklı çeşitte (resim, video vs.), istenmeyen içerikte (sakıncalı, spam, kopya vb.) veri olması bu işlemin yapılabilmesini zorlaştırmaktadır.
- **Zaman kısıtı:** Kullanıcıların aramalarının aradıkları ifadeyle ilgili olması artık yeterli değildir. Bunun yanında sonuçların belli bir zaman dilimi içinde olması gerekmektedir. Günümüzde bu değer 1 saniyeden bile azdır. Bunu sağlayabilmek için geliştirilen algoritmaların optimize edilmesi gerekmektedir.
- **Depolama kısıtı:** Arama algoritmalarının büyük veride çalışması, sonuçlarının hızlı alınmak istenmesi geniş bir disk alanı gereksinimi oluşturmaktadır fakat Dünya' daki bütün veri düşünülünce bunu karşılamanın pek mümkün olmadığı

anlaşılmaktadır. Bu sebepten algoritmanın az disk alanında gerekli süreci işletmesi gerekmektedir.

2.1.3 Sıralama Algoritmalarındaki Eksiklikler

Web' de halen milyarlarca belge bulunmakta ve bu sayı gün geçtikçe artmaktadır. Arama motorlarına gönderilen birçok sorgu ile milyonlarca belge eşleşmektedir. Bu belgelerden en yetkin ve ilgili olanların seçilmesi en temel problemlerden birisidir. Bu konuda kullanılan en önemli algoritmalardan birisi PageRank algoritmasıdır. PageRank algoritması Web' i bağlantılardan ve belgelerden oluşan bir çizge olarak modellemekte ve her sayfa için bir önem skoru hesaplamaktadır. Çok ve kaliteli bağlantı alan sayfaların önem skoru büyük olurken, az ve düşük kalitede bağlantı alan sayfaların önem skoru ise küçük olmaktadır. Hesaplanan bu önem skoru, bir sorgu ile eşleşen çok fazla sayıdaki belgeler arasından en kaliteli sayfaların seçilip kullanıcıya gösterilmesinde kullanılmaktadır.

Arama motorları için bir diğer temel problem ise spam Web sayfaları ile mücadeledir. Arama motoru sonuç sayfalarında haksız olarak yüksek sıralarda çıkmak isteyen Web siteleri ile arama motorları arasında süregiden bir mücadele vardır. Bu mücadele her gün yeni teknik ve yöntemlerle devam etmektedir.

Bazı Web sayfaları yazarları ise sayfalarının önem skorunu yükseltmek için sayfalarında kendilerine link vererek bunu sağlamaya çalışmaktadır. Arama motorları arama sonuçlarında bu tarz sayfaların önem skorunun haksız olarak yüksek olmasını engellemek için böyle sayfaları tespit etmeye çalışmaktadır. Böylece bu Web sayfalarının artık içeriği çekilmeyecektir.

2.1.4 Motivasyon

Bölüm 2.1.4' de bahsedilen olumsuzlukların aşılabilmesi için sıralama algoritmalarının iyileştirilmesi büyük önem taşımaktadır.

Sıralama algoritmaları genel anlamda yüzbinlerce, milyonlarca ve milyarlarca veriye en kısa yoldan, en hızlı bir biçimde, en az bellek kullanımı ve en az kaynak harcayarak ulaşmamızı sağlayan algoritmaların tümüdür. Sıralama algoritmaları bilgisayar biliminin

en temel işlemlerinden biridir. Bu algoritmalar elde var olan veriyi en uygun biçimde kullanarak en hızlı yoldan sonuca ulaşmamızı sağlarlar.

Bölüm 2.1.2' de açıklanan nedenlerden, sıralama algoritmalarının geliştirilme ihtiyacı artmaktadır. Arama motorları farklı sıralama algoritmaları kullanmaktadır. Ancak bu algoritmalar artan ve değişen ihtiyaca göre yeterli olmamaktadır.

Yukarıda bahsedilen nedenlerden, arama sonuçlarının kalitesini artırmak için sıralama algoritmalarının iyileştirilmesine ihtiyaç duyulmaktadır.

2.2 Tez Kapsamında Kullanılan Teknolojiler

2.2.1 Apache Nutch

Apache Nutch son derece esnek ve ölçeklenebilir, açık kaynak kodlu bir Web tarayıcısı yazılımı projesidir. Apache Nutch, ilk olarak 2003 yılında Doug Cutting ve Mike Cafarella tarafından geliştirilmiş, 2005 yılında Apache Lucene' in alt projesi olarak değerlendirilmiş, 2010 yılında ise Apache' nin bağımsız projelerinden biri olarak devam etmesine karar verilmiştir [26]. Daha sonra ise 1.x ve 2.x sürümleri olarak veri depolama konusunda farklı bir yol takip ettikleri için ayrı olarak geliştirilmeye devam etmiştir. Bu çalışma ise veritabanı ile gerçekleştirildiği için 2.x sürümü kullanılmıştır.

Apache Nutch, tamamen Java programlama dili ile kodlanmıştır. Geliştiricilere veri alma, sorgulama, kümeleme vb. işlemler için eklenti geliştirmelerine izin veren modüler bir mimariye sahiptir. Bunun yanında dağıtık olarak çalışabilmekte, sahip olduğu işlevler sayesinde arama motoru geliştirilebilmesine olanak sağlamaktadır. Bu çalışmada da Apache Nutch aracına eklenti geliştirilip arama sonuçlarının iyileştirilmesi konusu ele alınmıştır.

2.2.2 Apache Lucene ve Solr

Apache Solr, Apache Lucene projesi temel alınarak geliştirilmiş ileri düzey bir arama platformudur. Lucene, bir arama kütüphanesi sağlar ve pratikte kullanılabilmesi için bir program içerisinden ilgili metodların çağırılması gerekir. Profesyonel arama çözümü içeren Dünya üzerindeki binlerce kurum ya da şirket, arama altyapısını Lucene' e

dayandırmaktadır. Lucene kütüphanesi, Hadoop, Nutch gibi önemli projelere imza atan Doug Cutting [44] tarafından yazılmıştır. Orjinali Java ile yazılan Lucene daha sonra C#, C++, Perl ve Python gibi birçok programlama diline de çevrilmiştir [45]. Lucene, önceleri bir SourceForge projesi olarak başlatılmış, Eylül 2001' de Apache' nin Jakarta ailesine katılmıştır. Şubat 2005 itibari ile de Apache Vakfının üst seviye projelerinden biri olarak sunulmaya başlanmıştır. Bu tarihten günümüze kadar arama sistemlerinin büyük çoğunluğunun altyapısında Lucene kullanılmaktadır.

Lucene' ın indeksleme ve arama olmak üzere temel iki işlemi mevcuttur. Verilen doküman ya da girdi içeriği indekslenerek aranılabilir hale getirilmektedir. Daha sonra istenilen kriterlere göre bu indeks üzerinde arama yapılabilir. Solr ise, Lucene kütüphanesine dayalı ve doğrudan kullanıma hazır bir platformdur. Lucene arama kütüphanesi, dünyada önde gelen açık kaynak projeleri arasındadır ve Apache Software Foundation' ın üst düzey projelerinden biridir. Solr, Lucene projesinin en hızlı büyüyen alt projelerinden biridir. Arama yazılımı sektöründe, Lucene ve Solr özel lisansa sahip yazılımlara karşı güçlü bir alternatif sunmaktadır. Solr' ın sağladığı arama özellikleri, Türkiye' de ve dünyada pek çok firma ve internet sitesi tarafından kullanılmaktadır [46]. Örnekler arasında Instagram, Disney, Apple, Inc., SourceForge, GittiGidiyor.com ve HurriyetOto.com sayılabilir.

Solr, Java programlama dilinde geliştirilmiştir ve Jetty gibi bir sunucu tarafı barındırıcı (servlet container) içerisinde kendi başına çalıştırılabilir. REST benzeri HTTP/XML ve JSON uygulama programlama arayüzlerine sahiptir ve bu sayede neredeyse her programlama dili ile kullanılabilir. Solr, pek çok uygulama için programlama gerektirmeden konfigüre edilebilir. İleri düzeyde özelleştirme gerektiren durumlarda, Solr' ın kapsamlı eklenti mimarisi kullanılabilir.

Solr' ın önde gelen özellikleri arasında, güçlü metin arama desteği, aranan terimlerin vurgulanması, farklı alanlara göre arama, gerçek zamana yakın indeksleme, dinamik kümeleme, veritabanı entegrasyonu, zengin doküman desteği (Word, PDF, HTML vb.) ve coğrafi mekansal arama sayılabilir. Solr, yüksek güvenilirlik ve ölçeklenebilirliğe sahip hataya dayanıklı bir platformdur. Dağıtık indeksleme, replikasyon, yük dengeli

sorgulama, hata durumunda otomatik yük devretme ve kurtarma, merkezi konfigürasyon ve benzeri pek çok özellik sağlar.

Solr arama platformunun sağladığı temel özellikler şunlardır:

- Apache Tika projesi ile farklı türde dokümanların ayrıştırılabilmesi (PDF, Word ve HTML vb.)
- Veritabanı ve XML dosyalarından veri çekilebilmesi
- İleri tam-metin arama özellikleri
- Coğrafi mekansal arama
- Sorgu işleyiciler ekleyebilme
- Dağıtık arama desteğine sahip özelleştirilebilir bileşen tabanlı istek işleyiciler tanımlanabilmesi
- Dokümanların birden fazla kopyasının tespit edilebilmesi
- İndeks işleme adımlarının özelleştirilebilmesi
- Luke istek işleyici ile doküman derlemi hakkında bilgi alınabilmesi
- Çoklu arama indeksi tanımlanabilmesi

2.2.3 Apache HBase

Bu çalışmada veritabanı olarak kullanılan Apache HBase, ilişkisel veritabanı sistemlerine alternatif bir çözüm olarak ortaya çıkan, yatay olarak ölçeklendirilen bir NoSQL veri depolama sistemidir. Bunun yanısıra, sütun tabanlı veri yönetim sistemidir ve HDFS (Hadoop Distributed File System- Hadoop Dağıtılmış Dosya Sistemi) üzerinde çalışır [47]. Çok büyük veri kullanımları için uygundur. HBase, ilişkisel veritabanlarına benzemez ve yapısal sorgu dillerini desteklemez. HBase' in kendine ait bir dili vardır. NoSQL sistemlerinin ortaya çıkmasındaki en büyük sebep internetin hızlanması ile büyümekte olan ve yapısal olmayan verinin İlişkisel Veritabanı Yönetim Sistemlerinde performanslı bir şekilde işlenememesidir.

2.2.4 Apache Hadoop

Bu çalışmada büyük veri dosyalama ve büyük veri üzerinde dağıtık işlem kapasitesi sebebiyle kullanılan Hadoop, yaygın olarak kullanılmakta olan arama motoru

kütüphanesi Apache Lucene projesinin de geliştiricisi olan Doug Cutting tarafından geliştirilmiştir. Hadoop' un temelleri, Apache Lucene kullanılarak açık kaynak bir arama motoru geliştirilmesi fikriyle ortaya çıkan Apache Nutch projesinde atılmıştır. Nutch projesi, 2002 yılında başlamış ve proje kapsamında, bir içerik çekme sistemi (*crawler*) ve arama sistemi geliştirilmiştir. Ancak projenin başlangıç haliyle Web üzerindeki milyarlarca sayfaya ölçeklenmesi mümkün görülmemiştir. Bu noktada, Google arama motorunun kullandığı dağıtık dosya sisteminin (*Google File System, GFS*) mimarisinin anlatıldığı 2003 tarihli yayın [48] ve Google' ın geliştirdiği dağıtık veri işleme modeli olan MapReduce programlama modelinin anlatıldığı 2004 tarihli yayından [49] esinlenilerek, Nutch için dağıtık bir dosya sistemi ve MapReduce gerçekleştirimi yapılmıştır.

İlerleyen zamanlarda, Nutch için geliştirilen altyapının arama dışında başka alanlara da uygulanabileceği düşüncesiyle, Hadoop adı altında bağımsız bir alt proje oluşturulmuştur. Doug Cutting' in Yahoo' ya katılmasıyla, Hadoop' un geniş ölçekte bir sisteme dönüştürülmesi için gerekli kaynaklar sağlanmıştır. 2008 yılında, Yahoo, arama indeksinin 10.000 çekirdekli bir Hadoop kümesi tarafından oluşturulduğunu duyurmuştur. Hadoop, 2008 yılından beri üst seviye bir Apache projesi olarak yer almaktadır.

Hadoop' un büyük miktarda veri için genel amaçlı dağıtık veri depolama ve veri işleme konusunda sağladığı platform, endüstriden genel kabul görmüştür. Günümüzde, Oracle, Microsoft, HP gibi önde gelen pek çok bilişim firması, Hadoop desteği veya Hadoop ile entegrasyon çözümleri sunmaktadır. Ayrıca, birçok firmanın Hadoop dağıtımı vardır; bunların arasında Cloudera, Hortonworks, MapR gibi Hadoop firmalarının yanı sıra, IBM, Intel ve EMC gibi köklü firmalar da sayılabilir. Hadoop, Yahoo' nun yanı sıra Last.fm, Facebook, Twitter, New York Times, eBay, LinkedIn gibi pek çok şirket tarafından kullanılmaktadır [50], [51]. Örneğin, Facebook, Hadoop ve Hadoop ekosistemindeki ürünlerden Hive ve HBase' i veri ambarı ve gerçek zamanlı uygulamalar için kullanmaktadır. Twitter, Hadoop ve Hadoop ekosistemindeki Pig ve HBase' i veri analizi, görselleştirme, sosyal çizge analizi ve makine öğrenmesi gibi amaçlarla kullanmaktadır.

Hadoop, temelde iki ana kısımdan oluşur: HDFS (*Hadoop Distributed File System*) ve MapReduce. HDFS, Hadoop' un dağıtık dosya sistemidir ve dağıtık veri depolama altyapısı oluşturur. Hadoop MapReduce ise, dağıtık veri işleme için gerekli programlama modelini sağlar. Hadoop mimarisi, verinin bölümlenmesine ve büyük veri kümelerinin paralel bir şekilde işlenmesine izin verir. Bu işlemlerin detayları geliştiricilerden bağımsız olarak arka planda yürütülür.

Hadoop ve ekosistemindeki ürünler için tercih edilen sunucular, genel olarak "orta sınıf" olarak adlandırılan, nispeten düşük maliyetli ve kolaylıkla temin edilebilen sunuculardır (*commodity hardware*). Hadoop, hem hataya dayanıklı hem de yedekli bir mimari sağlar. Veri depolama ve veri işleme yapısı sayesinde, bir Hadoop kümesi yeni sunucuların eklenmesiyle kolaylıkla ölçeklenebilir ve petabaytlarca verinin içerildiği binlerce sunucuya erişilebilir.

2.2.4.1 Hadoop Distributed File System

Bir sunucunun veya bir bilgisayarın sahip olabileceği disk alanı sınırlıdır. Günümüzde pahalı disk sistemleri ile depolama çözümleri satılmaktadır. Ancak HDFS, sıradana yakın donanımlarla büyük dosyaları saklayabilmeyi sağlayan bir çözüm sunmaktadır [50]. Dağıtık dosya sistemlerindeki temel mantık, tek bir bilgisayarın saklayamayacağı hacimdeki veriyi birden fazla bilgisayar (sunucu) üzerinde dağıtık olarak saklamak olarak düşünülebilir.

Dağıtık dosya sistemleri, basit dosya sistemlerinin sunduğu hizmetleri bilgisayar ağları yardımıyla benzer şekilde sunmayı amaçlamaktadır. Dağıtık dosya sistemleri ağ aracılığıyla çalıştığından bilgisayar ağlarında meydana gelebilecek her türlü problem bu dosya sistemlerinin çalışmasını etkileyebilir. HDFS, bu problemler göz önünde bulundurularak tasarlanmıştır.

2.2.4.2 Namenode ve Datanode Kavramları

HDFS' de dağıtık dosya sisteminin tablosunu tutan makineye NameNode denilmektedir. DataNode ise asıl verinin yazıldığı, blokları tutan bilgisayarlardır. NameNode bütün özet bilgiyi tuttuğu için (hangi makinede hangi bayt adresinde hangi

bloğa ait bilgi olduğu) NameNode' daki bu bilginin kaybedilmesi, bozulması hatta Namenode' un tamamen erişilemez duruma gelmesi halinde bütün bilgi kaybedilmiş olacaktır. Bu sebeple Hadoop' ta datanode' ları orta sınıf makinelerle oluştururken, Namenode' u yedekliliği üst düzey bir makine ile oluşturmak gerekmektedir. Bunun yanında, Namenode bu özet bilgiyi (tabloyu) disk üzerinde saklamaktadır. Disk üzerindeki bu bilgi birden fazla dosya sistemine yazılabileceği gibi uzak bir bilgisayardaki NFS üzerine de yedeklenebilir.

2.2.5 MapReduce

MapReduce, büyük veriyi büyük kümelerde (clusters) işlemek üzere geliştirilmiş bir dağıtık programlama modelidir. Genel yapısı itibariyle MapReduce adından da anlaşılacağı üzere iki ana fonksiyondan oluşur. Birincisi map; bir yığının tüm üyelerini sahip olduğu fonksiyon ile işleyip bir sonuç listesi döndürür. İkincisi reduce; paralel bir şekilde çalışan iki veya daha fazla map fonksiyonundan dönen sonuçları harmanlar ve çözer. Hem map() hem de reduce() paralel bir şekilde çalışırlar, aynı anda aynı sistemde olmaları gerekmez.

Google, Web sayfalarını indekslemek için MapReduce kullanır. Birçok programlama dili kullanarak (Java, C++, Python, Perl, Ruby, C vs.) MapReduce programı geliştirilebilir.

Sonuç olarak, devasa boyuttaki yapısal olmayan verinin işlenmesi günümüz bilişim dünyasının üzerinde önemle durduğu zor konulardan biridir. Geleneksel yöntemlerle yapılan, belirli algoritmaların ve buluşsal yöntemlerin yönetilmesi çok zordur. Aynı zamanda çok hantal çalışan ve uzun süren programlar ortaya çıkmaktadır. MapReduce paralel sistemler üzerinde verinin işlenmesi ile alakalı çok başarılı çözümler sunmaktadır.

2.2.6 Apache ZooKeeper

ZooKeeper, dağıtık uygulamalar geliştirilmesine izin veren, dağıtık bir koordinasyon servisi olarak tanımlanabilir [52]. İlk olarak Yahoo' da şirket uygulamalarının, konfigürasyon bilgilerine hataya dayanıklı ve kolay bir şekilde ulaşabilmesini sağlamak için geliştirilmeye başlanmıştır [53]. Önceden Hadoop içerisinde bir alt proje olan

ZooKeeper, zamanla üst seviye bir Apache projesi haline gelmiştir. ZooKeeper' a dağıtık sunucu kümeleri üzerindeki işleri koordine etmeye yardımcı olacak çeşitli özellikler eklenmiştir. ZooKeeper, kullanıcılarına yüksek erişilebilirlik ve yüksek performans özellikleri sağlar. Aynı zamanda, geliştiricilere açık kaynak gerçekleştirimler içeren bir kütüphane sunar.

BÖLÜM 3

ARAŞTIRMA PROBLEMİNİN TANIMI

Bölüm 2' de verilen literatür taraması Web sayfalarındaki metin tabanlı atıfların PageRank değerlendirmesinde kullanılmadığını göstermektedir. Arama algoritmalarının geliştirilmesinde bağlantı tabanlı yöntemler, Web sayfalarının konulara ayrılması ve kişiselleştirme yöntemleri gibi yaklaşımlar kullanılmıştır.

Bu tezde popülarite tabanlı sıralama algoritmalarının dolaylı bağlantıları dikkate alarak nasıl geliştirilebileceği üzerinde durulmaktadır. Tezin kapsamında belirlenen problem ile ilgili araştırma soruları Bölüm 3.1' de detaylı olarak anlatılmaktadır. Bu araştırmanın özgün katkısı metin tabanlı atıfın PageRank değerlendirmesinde kullanılması ve yeni bir değerlendirme yönteminin sunulmasıdır.

3.1 Araştırma Soruları

Arama algoritmalarının geliştirimi için takip eden araştırma sorularına cevapların aranması gerekmektedir.

3.1.1 Dolaylı bağlantılar popülarite tabanlı algoritmaların geliştirilmesinde kullanılabilir mi? Dolaylı bağlantılar algoritma geliştirilmesinde yeterli midir?

Arama motorları algoritmalarının geliştirimi için birçok etken ortaya koymaktadır. Web sayfalarının birbirlerine olan bağlantıları, bu bağlantıların sayısı, Web sayfasının yaşı vb. gibi etkenler kullanılarak algoritmaların başarımı artırılabilir. Ancak, metin tabanlı atıf kullanılarak arama algoritmasının geliştirilmesi üzerinde yapılmış araştırmaların

eksikliği görülmektedir. Bu araştırma ile dolaylı bağlantı olarak kullanılan metin tabanlı atıfın arama algoritmasının başarısını artırmada etkili olup olmayacağı ortaya çıkarılacaktır. Günümüzdeki yaygın kullanımı nedeniyle PageRank algoritmasında geliştirme yapılmaktadır. Bu araştırmayla metin tabanlı atıfın PageRank algoritmasını iyileştirmede yeterli olup olmadığı irdelenmektedir.

3.1.2 Dolaylı bağlantılar kullanılarak popülarite tabanlı sıralama algoritmalarının doğruluk değerini artırmak için yöntem ne olabilir?

Bazı Web sayfalarında, o sayfa için önemli olan sayfaların sadece adı ya da kısaltması gibi ifadeler geçmektedir. Varolan PageRank algoritması ise sadece bağlantıları değerlendirmeye aldığı için dolaylı bağlantılar gözardı edilmektedir. Böylece bu Web sayfaları kullanıcılar arama yaptığında alt sıralarda çıkmaktadır. Fakat bu ifadelerin de bağlantılar gibi PageRank algoritmasının değerlendirilmesine katılması gerekmektedir.

3.1.3 Web sayfasının içinde dolaylı bağlantı tespiti nasıl yapılabilir?

Dolaylı bağlantıların tespiti için kullanılacak sistemin bu bağlantıları temsil edecek ifadeleri önceden öğrenmiş olması gerekmektedir. Bu da iki aşamalı çalışma gerektirmektedir. İlk olarak dolaylı bağlantıların metin ifadeleri öğrenilecek sonra ise bu ifadeler Web sayfalarında aranacaktır. Bulunduğunda ise bağlantı olarak o sayfanın giden bağlantı listesine eklenecektir.

3.1.4 Web sayfasının içinden elde edilen dolaylı bağlantılarla geliştirilmiş PageRank algoritması klasik PageRank ya da TF-IDF algoritmalarıyla karşılaştırıldığında elde edilen sonuçlar nedir?

Klasik PageRank (TF-IDF + klasik PageRank) ve TF-IDF sıralama algoritmalarıyla, bu tezde önerdiğimiz dolaylı bağlantıların kullanıldığı PageRank algoritması karşılaştırıldığında nasıl sonuçlar verdiği araştırılacaktır.

BÖLÜM 4

ÖNERİLEN ÇÖZÜM YÖNTEMİ

Bulmak istediğiniz, trilyonlarca Web sayfası değil, sorunuzun yanıtıdır. Sıralama algoritmaları, size tam olarak istediğiniz şeyi vermek için ipuçlarını araştıran bilgisayar programlarıdır.

Tipik bir sorgu için, yararlı bilgiye sahip milyonlarca olmasa bile binlerce Web sayfası bulunur. Sıralama algoritmaları, sorularınızı alıp yanıtlara dönüştüren bilgisayar işlemleri ve formülleridir. Günümüzde, Google' ın algoritmaları gerçekte neyi arıyor olabileceğinizi tahmin etmeyi mümkün kılan 200' den fazla benzersiz sinyale veya "ipucuna" dayanmaktadır. Bu sinyaller, Web sitelerindeki terimler, içeriğin güncelliği, bölgeniz ve PageRank gibi öğeleri içerir.

4.1 Klasik PageRank Algoritması

PageRank algoritması sayfaya verilen bağlantılar ve sayfadan giden bağlantılar ile ilgilenir. Örneğin *p* sayfasının PageRank değerini hesaplayalım.

d faktörü genelde 0.85 olarak tanımlanır [5][13]. Bu faktör tüm oranların 1'i geçmesini engellemek için kullanılmaktadır. Bir kişinin her adımda bir sonraki sayfaya tıklama olasılığını temsil etmektedir. *q->s*, *q* sayfasından giden bağlantı sayısı olsun. *p* sayfasının PageRank değeri denklem (4.1.1)' deki gibi hesaplanmaktadır:

$$R(p) = (1-d) + d \cdot \sum_{\{q:q\to p\}} \frac{R(q)}{|s: q \to s|} \qquad (4.1.1)$$

$R(p)$: *p* sayfasının PageRank değeri

$R(q)$: *q* sayfasının PageRank değeri

S: *q* sayfasından giden bağlantıların toplam sayısı

d: Rastsal Gezinti Faktörü (damping factor) , 0 <= *d* <=1 arasındadır.

PageRank, bağlantı veren sayfadan hedef sayfaya rastsal gezinti faktörü gözönüne alınarak eşit oranda bölünür. Şuna dikkat etmek gerekir: PageRank sayfa için geçerlidir, örneğin iletisim.html, firmamiz.html gibi. PageRank Web sitesi için değil Web sayfası için geçerlidir. Bu nedenle her sayfanın ayrı bir PageRank değeri vardır. Bu yüzden bağlantı verilirken PageRank bölündüğünde sayfadan sayfaya bölünmektedir.

4.2 Önerdiğimiz İyileştirilmiş PageRank Sıralama Algoritması

Bu çalışmada bir Web sayfasını başka bir Web sayfasına bağlayan metin tabanlı atıflar hesaba katılarak PageRank hesaplamasının doğruluğunu artırmak için bir yöntem sunulmaktadır. Literatürde varolan PageRank tabanlı sıralama algoritmaları sadece arama robotları tarafından takip edilen bağlantıları hesaba katmaktadır.

Bu tezde, bu bağlantılara ek olarak metin tabanlı atıflar da kullanılarak Web sayfalarının popülaritesinin daha doğru bir şekilde hesaplanacağını savunmaktayız. Bu yüzden gerçek bağlantı (hyper-link) olmayan metin tabanlı atıflar da gizli ya da dolaylı bağlantılar olarak hesaba katılarak PageRank algoritması denklem (4.2.1)‘ deki gibi genişletilmiştir.

$$R(p) = (1-d) + d \cdot \sum_{\{q:q\to p\}} \frac{R(q)\, W(q)}{|s: q \to s|} \qquad (4.2.1)$$

Burada PageRank algoritmasından farklı olarak $W(q)$ faktörü bulunmaktadır. Bu faktör, *q* sayfasından *p* sayfasına giden bağlantının gerçek ya da dolaylı bağlantı olmasına göre değişen, *q* sayfasının ağırlık değeridir.

4.3 Yöntemimizin Dayandığı Öngörüler

Bu yöntemde, dolaylı bağlantılar hesaba katılarak bir sayfadan diğer sayfalara giden bağlantı sayısı artırılmaktadır. Bir dolaylı bağlantı, bir Web sayfasının diğer Web

sayfasına gerçek bağlantı verilmeden yaptığı atıftır. Böylelikle, A Web sayfasının PageRank değeri A' ya gelen gerçek ve dolaylı bağlantılar ile hesaplanmaktadır.

Birçok Web sayfasında başka sayfalar, gerçek bağlantı verilmeden, sadece ismi ile veya o sayfayı temsil eden kelimeler kullanılarak bahsedilmektedir. Bu çalışmada, Web sayfalarındaki bu tip alıntılardan yola çıkılarak hedef Web sayfalarına verilen dolaylı bağlantıların tespit edildiği bir yöntem sunulmaktadır. Bu yöntem ile bulunan bu dolaylı bağlantıların adreslediği Web sayfalarının popülaritesini artırdığını iddia etmekteyiz. Bu nedenle, bulunan dolaylı bağlantılar gerçek bağlantı Web çizgesine dahil edilmektedir. Daha sonra PageRank skorları genişletilmiş yeni Web çizgesi üzerinde hesaplanmakta ve dolaylı bağlantıların eklenmesinin sıralama sonuçlarına olumlu etkisi olup olmadığı inceledik.

Bu çalışmada dolaylı bağlantıların Web sayfalarının içeriğinde gömülü olarak bulunan gerçek bağlantı metinleri olduğu öngörülmektedir. Gerçek bağlantı metni, görünen, seçilebilen bir gerçek bağlantı ifade eder. Gerçek bağlantı metni olarak elde edilen ifadeler o sayfanın arama motorları tarafından alacağı skoru etkilemektedir. Dolaylı bağlantıların Web sayfalarına atanabilmesi için ilk içerik çekme aşamasında bulunan gerçek bağlantıların metinleri, yeni içeriği çekilen Web sayfalarında aratılmaktadır. Burada eğer bir Web sayfası yazarı, sayfasına böyle bir metin eklediyse ilgili Web sayfasına atıf yapmak istediği öngörülmüştür. Diğer bir deyişle, hedef olan Web sayfasına dolaylı bağlantı vermiş olduğu öngörülmüştür.

Dolaylı bağlantıların önemi kaynağına göre değişebilmektedir. Mesela bir Web sayfasının başlığı ya da o sayfada sıklıkla kullanılan ifade, kalın olarak belirtilmiş metinler vb. gibi ifadeler önemli birer dolaylı bağlantı olabilirler. Dahası, dolaylı bağlantıların önemi gerçek bağlantıların öneminden farklı olabilir. Bu yüzden PageRank hesaplamasına ağırlık faktörü eklenmiştir. Fakat yöntemin doğrulamasını kolaylaştırmak için bu çalışmada dolaylı bağlantıların kaynağı olarak sadece gerçek bağlantı metinleri düşünülmüştür. Ayrıca dolaylı bağlantılar, gerçek bağlantılar gibi düşünülmüş ve ağırlıkları eşit alınmıştır.

4.4 İyileştirilmiş PageRank Sıralama Yöntemi için izlediğimiz adımlar

İçerik çekme işlemi, çekirdek Web sayfası (çekirdek URL) listesinden başlar ve başka sayfaları çekmek için aralarındaki bağlantıları kullanır. Bu işlem; içeriği çekilecek URL listesinin verilmesi, bu URL' lerin gruplanması, içeriklerinin alınması, içeriklerinin çözümlenmesi ve çözümlenen içeriğin veritabanına kaydedilme aşamalarını içerir. Yani toplamda 5 adıma sahiptir. İçerik çözümleme aşamasında çekirdek URL listesinden elde edilen bağlantılar ikinci içerik çekme işleminde tekrar ziyaret edilir. Bu işlem aşamalı olarak tekrar eder. Dolaylı bağlantılarla zenginleştirilmiş Web çizgesi oluşturmak için bu yöntemde iki aşama bulunmaktadır.

Yöntemin ilk adımında normal içerik çekme süreci takip edilmektedir. Fakat bu aşamadaki içerik çözümleme kısmında ikinci aşamada kullanılmak üzere çekirdek Web sayfa listesindeki URL' lerin giden bağlantılarının gerçek bağlantı metinleri ve bağlantıları ayrı bir veritabanına kaydedilmektedir. İlk aşamanın sonunda çok büyük boyutta içeriği çekilmiş veri elde edilmektedir. Bu veri temel alınarak Web sayfalarının düğüm, bağlantıların ise kenar olduğu bir Web çizgesi oluşturulmaktadır. Sayfanın içeriğini alma, sayfada bulunan bağlantıları çıkarma ve içeriğini kaydetme normal içerik çekme sürecindeki gibi gerçekleştirilmiş, sayfaların skorları içeriği çekilen veri temel alınarak oluşturulan Web çizgesi kullanılarak geliştirilmiş PageRank hesaplaması ile belirlenmiştir.

İkinci adım için içerik çekme aracının içerik çözümleme kısmına eklenti geliştirilmiştir. Normal içerik çözümleme işlemi tamamlandıktan sonra bu eklenti devreye girmektedir. Bu aşamada içerik çekme çözümleme işleminden başlanmaktadır. Bu işlemde içeriği çözümlenen her Web sayfasının metninde veritabanına kaydedilen her gerçek bağlantı metni aratılmaktadır. Eğer bu Web sayfasının metninde veritabanındaki herhangi bir metin eşleşirse bu dolaylı bağlantı olarak kabul edilmektedir ve veritabanında bu gerçek bağlantı metninin URL' i, ilgili Web sayfasının giden bağlantı listesine ve Web çizgesine eklenmektedir. Eklenen bu dolaylı bağlantıya gerçek bağlantı gibi değer verilmektedir. İkinci adımın sonunda Web sayfaları arasındaki dolaylı bağlantıları da kapsayan zenginleştirilmiş bir Web çizgesi elde edilmektedir. İkinci adımda zenginleştirilmiş Web çizgesi kullanılarak PageRank hesaplanmaktadır.

Girdi: **1.** İlk içerik çekme işleminin sonucu olan URL, gerçek bağlantı metni.

2. Şuan içeriği çözümlenen URL (Önceki URL listesinin giden bağlantılarının içeriğinin çözümlenmesi ile elde edilen) .

Çıktı: Dolaylı bağlantı atıflarıyla güncellenen Web sayfalarının zenginleştirilmiş giden bağlantıları.

GERÇEK BAĞLANTI METİNLERİ = {a_1,a_2, … , a_m}, girdi url kümesinin gerçek bağlantı metin listesi

BAĞLANTILAR = {l_1, l_2, … , l_m}, girdi url kümesinin gerçek bağlantı metnine ait bağlantıları

KELİMELER = {w_1, w_2, … , w_n} , şuan içeriği çözümlenen url' in metin içeriği

VAROLAN_GİDEN_BAĞLANTILAR = {Co_1, $Co_{2,\dots,}Co_n$} , şuan içeriği çözümlenen url' in giden bağlantı listesi

KELİMELERİ_KÖKÜNE_AYIR = { } , köküne ayrılmış kelime listesi

GERÇEK_BAĞLANTI_METİNLERİNİ_KÖKÜNE_AYIR= { } , köküne ayrılmış gerçek bağlantı metin listesi

İlklendir *KELİMELERİ_KÖKÜNE_AYIR;*

İlklendir *GERÇEK_BAĞLANTI_METİNLERİNİ_KÖKÜNE_AYIR;*

foreach $w_i \in$ *KELİMELER* **do**

KELİMELERİ_KÖKÜNE_AYIR.Ekle(köküne_ayır(w_i));

endfor

foreach $a_j \in$ *GERÇEK BAĞLANTI METİNLERİ* **do**

GERÇEK_BAĞLANTI _METİNLERİNİ_KÖKÜNE_AYIR. Ekle (köküne_ayır (a_j));

if *doğrula(GERÇEK_BAĞLANTI _METİNLERİNİ_KÖKÜNE_AYIR.At(j))* **then**

foreach $Sw_i \in$ *KELİMELERİ_KÖKÜNE_AYIR* **do**

if *eşleşirse(GERÇEK_BAĞLANTI _METİNLERİNİ_KÖKÜNE_AYIR.At(j), Sw_i)* **then**

VAROLAN_GİDEN_BAĞLANTILAR.Add(GERÇEK_BAĞLANTI_METİNLERİNİ_KÖKÜNE_AYIR.At(j), l_j);

endif

endfor

endif

endfor

Şekil 4.4.1 Önerdiğimiz dolaylı bağlantıların tespit algoritması

Yaklaşımın daha iyi anlaşılabilmesi için ikinci aşamada kullanılan algoritmanın sözde kodu (pseudo code) Şekil 4.4.1‘ de verilmektedir. Bu algoritma önceden içeriği çekilmiş veriden gerçek bağlantı metin bilgisini edinmektedir. Algoritma iki girdi kullanmaktadır. İlki gerçek bağlantı metni ve URL çifti listesidir. İkincisi ise şuan içeriği çözümlenen URL’ dir. İçeriği çekilecek bütün Web sayfaları için bu algoritma tekrar etmektedir. Algoritmanın çıktısı içeriği çözümlenen URL’ lerin güncellenen giden bağlantı listesidir. Yeni oluşturulan giden bağlantı listesine göre zenginleştirilmiş Web çizgesi oluşturulmaktadır. Gerçek bağlantı metinlerinin ve Web sayfasının içeriğindeki kelimelerin eşleşmesini garantiye almak için içeriği çözümlenen URL’ in metnindeki her kelime ve gerçek bağlantı metinlerinin herbiri köklerine ayrılmaktadır. Bu işlem Zemberek kütüphanesi [54] tarafından yapılmaktadır. Zemberek, açık kaynak kodlu, ortamdan bağımsız, genel amaçlı Doğal Dil İşleme kütüphanesi olan ve Türk dilleri -özellikle Türkçe- için araç seti olarak tasarlanmıştır.

Deneyimsiz Web sayfa yazarlarının sayısı oldukça fazla olduğu için birçok sayfa boşluk, özel karakter ya da sayı gibi metin olmayan gerçek bağlantı ifadeleri içermektedir. Bu ifadeler özensiz oluşturulan bağlantılar olarak düşünülmüş ve işleme tâbi tutulmamıştır. Ek olarak, Web sayfalarının kendi kendine bağlantı vermelerini engellemek için içeriği çözümlenen URL ile gerçek bağlantı metninin karşılığı olan adres eşleştiğinde bu URL giden bağlantı listesine tekrar eklenmemektedir. Ayrıca içeriği çözümlenen Web sayfasının giden bağlantı listesinde gerçek bağlantı metninin URL’ i bulunuyorsa bu URL giden bağlantı listesine tekrar eklenmemektedir.

Gerçek bağlantı metnini Web sayfasının içeriğinde ararken sadece kelimeler (metin tabanlı ifadeler) kullanılmaktadır. Arama yapılırken eğer gerçek bağlantı metni, içeriği çözümlenen URL’ in metninde bulunursa gerçek bağlantı metninin veritabanındaki adres karşılığı, o URL’ in giden bağlantı listesine eklenmektedir.

Yapılan çalışmanın daha iyi anlaşılabilmesi için Çizelge 4.4.1’ deki ifadeler örnek verilebilir.

Çizelge 4.4.1 Örnek metin-bağlantı eşleşmesi

No	Metin	Bağlantı
1	YTÜ	www.yildiz.edu.tr
2	Yıldız Teknik Üniversitesi	www.yildiz.edu.tr
3	TÜBİTAK	www.tubitak.gov.tr
4	Türkiye Bilimsel ve Teknolojik Araştırma Kurumu	www.tubitak.gov.tr

Örneğin ilk içerik çekme aşamasında veritabanımıza 'http://www.tubitak.gov.tr' URL' i ve 'TÜBİTAK' gerçek bağlantı metni bilgisi kaydedilmektedir. Çözümleme aşamasından başlayan ikinci içerik çekmesi işleminde 'TÜBİTAK' kelimesi 'W' Web sayfasında metin tabanlı bilgi olarak bulunduğunda ve veritabanımızdaki 'TÜBİTAK' gerçek bağlantı metni, bu kelime ile eşleştiğinde, 'http://www.tubitak.gov.tr' adresi 'W' Web sayfasının giden bağlantı listesine dolaylı bağlantı olarak eklenmektedir. Bu yüzden, o andan itibaren 'TÜBİTAK' kelimesi aratıldığında 'W' Web sayfası diğer ilgili sonuçlar arasında daha yukarıda çıkacaktır.

Bir atıfın dolaylı bağlantı sayılabilmesi için o atıfı oluşturan gerçek bağlantı metni – URL ikilisinin belli sayıda tekrarlanması gerekir. Bu çalışmada bu veri dikkate alınarak parametre olarak kullanılmıştır. Fakat olabildiğince fazla dolaylı bağlantı yakalayabilmek için bu sayı 1 olarak belirlenmiştir.

4.5 Veri Kümesi

Bu çalışmada TÜBİTAK' ta geliştirilen arama motoru kapsamında oluşturulan veri kullanılmıştır. Veri kümesini oluşturmak için açık kaynak kodlu Nutch aracı kullanılmıştır. Çalışmamız sadece Türkçe Web sayfaları ile ilgili olduğu için dil tespit eklentisi kullanılmıştır. Dil tespitinde en iyi bilinen teknik farklı dillerdeki metinler ile eğitilen ve eğitilen model temel alınarak her dokümanın puanlandırıldığı n-gram modellerini kullanmaktır. Çalışmamızda kullandığımız, Tübitak Projesi' nde Web içerik

çekme sistemi için istatiksel n-gram modellerinden yararlanarak dil tespiti yapmak için geliştirilen ve "lang-id" açık kaynak kodlu eklentiden faydalanılmıştır. Bu yöntem kullanılarak Web sayfalarının dili tespit edilmiştir. Bu da sayfa içeriğine özel işlem yapılabilmesini sağlamakta ve farklı diller için içerik çekilmesini önlemektedir.

Yöntemimizin ilk aşamasında bu yöntemi takip ederek 10 bin adres içeren çekirdek URL listesi ile Türkçe Web sayfalarının içeriği çekilmiştir. Bu çekirdek liste DMOZ' dan [55] elde edilmiştir. DMOZ, en büyük, en kapsamlı, Web' in insanlar tarafından düzenlenen bir dizinidir. Bu dizin gönüllülerden oluşan küresel bir topluluk tarafından oluşturulmuştur ve halen geliştirilmektedir.

4 derinlikle gerçekleştirilen ilk içerik çekme işleminin içerik çözümleme adımından sonra yine 10 bin Web sayfası elde edilmiş fakat yaklaşık 86 milyondan fazla bağlantı elde edilmiştir. İçerik çözümleme adımından sonraki veritabanı güncelleme adımında ise önceki aşamada elde edilen bağlantılar da Web sayfası olarak eklenmiş, böylelikle bu aşamanın sonunda yaklaşık 86 milyon Web sayfası elde edilmiştir. Fakat bağlantı sayısı yine 86 milyondan fazladır. Bu işlemin sonrasında Web sayfalarının düğüm, bağlantıların ise kenar olduğu bir Web çizgesi oluşturulmuştur.

İkinci aşamada, Şekil 4.4.1' de anlatıldığı gibi Web sayfalarından elde edilen dolaylı bağlantılar eklenmektedir. Bu aşamanın içerik çözümleme adımında yaklaşık 86 milyon Web sayfası üzerinden 410 milyondan fazla bağlantı elde edilmiştir. Bu bağlantıların yaklaşık 105 milyonu dolaylı bağlantı, 305 milyonu ise gerçek bağlantıdır. En son veritabanı güncelleme adımında ise elde edilen bağlantılar Web sayfası olarak eklenerek yaklaşık 410 milyon Web sayfası elde edilmiş, bağlantı sayısı ise aynı kalmıştır.

Bu çalışma Mayıs ve Kasım 2014 ayları arasında gerçekleştirilmiştir yani içerik çekme işlemi 5 ay sürmüştür. Bu süreçte ağda yaşanan sıkıntılar sebebiyle ikinci aşama, birden fazla olacak şekilde gerçekleştirilmiştir.

Bu süreçte sadece Türkçe sayfaların içeriği çekilmiştir. Farklı yerel ve bölgesel URL ağlarındaki Dünya çapında yer alan sunuculardan Web sayfalarının içeriği çekilmiştir fakat Web sayfalarının dili tespit edilebildiği için Web sayfalarını ".tr" uzantılı olacak şekilde ayırma ihtiyacı hissedilmemiştir.

BÖLÜM 5

UYGULAMA DETAYLARI

Üretilen büyük verinin depolanması, yönetilmesi, işlenmesi, bilgiye dönüştürülmesi ve anlamlandırılması noktasında birçok problemi karşımıza çıkarmaktadır. Bu büyüklükteki veri üzerinde geleneksel yöntemler yetersiz kalmaktadır. Kısaca; büyük veri ile baş edebilmek için yeni yöntemlere ve teknolojilere ihtiyaç vardır.

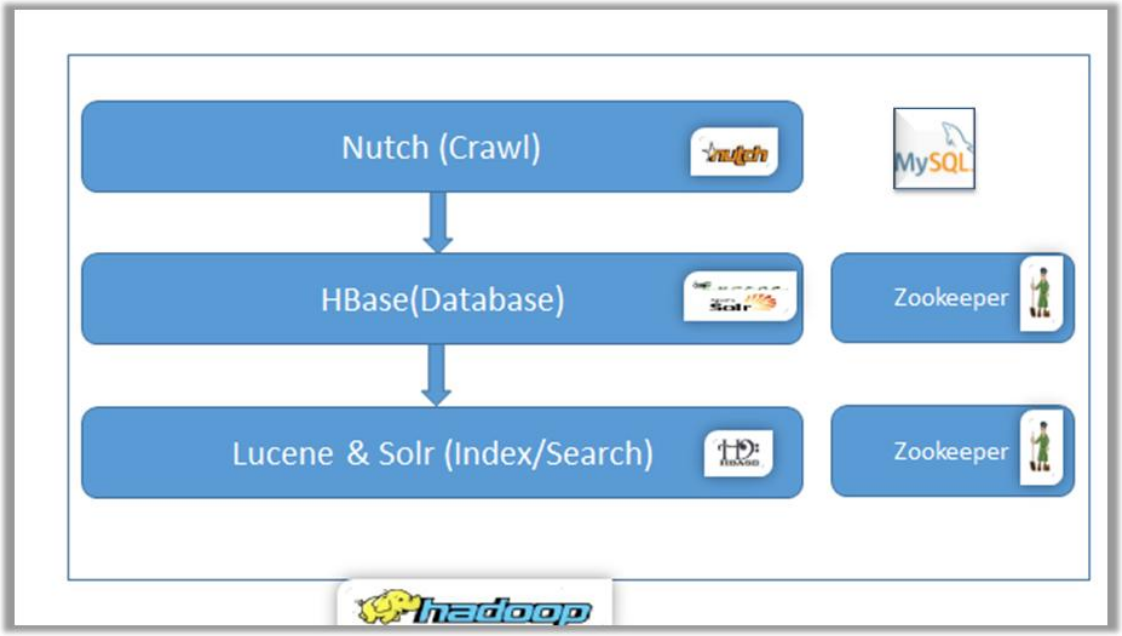
Büyük veri ile ilgili çalışmalar 2004 yılında Google çalışanlarından Jeffrey Dean ve Sanjay Ghemawat' ın "MapReduce: Simplied Data Processing on Large Clusters" başlıklı makalesi ile hız kazanmıştır. Bu makalenin yayınlanması sonrasında Apache tarafından Hadoop isimli bir proje başlatılmış, Hadoop projesinin ilk sürümü 4 Eylül 2007 tarihinde çıkmıştır. Fakat Hadoop' un temelleri, aslında içerik çekme aracı olan Apache Nutch projesinde atılmıştır. Bu kısma kadar açıklananlardan da anlaşılacağı üzere büyük veri ve bu alandaki çalışmalar yeni gereksinimler oluşturmuş, bu gereksinimleri karşılayan yeni araçlar geliştirilmiştir. Günümüzde kullanılan birçok büyük veri teknolojisi de Apache grubu çevresinde toplanmıştır. Büyük veri teknolojileri denildiğinde genellikle Apache ve ekosistemindeki teknolojiler kastedilmektedir.

Şekil 5.1.1' de çalışmanın yapıldığı ortam gösterilmiştir. İnternet sayfalarının içeriğini çekmek için arama robotu olarak Nutch 2.1 kullanılmıştır. Nutch aracının içerik çözümleme aşamasına, zenginleştirilmiş Web çizgesi oluşturabilmek için dolaylı bağlantıları de ekleyen bir eklenti geliştirilmiştir. Bir sonraki işlem olan 3 ayrı algoritmanın indeksleme ve sorgulama işlemleri için ise Solr 4.6.0 kullanılmıştır. Lucene 4.6.0 çalışma esnasında arama kütüphanesi olarak kullanılmış, elde edilen veri ise Apache HBase' de tutulmuştur. HBase' in bu çalışmada 3 yedekli, 7 RegionServer' ı ve

bir Master sunucusu bulunmaktadır. Bu araçların arasındaki koordinasyonu sağlamak için Zookeper kullanılmıştır. Araçların hepsi Hadoop sisteminde çalıştırılmıştır. Hadoop sisteminde ise 1 NameNode ve 7 DataNode bulunamaktadır. Her node' un 8.06 terabyte hdd, 64 GB ram, 3.0 GHz işlemci hızı ve 16 çekirdeği ile Centos işletim sistemi vardır.

Ayrıca Hadoop sisteminde gerçek bağlantı metni ve URL çiftlerini tutmak için ilişkisel veritabanı olarak MySQL kullanılmıştır. Bu veritabanında; asıl URL, giden bağlantı, gerçek bağlantı metni, gerçek bağlantı metin sayısı, gerçek bağlantı metin uzunluğu ve dolaylı bağlantı mı olmak üzere 6 nitelik bulunmaktadır. Asıl URL, giden bağlantı ve gerçek bağlantı metin alanları ilk içerik çekme işleminden elde edilmektedir. Asıl URL, çekirdek URL listesindeki adrestir. Giden bağlantı alanında çekirdek URL listesindeki sayfalardan giden bağlantılar bulunmaktadır. Gerçek bağlantı metni de bu giden bağlantıların ilgili sayfadaki ifadeleridir. Gerçek bağlantı metin sayısı, içerik çekilen sayfalarda ilgili metnin kaç defa geçtiği bilgisini tutar. Bu nitelikteki bilgi ile eğer bir gerçek bağlantı metni birden fazla sayfada geçiyorsa o metnin kullanılmaması için oluşturulmuştur. Gerçek bağlantı metin uzunluğu ise veritabanından en uzun gerçek bağlantı metni ile aramaya başlanması içindir. En son nitelik ise veritabanındaki metnin dolaylı bağlantı metni olup olmadığını temsil etmektedir.

5.1 Apache Nutch Üzerindeki Geliştirmelerimiz



Şekil 5.1.1 Çalışma ortamı

Apache Nutch, yerel dosya sistemi, iç internet ve evrensel internet olmak üzere üç farklı durumda da çalışabilmektedir. 2.x sürümünde Nutch' ın desteklediği Cassandra, HBase ve MySQL gibi veritabanları kullanılabilmektedir. Yapılan çalışmada da veritabanı olarak HBase ve MySQL kullanılmıştır.

Apache Nutch' ın içerik çekme işlemi 5 adımdan oluşmaktadır. Bu beş adımdan ilki sadece bir kez ve döngülerden hemen önce uygulanır. Diğer dört adım ise her içerik çekme döngüsünde birer kez gerçekleştirilir [56].

1. **Yerleştirme (Inject):** İçerik çekme işleminin başlayacağı çekirdek internet sayfası kümesi (ilk etapta içeriğinin çekilmesi istenen internet sayfaları) URL veritabanına yerleştirilir. Nutch 2.x' de veritabanı olarak "webpage" olarak adlandırılan bir tablo kullanılmaktadır. Bu tablo, içerik çekme adımlarının girdi ve çıktılar için kullandığı tek veri deposudur. Bu tablodaki her satır ziyaret edilen ya da edilecek olan bir internet sayfasını temsil etmektedir. Webpage tablosunda her internet sayfası için şu bilgiler tutulmaktadır [57].

Çizelge 5.1.1 Webpage tablosu alanları

Alan İsmi	**Açıklaması**
Id	Değiştirilmiş internet sayfası adresi
headers	Http sayfa başlığı
text	Çözümlenmiş metin alanı
status	Sayfanın içerik alınma durumu (örn. 1- henüz içeriği çekilmemiş, 2- içeriği çekilmiş ve 3- sayfa artık mevcut değil, 4- geçiçi yönlendirilmiş, 5- kalıcı yönlendirilmiş)
markers	Sayfanın yerleştirme, üretme, içerik alınma ve çözümleme imleyicileri (grup numaraları)
parseStatus	Çözümleme durumu (örn. Failed veya Success)
modifiedTime	Sayfa imzasının değiştiği son zaman
score	Sayfanın sıralama ve önem değeri
Typ	Mime/doküman tipi (örn. text/html ve application/pdf)
baseUrl	Taban URL adresi
content	Sayfanın içeriği
title	Sayfanın HTML başlığı
repUrl	Yönlendirme URL adresi
fetchInterval	İçerik alma sıklığı
prevFetchTime	Bir önceki içerik alınma zamanı
inlinks	Gelen bağlantılar
prevSignature	Önceki imza
outlinks	Giden bağlantılar
fetchTime	İçerik alma zamanı
retriesSinceFetch	İçerik alma için yeniden deneme sayısı

Çizelge 5.1.2 Webpage tablosu alanları (devamı)

Alan İsmi	**Açıklaması**
protocolStatus	Protokol durumu (örn. Access_Denied 17, Blocked 23 ve Exception 16)
signature	Sayfanın imzası (her içerik alma işleminde yeniden hesaplanır)
metadata	Sayfanın üst verisi

Webpage tablosundaki alanlardan "id" alanı birincil alandır. Aynı alan ismine sahip internet sayfası adreslerinin tabloda bir arada olabilmesi ve arama işlemlerinin gerekirse kolaylaşması için adresler tabloya yerleştirilirken (alan ismi:protokol:kapı:yol) formatını sağlayacak şekilde değiştirilir. Örneğin, "http://www.yildiz.edu.tr" adresi webpage tablosuna "tr.edu.yildiz.www:http/" olarak yerleştirilir.

Nutch 2.x sürümünde, çekirdek internet sayfaları kümesi bir metin dosyasından (seed.txt) okunur. Bu dosyada, her internet sayfası adresi tek bir satırda olacak şekilde olmalıdır. Aşağıda iki çekirdek internet sayfası içeren örnek bir dosya verilmiştir.

http://www.yildiz.edu.tr

http://www.tubitak.gov.tr

2. Üretme (Generate): URL veritabanındaki internet sayfalarından bir sonraki içerik çekme döngüsünde ziyaret edilecek sayfalar belirlenip bir grup (ziyaret grubu) oluşturulur. Bu grup webpage tablosundan oluşturulur ve bazı internet adresleri (satırlar) ziyaret edilecek sayfalar olarak işaretlenir. Bu süreçte Nutch tarafından gerçekleştirilen işlemler şöyle özetlenebilir:

a. Daha önceden ziyaret edilmiş olarak işaretlenmiş sayfalar atlanır.
b. İnternet adresleri normalize edilir ve tanımlanmış URL filtreleri uygulanır.

c. Sayfaların içerik alma zamanı ve sıralama/önem değerleri hesaplanır. Hesaplama işleminde aksi belirtilmedikçe OPIC (Online Page Importance Calculation) kullanılır.
d. Kullanıcı tarafından belirlenmiş olan içeriği alınacak internet sayfası sayısına ulaşılıp ulaşılmadığı kontrol edilir (Bu kontroller, aynı alan adresine sahip internet sayfalarının sayısının da kontrol altında tutulmasını sağlar.).
e. Sıradaki içerik çekme döngüsünde içeriğinin alınmasına karar verilen tüm internet sayfaları için bir ziyaret grubu numarası üretilir ve tabloda ilgili düzenlemeler gerçekleştirilir.

3. **İçeriği Almak (Fetch):** Ziyaret grubunda yer alan her internet sayfası ziyaret edilerek içeriği alınır. Bu süreçte Nutch tarafından gerçekleştirilen işlemler şöyle özetlenebilir:
   a. Tüm webpage tablosu tıpkı üretme işlemindeki gibi incelenir ve içeriğinin çekilmesi için işaretli sayfalar belirlenir.
   b. Her bir alan adresi için farklı bir içerik alma kuyruğu oluşturulur.
   c. Yapılan ziyaretler sırasında sunucuların belirlediği politikalara ve nezaket kurallarına riayet edilir.
   d. Webpage tablosunda içeriği alınan internet sayfalarının içerik alma zamanı (fetchTime) güncellenir.
4. **İçeriği Çözümleme (Parse):** İçeriği alınan internet sayfaları çözümlenir ve içerik metni, üst veri, tespit edilen internet bağlantı adresleri vb. gibi bileşenlere ayrıştırılır. Webpage tablosu bu işlemin sonucunda güncellenir.
5. **Veritabanı Güncelleme (UpdateDB):** URL veritabanındaki bilgi içerik çözümleme sırasında tespit edilen yeni internet adreslerinin eklenmesini takiben güncellenir. Webpage tablosundaki her satır, elde edilen bilgiler (örn. giden bağlantıların sıralama/önem değeri ve çengel metni) ile güncellenir.

Bu işlemler tek bir komutla gerçekleştirilebildiği gibi içerik çekmenin tüm alt adımları farklı komutlarla tek tek de çalıştırılabilir. Kullanılan komutlar veya parametreleri konusunda Nutch sürümleri arasında farklar mevcuttur [57]. Nutch 2.x sürümünde

içerik çekme işlemi önceden hazırlanmış bir çekirdek internet sayfası kümesi kullanılarak aşağıdaki komut ile çalıştırılabilir:

bin/nutch crawl <urlDir> [-solr <solrURL>] [-depth i] [-topN N]

- <urlDir>: Çekirdek internet sayfa adreslerinin olduğu seed.txt dosyasının yer aldığı klasör
- [-solr <solrURL>]: Apache Solr ile çekilen veri indekslenecekse Solr' ın çalıştığı adres (örn. –solr http://localhost:8983/solr/)
- [-depth i]: İçerik çekme sırasında kaç seviyeye inileceği, yani ziyaret edilen bir sayfadan kaç seviye giden bağlantıların takip edileceği
- [-topN N]: Ziyaret edilen sayfadan en fazla kaç giden bağlantının işleme tabi tutulacağı

Apache Nutch aracında içerik çekme işlemi yapılırken birçok eklenti çalışmaktadır. Yeni özellik eklenmesi (Örneğin içerik çözümleme sırasında ek bilgiler elde edilmesi, indekse yeni bir alan eklenmesi vb.) istendiği takdirde de eklenti geliştirilebilmektedir. Bunun için, "Extension Points" diye adlandırılan sınıflardan, istenilen yeni özellikleri sağlayan sınıflar türetilebilir. Nutch Extension Point sınıfları aşağıda listelenmiştir.

- Nutch Indexing Filter (org.apache.nutch.indexer.IndexingFilter)
- Nutch Content Parser (org.apache.nutch.parse.Parser)
- Parse Filter (org.apache.nutch.parse.ParseFilter)
- Nutch Protocol (org.apache.nutch.protocol.Protocol)
- Nutch URL Filter (org.apache.nutch.net.URLFilter)
- Nutch URL Normalizer (org.apache.nutch.net.URLNormalizer)
- Nutch Scoring (org.apache.nutch.scoring.ScoringFilter)

Yapılan çalışmada da Parse Filter sınıfından içerik çözümleme işlemine ek özellikler kazandıran yeni bir sınıf üretilmiştir. Bu yeni sınıfın eklendiği Nutch aracıyla, ikinci adımda, içerik çözümleme aşamasından başlanarak ilk içerik çekmede elde edilen veri işlenmiştir. Bu veri Parse Filter' a yazılan eklenti ile işlenerek yeni içeriği çözümlenen Web sayfalarında dolaylı bağlantı tespiti yapılmaktadır. En son aşama olan veritabanı

güncelleme adımında ise gerçek bağlantılarla eklenen dolaylı bağlantılar ile genişletilmiş Web çizgesi elde edilmiştir.

5.2 Apache Lucene ve Solr

Apache Solr, Apache Lucene projesi temel alınarak geliştirilmiş ileri düzey bir arama platformudur. Lucene de bir arama kütüphanesi sağlar. Bu çalışmada da arama kütüphanesi olarak Lucene kullanılmıştır.

Kullanıcıların yaptıkları sorgular Apache Solr' a yönlendirilmiş, gelen sonuçlar bir Web arayüzüyle kullanıcıya sunulmuştur. Bu çalışmada 3 farklı algoritma için 3 ayrı Solr servisi kullanılmış, sorgular aynı anda bu Solr servislerine gönderilmiştir. Solr, Web arayüzünden sorguların gönderildiği ve gelen sonuçların yine Web arayüzüne gönderildiği bir arama platformu olarak kullanılmıştır.

5.3 Apache HBase

Bu çalışmada büyük veri işleme yeteneği ve Hadoop Distirubuted File System- Hadoop Dağıtılmış Dosya Sistemi üzerinde çalışabilecek bir veritabanı olması sebebiyle veritabanı olarak Apache HBase kullanılmıştır.

5.4 Apache Hadoop

Bu çalışmada büyük veri dosyalama ve büyük veri üzerinde dağıtık işlem kapasitesi yeteneği sebebiyle Hadoop kullanılmıştır.

5.5 Apache ZooKeeper

ZooKeeper, yukarıda bahsedilen dağıtık çalışan araçların koordinasyonu için kullanılmıştır.

BÖLÜM 6

YÖNTEMİN DEĞERLENDİRİLMESİ

Bu çalışmada, verilen sorguya göre Web sayfalarında benzerlik açısından değerlendirme TF-IDF tabanlı algoritma kullanılarak yapılmıştır. TF, dokümandaki kelime sıklığıdır ve bir kelimenin o dokümanda ne kadar sık geçtiğini ölçmektedir. IDF ise ters doküman sıklığıdır ve ilgili kelimenin indeksteki kaç dokümanda geçtiğini hesaplamaktadır. Sadece TF-IDF algoritmasının kullanılması, Web sayfalarının aranan sorguya göre benzerliği dikkate alınarak sıralanmasını sağlar.

Ayrıca arama esnasında dokümanları sıralamak için bir boost değeri tanımlanmaktadır. Boost değeri, aranan sorgunun Web sayfalarında geçtiği kısımlar için farklı değerler bütünü olarak tanımlanabilir. Örneğin Web sayfalarının başlık kısmında bulunan sorgular için farklı bir değer, Web sayfasının URL' inde bulunan sorgular için farklı değer verilebilir. Bu da, Web sayfalarının bağlantı tabanlı sıralamasının yapılmasını sağlamaktadır. Boost değeri TF-IDF tabanlı sorgulamada eşleşen sonuçların tekrar sıralanması için kullanılmaktadır. Bu da Web sayfalarının yapılan sorguya göre hem benzerlik hem de bağlantı tabanlı sıralanmasını sağlar. Böylece, TF-IDF ve klasik PageRank algoritması iki sıralama yöntemini de kapsamaktadır.

İyileştirilmiş popülarite tabanlı sıralama algoritmasında ise bağlantı tabanlı (PageRank) kısım için geliştirme yapılmış, TF-IDF kısmı olduğu şeklinde kullanılmıştır. Böylece iyileştirilmiş PageRank algoritması da hem benzerlik hem de bağlantı tabanlı sıralama yapmaktadır. Bu algoritma klasik PageRank (hem TF-IDF hem klasik PageRank bütünleşik olarak) ve sadece TF-IDF tabanlı yöntem ile karşılaştırılmıştır.

6.1 Kullanıcı Değerlendirme Çalışması

Algoritmaların karşılaştırılabilmesi için Türkçe içeriğin çekilmesiyle elde edilen veri kümesinde bir kullanıcı değerlendirme çalışması gerçekleştirilmiştir. 410 milyon Türkçe Web sayfası ile yapılan bu çalışma için Java Server Pages kullanılarak kullanıcı arayüzleri oluşturulmuştur.

Kullanıcı bir sorgu arattığında, sonuçları getirmek için ise Solr arama mekanizması kullanılmıştır. TF-IDF tabanlı, TF-IDF ve klasik PageRank ile geliştirilen sıralama algoritması için Solr' a 3 ayrı sorgulama yapısı kurulmuştur böylece aynı sorgu, her algoritmanın Solr servisine aynı anda gitmektedir.

Algoritmaların değerlendirilmesi için toplam 20 gönüllü çalışmaya katılmış ve toplam 60 sorgu yapılmıştır. Her gönüllü, siyaset, trafik, yds, kariyer, devlet, ekonomi, tübitak, emlak, hepsiburada vb. gibi ifadeleri, hazırlanan arayüzde kişi başı 3 adet sorgu olacak şekilde aratmıştır.

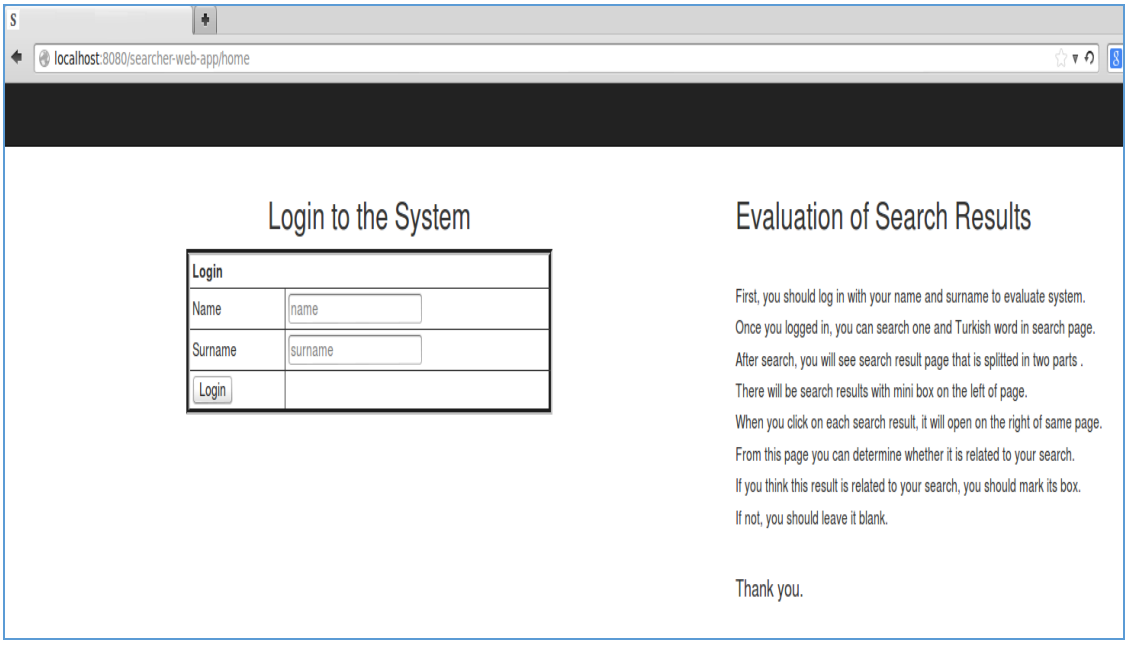
Tutturma ve bulma değerlerinde herhangi bir sapma olmaması için gönüllülere çalışmanın asıl fikrinden bahsedilmemiştir. Sadece yaptıkları aramayla ilgili olduğunu düşündükleri sonuçları belirtmelerinin beklendiği belirtilmiştir. Ayrıca 3 algoritmanın sonuçları gönüllülere karıştırılarak sunulmuştur. Kullanıcı hangi sonucun hangi algoritmaya ait olduğunu bilmemektedir.

Geliştirilen arayüzde ilk olarak gönüllülerden sisteme giriş yapmaları istenmiştir (Şekil 6.1.1). Bu her gönüllünün değerlendirmede bir kere yer almasını sağlamıştır. İkinci defa sisteme giriş yapılması engellenmiştir.

Gönüllü sisteme giriş yaptıktan sonra Şekil 6.1.2' deki gibi arama sayfasına yönlendirilmiştir. Sorgu girildikten sonra aranan ifade 3 farklı algoritmanın Solr' ına gönderilmiş, gelen sonuçlar karıştırılarak Şekil 6.1.3' deki gibi kullanıcıya sunulmuştur. Fakat her algoritmadan ilk 10 sonuç getirilmiş, toplamda 30 sonuç gösterilmiştir. Fakat 3 algoritmanın ilk 10 sonucundan herhangi biri eşleşirse sadece biri gösterilmiştir. Yani kullanıcıya gösterilen sonuç 10 ile 30 arasında değişmektedir.

Sonuç ekranında her sonucun yanında Şekil 6.1.3' deki gibi işaretleme kutusu bulunmaktadır. Gönüllü kişinin arama sonuçlarını değerlendirebilmesi için arama sonuçlarının geldiği sayfa ikiye bölünmüştür. Gönüllü kişi herhangi bir arama sonucuna tıkladığında sayfanın sağında ilgili sayfa Şekil 6.1.4' deki gibi görüntülenmektedir.

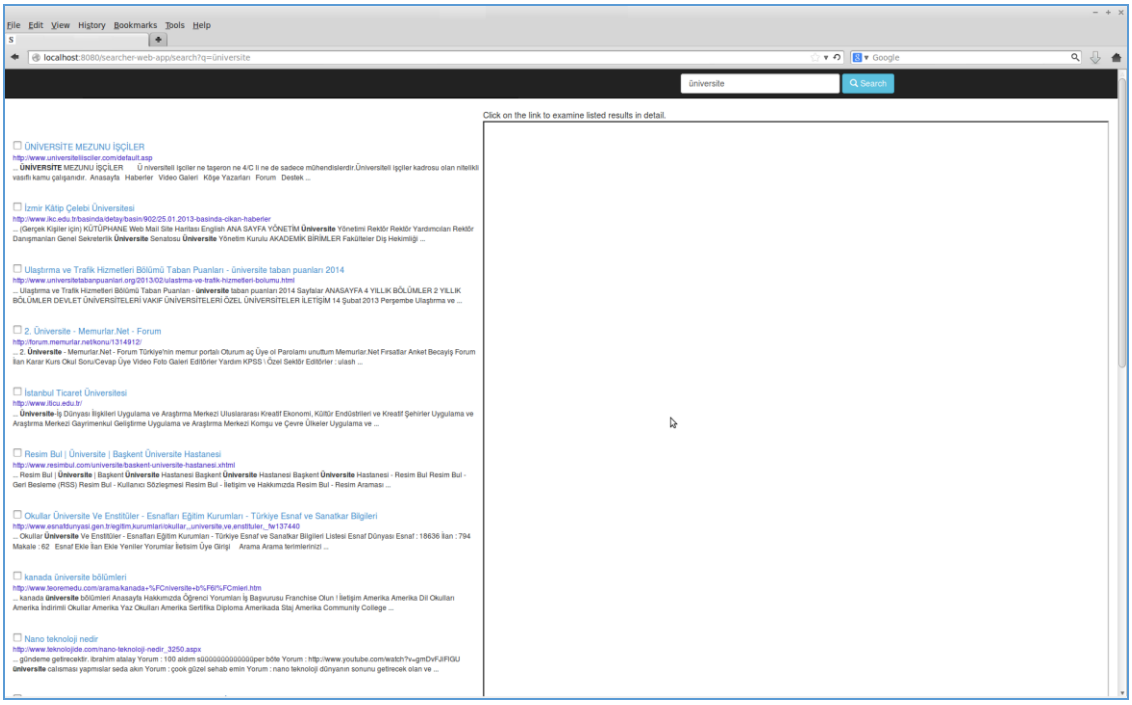
Gönüllü bu sayfaları inceledikten sonra ilgili sonucun yaptığı aramayla alakalı olup olmadığına karar vermektedir. Eğer alakalı olduğunu düşünürse sonucun solundaki işaretleme kutusunu seçmektedir. Gönüllü bu kutuyu seçtiğinde gönüllü kişinin bilgisini yanında aranan ifade, seçilen arama sonucunun URL' i, bu sonucun hangi algoritmaya ait olduğu ve ait olduğu algoritmada kaçıncı sırada olduğu kaydedilmektedir. Bu bilgi, algoritmaların karşılaştırılabilmesine olanak sağlamaktadır.



Şekil 6.1.1 Giriş sayfası



Şekil 6.1.2 Sorgu sayfası



Şekil 6.1.3 Sonuç sayfası



Şekil 6.1.4 Sonuç sayfası 2

6.2 Değerlendirme

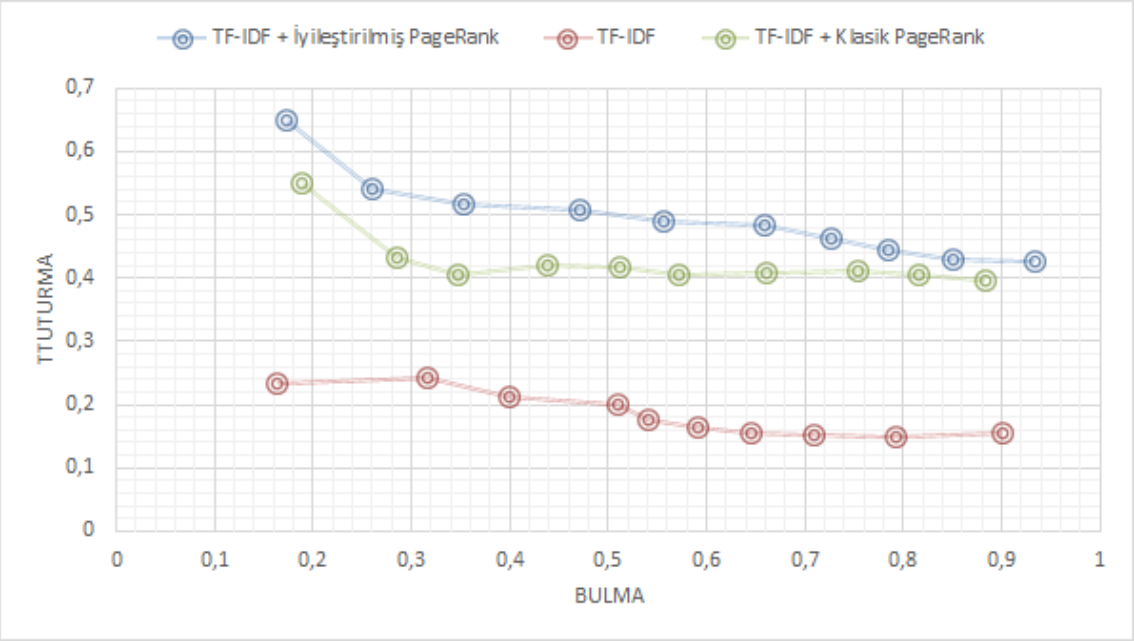
Gönüllü kişi bir sorgunun değerlendirmesini yaptığında, o sorgunun tutturma ve bulma değerleri aşağıdaki gibi hesaplanmaktadır. *h* etkeni, 3 algoritmadan elde edilen sonucun sırasını göstermektedir. *u* ise bir sonuç için gönüllü kişinin değerlendirmesini göstermektedir, 1 ya da 0 değerini almaktadır. Sonuç olarak her sıralama algoritması *r*, sorgu *q* ve *i* sırası için tutturma ve bulma değeri hesaplanmaktadır.

$$tutturma_r(i,q) = \frac{1}{i}\sum_{j=1}^{i} u(h(r,j,q)) \tag{6.2.1}$$

$$bulma_r(i,q) = \frac{1}{|h:u(h(q))=1|}\sum_{j=1}^{i} u(h(r,j,q)) \tag{6.2.2}$$

r algoritmasının *q* sorgusu için *j* sırasındaki sonucunun seçilip seçilmediğini *h(r,j,q)* göstermektedir.

3 sıralama algoritması ve 10 sonuç için Şekil 6.2.1' de tutturma ve bulma grafiği gösterilmektedir. Sonuçlar gönüllüler tarafından yapılan 60 sorgunun ortalaması alınarak hesaplanmıştır.



Şekil 6.2.1 Farklı sıralama algoritmalarının tutturma ve bulma grafiği

Şekil 6.2.1' de görüldüğü gibi PageRank tabanlı algoritmalar sade benzerlik tabanlı sıralamadan daha iyi sonuç vermiştir. Bu, ilgili sayfaları tespit etmede bağlantı analizinin katkısı düşünülürse beklenen bir sonuçtur. Şekilden görüldüğü üzere PageRank algoritmasının dolaylı bağlantılarla geliştirilmesi, bir arama motorunun sonuçlarının kalitesini artırmasını sağlayabilir. Sonuçlara dayanılarak, geliştirilen yöntemin dolaylı bağlantılar ile sayfaların tahmin edilen alakalarını ayarlayacak bir mekanizma sağladığı söylenebilmektedir.

BÖLÜM 7

SONUÇLAR VE GELECEKTE PLANLANAN ÇALIŞMALAR

Bu tez kapsamında Web' de bilgi bulma konusunda kullanılan en temel algoritmalardan birisi olan PageRank algoritmasının iyileştirilmesi gerçekleştirilmiştir. Bu algoritmanın iyileştirilmesinde kullanılacak yöntem Web' de bulunan atıfların dolaylı bağlantılar olarak değerlendirilmesi ve bu bağlantıların Web çizgesine dahil edilmesidir. Açık http bağlantılarından oluşturulan Web çizgesinin dolaylı bağlantılarla zenginleştirilmesi PageRank önem skorlarının daha doğru hesaplanmasını sağlamıştır. Yapmış olduğumuz literatür taramasında Web sayfalarındaki atıfların gizli bağlantılar olarak değerlendirilmesi ve bunların Web çizgesine dahil edilmesine yönelik yapılmış bir çalışma bulunamamıştır. Dolayısıyla bu problem daha önce çalışılmamış yeni ve önemli bir problem olarak değerlendirilmiştir.

Atıflardan oluşturulacak dolaylı bağlantıların PageRank hesaplamalarında kullanılmasının etkisinin en fazla olacağı alanlardan birisi, Web sitesi ana sayfası önem skoru hesaplamalarıdır. Bu tip sayfalara çok sayıda dolaylı bağlantının tespit edileceği, dolayısıyla yeni PageRank algoritmasıyla ana sayfa sıralamalarının daha doğru bir şekilde hesaplanacağını düşünmekteyiz. Arama motoru sıralaması online ticaret yapan firmalar için çok önemlidir. Firmalar arama motoru sonuçlarında ilk sıralarda çıkabilmek için Web sitelerini arama motoru dostu yapmakta ve bu konuya ciddi kaynaklar ayırmaktadır. Bu konuya yönelik Arama Motoru Optimizasyonu (Search Engine Optimization) adında yeni bir iş sektörü oluşmuştur. Dolayısıyla ana sayfa sıralamalarının objektif ve doğru bir şekilde yapılmasına imkân verecek yöntemlerin önemi büyüktür.

Geliştirmiş olduğumuz yeni PageRank algoritmasını bağımsız bir kullanıcı grubu tarafından değerlendiriyoruz. Bu yöntem Web araştırmalarında yeni algoritma değerlendirmesinde sıklıkla kullanılmaktadır. Biz de geliştirmiş olduğumuz dolaylı bağlantılara duyarlı, iyileştirilmiş PageRank algoritmasını test etmek için öncelikle kullanıcılar tarafından girilmiş sorgu kümeleri ile test yapıyoruz. Her sorgu arama motorunun bir alandaki sayfa bulma kapasitesini ölçecek şekilde seçilmiştir. Bu sorgu kümelerine her algoritma ile geliştirilen sistemin döndürdüğü sonuçların bağımsız denekler tarafından değerlendirilmesini sağlıyoruz.

Araştırmada Bölüm 4.5' de detaylı olarak açıkladığımız veri kümesi üzerinde bağımsız kullanıcı grubu değerlendirmesi gerçekleştirilmiştir. Yapılan değerlendirme sonucunda elde edilen sonuçlar, tez kapsamında iyileştirdiğimiz PageRank algoritmasının klasik PageRank (TF-IDF + klasik PageRank) algoritmasının doğruluk yüzdesini artırdığı görülmektedir.

7.1 Araştırma Sorularına Cevaplar

Bölüm 3.1' de ortaya konulan araştırma soruları, Bölüm 6' da yapılan değerlendirmeler sonucunda bulunan bulgular ile takip eden kısımlarda yanıtlanmaktadır.

7.1.1 Dolaylı bağlantılar popülarite tabanlı algoritmaların geliştirilmesinde kullanılabilir mi? Dolaylı bağlantılar algoritma geliştirilmesinde yeterli midir?

Geliştirilen algoritma ile dolaylı bağlantıların popülarite tabanlı algoritmaların geliştirilmesinde kullanılabileceği kanıtlanmaktadır. Bölüm 6.2' de yapılan değerlendirmeler sonucunda dolaylı bağlantıların PageRank algoritmasını iyileştirdiği görülmektedir.

7.1.2 Dolaylı bağlantılar kullanılarak doğruluk değerini artırmak için yöntem ne olabilir?

Bölüm 6' da belirtilen sonuçlar doğrultusunda, dolaylı bağlantılar kullanılarak ilgili Web sayfalarına giden bağlantı sayısı artırılarak o sayfanın popülaritesi artırılabilmektedir. Bunun nasıl yapılacağı ayrıntılı olarak Bölüm 4' de ayrıntılı olarak anlatılmıştır.

7.1.3 Web sayfasının içinde dolaylı bağlantı tespiti nasıl yapılabilir?

Dolaylı bağlantı listesini oluşturmak için kullanılacak farklı yöntemler bulunmaktadır. Bir atıf metninin bir hedef sayfası için dolaylı bağlantı olabilmesinin temel kuralı, o atıf metninin bağlantılarda hedef sayfa için sıklıkla kullanılması ve diğer sayfalar için ya nadiren kullanılması veya hiç kullanılmamasıdır. Bir atıf metninin sadece tek bir URL adresi için sıklıkla kullanıldığı tespit edildiğinde, o atıf metni o sayfa için dolaylı bağlantı olarak seçilecektir. Örneğin "Yıldız Teknik Üniversitesi" atıf metninin çoğunlukla "http://www.yildiz.edu.tr" adresi için kullanıldığı tespit edildiğinde, "Yıldız Teknik Üniversitesi" atıf metni "http:// www.yildiz.edu.tr" adresi için muhtemel bir dolaylı ya da gizli bağlantı olarak seçilecektir. Diğer yöntem ise Web sayfalarının başlıklarının tespit edilmesi ve bu başlıkların o sayfalar için muhtemel gizli bağlantı olarak seçilmesidir. Birçok Web sayfasında gizli bağlantı ifadesi olabilecek başlıklar vardır. Örneğin kurumsal sitelerin çoğu kurum adını ana sayfa başlığı olarak kullanmaktadır. Bu metinler gizli bağlantı metni olabilecek ifadelerdir. Web sayfalarının başlıkları Web sayfası içeriklerinin incelenmesi ve işlenmesi sonucunda tespit edilecektir. Web sayfalarında başlıklar bazı durumda HTML belgesinin <title> kısmında olsa da, diğer birçok durumda Web sayfası içeriğinde bulunmaktadır.

Bu tez kapsamında dolaylı bağlantıların tespiti için bağlantılarda yer alan atıf metinleri kullanılmıştır. Bağlantılardaki atıf metin bilgisi yeni çekilen Web sayfası içeriğinde aratılıp belirli bir sıklıkta tekrarlanan eşleşme bulunduğunda ilgili atıf metninin Web sayfasının giden bağlantı listesine dolaylı bağlantı olarak eklenmektedir. Bağlantılardaki atıf metinlerinin aratılmasıyla ilgili ayrıntılar Bölüm 4' de anlatılmıştır. Bölüm 6' da detayları verilen değerlendirmeler sonucu geliştirilen algoritmanın TF-IDF + klasik PageRank ve TF-IDF algoritmalarından daha doğru sonuçlar verdiği görülmektedir.

7.1.4 Web sayfasının içinden elde edilen dolaylı bağlantılarla geliştirilmiş sıralama algoritması klasik PageRank ya da TF-IDF algoritmalarıyla karşılaştırıldığında elde edilen sonuçlar nedir?

TF-IDF tabanlı yöntem, aranılan sorgunun indekslenen dokümanlardaki sıklığına bakmaktadır. Bu yöntemde bağlantı tabanlı herhangi bir uygulama bulunmamaktadır. Fakat PageRank algoritmasında TF-IDF' in yanında Web sayfalarının birbirleriyle olan

bağlantıları da incelenmektedir. Bu da aranılan sorgu için daha alakalı sayfaların gelmesini sağlamaktadır. Geliştirilen yöntem de ise Web sayfalarının içeriklerinde geçen ifadeler de PageRank hesaplamasına katılmaktadır. Kullanıcının arattığı ifade için bu içeriklerde geçen ifadeler de değerlendirmeye katılmakta ve sonuçlar buna göre iyileştirilmektedir. Bölüm 6' daki sonuçlar göstermiştir ki; bu tezde önerilen yöntemin sonuçları diğer yöntemlerin sonuçlarından daha iyidir.

7.2 Gelecekte Yapılacak Çalışmalar

Gelecek çalışma olarak dolaylı bağlantılara farklı ağırlıklar verilmesi, Web sayfa başlığı gibi farklı dolaylı bağlantı kaynaklarının kullanılması planlanmaktadır. Ayrıca yapılan kullanıcı değerlendirme çalışmasının daha fazla gönüllüyle ve daha fazla sorguyla genişletilerek sonuçların doğrulanması planlanmaktadır.

KAYNAKLAR

[1] Liu, B., (2011). Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data by Bing Liu, Springer, Second Edition, Springer.

[2] Chip Online, Google günlük 20 Petabyte veri işliyor, http://www.chip.com.tr/haber/google-gunluk-20-petabyte-veri-isliyor_5368.html, 18 Kasım 2014.

[3] Salton, G., (1989). Automatic text processing: the transformation, analysis and retrieval of information by computer, Addison-Wesley.

[4] Justin, B., Dayne, F. ve Thorsten J., (1996). "A machine learning architecture for optimizing web search engines", In Proceedings of the AAAI Workshop on Internet-Based Information Systems, 5 August 1996, Oregon.

[5] Sergey, B. ve Lawrence, P. (1998). "The anatomy of a large-scale hypertextual web search engine", In Seventh International World Wide Web Conference, 14-18 April 1998, Brisbane.

[6] Brian, D., D., Apostolos, G., Konstantinos, K., Yingfang, .L, Hyun-ju, S., Wei, W. ve Baohua, W., (1999). "DiscoWeb: Applying link analysis to web search", In Proceedings of the Eighth International World Wide Web Conference, 11-14 May 1999, Toronto.

[7] Jon, M. K., (1998). "Authoritative sources in a hyperlinked environment", In Proceedings ACM-SIAM Symposium on Discrete Algorithms, 25–27 January 1998, San Francisco, 668–677.

[8] Steve, L. ve Giles, C. L., (1998). "Context and page analysis for improved web search", IEEE Internet Computing, 2(4):38–46.

[9] Barry, C. L., (1993). The Identification of User Criteria of Relevance and Document Characteristics: Beyond the Topical Approach to Information Retrieval. Doktora Tezi, Syracuse University, New York.

[10] Mizzaro, S., (1997). "Relevance: The whole history", Journal of the American Society for Information Science, 48(9):810–832.

[11] Krishna, B. ve Monika, R.H. (1998). "Improved Algorithms for Topic Distillation in a Hyperlinked Environment", ACM-SIGIR, 24-28 August 1998, Melbourne.

[12] Soumen, C., Byron, D., David, G., Jon, K., Prabhakar, R. ve Sridhar, R., (1998). "Automatic Resource Compilation by Analyzing Hyperlink Structure and

Associated Text", Proc. Seventh Int'l World Wide Web Conf., 18-18 April 1998, Brisbane.

[13] Lawrence, P., Sergey, B., Motwani, R. ve Winograd, T., (1998). "The PageRank Citation Ranking: Bringing Order to the Web", Stanford Digital Libraries Working Paper, Stanford InfoLab.

[14] Haveliwala, T.H., (2003). "Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search", IEEE Transactions on Knowledge and Data Engineering.

[15] Wei, H., ve Bin, L., (2011). "An Improved Method for the Computation of PageRank", 2011 International Conference on Mechatronic Science, Electric Engineering and Computer, 19-22 August 2011, Jilin.

[16] Decai, H., Huachun, Q., Yuan, Y. ve Yue-feng, Z., (2006). "TC-PageRank Algorithm Based on Topic Correlation", Proceedings of the 6th World Congress on Intelligent Control and Automation, 21-23 June 2006, Dalian.

[17] Chia-Chen, Y., Jih-Shih, H., (2009). "Pagerank Algorithm Improvement by Page Relevance Measurement", FUZZ-IEEE 2009, 20-24 August 2009, ICC Jeju.

[18] Apache Tika, http://tika.apache.org/1.4/detection.html, 11 Kasım 2014.

[19] Language Detection API, http://detectlanguage.com, 11 Kasım 2014.

[20] Fast text language identification library in Java, http://code.google.com/p/lang-id, 11 Kasım 2014.

[21] Canan, G., Hayati, G., Fatma, C.P.M., Şeniz, D., Ahmet, A.A. ve Murat, O., (2014). "Büyük Veride Dil Odaklı İçerik Çekme", SIU 2014, 23-25 Nisan, Trabzon.

[22] Mehmet, A., Mehmet, A.N. ve Filippo, M., (2004). "Using hyperlink features to personalize web search, WebKDD 2004, Seattle.

[23] Aktas, M., Nacar, M. ve Menczer, F., (2006). Using Hyperlink Features to Personalize Web Search, Advances in Web Mining and Web Usage Analysis, Springer, 104-115.

[24] Ahu, S., Bamshad, M. ve Robin, B., (2007). "Web search personalization with Ontological User Profiles", Proc. 16th ACM Conf. Information and Knowledge Management (CIKM'07), 6-10 November 2007, Lisbon, 525-534.

[25] Web Site: Google's personalized search web site: http://www.google.com/psearch, 23 Ekim 2014.

[26] Nutch Wiki, http://wiki.apache.org/nutch/Nutch2Crawling, 15 Kasım 2014.

[27] Rappoport, R., (2012), Search Engine Watch, Anatomy of a Search Engine: Inside Google, http://searchenginewatch.com/article/2064446/Anatomy-of-a-SearchEngine-Inside-Google, 12 Ekim 2012.

[28] Deutsch, P., (2000). "Archie—A Darwinian Development Process", IEEE Internet computing, 69-71.

[29] Zhang, D. ve Dong, Y., (2003).“Semantic, hierarchical, online clustering of web search results”, APWeb’04, 14-17 April 2004, Hamhzou, 69–78.

[30] Wikipedia, User:Oscar.nierstrasz/W3Catalog, http://en.wikipedia.org/wiki/User:Oscar.nierstrasz/W3Catalog, 10 Kasım 2014.

[31] Wandex, http://encyc.org/wiki/Wandex, 26 Kasım 2014.

[32] Wikipedia, Aliweb, http://en.wikipedia.org/wiki/Aliweb, 10 Kasım 2014.

[33] Wikipedia, JumpStat’on\ http://en.wikipedia.org/wiki/JumpStation\, 10 Kasım 2014.

[34] Pinkerton, B., (1994). “Finding what people want: Experiences with the WebCrawler”, In Proceedings of the First World Wide Web Conference, 25-27 May 1994, Geneva.

[35] Wikipedia, History of Yahoo, http://en.wikipedia.org/wiki/History_of_Yahoo!, 15 Kasım 2014.

[36] Yandex, http://sirket.yandex.com.tr/general_info/history.xml, 15 Kasım 2014.

[37] Brandt, Starting Up. How Google got its groove, https://alumni.stanford.edu/get/page/magazine/article/?article_id=35436, 14 Ekim 2014.

[38] Baidu, The Baidu Story, http://ir.baidu.com/phoenix.zhtml?c=188488&p=irol-homeprofile, 14 Ekim 2014.

[39] Redmond, W., (2009), Microsoft, Microsoft’s New Search at Bing.com Helps People Make Better Decisions: Decision Engine goes beyond search to help customers deal with information overload, 28 May 2009.

[40] Nutch Wiki, http://wiki.apache.org/nutch/NutchTutorial, 18 Kasım 2014.

[41] Buettcher, S., Clarke, C.L.A. ve Cormack, G.V., (2010). Information Retrieval: Implementing and Evaluating Search Engines, MIT Press.

[42] Baeza-Yates, R. ve Ribeiro-Neto, B., (2011). Modern Information Retrieval: The Concepts and Technology Behind Search, Second Edition, Addison-Wesley.

[43] Croft, B., Metzler, D. ve Strohman, T., (2010). “Search Engines: Information Retrieval in Practice”, First Edition, Addison-Wesley.

[44] Doug Cutting, http://en.wikipedia.org/wiki/Doug_Cutting, 18 Kasım 2014.

[45] McCandless, M., Hatcher, E. ve Gospodnetić O., (2010). Lucene In Action, Second Edition, Manning.

[46] Solr Wiki, http://wiki.apache.org/solr/PublicServers, 18 Kasım 2014.

[47] Warden, P., (2011). Big Data Glossary, NoSQL Databases, Chapter 2, O'reilly.

[48] Sanjay, G., Howard, G. ve Shun-Tak, L., (2003). “The Google File System”, SOSP’03, October 19–22, 2003, New York.

[49] Jeffrey, D. ve Sanjay G., (2004). “MapReduce: Simplified Data Processing on Large Clusters”, OSDI, 5 December 2004, San Francisco.

[50] White, T., (2012). Hadoop - The Definitive Guide, Third Edition, O'reilly.

[51] Holmes, A., (2012). Hadoop in Practice, Second Edition, Manning.

[52] Apache ZooKeeper Wiki, https://cwiki.apache.org/confluence/display/ZOOKEEPER/Index, 18 Kasım 2014.

[53] White, T., (2012). Hadoop - The Definitive Guide, Zookeeper, Chapter 14, Third Edition, O'reilly.

[54] Zemberek, http://code.google.com/p/zemberek , 18 Kasım 2014.

[55] DMOZ, http://www.dmoz.org/World/T%C3%BCrk%C3%A7e/, 18 Kasım 2014.

[56] Search and NLP for CJK, Understanding the columns/fields in Nutch 2.0 Webpage, http://nlp.solutions.asia/?p=232, 20 Kasım 2014.

[57] Nutch Wiki, http://wiki.apache.org/nutch/CommandLineOptions, 20 Kasım 2014.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Sinem KIRTAÇ

Doğum Tarihi ve Yeri : 23.09.1986 – Kocaeli

Yabancı Dili : İngilizce

E-posta : sinem.aykanat@tubitak.gov.tr

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Bilgisayar Mühendisliği	İstanbul Kültür Üniversitesi (ÇAP)	2009
Lisans	Endüstri Mühendisliği	İstanbul Kültür Üniversitesi	2008
Lise	Fen Bilimleri	24 Kasım Anadolu Lisesi	2004

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2010	Tübitak	Test Mühendisi

YAYINLARI

Bildiri

1. Sefer, E. Aykanat, S., “An Evaluation of Client-Side Dependencies of Search Engines by Load Testing”, The Fifth International Conference on Advances in System Testing and Validation Lifecycle, VALID, 2013.

2. Özalp, N. Aykanat, S. Erdem, Z. Ayan, U., “Improved Naïve Bayesian Algorithm for Author Detection, SIU, 2013.