T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAMA MOTORLARINDA KULLANILAN ARAMA ROBOTU MİMARİLERİNİN İNCELENMESİ VE YENİ BİR YAKLAŞIM SUNULMASI

EYÜP CAN DÜNDAR
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
Danışman : Aydın CARUS
2009
EDİRNE

T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAMA MOTORLARINDA KULLANILAN ARAMA ROBOTU MİMARİLERİNİN İNCELENMESİ VE YENİ BİR YAKLAŞIM SUNULMASI

Eyüp Can DÜNDAR

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez 15/01/2010 tarihinde aşağıdaki juri tarafından kabul edilmiştir.

Yrd.Doç.Dr. Aydın CARUS
Danışman

Yrd.Doç.Dr. Nurşen SUÇSUZ
Üye

Yrd. Doç. Dr. Hilmi KUŞÇU
Üye

ÖZET

Yüksek Lisans Tezi, Arama Motorlarında Kullanılan Arama Robotu Mimarilerinin İncelenmesi ve Yeni Bir Yaklaşımın Sunulması, T.C. Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı

Internet kullanımının yaygınlaşması ile birlikte, Web üzerinde yayımlanan bilgiler içerisinde doğru bilgiye ulaşmak için arama motorları geliştirilmiştir. Arama motorlarının etkin bilgiye erişebilmesi için farklı metotlar mevcuttur. Arama motorlarının sayfaları taraması için geliştirilen metotlar incelenmekte ve karşılaştırma yapılabilmektedir.

Bu tez 2009 yılında yapılmıştır ve 111 sayfadan oluşmaktadır.

ANAHTAR KELİMELER : Crawler, Web Robotları, Cluster, HTML, XML, Arama Motoru

ABSTRACT

Graduate Thesis, Search Engine Structure And Web Robots, T.C. Trakya University, Graduate School Of Natural And Applied Sciences, Department Of Computer Engineering

The World Wide Web has become the biggest digital library. With the introduction of HTML, the web has become the largest accessible store of information. Search engines use crawlers to create a copy of the visited pages for later processing that will index the downloaded pages to provide fast searches. In this work, the differences of web robots, their performance and similarities are given.

This work is done in 2009 and consist 111 Pages.

KEYWORDS : Crawler, Web Robots, Cluster, HTML, XML, Search Engine

İÇİNDEKİLER

1. Giriş

Bilgi, "subje (insan) ile obje (nesne) arasındaki ilişkiden çıkan bir sonuç" olarak tanımlanmıştır. İnsanoğlunun gelişim sürecinde bilginin etkinliği en önemli faktörlerden biridir. Bireyler, değerli bilginin paylaşımını sağladığı dönemde, medeniyet sürecinde önemli ilerlemeler kaydedilmiştir.

"Sanayi Devrimi"nde üretim araçları ile insanın emek gücü bütünleşerek insanların ihtiyaçlarını karşılayacak ürün ve hizmetlerin üretimi artmıştır. Emek gücü, akıl ve bilgiden yoksun olduğu sürece doğru ve verimli bir şekilde kullanılması mümkün olmamaktadır. Aklı kullanmak emeği mekanikleşmekten uzaklaştırır. Günümüzde birçok makine bizim yerimize mekanikleşerek, ta ki fişi çekilinceye kadar veya kendi yazılımları ihtiyacı karşılayamaz hâle gelinceye kadar çalışırlar. Bugün gelinen durumda insanlık, her alanda bilgi, bilişim ve iletişim teknolojilerinden, yenilikçi fikirlerden yaralanarak geçmişe göre, medeniyet yolunda daha hızlı yol almaktadır.

İnternetin hayatımızda yer almaya başlaması ile birlikte, insanoğlunun istediği bilgiye hızlı bir şekilde erişebilmesi mümkün hale gelmiştir. Bilgisayarların icadından bugüne oldukça büyük bir değişim geçirdikleri aşikâr, eskiden oda büyüklüğündeki bilgisayarlar kullanılırken, bugün artık ceplerimize sığmaktadır. İnsanoğlu, istediği bilgiye cebinde taşıdığı bilgisayar veya cep telefonu sayesinde hızlı ve kolayca ulaşabilmektedir.

1995 yılında 16 milyon olan küresel internet kullanıcısının günümüzde 1.407 milyar olduğu tahmin edilmektedir. Yaklaşık olarak dünya nüfusunun %21,1'ine denk gelmektedir ki bu da her 5 kişiden birinin interneti kullandığını göstermektedir. Bilginin internet üzerinde paylaşımının yaygınlaşması ile birlikte, internet üzerindeki verinin büyüklüğü artmıştır. Mevcut durumda internet üzerinde herhangi bir organize yapı ya da indeksleme metodu olmadığından dolayı internet üzerindeki verinin boyutu tahmin edilememektedir. [1] İnternetin yaygınlaşması ile birlikte bilgiye ulaşım ilk icadındaki durumuna göre zorlaşmıştır, internet üzerindeki bilgiye kolayca ulaşabilmek için arama motorları geliştirilmiştir. Arama motorları sayesinde bilginin hangi kaynakta olduğunun tespit edilerek bilgiye ulaşmak son yıllarda yaygın olarak kullanılan yöntemdir.

2. Arama Motorları

Arama motoru, internet üzerinde bulunan içeriği aramak için kullanılan bir araçtır. Dünyadaki hemen hemen tüm web sitelerinin listelendiği, kategorize edilebilen, aradığımız bilgilere hızlı bir şekilde ulaşmamızı sağlayan web siteleridir. Arama motoru bilgiye erişme sistemi olarak ta tanımlanabilir. Bir arama motoru temelde üç ana bileşenden oluşmaktadır: "Web Robotu", "Arama İndeksi" ve "Kullanıcı Arabirimi".

Internet üzerinde birçok arama motoru bulunmaktadır. Bazıları yerel bazda hizmet verirken bazı arama motorları da belli bir konu üzerinde oluşturulmuş dizin bazında aramaya izin vermektedir.

Şu anda en çok bilinen ve kullanılan arama motorları google.com, yahoo.com, live.com'dur. Bunların yanında internet üzerinde hizmet veren birçok arama motorları da aynı işlevi yerine getirmektedir. Arama motorları, web robotu (crawler) yazılımlarla web sayfalarının incelerler ve içerisinde bulunan kelimeleri, cümleleri, resimleri vb. içeriği kendi veritabanları üzerine kaydederek bunlardan indeks oluştururlar. Bu işlemi belirli aralıklarda sürekli yaparak güncel bir indekse sahip olurlar. Kullanıcılar ise istemci web arabirimini kullanarak veya izin verilen başka erişim yöntemleri ile arama motorlarının indekslerinden anahtar kelimeleri veya bir cümleyi girerek aradıkları bilgilerin nerede olduğunu tespit ederler.

Arama motorlarının indeksleme mantığı kelimelerle birlikte, sitenin tümünün değerlendirilmesiyle yapılması durumunda kelimelerin anlamlarının belirli bir oranda değerlendirilmesi sağlanılarak kullanıcılara aradıkları bilgiye daha kolay ulaşılabilmesi sağlanır.

2.1 Arama Motorunun Tarihçesi

İnternetin resmen ilk kullanılmaya başlandığı 1970 yılında, mevcut web sunucularının listesinin sistemde saklanması sayesinde aranan bilgiye erişim sağlanmaktaydı. Bu liste Tim Berners Lee tarafından hazırlanıp CERN sunucuları üzerine kaydediliyordu. İnternetin yaygınlaşması ve web sunucularının sayısının artması ile birlikte bu liste takip edilemez hale geldi, NCSA (National Center for

Supercomputing Applications) sitesinde sadece yeni web sunucularının isimleri "What's New" bölümünde gösteriliyordu. [2]

Internet üzerinde bilginin aranması için kullanılan ilk araç "Archie"dir. [3] 1990 yılında Alan Emtage tarafından geliştirilen Archie, FTP sitelerindeki erişilebilen klasörlerin ve dosyaların listesini getiriyordu. Archie, listeyi kullanıcılara sağlıyorken, bu siteler üzerindeki herhangi bir içeriği indeksleme işlemi gerçekleştirmiyordu.

Minnesota Üniversitesi'nde Mark McCahill'in 1991 yılında Gopher'ı ortaya çıkarması ile birlikte Veronica ve Jughead adında iki arama programı doğdu. Arcihe'de olduğu gibi bu iki arama programı Gopher Index Sistemi'ndeki dosya isimlerini ve klasörlerini arayarak Gopher listelerini oluşturuyordu. Veronica, bu Gopher listelerinde kelime aramasını sağlıyordu. Jughead ise Gopher sunucularındaki menü bilgilerini bulmaya yarayan bir araçtı. [4]

1993 yılında Matthew Gray, bilinen ilk web robotunu geliştirdi. Perl üzerinde geliştirilen "Word Wide Web Waderer", "Wandex" indeksini oluşturmak için kullanılıyordu. "Wandex"in temel amacı internetin büyüklüğünün belirlenmesiydi. 1995 yılına kadar Wandex bu işlemi başarıyla gerçekleştirdi. [5]

Kasım 1993'te Aliweb yayınlandı, Aliweb herhangi bir web robotu kullanmıyordu, bunun yerine belirli bir formatta ilgili web sitesinin yöneticisi tarafından indeks dosyasına kayıt yapılması gerekiyordu.

Jumpstation arama motoru Aralık 1993'te geliştirildi. Jumpstation web robotu kullanarak sayfaları indeksliyor ve bir sorgu sayfası üzerinden arama yapılabilmesi sağlanıyordu. Böylece arama motorlarının temel üç özelliğini ("Web Robotu", "Arama İndeksi", "Kullanıcı Arabirimi") kullanan ilk arama motoru ortaya çıkmış oluyordu. Jumpstation'ın çalışmış olduğu platform üzerindeki kaynak yetersizliğinden dolayı, web robotunun ulaşmış olduğu sayfaların sadece başlıkları indekslenebiliyordu.

Crawler tabanlı full-text tarama özelliği bulunan ilk arama motoru 1994 yılında ortaya çıkan WebCrawler'dır. Daha önceki uyarlamalarına göre web sayfası üzerindeki tüm kelimeler üzerinde arama yapabilme imkânı sağlıyordu. Bu özellik zamanla arama motorlarının en temel standardı olarak gerçekleşmiştir. Genel olarak internet üzerinde kullanıma açılan ilk arama motoruydu. Aynı zamanda 1994 yılında Carnegie Mellon Üniversitesi'nde yayımlanan Lycos ilk ticari olarak ortaya çıkan arama motorudur.

1994 yılından hemen sonra, arka arkaya birçok arama motoru ortaya çıktı ve popüler olmak için adeta birbirleriyle yarışıyorlardı: Magellan, Excite, Infoseek, Inktomi, Northern Light, AltaVista ve Yahoo. Ancak bunların içinde bulunan, David Filo ve Jerry Yang'ın kurduğu Yahoo, insanların ilgisini diğerlerinden daha çok çekerek bilgiye ulaşmanın en popüler yolu olarak kullanılan arama motorları arasında yer aldı. Yahoo arama işlemini web sayfalarının full-text kopyalarından ziyade, Yahoo'nun kendi içerisinde bulunun web dizininde gerçekleştiriyordu. Kelime bazlı aramanın yanı sıra, kullanıcılara konu dizinleri içerisindeki listelere ulaşma imkânı sağlıyordu.

1998 yılında Google'u kuran Larry Page ve Sergey Brin, Pagerank [7] teknolojisini satmak isteseler de alıcı olmadığı için bunu başaramadılar. Internet ağındaki sayfaları puanlayan bu sistem, o sayfaya ne kadar çok link verildiyse ve link verenlerin puanı ne kadar çoksa, o sayfaya yüksek puan verme mantığından oluşuyordu. Daha sonra bu sistemi satamayınca 1999 yılında "Google Search"ü kurdular.

Hitbox'a göre Google, 2008 yılında dünya çapında % 82.7 oranında bir popülerliğe sahip durumda olmasına rağmen 2009 yılında bu oranın %78.4'e düştüğü görülüyor. Google'un 2009 yılında popülerliğinin düşmesinde en önemli etken Çin'de yerel arama motoru Baidu'nun kullanılmasının yaygınlaşmasıdır. [8]

2.2 Arama Motorunun Özellikleri

Internet üzerinde veriye kullanıcıların kolay bir şekilde ulaşmasını sağlayan arama motorlarının üç temel bileşeni vardır: "Web Robotu", "Arama İndeksi" ve "Kullanıcı Arayüzü". Arama motorları internetin yaygınlaşması ile birlikte, yapısal olarak bazı değişikliklere uğramıştır. İlk önceleri kullanıcının aradığı kelime anlık olarak internet üzerinden sorgulanarak kullanıcılara cevap dönülürken, günümüzde arama motorlarının etkin bir veritabanı indeksleme yöntemi bulunmaktadır. Web robotlarının taradığı sayfalar, arama indeksine kaydedilerek kullanıcılara hızlı bir şekilde sonuç döndürülmesi sağlanmaktadır.

2.2.1 Web Robotu

Internet üzerindeki web sayfalarını otomatik olarak tarayan yazılımlara verilen genel addır. Verilen ilk sayfadan başlayarak belirli bir algoritma ile web sayfalarını tarayarak, arama indeksine kaydedilebilmesi için web sayfalarını tarar. Web robotunun iyi optimize edilmiş bir yapısı ve etkin bir algoritması olması gerekmektedir. Google web robotu olarak Googlebot'u kullanırken, Altavista'nın web robotunun adı da "Scooter"dır.

2.2.2 Arama İndeksi

Web Robotu tarafından taranıp kaydedilen web sayfalarını, arama motorunun veritabanına belirli bir algoritma ile kaydedilmesini sağlar. Kullanıcıların yaptıkları sorgulamalarda aradıkları web sitelerine en kısa zamanda ulaşabilmesi için indeksleyicinin algoritması büyük önem arz etmektedir. Web sayfalarının en etkin bulunacak şekilde indekslenmesini amaçlar. Arama sonuçlarının kelime üzerinden yapılarak, sitenin tümünün değerlendirilmesinin sağlanması amacıyla farklı web robotu yaklaşımları geliştirilmiştir. "Subject – Focussed Crawler" , "Subject – Specific Crawler" gibi.

2.2.3 Kullanıcı Arabirimi

Arama motorunun veritabanı üzerinden, indekslenmiş verinin kullanıcılar tarafından sorgulanabilmesi sağlar.

3. Web Robotları

Internet üzerinde bulunan büyük boyuttaki verilerden birçok bilgiye ulaşabilmemiz mümkündür. Günün haberlerine, beklediğimiz bir kargonun şu anda nerede olduğuna, banka hesaplarımıza, almak bir istediğimiz ürünün fiyatına ve araştırmak istediğimiz konu gibi. Bütün bunları yapmak için bilgisayarımızı açıp internet tarayıcımızın üzerine mouse ile tıklamamız yeterlidir.

Internet üzerindeki bilgiler insanların kullanımı için hazırlanmıştır. Web siteleri onları ziyaret edecek kullanıcılara açıktır. Peki, web siteleri sadece insanlar tarafından web tarayıcıları vasıtasıyla mı erişilirler? Aslında, tüm web sitelerine erişim sağlayan kullanıcıların tamamı bir bilgisayar programıdır. Kullandığımız web tarayıcıda temelde sayfaların görüntülenmesini sağlayan bir bilgisayar programıdır. Bunun yanı sıra web robotları da sayfaları tarayan diğer programlardır. Web robotları sayfaları tarayarak, indeksleyicinin arama motoruna gerekli verinin indekslenmesi için sayfaları kaydederler.

Web robotları, kullanıcının verdiği ilk adresten başlayarak site üzerindeki linkleri takip ederek, web sitelerini arama motorunun indekslemesi için kaydeder. Bu işleme kullanıcının vermiş olduğu maksimum sayfa sayısına erişinceye kadar veya tüm linkler taranana kadar devam eder.

Web robotları arama motorları tarafından önceden beri kullanılmaktadır. Arama motorları kullanıcının verdiği kelimeleri içeren web sitelerini kullanıcıya listeler. İlk başlarda bu işlemi gerçekleştirmek için, kullanıcının verdiği kelimeyi, bir yandan web robotları ile siteleri tararken diğer tarafta sonuç vermek için kullanıcı bekletilirdi. [9] Belli bir süre sonra WWW (World Wide Web) kullanımının artmasıyla birlikte büyüyen veri karşısında bu yöntem etkin sonuç vermez hâle geldi. Günümüzde arama motorları web robotları vasıtasıyla taradıkları siteden geçen bilgileri kendi veritabanları üzerine kaydedip, sorgulama işlemlerine bu veritabanı üzerinden yanıt verirler.

Web robotları, web madenciliği başta olmak üzere birçok konuda kullanılabilmektedir. Web robotlarının kullanım alanlarından biride "Site Haritası" oluşturulmasının sağlanmasıdır. Web robotu verilen sitenin ana sayfasından başlayarak bu sayfa üzerindeki linkleri takip edip, "Site Haritası"nın belirlenmesini sağlar. Ayrıca web robotları site üzerindeki yazım hatalarını ve hatalı linkleri bulmak içinde kullanılmaktadır.

İnternet üzerinden birkaç sayfayı bir saniye içinde indirmek kolay olmasına rağmen, sağlam, mükemmel, G/Ç ağ yapılandırmasını denetleyen, birkaç hafta içinde

milyonlarca sayfayı tarayıp, büyük boyuttaki veriyi depolayacak yüksek performanslı bir web robotu tasarlamak oldukça zordur.

Web robotu işleme başlarken, kullanıcı tarafından programa bir veya daha fazla başlangıç sitesi verilir. Verilen bu sayfalar "çekirdek sayfalar" olarak tanımlanırlar. Web robotu, indirdiği sayfanın bir kopyasını depolama birimine kaydeder.

Gelen sayfanın html kodunu yorumlayarak, sayfaya üzerindeki linkleri bulur. Her web robotu kendi içinde bir "Html Parser" içerir. Gelen linkler üzerinde, farklı stratejilere göre, bir sonraki linki seçer. Web robotları url'leri belirlerken "Url Düzenleme" işlemini gerçekleştirirler. Link-based crawler, sayfa üzerindeki bütün linkleri tek tek gezerken, Konu odaklı web robotu , <a href …> tagının yazısında veya linkinde verilen kelimelerden herhangi biri varsa ilerler. Ek A'da web robotu geliştirilmesinde kullandığımız "Html düzenleyici" kod düzeneği bulunmaktadır.

Web robotlarının sürekli olarak sayfaları taraması, sayfaların bulunduğu sunuculardaki yoğunluğu arttırmakla beraber, crawlerlar büyük hacimli disklere ihtiyaç duymaktadırlar. Ayrıca internet üzerinde sayfaların sürekli güncellenmeler olduğundan, yeni bir sayfa eklenebileceği gibi, mevcut bir sayfanın yayından kaldırılabilmesi mümkün olduğundan web robotlarının daha verimli çalışabilmesi için belirli ilkeler belirlenmiştir.

“Selection Policy” : Web robotlarının hangi sayfaları indireceği belirlenir. Web robotu stratejileri bu ilke dâhilindedir.

“Re-visit Policy” : Sayfa üzerindeki değişikliklerin ne kadar zaman aralıklarında yapılacağını belirler.

“Politeness Policy” : Web robotu, bir insanın o sayfayı ziyaret etmesine ziyade, server üzerinde çok hızlı işlem yapabilmektedir. Bu yüzden “Robot Protokolü” geliştirilmiştir.

“Parallelization Policy” : Web robotu, çoklu işlemci yapısında çalıştıkları için, crawlerların aynı sayfayı tekrar indirmemesini belirten ilkedir. [10]

Yüksek performanslı crawler tasarlamanın ilk adımı, iyi bir strateji tanımlamaktır. Web robotlarını arama motorlarının merkezidirler.

3.1 Tarama İşleminde Kullanılan Araçlar

Basit olarak düşündüğümüzde web robotlarının sayfalara erişim sağlayarak, internet üzerindeki veriyi topladığını düşünebiliriz. Fakat web robotu bir kişinin internet tarayıcısı ile sayfayı ziyaret etmesinden çok daha hızlı bir şekilde işlemlerini gerçekleştirebilmektedir. Diğer tarafta web sayfalarını geliştiren webmasterlarında

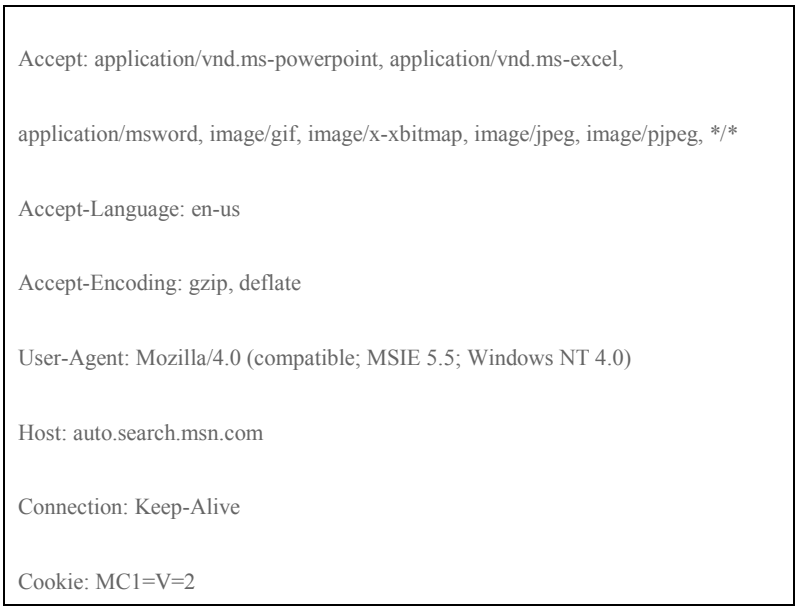
beklentilerini web robotu göz önünde bulundurmak zorundadır. Web robotu bazı kurallar çerçevesinde işlemlerini gerçekleştirmelidir.

Web robotu, sayfaları çok hızlı bir şekilde tarayabildiği için, web sunucusu üzerinde, ağ trafiğini oldukça arttırabilir. Web sunucusu web robotunun isteklerine cevap verebilmek için, diğer kullanıcılara istenilen zaman içerisinde cevap veremeyebilir. Web robotu aynı sunucu üzerindeki sayfaları ziyaret ederken belli bir frekans ile sayfaları tarama işlemini yapmalıdır. Web sunucu üzerindeki bant genişliğini yine aynı şekilde belli bir eşik değer çerçevesinde kullanmalıdır.

Web robotları, tarama işlemini belirli bir sayfa sınırlaması olmadığı sürece, ziyaret edilecek hiçbir link kalmayana kadar devam edebilir. İnternetin büyüklüğünün artmasıyla birlikte ziyaret edilecek link sayısı belirsiz durumda olabilir. Dolayısıyla web robotları işlemlerine başlamadan önce ziyaret edilecek maksimum web sayfası sayısı belirlenmelidir.

Web tarayıcıları web sunucularına erişim sağlarken http header nesnesi içerisinde kendilerini tanımlayan benzersiz bir isim kullanırlar. Web robotlarının da sunuculara erişim sağlarken kendini tanımlayıcı benzersiz bir isim kullanması gerekmektedir. Web sunucuya her bir istek geldiğinde istemci tarafından web sunucusuna bir http header nesnesi gönderilmektedir.

```
GET /response.asp?MT=www.gsu1.com&srch=5&prov=&utf8 HTTP/1.1
```



Şekil 1: HTTP Header Nesnesi.

Şekil 1'de de görebileceği üzere user-agent bölümünde "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0)" bulunmaktadır. Bu isim "Microsoft Internet Explorer" ait olan benzersiz tanımlayıcıdır. Web robotu da web sunucularına erişim sağlarken "user-agent" bölümünde kendi benzersiz tanımlayıcısını bulundurmalıdır.

3.1.1 Robot Protokolü

Webmasterlar sitelerini düzenlerken web robotlarının sitelerini ziyaret etme politikasını belirleyebilirler. Web robotlarının sitenin belli bir bölümüne ya da tamamını incelemesini engelleme hakkına sahiptirler. Robots.txt ile web sitesi yöneticileri arama motorunun sitenin hangi bölümlerini indeksleyeceğini belirlerler. Web robotlarının sitelerini tarama zamanlarını ve tarama frekanslarını da web sitesi sahipleri robots.txt ile

belirleyebilirler. Robots.txt site adresinin kök dizini içerisinde bulunmalıdır. Örneğin: http://www.benimsitem.com/robots.txt şeklinde erişim sağlanmalıdır.

Eğer site yöneticisi, web sitesinin arama motoru tarafından indekslenmesini istemiyor ise robots.txt dosyasını şekil 2'de olduğu gibi oluşturmalıdır:

```
User-agent: *

Disallow:
```

Şekil 2 : Robots.Txt Örneği

User-Agent bölümünde HTTP Header nesnesi içerisinde gönderilen web robotunun ismi tanımlanmaktadır. User-Agent kısmında "*" karakteri bulunması belirtilen kuralın tüm web robotları için geçerli olduğunu belirtir.

```
User-agent: Googlebot

Disallow:

User-agent: Msnbot

Disallow: /nesne.html
```

Şekil 3: Robots.Txt Örneği

Web sitesi yöneticisi belirli sayfaların arama motoru tarafından indekslenmesini istemiyorsa robots.txt aşağıdaki gibi oluşturulmalıdır.

```
U User-agent: *

Disallow: /resimlerim/gizliresim.html

Disallow: /projelerim/

Disallow: /ozelklasor/
```

Şekil 4: Robots.Txt Örneği

Disallow bölümünde arama motoru tarafından indekslenmesi istenmeyen klasörlerde belirtilmelidir. Robot protokolünün dezavantajı, web robotunun web sitesini ziyaret sırasında herhangi bir zorunluluk içermeden sayfaları ziyaret edebilmesidir. Web robotu robots.txt dosyasında belirtilen kurallara uymasa da, site üzerindeki sayfaları ziyaret edebilir. Robots.txt etik kurallar çerçevesinde site yöneticisi ve web robotu arasında belirlenmiş olan tanımlamalardır.

3.1.2 Site Haritası (Sitemap.xml)

Arama motorlarının kullandıkları web robotları ile site yöneticileri arasından iletişim sağlanması amacıyla geliştirilen bir diğer yöntem de "site haritası protokolünün oluşturulmasıdır.

Site yöneticileri, sitemap.xml içerisinde web siteleri ile ilgili olarak tüm URL'leri listeleyen bir doküman hazırlarlar. Bu sayede web robotları, siten üzerinde daha etkin bir kontrolü sağlanmış olur. Ayrıca web robotları gereksiz yere web sitesi sunucuları

üzerinde ağ trafiğini yoğunlaştırıcı işlem yapmaktan uzaklaşırlar. Web robotları bu sitemap.xml üzerinde ne sıklıkla değiştiği, en son ne zaman değiştiği, hangi sayfanın diğer sayfadan daha önemli olduğu gibi bilgilerine buradan ulaşabilirler. Böylece web robotlarına daha etkin bir şekilde siteyi tarama imkânı sağlanabilmektedir. Site haritası protokolü, robot protokolünde bulunan hariç tutma özelliğinin tam tersi olarak dâhil etme mantığı ile çalışır.

Sitemap.xml , site içerisinde ana klasörde bulunması tercih edilen bir yöntemdir. Örn.:www.benimsayfam.com/sitemap.xml gibi. Bunun yanı sıra sitemap.xml'i robots.txt içerisine konumunu belirtecek şekilde yazılabilmektedir. Şekil 5'de robots.txt içerisinde örnek bir sitemap.xml bildirimi gösterilmektedir.

```
Sitemap: www.benimsayfam.com/sitemap/sitemap.xml
```

Şekil 5: Robots.txt üzerinde Sitemap.xml Belirtimi

Günümüzde etkin olarak kullanılan arama motorları siteyi tarama işlemi gerçekleştirmek istediklerinde ilk başvurdukları protokol site haritası protokolüdür. Şekil 6'da örnek bir sitemap.xml dosyası görülebilmektedir.

```
<urlset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://www.fem14.tr.gg/schemas/sitemap/0.9
```

```
        http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd"
        xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

   <url>
     <loc>http://www.google.com.tr/</loc>
     <lastmod>2007-10-10</lastmod>
     <changefreq>daily</changefreq>
     <priority>0.9</priority>
   </url>
   <url>
     <loc>http://www.google.com.tr/sitemap.xml</loc>
     <lastmod>2007-10-10</lastmod>
     <changefreq>monthly</changefreq>
     <priority>0.5</priority>
   </url>
</urlset>
```

Şekil 6: Sitemap.xml Örneği

Sitemap.xml'de bulunan "changefreg" alanında sayfanın değişim frekansı "never", "yearly", "monthly", "weekly" , "daily" , "hourly" , "always" olarak belirtilebilmektedir. Burada belirtilen değerler doğrultusunda web robotu hangi aralıklar bu sayfayı tekrar kontrol etmesi gerektiğini belirleyebilmektedir. "Priotry" bölümü

sayfanın önceliğinin atandığı bölümdür. Sitede bulunan sayfanın diğer sayfalara göre hangisinin öncelikle taranmasını istendiği konusunda web robotunu yönlendirecektir.

Web sitesi yöneticileri, arama motorlarına sitemap.xml dosyasını değiştiğini bildirebilirler. Örneği google arama motoru için "http://www.google.com/webmasters/sitemaps/ping?sitemap=" adresinden bildirim yapılabilmektedir.

Sitemap.xml dosyasının oluşturulmasını bir avantajı da, Flash gibi arama motorları tarafından içeriği taranamayan sitelerin, web robotu tarafından bilinirliliğini sağlar.

3.1.3 URL Düzenleme

Internet üzerindeki belgeler birbirlerine linkler üzerinden bağlantı sağlarlar. Bir sayfadaki HTML "<a" ile başlayan etiketin "href" özelliğinde link verilen sayfanın URL'si bulunmaktadır. URL'ler relative (bağıl) ve absolute (mutlak) olmak üzere ikiye ayrılırlar. Mutlak URL'lerde link üzerinde web sayfasının adresi ve dosya ismi bulunmaktadır. Örneğin; http://www.example.com/dosya.html. www.example.com web sunucunun adresi, dosya.html'de dosyanın adını göstermektedir. Bağıl URL'ler, mutlak URL'lerin aslında bir kısmını göstermektedir. Link sadece "/dosya.html" şeklinde http etiketinin içerisinde verilirse bu link bağıl bir linktir.

Web robotları, sayfa üzerindeki linkleri inceleme işlemi sırasında link listesini belirlerken, bağıl URL'leri, mutlak URL'ye çevirme işlemini gerçekleştirmelidir. Web robotlarının kullandığı bu işleme "URL Düzenleme" işlemi denilmektedir. URL düzenleme işlemi sırasında web robotu şu adımları uygular:

URL'ler büyük küçük harfe duyarlı değildir. HTML etiketinde büyük harfle verilmiş olan bir link web robotu küçük harfe çevirir.

HTTP://www.Example.com/ → http://www.example.com/

İndeks sayfalarını URL içerisinde ayrıca belirtmeye gerek yoktur. Eğer linkte verilen URL bir indeks sayfasına yönlendirme yapıyorsa, web robotu indeks sayfasını linkte göstermez.

http://www.example.com/default.asp → http://www.example.com/

http://www.example.com/a/index.html → http://www.example.com/a/

Eğer verilen link bir fragment içeriyorsa, URL düzenleme işlemi sırasında fragmentlar link üzerinden kaldırılır.

http://www.example.com/bar.html#section1 → http://www.example.com/bar.html

HTML sayfaları HTTP ‘nin default portu 80 üzerinden yayımlanırlar. Eğer link üzerinden default port olan 80 belirtilmişse, web robotu 80 portunu link üzerinden kaldırır.

http://www.example.com:80/bar.html → http://www.example.com/bar.html

RFC 3986 algoritmasına göre link üzerinde belirtilen “..” ve “.” bölümleri web robotu tarafından kaldırılır.

http://www.example.com/../a/b/../c/./d.html→http://www.example.com/a/c/d.html

Bazı web sayfaları linkler içerisinde birden fazla değişken içerebilirler. URL düzenleme işleminde bu değişken alfabetik olarak sıraya dizilirler.

http://www.example.com/display?lang=en&article=fred

http://www.example.com/display?article=fred&lang=en

URL satırında sorgu kısmı boş olarak verilmiş ise, “?” işareti URL düzenleme işlemi sırasında kaldırılır.

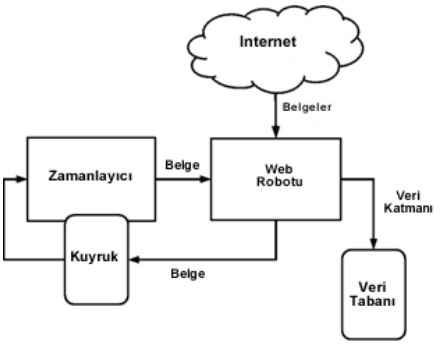
http://www.example.com/display? → http://www.example.com/display

3.1.4 HTML Belgenin XML Belgeye Dönüştürülmesi

Bu çalışmada internet üzerinden web robotları tarafından indirilen HTML sayfalarının içindeki verilerin, standart olarak hiyerarşik ve düzenli bir yapıya sahip XML biçimine dönüştürülmesini sağlayan bir uygulama geliştirilmiştir. Geliştirilen bu uygulama ile HTML belge içindeki verilerin indeksleyici gibi internet üzerindeki verileri işleyen programlara daha etkin olarak sunulması sağlanmıştır.

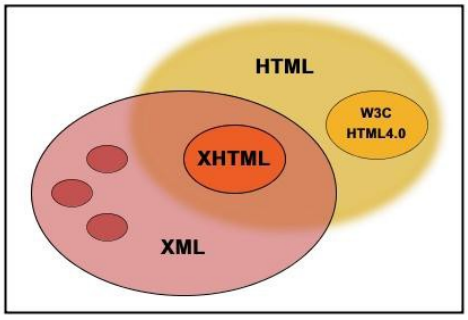
Günümüzde internet kullanımının yaygınlaşması ile birlikte internet üzerindeki belgelerin sayısı gün geçtikçe artmaktadır. Bu belgelerin büyük bir çoğunluğu HTML (Hypertext Markup Language) standardında hazırlanıp yayınlanmaktadır. HTML, belgelerin birbirine nasıl bağlanacağını ve belge içindeki metin ve resimlerin nasıl yerleşeceklerini belirleyen ve etiket olarak isimlendirilen kod parçalarını içeren bir işaretleme dilidir. Tarayıcılar, HTML kodunu yorumlayarak kullanıcıya bilginin sunulmasını sağlar. HTML standardında, kesin belirleyici bir düzenin olmayışı farklı tarayıcıların aynı kodu farklı yorumlamasına sebep olabilmektedir. Ayrıca bu durum, HTML belgeleri üzerinde gösterilen verinin işlenmesini zorlaştırmaktadır. Verilerin çeşitli çalışma ortamları arasında paylaşımının etkin olarak sağlanabilmesi için XML (Extended Markup Language) standardı geliştirilmiştir. XML'in amacı farklı sistemler arası veri aktarımını sağlamaktır. Verinin tanımlanması ve betimlemesi için kullanılır. HTML'de kullanılacak etiketlerin önceden tanımlı olması gerekirken, XML'de kullanılacak etiketler önceden tanımlı olmak zorunda değildir. XML belgelerinin en önemli özelliği, belgelerin ağaç yapısında olmasıdır.

İnternet üzerinden bir bilgiye ulaşmak istediğimizde, bu bilgiyi milyonlarca web sitesi içerisinden sayfaları inceleyerek ulaşmak büyük bir çaba gerektirir. Bu sorunu çözmek için arama motorları geliştirilmiştir. Arama motorları, internet üzerindeki web sayfalarını tarayıp, onları kendi içinde indeksleyip, kullanıcıların indekslenmiş veriler üzerinde arama yapmasına olanak sağlayan sistemlerdir. Arama motorları sayesinde kullanıcıların istedikleri bilgiye daha hızlı ulaşması sağlanmaktadır. Arama motorlarında web robotu olarak isimlendirilen programlar kullanılarak, internet üzerindeki sayfalar otomatik olarak kayıt edilir. Web Robotu taradığı tüm sayfaların bir kopyasını alır. İndeksleyici olarak isimlendirilen program, Web Robotunun indirmiş olduğu bu sayfaları çeşitli yöntemler kullanarak, arama motorunun veri tabanına kaydeder. Bunun yapılması sonucunda kullanıcıların istedikleri bilgiye, veritabanı üzerinden sorgulamalar yaparak verinin bulunduğu sayfalara çok kısa sürede ulaşabilmesi sağlanmaktadır. Şekil-7 de gösterildiği gibi Web Robotu, web sayfaları üzerindeki bağlantıları belirli bir yöntemle takip ederek sayfalar arasında geçişi sağlamaktadır. Bir Web Robotu için önemli olan veri sayfa içindeki bağlantılardır. Web Robotu sayfanın bir kopyasını oluşturduktan sonra, belge içindeki kelimeleri veritabanına kaydeden indeksleyici için resmin nerede gösterildiği veya yazının kalın mı yazıldığı gibi belgenin gösterimine yönelik ayrıntılar önemli değildir. İndeksleyici, belge içindeki kelimeleri kullanmaktadır. Belge üzerindeki kelimeleri belirli bir metotla indekslemektedir. Web Robotunun belgeyi belirli bir standartta ve HTML'in düzensizliğinden arındırılmış olarak sunması, indeksleyicinin daha etkin olarak indeks verilerine ulaşmasını sağlayacaktır. [11]



Şekil 7: Web Robotu Yapısı

HTML'in XML'e çevrimi konusunda daha önceden geliştirilmiş çalışmalar mevcuttur. Bu çalışmalar HTML üzerindeki verinin okunarak XML'e çevrilmesinden öte, HTML belgeyi XML formatına dönüştürerek HTML belgenin XHTML formatına aktarılmasını yapmaktadırlar. Şekil 8'de XML, HTML ve XHTML arasındaki ilişki verilmektedir. Bu çalışmada HTML üzerindeki veri katmanının XML'e çevrilmesi işlemini gerçekleştiren bir uygulama geliştirilmiştir.



Şekil 8 : HTML ve XML ilişkisi

HTML (Hypertext Markup Language), belgelerin birbirine nasıl bağlanacağını ve belge içindeki metin ve resimlerin nasıl yerleşeceklerini belirleyen ve etiket (tag) denilen kod parçacıklarından oluşan bir işaretleme dilidir. [12]

```
<html>
 <head>
   <title>Bu Bir Başlık</title>
 </head>
 <body>
   <p>
     Merhaba
     <b>KALIN</b>
     <i>ITALIK</i>
     <b><i>Kalın ve italik</i></b>
   </p>
 </body>
</html>
```

Şekil 9*: HTML Belgenin Yapısı.*

HTML bir programlama dili olarak değerlendirilemez. Programlama dili seri prosedür ve açıklamalardan oluşur ve genelde bir dış veriye ulaşmayı hedefler. HTML ise başlı başına verinin kendisidir.

HTML birçok farklı etiketlerden oluşmaktadır. Web robotları bu etiketlerin sadece veri ve bağlantıları içeren kısımları ile ilgilenir. İndeksleme işlemi sürecinde, indeksleyici taranmış sayfaları inceleyerek, sayfa üzerindeki kelimeleri indeksler, dolayısıyla HTML düzenleme işleme arama motoru oluşturman önemli adımlarından biridir. Web robotu site üzerindeki linkleri belli bir algoritma içerisinde takip eder. HTML yapısında link etiketi <a href> etiketi ile belirtilir.

 Buraya Tıklayınız

Yukarıdaki örnekte kullanıcı sadece “Burayı Tıklayınız” yazısını görmektedir. Alt özelliği ile belirtilen açıklama internet tarayıcının link üzerinde geldiğinde küçük bir pencere ile belirtilir.

HTML’de etiketler hazırlanırken, etiketlerin </> bitiş işareti ile bitirilme zorunluluğu bulunmamaktadır. HTML kesin olarak bir yapısal düzene sahip olmamasının yanı sıra, bu sayede yapılan hataları gözardı eder. HTML etiketlerinde, etikete ait nitelikler “attribute” olarak adlandırılır. “name=’id’ ” biçiminde gösterir. Arada boşluk bırakılarak birden fazla nitelik tanımlanabilir. [13]

HTML belgesinde açıklamalar “<-- açıklama -->” şeklinde gösterilirler. Açıklama metinleri dokümanın yapısına müdahale edecek herhangi bilgi içermez. Açıklama bölümleri birde internet tarayıcının desteklemediği JavaScript’leri gizlemek için kullanılır. Açıklama alanları web robotları tarafından işleme alınmamaktadır.

```
<script>
<!--
var popupwindow _;
function fonksiyon(parametre)
{
if(popupwindow_&&! popupwindow _.closed)
{
popupwindow _.location.href= parametre;
```

```
}

wpop_.focus();

}

//-->

</script>
```

Şekil 10: Script Örneği

HTML tutarlı bir biçimleme dili değildir. Örneğin, bir belgede başlangıç etiketleri, bitiş etiketleri, diğer etiketler ve metin ile karşılaşılmasına rağmen her başlangıcı olan etiketin bir bitişi olması zorunluluğu yoktur. HTML belgeleri metin, açıklama, basit etiketler ve sonlu etiketler bileşenlerinden oluşur.

HTML üzerindeki etiketler iki gruba ayrılabilir, ilk grup etiketler HTML üzerindeki verileri içeren gruptur. Bu grupta paragraflar, yazılar, resimler ve linkler bulunmaktadır. Diğer grup ise verinin altyapısı olarak adlandırılan, verilerin tarayıcılar tarafından nasıl gösterileceğini belirten etiketlerdir. Arama motorları verinin nasıl gösterileceği ile değil ne olduğu ile ilgilenmektedir. Dolayısıyla aslında Web Robotlarının indeksleyici için hazırlayacağı bilgi ilk grupta belirtilen verilerdir.

XML, HTML ile pek çok açıdan benzerlik gösteren bir işaretleme dilidir. Verinin tanımlanması ve betimlenmesi için kullanılır. HTML'deki yapının aksine XML'de kullanılacak olan etiketler önceden tanımlı değildir. Yani bir XML dokümanının yapısı

tamamıyla kullanıcı tarafından oluşturulur. Verinin betimlenmesi için DTD (Document Type Definition) adı verilen yapılar kullanılmaktadır. [14]

XML ve HTML arasındaki en belirgin fark XML'in verinin kendisiyle ilgilenmesi HTML'in ise verinin sunumuyla ilgilenmesidir. HTML dokümanları veriye ilişkin gösterim bilgilerini içerirken XML dokümanları ise verinin tanım bilgilerini içermektedir. XML'in tasarım amaçlarından biri de verinin taşınmasıdır. Bu özellikleri incelendiğinde XML'in bir çok önemli işlevi yerine getirdiği görülmektedir.

Günümüz bilişim uygulamalarında XML bir çok farklı alanda kullanılmaktadır. Bu nedenle XML'i bir anlamda geleceğin web dili olarak tanımlamak mümkündür.

```
<mesaj>
  <kime>Ali</kime>
  <kimden>Ayşe</kimden>
  <baslik>Nice Yıllara</baslik>
  <yazi>Doğum Günün Kutlu Olsun!</yazi>
</mesaj>
```

Şekil 11 : XML Belgenin Yapısı

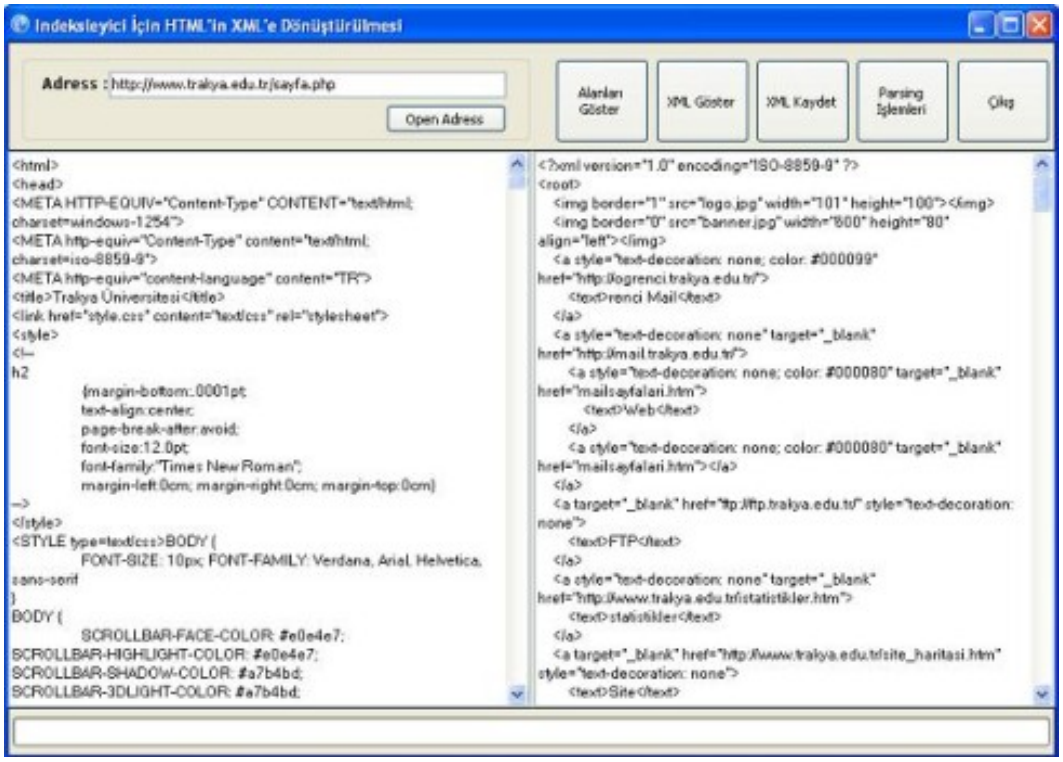
HTML bir sözcüğü etiketler arasına alarak metnin koyu ya da italik yazılmasını sağlar. Oysa XML ise yapısal verilerin etiketlenmesi için bir iskelet yapıyı sağlar. XML, HTML'nin yerine geliştirilmemiştir. Farklı amaçlara sahiptir. XML daha çok verinin taşınması, dönüştürülmesi gibi verinin kendisine odaklıdır.

SOAP (Simple Object Access Protocol) internet üzerinde bilginin XML protokolü kullanarak paylaşılmasını sağlayan bir uygulamadır.[1] SOAP uygulamasının birincil fonksiyonu Web Robotları ile çok benzerdir. Web Robotları siteleri tarayarak, indeksleyici için gerekli olan bilgiyi siteden alıp getirir, SOAP adresleri de web siteden istenen veriyi getirmek için kullanılır. XML biçimi, HTTP üzerinden istenilen bilgilerin alındıktan sonra, farklı sistemlere aktarmak üzere kullanılması için uygundur. [15]

İndeksleyici, Web Robotlarının getirdiği belgeleri yorumlayarak, arama motorunun veri tabanına kaydeden programdır. İndeksleyici, sadece veri içermeyen karışık HTML kodları içinde arama motorunun veri tabanı için gerekli olan bilgileri alıp belirli bir algoritma dâhilinde bu bilgileri arama motorunun veri tabanına kaydeder. Arama motorları, sayfalar üzerindeki bağlantıları, kelimeleri, resimlerin adreslerini ve resimlerin yazılarını veri tabanına kaydeder.

Geliştirilen uygulama sayesinde indeksleyicinin düzensiz ve karmaşık HTML kodlarından metinleri ve bağlantıları bulmak için zaman harcaması yerine, indeksleyiciye gerekli olan bilginin XML formatında sunulması sağlanmıştır. Bu şekilde, indeksleyicinin HTML üzerindeki düzensiz kodları tasnif etmesine gerek kalmayacaktır.

[1] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

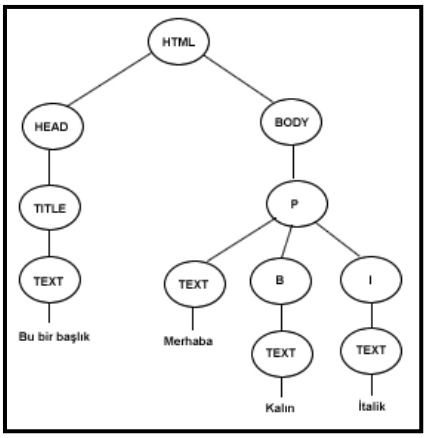


Şekil 12 : Geliştirilen Uygulama Arayüzü

Geliştirilen uygulamada web belgelerinin incelenmesinde, HTML ve XML belgeleri için yüksek düzeyli, etkin uygulamalar geliştirilmesini sağlayan DOM (Document Object Model) yapısı[2] kullanılmıştır. Geliştirilen uygulamanın arayüzü şekil 5'te gösterilmektedir. Uygulamada HTML belgenin XML belge olarak dönüştürülmesi için önce HTML belge ağaç yapısında ifade edilmektedir. HTML belgelerde açılan etiketlerin sonlandırılma zorunluluğu yoktur. HTML belgenin, bir ağaç yapısı olarak tanımlanabilmesi için HTML belge içindeki sonlandırılmayan etiketlere, sonlandırma etiketleri eklenir, varsa başlangıç etiketi bulunmayan etiketler de kaldırılır, böylece HTML belgeye XML'deki ağaç yapısına uygun hale getirilir.Bu işlem sonucunda şekil

[2] http://www.w3.org/DOM/

6'da gösterildiği gibi HTML belgesini ağaç yapısı şeklinde gösterebiliriz. HTML belgesinin kök düğümü <HTML> </HTML> etiketi olmaktadır.



Şekil 13: HTML Belgenin Ağaç Yapısında İfade Edilmesi

XML belgelerin ağaç yapısı şeklinde tanımlandığına göre HTML belgelerinde yukarıda belirtilen düzenlemeleri yaptıktan sonra şekil 7 de verilen algoritmayı kullanarak belgeler XML formatına çevrilmektedir.

```
Verilen Web Sayfasının HTML kodlarını İndir
Yap
{
  ilk etiketi oku
  Eğer (etiketin bitiş noktası yoksa)
  {
     etikete bitiş noktası ekle
  }
```

```
    Eğer ( etiketin başlangıç noktası yoksa)
    {
       etiketi sil
    }
  } DevamEt (son etikete kadar oku)
  XML belgeyi tanımla
  Yap
  {
    HTML etiketini XML etiketine çevir
  } DevamEt (son etikete kadar oku)
```

Şekil 14 : HTML belgenin XML belgeye dönüştürülme algortiması

Yapılan çalışma ile amacımız, HTML belgeyi XML belge olarak ifade etmekten çok indeksleyici için HTML içindeki gerekli verinin hazırlanmasını sağlamaktır. İndeksleyici "href" , "img" etiketleri ve belge üzerindeki metinleri girdi olarak almaktadır. İndeksleyicinin girdi olarak aldığı bu bilgiler Şekil 8 de verilen algoritma kullanılıp XML formatına çevrilerek indeksleyici için hazır hale getirilmektedir. Daha sonra indeksleyici, arama motorunun veritabanına XML olarak ifade edilmiş anlamlı bilgiyi veri tabanına kayıt edecektir. [16]

```
Verilen Web Sayfasının HTML kodlarını İndir
Yap
```

```
   {
     ilk etiketi oku
     Eğer (etiketin bitiş noktası yoksa)
     {
        etikete bitiş noktası ekle
     }
     Eğer ( etiketin başlangıç noktası yoksa)
     {
        etiketi sil
     }
   } DevamEt (son etikete kadar oku)
   XML belgeyi tanımla
   Yap
   {
     Etiketi oku
     Eğer (etiket = bağlantı) veya (etiket=yazi) veya
(etiket=resim)
     {
        XML belgede düğümünü oluştur
     }
   } DevamEt (son etikete kadar oku)
```

Şekil 15: İndeksleyici için HTML deki bilgilerin XML formatına dönüştürülmesi

HTML belge ve XML belge incelendiğinde dönüşüm işleminin indeksleyici için gereksiz olarak nitelendirilen fazla verilerden arındırılmış doğru bir XML belge haline geldiği ve çok karmaşık HTML dokümanlarda bile indeksleyici için uygun olarak dönüşüm yaptığı görülmüştür.

```
<p><b>

<font size="1" face="Verdana">
<img border="0" src="yeni_bullet.gif" width="8" height="6"></font><font size="1" face="Verdana" color="#2388EF">
<a target="_blank" href="http://www.trakya.edu.tr/bidb/">BİDB</a><br>
</font>
<font size="1" face="Verdana">
<img border="0" src="yeni_bullet.gif" width="8" height="6"></font><font size="1" face="Verdana" color="#2388EF">
<a target="_blank" href="http://www.trakya.edu.tr/bidb/internet_hiz_kul_kurallari.htm">
T.Ü. İnternet Hizmetleri Kullanım  Kuralları </a></font></b><br>
<b>
<font size="1" face="Verdana">
<img border="0" src="yeni_bullet.gif" width="8" height="6"></font><font color="#0B4460"><font face="Verdana" size="1">
  <a target="_blank" href="http://www.trakya.edu.tr/yeni/kisisel_web.php">Kişisel Web Siteleri</a></font><a style="text-decoration: none; color
  </a> </font></b><br>
<b>
<font size="1" face="Verdana">
<img border="0" src="yeni_bullet.gif" width="8" height="6">
  <a target="_blank" href="eskiwebler.htm">Eski
  Web Siteleri</a></font></b></td>
</tr>
<tr>
<td class="kaderspecial" bgcolor="#FFFFFF">
  <table border="0" width="100%" id="table17">
<tr>
<td>
<p align="center"><a target="_blank" href="fulbright.htm">
<img border="0" src="fulbright.gif" width="152" height="75"></a></td>
</tr>
</table>
```

Şekil 16 - Örnek bir HTML Belgesi.

Şekil 17 - HTML belgenin XML'e dönüştürülmüş şekli.

HTML içindeki verilerden, indeksleyici için gerekli olan verileri bulmak için XML formatından ifade edilmesinin oldukça fayda sağladığı görülmektedir. Ayrıca HTML belgelerin XML belgeye çevrilerek işlenmesi arama motorlarının sürekli güncel kalması dışında başka bir çok amaçla da kullanılabilir. XML belgenin yeniden kullanılabilirliği XML biçimli belgenin ağaç yapısında olmasından dolayı daha uygundur. Ek B'de "HTML'den XML'e Çevirici" programının kodları görülebilir.

HTML belgeler üzerindeki verilerin XML formatına dönüştürülmesi, internet üzerinden otomatik olarak bilgi alışverişi yapılan tüm uygulamalar için kullanılabilir.

Gelecekte web sayfalarında SOAP desteğinin yaygınlaşması ile birlikte belki de her internet sitesi kendi veri katmanını XML formatında kullanıcılara sunan servisler içerecektir.

3.2 Web Robotu Türleri

Arama kriterine herhangi bir kelime yazıldığında arama motoru aranan kelime ile ilgili olarak birçok web sitesini listelemektedir. Günümüzde internetin bilgi paylaşımının temel aracı olarak kullanılması ile birlikte arama sonuçlarında binlerce web sitesi listelenebilmektedir. Bunun yanı sıra arama motorları internet üzerindeki sayfalarda bulunan kelimeleri ve linkleri kullandığı için, bazen art niyetli olarak geliştirilmiş sitelerden dolayı getirdiği sonuçlar verilen kelimeyi içersede aranan konu ile alakalı bağlantısı olmayabilir. Dolayısıyla arama motorlarının getirdiği sonuçlar için kullanıcılar kişisel yetileriyle tekrar arama yapmak zorunda kalmaktadırlar. Bunun sonucu olarak arama motorlarının istenilen bilgiye erişmesinde, farklı algoritmalar geliştirilmiştir. Konu Tabanlı, Konu Odaklı web robotlarının oluşmasının temel neden buradan kaynaklanmaktadır.

3.2.1 Link – Odaklı Web Robotu

Link – odaklı web robotu, sayfa üzerindeki linkleri takip ederek, işlem yapan web robotudur. Link – odaklı web robotuna, giriş olarak bir sayfa adresi verilir. Link –

odaklı web robotu verilen sayfanın HTML kodunu tarayarak, sayfa üzerindeki linkleri bulur. Sayfayı depolama birimine kaydeder.

Girdi: Başlangıç Sayfası, Maksimum Sayfa İndirme Sayısı

Tarama: Sayfadaki İlk Linki Tara

Çıktı: Tarama Sonuçları

Link - odaklı web robotu, verilen sayfalar üzerinde maksimum sayfa sayısına ulaşılan kadar tüm sayfaları kaydeder.

Link- odaklı web robotu, maliyetli bir web robotudur. Sayfaların anlamını, içindeki kelimelerin geçerliliğini kontrol etmediği için değerli değersiz tüm sayfaları kaydetmektedir.

Sayfa üzerindeki tüm linkleri hafızasına kaydeder, herhangi bir tıkanma ile karşılaşıldığında, derinlik öncelikli algoritmayı kullanarak işlemlerine elinde hiçbir link kalmayana kadar ya da maksimum sayfa sayısına ulaşıncaya kadar devam eder. Yapılan deneylerde Ek C'de bulunan link odaklı web robotu kullanılmıştır.

3.2.2 Konu Seçimli Web Robotu

Konu seçimli web robotu, internet üzerindeki sayfaların üzerindeki linkleri verilen konuya göre inceleyip kaydeden web robotudur.

Girdi: Başlangıç Sayfası, Maksimum Sayfa Sayısı, Sorgu Kelimeleri

Tarama: Sadece link adresinde veya link adında verilen kelimeler geçiyorsa sayfayı tara

Çıktı: Tarama Sonuçları

Konu seçimli web robotunda kullanıcı tarafından kelimeler verilir. Konu seçimli web robotunda verilen kelimelerin önemi büyüktür. Kelimeler konuyu tam olarak yansıtacak şekilde seçilmelidir. Konuya özel kelimeler vermemiz gerekir. Mümkün olduğu kadar konu ile ilgili eşsesli kelimeler varsa bunlardan kaçınılmalıdır.

Bu web robotu, gelen sayfa üzerindeki linkleri inceleyerek, linkleri yazısında ya da başlığında belirtilen kelimelerden herhangi biri varsa o linki takip ederek tarama işlemini gerçekleştirir.

Konu seçimli, sayfa üzerindeki geçerli tüm linkleri hafızaya kaydeder. Herhangi bir sayfada tıkanma olduğunda, derinlik öncelikli mantığı kullanarak tarama işlemine elinde hiçbir link kalmayıncaya kadar ya da maksimum sayfa sayısına ulaşıncaya kadar devam eder.

Konu seçimli web robotu ile sadece belirlediğimiz konudaki sayfaları kaydedebiliriz. Böylece boyuttan kazancımız olmakla beraber, konuya özel arama motorunda gereksiz orandaki sayfa sayısı düşük olacaktır (Çöp sayfalar).

3.2.3 Konu Odaklı Web Robotu

Konu odaklı web robotu, internet üzerindeki sayfalarda bulunan linkleri inceleyerek link tanımlarında kullanılan kelimeleri kullanım sayılarına göre sıralar.

Sıralanan kelimelerden programa girdi olarak verilen eşik değerin üstündeki kelimeleri içerek linkler üzerinden tarama işlemini gerçekleştirir.

Girdi: Başlangıç Sayfası, Kelime Eşik Değeri

Tarama: Kelime Eşik Değerinin üstünde frekansa sahip kelimeler içeren linkleri tara

Çıktı: Tarama Sonuçları

Konu odaklı web robotu, art niyetli site sahiplerinin, HTML sayfalar üzerinde zemin rengi ve yazıtipi rengini aynı yapması sonucu oluşan reklam amaçlı gereksiz bilginin indekslenmesini engeller. Kelime eşik değeri ile web sayfasının yoğunlaştığı konu ortaya çıkartılarak, web robotunun konu odaklı sayfalar üzerinden devam etmesini sağlar.

Bu web robotu, gelen sayfa üzerindeki linkleri inceleyerek, eşek değerin üstündeki kelimeleri, linklerin yazısında ya da başlığında içeren linkler üzerinden tarama işlemini gerçekleştirir.

Konu odaklı web robotu, sayfa üzerindeki geçerli tüm linkleri hafızaya kaydeder. Herhangi bir sayfada tıkanma olduğunda, derinlik öncelikli mantığı kullanarak tarama işlemine elinde hiçbir link kalmayıncaya kadar ya da maksimum sayfa sayısına ulaşıncaya kadar devam eder.

Konu odaklı web robotu ile yoğun text içeren web siteleri taranabilir. Eğer resim yoğunluklu bir sitede kelimelerin frekansı eşik değerinin altında kalması muhtemel

olacaktır. Böylece boyuttan kazancımız olmakla beraber, arama motorunda gereksiz orandaki sayfa sayısı da düşük olacaktır.

Konu seçimli web robotundan farklı olarak, kullanıcı tarafından herhangi bir konuyu belirleyici kelimeler verilmez, taranan siteler üzerinden ilerlenecek konu belirlenir. Ek D'de konu odaklı web robotunun program kod düzeneği görülebilir.

3.2.4 RSS Özellikli Web Robotu

Arama motorları sık güncellenen siteleri web robotları ile sıklıkla ziyaret ederler. Haber sağlayıcı web siteleri sık güncellenen web sitelerinden biridir. Günümüzdeki birçok haber sağlayıcının RSS servisi bulunmaktadır. Her rss servisi aslında bir XML dosyasıdır. Ek E'de RSS okuyucu kod sınıfı görülebilir.

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Liftoff News</title>
    <link>http://liftoff.msfc.nasa.gov/</link>
    <description>Liftoff to Space Exploration.</description>
    <language>en-us</language>
    <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>
    <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</lastBuildDate>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
```

```
    <generator>Weblog Editor 2.0</generator>
    <managingEditor>editor@example.com</managingEditor>
    <webMaster>webmaster@example.com</webMaster>

    <item>
      <title>Star City</title>
      <link>http://liftoff.msfc.nasa.gov/news/2003/news-
starcity.asp</link>
      <description>How do Americans get ready to work with Russians
aboard the
        International Space Station? They take a crash course in
culture, language
        and protocol at Russia's Star City.</description>
      <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>

<guid>http://liftoff.msfc.nasa.gov/2003/06/03.html#item573</guid>
    </item>

    <item>
      <title>Space Exploration</title>
      <link>http://liftoff.msfc.nasa.gov/</link>
      <description>Sky watchers in Europe, Asia, and parts of Alaska
and Canada
```

```
        will experience a partial eclipse of the Sun on Saturday, May
31st.</description>
      <pubDate>Fri, 30 May 2003 11:06:42 GMT</pubDate>

<guid>http://liftoff.msfc.nasa.gov/2003/05/30.html#item572</guid>
    </item>

    <item>
      <title>The Engine That Does More</title>
      <link>http://liftoff.msfc.nasa.gov/news/2003/news-
VASIMR.asp</link>
      <description>Before man travels to Mars, NASA hopes to design
new engines
        that will let us fly through the Solar System more quickly.
The proposed
        VASIMR engine would do that.</description>
      <pubDate>Tue, 27 May 2003 08:37:32 GMT</pubDate>

<guid>http://liftoff.msfc.nasa.gov/2003/05/27.html#item571</guid>
    </item>

    <item>
      <title>Astronauts' Dirty Laundry</title>
```

```
      <link>http://liftoff.msfc.nasa.gov/news/2003/news-
laundry.asp</link>
      <description>Compared to earlier spacecraft, the International
Space
        Station has many luxuries, but laundry facilities are not one
of them.
        Instead, astronauts have other options.</description>
      <pubDate>Tue, 20 May 2003 08:56:02 GMT</pubDate>

<guid>http://liftoff.msfc.nasa.gov/2003/05/20.html#item570</guid>
    </item>
  </channel>
</rss>
```

Şekil 18 - Örnek RSS Dosyası

Birçok RSS dosyasında, haberlerin sınıflandırılması konu gruplu olarak yapılmıştır. RSS Özellikli Web Robotu, konu gruplu arama motorlarında RSS üzerinde verilen linkleri tarayarak, arama indeksine RSS'te belirtilen konu üzerinde sayfanın kaydedilmesini sağlar.

RSS özellikli web robotu kullanılması sayesinde, sayfa üzerinden herhangi bir konu belirtimine yada konu seçimine gerek kalmamaktadır. RSS'in belirli bir konu ile

getirdiği web sayfaları indeksleyici tarafından kolaylıkla arama veritabanına kaydedilebilir. Ek F'de RSS Özellikli Web Robotu'nun program kod düzeneği görülebilir.

Sık güncellenen web sitelerinde arama motoru kullanıcılarına güncel sonuç verilebilmesi için, yoğun bir ağ trafiği oluşturmamak için RSS özelliği kullanılabilir.

3.2.5 Evrensel Web Robotu

Internet üzerinde bilgiler birçok farklı dilde bulunabilmektedir. Günümüzde, makine ile ilgili araştırma yaparken Almanca kaynakları, matematik ile ilgili araştırma yaparken Rusça kaynakları incelemek yaptığımız çalışmada büyük katkı sağlayacaktır. Bu bağlamda arama motorlar sadece kelime üzerine odaklanarak işlemlerini gerçekleştirmektedir. Aynı kelimenin farklı dillerdeki karşılıklarını içeren sonuçları kullanıcılara getirememektedir.

Evrensel arama motorunun geliştirilmesinde çevirici önemli bir rol oynamaktadır. Çevirici kişinin kendisi vasıtasıyla geliştirilmiş olabileceği gibi üçüncü parti yazılımlar ile de sağlanabilir. Evrensel web robotunun geliştirilmesinde ücretsiz olarak sunulan http://141.85.5.146/WebServices/LangIdWebService.asmx?wsdl adresindeki web servisi kullanılmıştır.

Evrensel web robotunda link odaklı web robotunda olduğu gibi 5 farklı dilde hazırlanmış web sitesi başlangıç sayfası olarak verilmiştir. Web robotu başlangıç sayfalarından her biri için ayrı bir thread başlatarak, maksimum sayfa sayısına ulaşana kadar tarama işlemini gerçekleştirmiştir.

Evrensel web robotu tarama işlemi sırasında, sitenin içerisindeki veriyi İngilizceye çevirerek indeksleyiciye hazırlamıştır. Dolayısıyla tüm sitelerdeki elde edilen veriler İngilizceye tercüme edilmiş halde arama motorunun veritabanına eklenmiştir.

Konu	Web Site Erişim Sayısı	Konu Seçimli
İngilizce	10323	69%
Almanca	2765	19%
Fransızca	1243	8%
Rusça	562	4%

Şekil 19: Dil Bazında site erişim karşılaştırması

Şekilde de görüleceği üzere internet üzerinde İngilizce olarak hazırlanmış web sitelere erişim oranı diğer dillere göre yüksektir. Burada web robotunun başlangıç olarak kullandığı web sitesinin diğer sitelere verdiği link sayısı da sonuçları

etkilemektedir. Yine de internet üzerinden İngilizce veriye erişim diğer dillere göre yoğun bir şekilde sağlanabilmektedir.

3.3 Web Robotlarının Değerlendirilmesi

Konu	Link Odaklı	Konu Seçimli
Felis	7%	93%
Computer	4%	96%
Health	4%	99%
Instruments	1%	99%
Travel	1%	97%

Şekil 20: Karşılaştırma

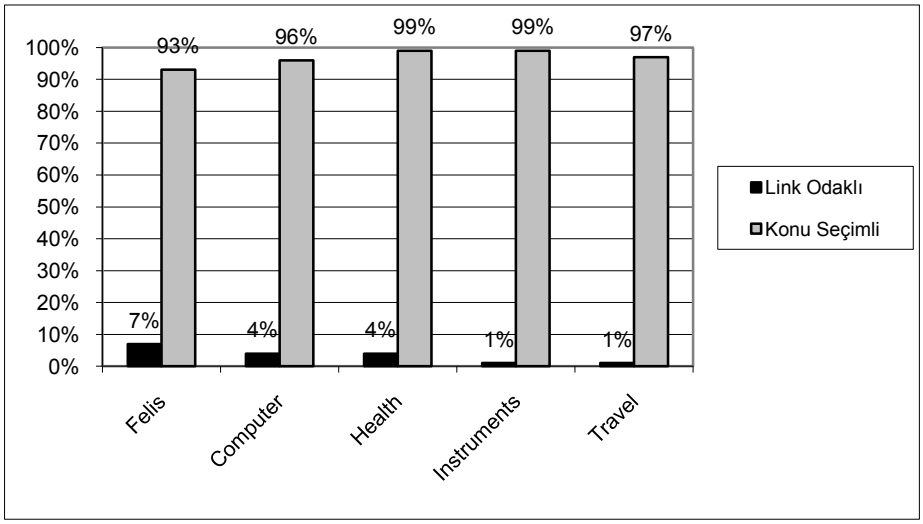
Belirlediğimiz beş konuda her iki web robotunu karşılaştıralım. Computer, Felis, Health, Insruments, Travel .

Computer konusunda genel bilgi içeren soysal bir siteden, sadece bilgisayar ile ilgili olan, yazılım veya donanım bilgisi içeren linkleri topladık.

Felis konusunda hayvanlar âlemi hakkında belgesel konusunda bilgi içeren bir siteden, sadece kedigillere ait olan bilgiler elde ettik.

Health konusunda yine genel bilgi içeren sosyal bir siteden sadece, sağlık ile ilgili olan linkleri toplayarak bu linkler üzerinde tarama işlemini gerçekleştirdik.

Instruments konusunda bir alışveriş sitesinde sadece müzikal enstrümanlara ulaşabileceğimiz sayfalar üzerinden tarama işlemini gerçekleştirdik.



Şekil 21 – Link Odaklı ve Konu Seçimli Web Robotu

Travel konusunda tüm dünya üzerindeki tatil yörelerini referans alan bir sitede, Türkiye'de tatil yapmak konusunda bize referans olabilecek siteleri yakalamaya çalıştık.

Konu seçimli web robotunu kullanarak, konuya özel arama motorları oluşturabiliriz. Kedigiller konusunu ele aldığımızda, link-odaklı web robotu ile gelen sayfaların %7'si kedigiller ile ilgiliyken, Konu seçimli web robotu da gelen sayfaların % 93 ü kedigiller ile ilgilidir. Konu seçimli web robotu da gelen sayfaların tümü

kedigiller ile ilgili bilgi içermektedir. Konu seçimli web robotu kullanılarak arama motorunda gelen sayfaların gruplandırılması yapılabilir.

Örneğin “cat” kelimesini arama motoruna girdiğimizde, bize yüzlerce sayfa getirmektedir. Gelen bu sayfalar içinde, teknoloji şirketlerinden tutunda mimarlık şirketlerine kadar yüzlerce sayfa gelmektedir. Aslında biz konu olarak bildiğimiz hayvan kedileri arıyoruz. Dolayısıyla konu bazlı bir gruplama yapıldığında arama motorlarında faydalı olacaktır. Bunun için konu seçimli web robotu kullanılabilir.

Yapılan test sonuçları bize şunu göstermektedir ki, link-odaklı web robotu ile yapılan taramadaki başarı oranı, bize verilen konu o sitenin yüzde kaçını oluşturduğunu göstermektedir. Maksimum sayfa sayısını 100 verdiğimizde gelen sitelerin konumuzla ilgi oranı %1'dir. Maksimum sayfa sayısını 10000 verdiğimizde de gelen sayfaların konumuzla ilgi oranı %1,002’dir. Dolayısıyla link odaklı web robotu konuyla ilgi oranı web sitesinin o konu hakkında bilgi içeren bölümünün yüzdesini vermektedir.

3.4 Sonuç

Eğer konu bazlı arama motoru yapılacaksa, konu seçimli web robotları kullanılmalıdır. Eğer arama motorumuzda konuları gruplayacak şekilde veri yapısı oluşturacaksak konu seçimli web robotu kullanılmalıdır. Konu seçimli web robotları indeksleme adımı içinde kolaylık oluşturmaktadır.

Internet kullanımının artması ile birlikte internet üzerinde bulunan yararlı ya da yararsız birçok bilginin, kullanıcıya arama motoru aracılığıyla erişmesi sağlanırken, kullanıcıya verilecek olan sonuçların etkin bir şekilde sağlanabilmesi arama motorunun etkin bir algoritma kullanması gerekliliği deney sonuçlarında görülmüştür. Link odaklı web robotu ile ortalama % 3.2 başarı oranı yakalanırken, konu seçimli web robotu ile %99 ortalama başarı sağlanabilmektedir. Arama motorları üzerinde kişinin girdiği kelimeler doğrultusunda gruplama yapılarak sonuçların getirilmesinin sağlanabilmesi veya sadece konuya özel arama motorlarının geliştirilebilmesi için konu seçimli veya konu odaklı web robotlarının, arama motoru veritabanı oluşturulurken kullanılması gerekmektedir. Arama motorlarının en temel amacı, kullanıcıya istediği bilgiyi en kısa zamanda ve en güncel halinde ulaştırabilmesini sağlamaktadır. Bu doğrultuda web robotları internet siteleri üzerinde tarama işlemini gerçekleştirirken, web sitesi sahiplerinin reklam amaçlı olarak ya da web robotlarını yanıltmak için hazırlamış olduğu yazıtipi ve site arka fon renginin aynı durumda olması gibi amaçların web robotu tarafından etkin bir şekilde incelemesi gerekmektedir. Web robotları siteleri tararken kullanıcının ekranda görmüş olduğu içeriği kendine referans olarak kullanmalıdır.

Web robotları internet üzerinde işlemlerini gerçekleştirirken, web sitesi sunucuları arasında yoğun bir ağ trafiği oluşturabilmektedirler. Dolayısıyla web robotu geliştirilirken yazılımların etik kurallar çerçevesinde “web robotu protokolü” ve “site haritası” protokolüne bağlı kalmalıdır. Bunun yanı sıra arama motorları veritabanlarını güncel tutarak, kullanıcıların en yeni bilgiyi sunabilmesi için sık değişiklik yapılan

sitelerde, örneğin haber siteleri, RSS desteği var ise RSS tabanlı web robotunu kullanarak arama motorunun veritabanının güncelliğini korumasını sağlayabilir. RSS tabanlı web robotlarının kullanımında RSS özelliği kullanılan sitenin etkin bir RSS düzeneğinin bulunması gereklidir. Eğer RSS özelliği ve site içeriği birbirinden bağımsız olarak çalışıyorsa, kullanıcının arama motoru vasıtasıyla bilgiye erişimini etkileyecektir.

Web robotları sistemin sadece HTML bölümünü incelemektedirler. Flash veya Java ile oluşturulmuş linkleri inceleyememektedirler. Flash özelliğinin içeriğinin sadece geliştiren firma tarafından biliniyor olması dolayısıyla web robotları flash nesnelerini inceleyememektedir. Arama motorunda, flash nesnesi yoğun olarak kullanılan sitelere ait veri indekslemesi yeteri düzeyde yapılamamaktadır.

Arama motorlarının dilden bağımsız bir şekilde çalışabilmesi için Evrensel Arama Motoru kullanılabilir. Farklı dillerdeki verilerin erişiminde evrensel arama motoru vasıtasıyla ortak bir dilde tercüme yapılarak ilgili veriler indekslenir. Kullanıcının arama yapmak istediği verilere yine tercüme yapılarak ortak dil ile oluşturulmuş veritabanında arama yapılması sağlanabilir. Evrensel web robotunun başarısında çeviri işleminin etkinliği önem arz etmektedir. Çeviri özelliği yeteri düzeyde iyi olmadığı durumlarda kullanıcıya etkin sonuç sağlanamayabilir. Eğer arama motoru veritabanı hazırlanmasında Evrensel web robotu kullanılacak ise etkin bir çevirici kullanılmalıdır.

Arama motorlarında kullanıcıya etkin bir sonuç verilebilmesi için, konu seçimli ve konu odaklı web robotlarının kullanılarak arama indeksi üzerinde konu bazlı gruplandırma yapılabilmelidir. Kullanıcı arama sonuçlarında tekrar kendi bir arama

işlemi gerçekleştirmeksizin ilgili kelime ile alakalı olarak listelenen konulardan erişmek istediğini tıklayarak, etkin bir sonuç listesine ulaşabilir. Böylece hem kullanıcıya en kısa zamanda bilgiye erişim sağlanılmış, hem de gereksiz ağ trafiğinin oluşması engellenmiş olacaktır.

KAYNAKLAR

[1] V. Shkapenyuk and T. Suel**,** Design and Implementation of a High-Performance Distributed Web Crawler, IEEE International Conference on Data Engineering, February 2002

[2] S. H. Lee, S. J. Kim and S. H. Hong, On URL Normalization, Proceedings of the International Conference on Computational Science and its Applications, 2005

[3] E. G. Coffman, Z. Liu and R. Weber, Optimal Robot Scheduling for Web Search Engines, Journal of Scheduling, 1998

[4] Cho J. and H. Garcia-Molina, Parallel Crawlers , In Proceedings of the 11th International Conference on World Wide Web, USA, 2002

[5] S. Brin and L. Page, The Anatomy of a Large-Scale Hyper-Textual Web Search Engine, Computer Networks and ISDN Systems, 1998

[6] S. Chakrabarti, Mining the Web, Morgan Kaufmann Publishers, ISBN 1-55860-754-4, 2003

[7] J. Tsay, AuToCrawler: An Integrated System for Automatic Topical Crawler, Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05), 2005

[8] Raggett, Dave. Le Hours, Arnaud. Jacobs, Ian. HTML 4.01 Specification World Wide Web Consortium, 1999. http://www.w3.org/TR/1999/PR-html40-19990824/

[9] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C.M.; Maler, Eve; Yergeau, François, editors. Extensible Markup Language (XML) 1.0 (3rd Edition). World Wide Web Consortium, 2006. http://www.w3.org/TR/2006/REC-xml-20060816/

[10] Cho J. and H.Garcia-Molina, Parallel Crawlers , In Proceedings of the 11th International Conference on World Wide Web, USA, 2002

[11] J. Tsay, AuToCrawler: An Integrated System for Automatic Topical Crawler, Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05), 2005

[12] Naper HTML to XML Conversion Utility.Naper Solutions, 2003.http://www.napersolutions.com/htmltoxml.html [02/24/2005]

[13] Potok, Thomas;Elmore, Mark;Reed Stylus Studio HTML to XML Importer. Stylus Studio, 2004. http://www.stylusstudio.com/html_to_xml.html [02/24/2005]

[14] Raggett, Dave. XHTML: The Extensible Hypertext Markup Language by Dave Raggett, W3C LA event in Stockholm, 24 March 1999.

[15] Box, Don. Enhebuske David, Kakivaya,Gopal. Leyman, Andrew. Mendelsohn, Noah. Nielsen H. Frystk, Thatte,Satish. Winer, Dave. Simple Process Access Control (SOAP), World Wide Web Consortium, 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[16] Le Hegarret, Philippe. Whitmer, Ray. Wood, Lauren. Document Object Model (DOM), http://www.w3.org/DOM/, 2006/06/12

TEŞEKKÜR

Tez konumun seçiminde ve geri kalan süreçte danışmanlığımı yapan, desteğini esirgemeyen, bana her konuda yardımcı olan Yrd. Doç. Dr. Aydın CARUS'a teşekkürlerimi sunarım.

ÖZGEÇMİŞ

Kişisel

Adı Soyadı	: Eyüp Can DÜNDAR
Doğum Tarihi	: 28.06.1983
Doğum Yeri	: Çorlu
Medeni Hali	: Bekâr
T.C. Kimlik No	: 42637181462

Eğitim

2006 -	Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümü
2006 – 2001	Trakya Üniversitesi Mühendislik Mimarlık Fakültesi Bilgisayar Mühendisliği
2001 – 1994	Çorlu, Mehmet Akif Ersoy Anadolu Lisesi

İş Deneyimi

2005 -	Prestij Bilgisayar Sistemleri, Genel Md. Yrd.

Dil

İyi derece İngilizce (Okuma, Konuşma, Yazma)

EK A – HTML DÜZENLEYİCİ SINIF

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.IO;
using System.Collections;
namespace MyParser
{
    public class HTMLParser
    {
        public DateTime HTMLLastModified;
        public bool HTMLEmpty = true;
        public string HTMLString = "";
        public string HTMLEncoding = "";
        public ArrayList HTMLLinks = new ArrayList();
        public ArrayList HTMLControlWords = new ArrayList();
        public string HTMLRealAdress = "";
        public string HTMLAdress = "";
        public string HTMLServer = "";
        public string HTMLTitle = "";
        private bool HTMLParsed;
        private void BlankPage()
        {
            HTMLString = "";
            HTMLEncoding = "";
            HTMLRealAdress = "";
            HTMLEmpty = true;
            HTMLParsed = false;
        }
        public HTMLParser(string HTMLStr, ArrayList HTMLWords)
        {
            if (HTMLStr.Trim() != "")
```

```
            {
                try
                {
                    HTMLAdress = HTMLStr;
                    HttpWebRequest req =
(HttpWebRequest)WebRequest.Create(HTMLStr);
                    HttpWebResponse resp = (HttpWebResponse)req.GetResponse();
                    Stream istrm = resp.GetResponseStream();
                    StreamReader rdr = new StreamReader(istrm,
System.Text.Encoding.GetEncoding(resp.CharacterSet));
                    HTMLString = rdr.ReadToEnd();
                    HTMLEncoding = resp.CharacterSet;
                    string str;
                    int i;
                    str = "http://" + resp.ResponseUri.Host;
                    for (i = 0; i < 1; i++)
                    {
                        if (resp.ResponseUri.Segments[i].EndsWith("/") ==
true)
                        {
                            str = str +
resp.ResponseUri.Segments[i].ToString();
                        }
                        else
                        {
                            break;
                        }
                    }
                    HTMLRealAdress = str;
                    HTMLAdress = HTMLRealAdress;
                    HTMLServer = resp.Server.ToString();
                    HTMLLastModified = resp.LastModified;
                    HTMLEmpty = false;
                    HTMLParsed = false;
                    HTMLControlWords = HTMLWords;
```

```
                }
                catch
                {
                    BlankPage();
                }
            }
            else
            {
                BlankPage();
            }
        }
        public bool ParseHTML()
        {
            HTMLTitle = GetTitle();
            GetLinks();
            HTMLParsed = true;
            return HTMLParsed;
        }
        private string GetTitle()
        {
            int titlestart, titleend;
            titlestart = HTMLString.IndexOf("<title>");
            titleend = HTMLString.IndexOf("</title>");
            if ((titlestart != -1) || (titleend != -1) && (((titleend -
titlestart) - 7) > 0))
            {
                return HTMLString.Substring(titlestart + 7, (titleend -
titlestart) - 7);
            }
            else
            {
                return "";
            }
        }
```

```
        private void GetLinks()
        {
            int point = 0;
            int start = 0;
            int stop = 0;
            do
            {
                try
                {
                    start = HTMLString.ToLower().IndexOf("<a href", point);
                    stop = HTMLString.ToLower().IndexOf("</a>", start);
                    string str = "";
                    str = HTMLString.Substring(start, stop - start +
4).Replace("\"", "'");
                    int i;
                    bool hasword = false;
                    if (HTMLControlWords.Count == 0)
                    {
                        hasword = true;
                    }
                    else
                    {
                        for (i = 0; i<HTMLControlWords.Count; i++)
                        {
                            if
((str.ToLower().IndexOf(HTMLControlWords[i].ToString().ToLower()) != -1))
                            {
                                hasword = true;
                                break;
                            }
                        }
                    }
                    if (hasword == true)
                    {
```

```
                        str = LinkToStr(str);
                        if ((str.IndexOf("javascript") == -1) && (str != "")
&& (str.IndexOf("#") == -1) && (HTMLAdress.Replace("/",
"").EndsWith(str.Replace("/", "")) == false))
                        {
                            if (HTMLLinks.Contains(str) == false)
                            {
                                HTMLLinks.Add(str);
                                veritabaninaekle(str, hasword);
                            }
                        }
                    }
                    point = stop+4;
                }
                catch
                {
                    start = -1;
                    stop = -1;
                }
            } while ((start != -1) || (stop != -1));
        }
        private void veritabaninaekle(string str, bool deger)
        {
            string sayi = "";
            try
            {
                if (deger==true)
                {
                    sayi = "1";
                }
                else
                {
                    sayi = "0";
                }
```

```
                System.Data.OleDb.OleDbConnection con = new
System.Data.OleDb.OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=c:\\vt1.mdb");

                con.Open();

                System.Data.OleDb.OleDbCommand cmd = new
System.Data.OleDb.OleDbCommand("insert into links(link,hasword) values('" +
str + "','" + sayi + "')", con);

                cmd.ExecuteNonQuery();

                con.Close();

            }

            catch

            {

            }

        }

        private string LinkToStr(string str)

        {

            try

            {

                int start = 0;

                int stop = 0;

                //javascript:openWin(http://sondakika.haber7.com,,toolbar=0,

                str = str.Replace("javascript:openWin(", "").Replace(",", "
").Replace(">"," ");



                start = str.ToLower().IndexOf("f=") + 2;

                stop = str.ToLower().IndexOf(" ", start);

                str = str.Substring(start, (stop - start)).Replace("'", "");

                str = str.Replace("'", "").Replace("\"", "");

                if (str.Length < 3)

                {

                    str = "";

                    return "";

                }

                if (HTMLAdress.EndsWith(str.Replace("\\", "").Replace("//",
"/").Replace(":/", "://").Replace("//", "/").Replace("http:/", "http://")))

                {
```

```
                str = "";
                return "";
            }
            if (str.IndexOf("admentor") != -1)
            {
                str = str;
            }
            if (str.IndexOf("//") == -1)
            {
                if (HTMLAdress.EndsWith(str) == true)
                {
                    str = "";
                    return "";
                }
                else
                {
                    str = HTMLAdress + '/' + str;
                }
            }
            str = str.Replace("\\", "").Replace("//", "/").Replace(":/",
"://").Replace("//", "/").Replace("http:/", "http://"); ;
            if ((str.IndexOf("mailto") == -1) && (str.IndexOf(".exe") == -
1) && (str.IndexOf(".jpg") == -1) && (str.IndexOf(".bmp") == -1) &&
(str.IndexOf(".png") == -1) && (str.IndexOf(".gif") == -1))
            {
                return str;
            }
            else
            {
                return "";
            }
        }
        catch
        {
            return "";
```

```
            }
        }

        public bool  DownloadHTML(string FileName,string directory)
        {
           string url;
            bool sonuc=false;

            url = FileName.Replace("http://", "").Replace("/",
"").Replace("?", "").Replace(":", "").Replace("=", "").Replace("*","") +
".html";
            try
            {
                try
                {
                   /* FileStream file = new FileStream(directory + url,
FileMode.CreateNew, FileAccess.Write);
                    StreamWriter sw = new StreamWriter(file);
                    sw.Write(HTMLString);
                    sw.Close();
                    file.Close(); */
                    try
                    {
                       /* FileStream file1 = new FileStream(directory +
"links.txt", FileMode.Append , FileAccess.Write);
                        StreamWriter sw1 = new StreamWriter(file1);
                        sw1.WriteLine(FileName);
                        sw1.Close();
                        file1.Close();*/
                    }
                    catch
                    {
                    }
                    sonuc= true ;
```

```
                }
                catch
                {
                    /*FileStream file = new FileStream(FileName,
FileMode.CreateNew, FileAccess.Write);
                    StreamWriter sw = new StreamWriter(file);
                    sw.Write("");
                    sw.Close();*/
                    sonuc= false;
                }
            }
            catch { }
            return sonuc;
        }
    }
}
```

EK B – HTML'DEN XML'E ÇEVİRİCİ

```
Imports System.Net

Imports System.Xml


Public Class Form1

    Dim chr1 As String

    Private Sub btnopen_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnopen.Click

        Dim req As System.Net.HttpWebRequest

        req = System.Net.WebRequest.Create(txtadress.Text)

        Dim resp As HttpWebResponse

        resp = req.GetResponse()

        Dim istrm As System.IO.Stream

        istrm = resp.GetResponseStream()

        Dim rdr As System.IO.StreamReader

        rdr = New System.IO.StreamReader(istrm,
System.Text.Encoding.GetEncoding(resp.CharacterSet))

        txthtml.Text = rdr.ReadToEnd()

        chr1 = resp.CharacterSet

    End Sub


    Private Sub txtadress_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles txtadress.TextChanged

        If txtadress.Text.Trim = "" Then btnopen.Enabled = False Else
btnopen.Enabled = True

    End Sub


    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        ProgressBar1.Maximum = txthtml.Text.Length + 1

        Dim status, sol, sag As Integer

        status = 0

        Dim str As String

        Try

            str = txthtml.Text.Replace(Chr(13), "").Replace(Chr(10), "")
```

```
        Catch ex As Exception

            str = txthtml.Text

        End Try


        Do

            sol = str.IndexOf("<", status)

            If sol > status + 1 Then

                If str.Substring(status, sol - status).Replace(Chr(13),
"").Length > 0 Then

                    TextBox2.Text = TextBox2.Text & str.Substring(status, sol
- status) & vbNewLine

                End If

            End If

            sag = str.IndexOf(">", sol)

            If str.Substring(sol, sag - sol + 1).Replace(Chr(13), "").Length >
0 Then

                TextBox2.Text = TextBox2.Text & str.Substring(sol, sag - sol +
1) & vbNewLine

            End If

            status = sag + 1

            'Application.DoEvents()

            ProgressBar1.Value = status

            ProgressBar1.Refresh()

        Loop While (str.Length > status)

    End Sub


    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click

        End

    End Sub

    Dim h As New Chilkat.HtmlToXml

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

        h.UnlockComponent("Anything begins the 30-day trial")

        h.XmlCharset = chr1

        h.Html = txthtml.Text
```

```
        TextBox2.Text = h.ToXml()

        h.WriteStringToFile(TextBox2.Text, "c:\html.xml", chr1)

    End Sub


    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click

        If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then

            h.WriteStringToFile(TextBox2.Text, SaveFileDialog1.FileName, chr1)

        End If

    End Sub

    Dim xdoc As New Xml.XmlDocument

    Dim xmlyeni As XmlTextWriter


    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click

        Dim objReader As New System.IO.StreamReader("c:\html.xml")

        Dim reader As New System.Xml.XmlTextReader("c:\html.xml")




        Dim str, str2 As String

        While Not objReader.EndOfStream

            str = objReader.ReadLine()

            If (str.Trim().StartsWith("<img")) Or
(str.Trim().StartsWith("</img")) Or (str.Trim().StartsWith("<a")) Or
(str.Trim().StartsWith("</a")) Or (str.Trim().StartsWith("<text")) Or
(str.Trim().StartsWith("</text")) Then

                If str.Contains("CDATA") = False Then str2 = str2 & str.Trim()

            End If

        End While

        TextBox2.Text = str2



        str2 = str2.Replace("<text>", "")

        str2 = str2.Replace("</text>", "")




        Dim h1 As New MyNameSpace.HtmlToXml
```

```
        h1.XmlCharset = chr1

        h1.Html = str2

        h1.DropCustomTags = True

        str2 = h1.ToXml()

        TextBox2.Text = str2

        h1.WriteStringToFile(str2.Replace("<!", "").Replace("[CDATA",
"").Replace("[", "").Replace("]]>", ""), "c:\bc.xml", chr1)


    End Sub

    Dim soneleman As Xml.XmlElement

    Dim xdoc2 As New Xml.XmlDocument

End Class
```

EK C – LINK TABANLI WEB ROBOTU

```
import java.applet.Applet;
import java.text.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.io.*;
public class WebCrawler extends Applet implements ActionListener, Runnable {
    public static final String SEARCH = "Search";
    public static final String STOP = "Stop";
    public static final String DISALLOW = "Disallow:";
    public static final int    SEARCH_LIMIT = 50;
    Panel   panelMain;
    List    listMatches;
    Label   labelStatus;
    // URLs to be searched
    Vector vectorToSearch;
    // URLs already searched
    Vector vectorSearched;
    // URLs which match
    Vector vectorMatches;
    Thread searchThread;
    TextField textURL;
    Choice    choiceType;
    public void init() {
       // set up the main UI panel
       panelMain = new Panel();
       panelMain.setLayout(new BorderLayout(5, 5));

       // text entry components
       Panel panelEntry = new Panel();
       panelEntry.setLayout(new BorderLayout(5, 5));
```

```
Panel panelURL = new Panel();
panelURL.setLayout(new FlowLayout(FlowLayout.LEFT, 5, 5));
Label labelURL = new Label("Starting URL: ", Label.RIGHT);
panelURL.add(labelURL);
textURL = new TextField("", 40);
panelURL.add(textURL);
panelEntry.add("North", panelURL);

Panel panelType = new Panel();
panelType.setLayout(new FlowLayout(FlowLayout.LEFT, 5, 5));
Label labelType = new Label("Content type: ", Label.RIGHT);
panelType.add(labelType);
choiceType = new Choice();
choiceType.addItem("text/html");
choiceType.addItem("audio/basic");
choiceType.addItem("audio/au");
choiceType.addItem("audio/aiff");
choiceType.addItem("audio/wav");
choiceType.addItem("video/mpeg");
choiceType.addItem("video/x-avi");
panelType.add(choiceType);
panelEntry.add("South", panelType);
panelMain.add("North", panelEntry);
// list of result URLs
Panel panelListButtons = new Panel();
panelListButtons.setLayout(new BorderLayout(5, 5));
Panel panelList = new Panel();
panelList.setLayout(new BorderLayout(5, 5));
Label labelResults = new Label("Search results");
panelList.add("North", labelResults);
Panel panelListCurrent = new Panel();
panelListCurrent.setLayout(new BorderLayout(5, 5));
```

```
    listMatches = new List(10);
    panelListCurrent.add("North", listMatches);
    labelStatus = new Label("");
    panelListCurrent.add("South", labelStatus);
    panelList.add("South", panelListCurrent);
    panelListButtons.add("North", panelList);
    // control buttons
    Panel panelButtons = new Panel();
    Button buttonSearch = new Button(SEARCH);
    buttonSearch.addActionListener(this);
    panelButtons.add(buttonSearch);
    Button buttonStop = new Button(STOP);
    buttonStop.addActionListener(this);
    panelButtons.add(buttonStop);
    panelListButtons.add("South", panelButtons);
    panelMain.add("South", panelListButtons);
    add(panelMain);
    setVisible(true);
    repaint();
    // initialize search data structures
    vectorToSearch = new Vector();
    vectorSearched = new Vector();
    vectorMatches = new Vector();
    // set default for URL access
    URLConnection.setDefaultAllowUserInteraction(false);
}
public void start() {
}
public void stop() {
    if (searchThread != null) {
        setStatus("stopping...");
        searchThread = null;
    }
```

```
}
public void destroy() {
}
boolean robotSafe(URL url) {
   String strHost = url.getHost();

   // form URL of the robots.txt file
   String strRobot = "http://" + strHost + "/robots.txt";
   URL urlRobot;
   try {
       urlRobot = new URL(strRobot);
   } catch (MalformedURLException e) {
       // something weird is happening, so don't trust it
       return false;
   }

   String strCommands;
   try {
       InputStream urlRobotStream = urlRobot.openStream();

       // read in entire file
       byte b[] = new byte[1000];
       int numRead = urlRobotStream.read(b);
       strCommands = new String(b, 0, numRead);
       while (numRead != -1) {
          if (Thread.currentThread() != searchThread)
              break;
          numRead = urlRobotStream.read(b);
          if (numRead != -1) {
              String newCommands = new String(b, 0, numRead);
              strCommands += newCommands;
          }
       }
```

```
            urlRobotStream.close();
        } catch (IOException e) {
            // if there is no robots.txt file, it is OK to search
            return true;
        }

        // assume that this robots.txt refers to us and
        // search for "Disallow:" commands.
        String strURL = url.getFile();
        int index = 0;
        while ((index = strCommands.indexOf(DISALLOW, index)) != -1) {
            index += DISALLOW.length();
            String strPath = strCommands.substring(index);
            StringTokenizer st = new StringTokenizer(strPath);

            if (!st.hasMoreTokens())
               break;

            String strBadPath = st.nextToken();

            // if the URL starts with a disallowed path, it is not safe
            if (strURL.indexOf(strBadPath) == 0)
               return false;
        }

        return true;
    }

    public void paint(Graphics g) {
        //Draw a Rectangle around the applet's display area.
        g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);

        panelMain.paint(g);
```

```
    panelMain.paintComponents(g);
    // update(g);
    // panelMain.update(g);
}

public void run() {
    String strURL = textURL.getText();
    String strTargetType = choiceType.getSelectedItem();
    int numberSearched = 0;
    int numberFound = 0;

    if (strURL.length() == 0) {
        setStatus("ERROR: must enter a starting URL");
        return;
    }

    // initialize search data structures
    vectorToSearch.removeAllElements();
    vectorSearched.removeAllElements();
    vectorMatches.removeAllElements();
    listMatches.removeAll();

    vectorToSearch.addElement(strURL);

    while ((vectorToSearch.size() > 0)
      && (Thread.currentThread() == searchThread)) {
        // get the first element from the to be searched list
        strURL = (String) vectorToSearch.elementAt(0);

        setStatus("searching " + strURL);

        URL url;
        try {
```

```
   url = new URL(strURL);
} catch (MalformedURLException e) {
   setStatus("ERROR: invalid URL " + strURL);
   break;
}

// mark the URL as searched (we want this one way or the other)
vectorToSearch.removeElementAt(0);
vectorSearched.addElement(strURL);

// can only search http: protocol URLs
if (url.getProtocol().compareTo("http") != 0)
   break;

// test to make sure it is before searching
if (!robotSafe(url))
   break;

try {
   // try opening the URL
   URLConnection urlConnection = url.openConnection();

   urlConnection.setAllowUserInteraction(false);

   InputStream urlStream = url.openStream();
   String type
     = urlConnection.guessContentTypeFromStream(urlStream);
   if (type == null)
       break;
   if (type.compareTo("text/html") != 0)
       break;

   // search the input stream for links
```

```
// first, read in the entire URL
byte b[] = new byte[1000];
int numRead = urlStream.read(b);
String content = new String(b, 0, numRead);
while (numRead != -1) {
    if (Thread.currentThread() != searchThread)
       break;
    numRead = urlStream.read(b);
    if (numRead != -1) {
       String newContent = new String(b, 0, numRead);
       content += newContent;
    }
}
urlStream.close();

if (Thread.currentThread() != searchThread)
    break;

String lowerCaseContent = content.toLowerCase();

int index = 0;
while ((index = lowerCaseContent.indexOf("<a", index)) != -1)
{
    if ((index = lowerCaseContent.indexOf("href", index)) == -1)
       break;
    if ((index = lowerCaseContent.indexOf("=", index)) == -1)
       break;

    if (Thread.currentThread() != searchThread)
       break;

    index++;
    String remaining = content.substring(index);
```

```
                    StringTokenizer st
                      = new StringTokenizer(remaining, "\t\n\r\">#");
                    String strLink = st.nextToken();

                    URL urlLink;
                    try {
                       urlLink = new URL(url, strLink);
                       strLink = urlLink.toString();
                    } catch (MalformedURLException e) {
                       setStatus("ERROR: bad URL " + strLink);
                       continue;
                    }

                    // only look at http links
                    if (urlLink.getProtocol().compareTo("http") != 0)
                       break;

                    if (Thread.currentThread() != searchThread)
                       break;

                    try {
                       // try opening the URL
                       URLConnection urlLinkConnection
                         = urlLink.openConnection();
                       urlLinkConnection.setAllowUserInteraction(false);
                       InputStream linkStream = urlLink.openStream();
                       String strType
                         =
urlLinkConnection.guessContentTypeFromStream(linkStream);
                       linkStream.close();

                       // if another page, add to the end of search list
                       if (strType == null)
```

```
                break;
            if (strType.compareTo("text/html") == 0) {
                // check to see if this URL has already been
                // searched or is going to be searched
                if ((!vectorSearched.contains(strLink))
                  && (!vectorToSearch.contains(strLink))) {

                   // test to make sure it is robot-safe!
                   if (robotSafe(urlLink))
                       vectorToSearch.addElement(strLink);
                }
            }

            // if the proper type, add it to the results list
            // unless we have already seen it
            if (strType.compareTo(strTargetType) == 0) {
                if (vectorMatches.contains(strLink) == false) {
                   listMatches.add(strLink);
                   vectorMatches.addElement(strLink);
                   numberFound++;
                   if (numberFound >= SEARCH_LIMIT)
                       break;
                }
            }
        } catch (IOException e) {
            setStatus("ERROR: couldn't open URL " + strLink);
            continue;
        }
    }
} catch (IOException e) {
    setStatus("ERROR: couldn't open URL " + strURL);
    break;
}
```

```
            numberSearched++;
            if (numberSearched >= SEARCH_LIMIT)
               break;
        }

        if (numberSearched >= SEARCH_LIMIT || numberFound >= SEARCH_LIMIT)
            setStatus("reached search limit of " + SEARCH_LIMIT);
        else
            setStatus("done");
        searchThread = null;
        // searchThread.stop();
    }

    void setStatus(String status) {
        labelStatus.setText(status);
    }

    public void actionPerformed(ActionEvent event) {
        String command = event.getActionCommand();

        if (command.compareTo(SEARCH) == 0) {
            setStatus("searching...");

            // launch a thread to do the search
            if (searchThread == null) {
               searchThread = new Thread(this);
            }
            searchThread.start();
        }
        else if (command.compareTo(STOP) == 0) {
            stop();
        }
```

```
    }
        public static void main (String argv[])
        {
                Frame f = new Frame("WebFrame");
                WebCrawler applet = new WebCrawler();
             f.add("Center", applet);

/*           Behind a firewall set your proxy and port here!
*/
                Properties props= new Properties(System.getProperties());
                props.put("http.proxySet", "true");
             props.put("http.proxyHost", "webcache-cup");
             props.put("http.proxyPort", "8080");

                Properties newprops = new Properties(props);
                System.setProperties(newprops);
/**/


                applet.init();
                applet.start();
                f.pack();
                f.show();
        }

}
```

EK D – KONU SEÇİMLİ WEB ROBOTU

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.IO;
using System.Text.RegularExpressions;
using System.Collections;
using System.Threading;
using System.Collections;



namespace ParserDeneme
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        public static ArrayList adresses = new ArrayList();
        public static ArrayList setwords = new ArrayList();

        private void txtlink_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                btnopen_Click(sender,null);
```

```
    }
}
private void txthtml_KeyDown(object sender, KeyEventArgs e)
{
    e.Handled = true;
}
private void txthtml_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true;
}
private void button1_Click(object sender, EventArgs e)
{
    if (textBox2.Text.Trim() != "")
    {
        int i;
        Boolean kelimevar = false;

        for (i = 0; i < listBox1.Items.Count; i++)
        {
            if (listBox1.Items[i].ToString() == textBox2.Text.Trim())
            {
                kelimevar = true;
            }
        }
        if (kelimevar == false)
        {
            listBox1.Items.Add(textBox2.Text.Trim());
        }
    }
    textBox2.Text = "";
}
private void listBox1_KeyDown(object sender, KeyEventArgs e)
{
```

```
    try
    {
        if ((e.KeyCode == Keys.Delete) && (listBox1.Items.Count > 0))
        {
            listBox1.Items.RemoveAt(listBox1.SelectedIndex);
        }
    }
    catch { }
}
private void button3_Click(object sender, EventArgs e)
{
    if (folderBrowserDialog1.ShowDialog()==DialogResult.OK )
    {
        textBox1.Text = folderBrowserDialog1.SelectedPath.ToString();
    }
}
private void textBox1_KeyDown(object sender, KeyEventArgs e)
{
    e.Handled = true;
}
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true;
}
string CrawlerDirectory = "";
TimeSpan starttime;
ArrayList backuplinks = new ArrayList();
ArrayList words = new ArrayList() ;
int linkcount = 0;
int downloaded = 0;
int maximum = 0;
Thread Crawler;
private void button2_Click(object sender, EventArgs e)
```

```
        {
            if (txtlink.Text.Trim() == "") { MessageBox.Show("Start adress is
empty !"); return; }
            if (textBox1.Text.Trim() == "") { MessageBox.Show("Please select
folder !"); return; }
            groupBox4.Enabled = false;
            groupBox3.Enabled = false;
            groupBox6.Enabled = false;
            maximum = (int)numericUpDown1.Value + 1;
            CheckForIllegalCrossThreadCalls = false;

            timer1.Enabled = true;
            if (System.IO.Directory.Exists(textBox1.Text) == false)
            {
                System.IO.Directory.CreateDirectory(textBox1.Text);
            }
            CrawlerDirectory = textBox1.Text + "/";
            int i;
            for (i = 0; i < listBox1.Items.Count; i++)
            {
                words.Add(listBox1.Items[i].ToString());
            }
            Crawler = new Thread(new ThreadStart(crawl));
            Crawler.Start();
            lblstarttime.Text = DateTime.Now.ToLongTimeString();
            lblthreadcount.Text = "1";
            lbllastpage.Text = txtlink.Text;
            starttime = DateTime.Now.TimeOfDay;
            button2.Enabled = false;
            button5.Enabled = true;
            button4.Enabled = true;

        }
```

```
        private void crawl()
        {
            int Links=0;
            string url;
            url = txtlink.Text;
            ArrayList linkler = new ArrayList();
            do
            {
                if (url == "")
                {
                    url = backuplinks[0].ToString();
                    backuplinks.RemoveAt(0);
                }
                if (linkler.Contains(url) == false)
                {
                    linkler.Add(url);
                    if (File.Exists(CrawlerDirectory + url.Replace("http://",
"").Replace("/", "").Replace("?", "").Replace(":", "").Replace("=", "") +
".html") == false)
                    {
                        try
                        {
                            lbllastpage.Text = url;
                            lblbackuplinks.Text =
backuplinks.Count.ToString();
                            lbllinkscount.Text = linkcount.ToString();
                            lbldowloaded.Text = downloaded.ToString();
                        }
                        catch { }
                        try
                        {
                            MyParser.HTMLParser z = new
MyParser.HTMLParser(url, words);
                            z.ParseHTML();
                            if (z.HTMLEmpty == false)
```

```
                {
                    if (z.DownloadHTML(url, CrawlerDirectory) ==
true)
                    {
                        downloaded += 1;
                    }
                    linkcount += 1;
                    url = z.HTMLLinks[0].ToString();
                    z.HTMLLinks.RemoveAt(0);
                    if (z.HTMLLinks.Count > 0)
                    {
                        int i;
                        for (i = 0; i < z.HTMLLinks.Count; i++)
                        {
                            if (File.Exists(CrawlerDirectory +
z.HTMLLinks[i].ToString().Replace("http://", "").Replace("/", "").Replace("?",
"").Replace(":", "").Replace("=", "") + ".html") == false)
                            {
                                int j;
                                if
(backuplinks.Contains(z.HTMLLinks[i].ToString()) == false)
                                {

backuplinks.Add(z.HTMLLinks[i].ToString());
                                    linkcount += 1;
                                }
                            }
                        }
                    }
                }
                else
                {
                    Links = 0;
                    url = "";
                }
            }
```

```
                    catch
                    {
                        Links = 0;
                        url = "";
                    }
                }
                else
                {
                    Links = 0;
                    url = "";
                }
            }
            else
            {
                url = "";
            }
        } while (((backuplinks.Count > 0) || (url!="")) &&
(maximum+1)>downloaded);
        button4.Enabled = false;
        button5.Enabled = false;
        timer1.Enabled = false;
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        lblrunningtime.Text =
DateTime.Now.TimeOfDay.Subtract(starttime).ToString().Substring(0,8);
    }

    private void button5_Click(object sender, EventArgs e)
    {
        Crawler.Abort();
        button5.Enabled = false;
        timer1.Enabled = false;
```

```
        }


        private void button4_Click(object sender, EventArgs e)

        {

            if (button4.Text.Trim() == "PAUSE") { Crawler.Suspend();
button4.Text = "PLAY"; button5.Enabled = false; }

            if (button4.Text.Trim() == "PLAY") { Crawler.Resume();
button5.Enabled = true; }

        }



        private void button6_Click(object sender, EventArgs e)

        {



            listBox1.Items.Clear();

            int i;

            for (i = 0; i < setwords.Count ; i++)

            {

                if (setwords[i].ToString().Replace("\n", "").Replace(":",
"").Replace("?", "").Replace("\t","").Length > 4)

                {

                    listBox1.Items.Add(setwords[i].ToString().Replace("\n",
"").Replace(":", "").Replace("?", "").Replace("\t",
"").Replace("&","").Replace("#",""));

                }

            }

            txtlink.Text = adresses[0].ToString();

            for (i = 1; i < adresses.Count; i++)

            {

                backuplinks.Add(adresses[i].ToString());

            }

        }

    }

}
```

EK E – RSS OKUYUCU SINIF

```
using System;

using System.Xml;

using System.IO;

using System.Collections;

namespace RSSReader

{

#region Event datatype/delegate

public class RssReaderErrorEventArgs : EventArgs

{

public string Message

{

get

{

return this.message;

}

set

{

this.message = value;

}

}

private string message;

}

public delegate void RssReaderErrorEventHandler(object sender,
RssReaderErrorEventArgs e);

#endregion

#region RssReader class

public class RssReader

{

public event EventHandler FeedLoaded;

public event EventHandler RssNodeFound;

public event EventHandler ChannelNodeFound;

public event EventHandler ItemAdded;
```

```
public event RssReaderErrorEventHandler Error;


public string RootNodeName
{
get
{
return this.rootNodeName;
}
set
{
this.rootNodeName = value;
}
}
public string ChannelNodeName
{
get
{
return this.channelNodeName;
}
set
{
this.channelNodeName = value;
}
}
public bool RdfMode
{
get
{
return this.rdfMode;
}
set
{
if ( value )
```

```
{
this.rootNodeName = "rdf:RDF";
}
Else
{
this.rootNodeName = "rss";
}
this.rdfMode = value;
}
}


private string rootNodeName = "rss";
private string channelNodeName = "channel";
private bool rdfMode = false;
public static RssFeed GetFeed(string Url,bool RdfFormat)
{
RssReader rssReader = new RssReader();
rssReader.RdfMode = RdfFormat;
return rssReader.Retrieve(Url);
}
public static RssFeed GetFeed(string Url)
{
RssReader rssReader = new RssReader();
return rssReader.Retrieve(Url);
}
public static string CreateHtml(RssFeed Feed,string Template,string
ItemPrefix,string ItemSuffix)
{
return new
RssHtmlMaker().GetHtmlContents(Feed,Template,ItemPrefix,ItemSuffix);
}
public static string CreateHtml(RssFeed Feed,string Template,string
ItemPrefix,string ItemSuffix,int MaxItems)
{
RssHtmlMaker rssHtmlMaker = new RssHtmlMaker();
```

```
rssHtmlMaker.MaxItems = MaxItems;

return rssHtmlMaker.GetHtmlContents(Feed,Template,ItemPrefix,ItemSuffix);

}

public RssFeed Retrieve(string Url)

{


RssFeed rssFeed = new RssFeed();

rssFeed.Items = new RssItems();

XmlTextReader xmlTextReader = new XmlTextReader(Url);

XmlValidatingReader xmlValidatingReader = new
XmlValidatingReader(xmlTextReader);

xmlValidatingReader.ValidationType = ValidationType.None;

XmlDocument xmlDoc= new XmlDocument();

try

{

xmlDoc.Load(xmlTextReader);


if ( this.FeedLoaded != null )

{

this.FeedLoaded(this, new EventArgs());

}


XmlNode rssXmlNode = null;


for (int i=0;i < xmlDoc.ChildNodes.Count;i++)

{

System.Diagnostics.Debug.Write("Child: " +xmlDoc.ChildNodes[i].Name);

System.Diagnostics.Debug.WriteLine(" has "
+xmlDoc.ChildNodes[i].ChildNodes.Count+" children");


if ( xmlDoc.ChildNodes[i].Name == this.rootNodeName &&
xmlDoc.ChildNodes[i].ChildNodes.Count > 0 )

{

rssXmlNode = xmlDoc.ChildNodes[i];
```

```
// Fire the found event
if ( this.RssNodeFound != null )
{
this.RssNodeFound(this,new EventArgs());
}
break;
}
}

if ( rssXmlNode != null )
{
XmlNode channelXmlNode = null;

for (int i=0;i < rssXmlNode.ChildNodes.Count;i++)
{
System.Diagnostics.Debug.WriteLine("Rss child:
"+rssXmlNode.ChildNodes[i].Name);
if ( rssXmlNode.ChildNodes[i].Name == this.channelNodeName &&
rssXmlNode.ChildNodes[i].ChildNodes.Count > 0 )
{
channelXmlNode = rssXmlNode.ChildNodes[i];

if ( this.ChannelNodeFound != null )
{
this.ChannelNodeFound(this,new EventArgs());
}

break;
}
}
if ( channelXmlNode != null )
{
for (int i=0;i < channelXmlNode.ChildNodes.Count;i++)
{
```

```
System.Diagnostics.Debug.WriteLine(channelXmlNode.ChildNodes[i].Name);
switch ( channelXmlNode.ChildNodes[i].Name )
{
case "title":
{
rssFeed.Title = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "description":
{
rssFeed.Description = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "language":
{
rssFeed.Language = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "copyright":
{
rssFeed.Copyright = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "webmaster":
{
rssFeed.Webmaster = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "pubDate":
{
rssFeed.PubDate = channelXmlNode.ChildNodes[i].InnerText;
break;
}
```

```
case "lastBuildDate":
{
rssFeed.LastBuildDate = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "category":
{
rssFeed.Category = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "generator":
{
rssFeed.Generator = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "ttl":
{
rssFeed.Ttl = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "rating":
{
rssFeed.Rating = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "skipHours":
{
rssFeed.Skiphours = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "skipDays":
{
rssFeed.Skipdays = channelXmlNode.ChildNodes[i].InnerText;
```

```
break;
}
case "managingEditor":
{
rssFeed.ManagingEditor = channelXmlNode.ChildNodes[i].InnerText;
break;
}
case "item":
{
rssFeed.Items.Add( this.getRssItem(channelXmlNode.ChildNodes[i]) );
if ( this.ItemAdded != null )
{
this.ItemAdded(this,new EventArgs());
}

break;
}
}

}

if ( this.RdfMode )
{
for (int i=0;i < rssXmlNode.ChildNodes.Count;i++)
{
switch ( rssXmlNode.ChildNodes[i].Name )
{
case "item":
{
rssFeed.Items.Add( this.getRssItem(rssXmlNode.ChildNodes[i]) );
if ( this.ItemAdded != null )
{
this.ItemAdded(this,new EventArgs());
```

```
}
break;
}
}
}
}
}
else
{
rssFeed.ErrorMessage = "Unable to find rss <seehannel> node";
if ( this.Error != null )
{
RssReaderErrorEventArgs args = new RssReaderErrorEventArgs();
args.Message = rssFeed.ErrorMessage;
this.Error(this,args);
}
}
}
else
{
rssFeed.ErrorMessage = "Unable to find root <rss> node";
if ( this.Error != null )
{
RssReaderErrorEventArgs args = new RssReaderErrorEventArgs();
args.Message = rssFeed.ErrorMessage;
this.Error(this,args);
}
}
}
catch (XmlException err)
{
//
rssFeed.ErrorMessage = "Xml error: " +err.Message;
```

```
// Fire the error event
if ( this.Error != null )
{
RssReaderErrorEventArgs args = new RssReaderErrorEventArgs();
args.Message = rssFeed.ErrorMessage;
this.Error(this,args);
}
return rssFeed;
}
return rssFeed;
}
private RssItem getRssItem(XmlNode xmlNode)
{
RssItem rssItem = new RssItem();

for (int i=0;i < xmlNode.ChildNodes.Count;i++)
{
switch ( xmlNode.ChildNodes[i].Name )
{
case "title":
{
rssItem.Title = xmlNode.ChildNodes[i].InnerText;
break;
}
case "description":
{
rssItem.Description = xmlNode.ChildNodes[i].InnerText;
break;
}
case "link":
{
rssItem.Link = xmlNode.ChildNodes[i].InnerText;
```

```
break;
}
case "author":
{
rssItem.Author = xmlNode.ChildNodes[i].InnerText;
break;
}
case "comments":
{
rssItem.Comments = xmlNode.ChildNodes[i].InnerText;
break;
}
case "pubdate":
{
rssItem.Pubdate = xmlNode.ChildNodes[i].InnerText;
break;
}
case "guid":
{
rssItem.Guid = xmlNode.ChildNodes[i].InnerText;
break;
}
}
}
return rssItem;
}
}
#endregion
#region Html creator class
public class RssHtmlMaker
{
public int MaxItems
{
```

```
get

{

return this.maxItems;

}

set

{

this.maxItems = value;

}

}

private int maxItems = 0;

public string GetHtmlContents(RssFeed Feed,string Template,string
ItemPrefix,string ItemSuffix)

{

string result = Template;

result = result.Replace("%Title%",Feed.Title);

result = result.Replace("%Description%",Feed.Description);

result = result.Replace("%Link%",Feed.Link);

result = result.Replace("%Language%",Feed.Language);

result = result.Replace("%Copyright%",Feed.Copyright);

result = result.Replace("%Webmaster%",Feed.Webmaster);

result = result.Replace("%PubDate%",Feed.PubDate);

result = result.Replace("%LastBuildDate%",Feed.LastBuildDate);

result = result.Replace("%Category%",Feed.Category);

result = result.Replace("%Generator%",Feed.Generator);

result = result.Replace("%Ttl%",Feed.Ttl);

result = result.Replace("%Rating%",Feed.Rating);

result = result.Replace("%Skiphours%",Feed.Skiphours);

result = result.Replace("%Skipdays%",Feed.Skipdays);

result = result.Replace("%Skipdays%",Feed.ManagingEditor);

string itemsContent = "";

string tempContent = "";

if ( maxItems == 0 || maxItems > Feed.Items.Count )

{

maxItems = Feed.Items.Count;
```

```
}
for (int i=0;i < maxItems;i++)
{
tempContent  = ItemPrefix;
tempContent  = tempContent.Replace("%Title%",Feed.Items[i].Title);
tempContent  = tempContent.Replace("%Description%",Feed.Items[i].Description);
tempContent  = tempContent.Replace("%Link%",Feed.Items[i].Link);
tempContent  = tempContent.Replace("%Author%",Feed.Items[i].Author);
tempContent  = tempContent.Replace("%Comments%",Feed.Items[i].Comments);
tempContent  = tempContent.Replace("%Pubdate%",Feed.Items[i].Pubdate);
tempContent  = tempContent.Replace("%Guid%",Feed.Items[i].Guid);
itemsContent += tempContent;
tempContent  = ItemSuffix;
tempContent  = tempContent.Replace("%Title%",Feed.Items[i].Title);
tempContent  = tempContent.Replace("%Description%",Feed.Items[i].Description);
tempContent  = tempContent.Replace("%Link%",Feed.Items[i].Link);
tempContent  = tempContent.Replace("%Author%",Feed.Items[i].Author);
tempContent  = tempContent.Replace("%Comments%",Feed.Items[i].Comments);
tempContent  = tempContent.Replace("%Pubdate%",Feed.Items[i].Pubdate);
tempContent  = tempContent.Replace("%Guid%",Feed.Items[i].Guid);
itemsContent += tempContent;
}
result = result.Replace("%Items%",itemsContent);
return result;
}
}
#endregion

#region Data structures
[Serializable()]
public struct RssFeed
{
public string Title;
```

```
public string Description;
public string Link;
public string Language;
public string Copyright;
public string Webmaster;
public string PubDate;
public string LastBuildDate;
public string Category;
public string Generator;
public string Ttl;
public string Rating;
public string Skiphours;
public string Skipdays;
public string ManagingEditor;
public RssItems Items;
public string ErrorMessage;
}
[Serializable()]
public struct RssItem
{
public string Title;
public string Description;
public string Link;
public string Author;
public string Comments;
public string Pubdate;
public string Guid;
}

[Serializable()]
public class RssItems : CollectionBase
{
public RssItem this[int item]
```

```
{
get
{
return this.getItem(item);
}
}
public void Add(RssItem rssItem)
{
List.Add(rssItem);
}
public bool Remove(int index)
{
if (index > Count - 1 || index < 0)
{
return false;
}
else
{
List.RemoveAt(index);
return true;
}
}
private RssItem getItem(int Index)
{
return (RssItem) List[Index];
}
}
#endregion
}
```

EK F – RSS TABANLI WEB ROBOTU

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using mshtml;
using System.Net;
using System.IO;
using System.Threading;
using System.Data.OleDb;
using net.zemberek.erisim;
using net.zemberek.yapi;
using net.zemberek.tr.yapi;
namespace Okuyucu
{
    public partial class Form1 : Form
    {
        Thread saatlik;

        RSSReader.RssList Liste = new RSSReader.RssList();
        public Form1()
        {
            InitializeComponent();
        }
        private void TMHide_Tick(object sender, EventArgs e)
        {
            TMHide.Enabled = false;
            this.Hide();
            TMSaat_Tick(null, null);
        }
```

```
        private void TMSaat_Tick(object sender, EventArgs e)
        {
            CheckForIllegalCrossThreadCalls = false;
            saatlik = null;
            saatlik = new Thread(new ThreadStart(SaatlikTaramayiBaslat));
            saatlik.Start();
            notifyIcon1.ShowBalloonTip(1000, "Saatlik RSS tarama başladı...",
"Rss Takip", ToolTipIcon.Info);
        }
        private void SaatlikTaramayiBaslat()
        {
            try
            {
                RSSReader.RssReader read = new RSSReader.RssReader();
                RSSReader.RssFeed feed;
                int i;
                foreach (string str in Liste.Ekonomi)
                {
                    try
                    {
                        feed = read.Retrieve(str);
                        foreach (RSSReader.RssItem adres in feed.Items)
                        {
                            try
                            {
                                RssOku("Ekonomi", adres.Link);
                            }
                            catch { }
                        }
                    }
                    catch { }
                }
```

```
                notifyIcon1.ShowBalloonTip(1000, "Ekonomi RSS tarama
tamamlandı...", "Rss Takip", ToolTipIcon.Info);

                foreach (string str in Liste.Saglik)

                {

                    try

                    {

                        feed = read.Retrieve(str);

                        foreach (RSSReader.RssItem adres in feed.Items)

                        {

                            try

                            {

                                RssOku("Sağlık", adres.Link);

                            }

                            catch { }

                        }

                    }

                    catch { }

                }

                notifyIcon1.ShowBalloonTip(1000, "Sağlık RSS tarama
tamamlandı...", "Rss Takip", ToolTipIcon.Info);

                foreach (string str in Liste.Siyaset)

                {  try

                    {

                    feed = read.Retrieve(str);

                    foreach (RSSReader.RssItem adres in feed.Items)

                    {  try

                    {

                        RssOku("Siyaset", adres.Link);

                        }

                    catch { }

                    }

                }

                    catch { }

                }
```

```
                NotifyIcon1.ShowBalloonTip(1000, "Siyaset RSS tarama
tamamlandı...", "Rss Takip", ToolTipIcon.Info);

                foreach (string str in Liste.Spor)

                {  try

                    {

                    feed = read.Retrieve(str);

                    foreach (RSSReader.RssItem adres in feed.Items)

                    {  try

                    {

                        RssOku("Spor", adres.Link);

                        }

                    catch { }

                    }

                }

                    catch { }

                }

                notifyIcon1.ShowBalloonTip(1000, "Spor RSS tarama
tamamlandı...", "Rss Takip", ToolTipIcon.Info);

                foreach (string str in Liste.Teknoloji)

                {

                    try

                    {

                        feed = read.Retrieve(str);

                        foreach (RSSReader.RssItem adres in feed.Items)

                        {

                            try

                            {

                                RssOku("Teknoloji", adres.Link);

                            }

                            catch { }

                        }

                    }

                    catch { }

                }
```

```
                notifyIcon1.ShowBalloonTip(1000, "Teknoloji RSS tarama
tamamlandı...", "Rss Takip", ToolTipIcon.Info);

            }

            catch(Exception ex)

            {

                notifyIcon1.ShowBalloonTip(1000, ex.Message, "Hata Oluştu
!!!", ToolTipIcon.Error);

            }

        }



        Boolean IsNumber(string str)

        {

            Boolean sonuc = false;

            for (int i=0; i<str.Length; i++)

            {

                if (Char.IsNumber(str[i]))

                {

                    sonuc = true;

                    return sonuc;

                }

            }

            return sonuc;

        }

        private void RssOku(string konu, string url)

        {

            try

            {

                //Değişkenler

                Boolean urlgezildi = false;

                //Access Bağlantısı

                OleDbConnection OleCon = new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Application.StartupPath  + "\\okuyucu.mdb");

                OleCon.Open();

                //URL Ziyaret Edildi Mi?
```

```
                OleDbCommand cmd = new OleDbCommand("select count(*) as sayi
from urllist where url='" + url + "'",OleCon);

                OleDbDataReader reader = cmd.ExecuteReader();

                reader.Read();

                if ((int)reader[0] > 0)

                {

                    urlgezildi = true;

                }

                else

                {

                    urlgezildi = false;

                }

                try { cmd.Dispose(); cmd = null; }

                catch { }

                try { reader.Close(); reader.Dispose(); reader = null; }

                catch { }

                //URL Daha Önceden Ziyaret Edilmediyse URL yi İncele

                if (!urlgezildi)

                {

                    //URL deki sadece yazilarin tutulduğu değişken

                    string urlstr;

                    //Sadece Yazıları Al

                    HttpWebRequest req;

                    req = (HttpWebRequest)HttpWebRequest.Create(url);

                    req.Method = "GET";

                    req.Timeout = 10000;

                    HttpWebResponse res;

                    res = (HttpWebResponse)req.GetResponse();

                    Stream s = res.GetResponseStream();

                    StreamReader sr = new StreamReader(s, Encoding.Default);

                    urlstr = sr.ReadToEnd();

                    IHTMLDocument2 doc = new HTMLDocumentClass();

                    doc.write(new Object[] { urlstr });

                    doc.close();
```

```
                    urlstr = doc.body.innerText.Replace((char)10, (char)"
"[0]).Replace((char)13, (char)" "[0]).Replace("Ã§","Ç");

                    try { doc.clear(); doc = null; }

                    catch{ }

                    try { sr.Close(); sr = null; }

                    catch { }

                    try { s.Close(); s = null; }

                    catch { }

                    try { res.Close(); res = null; }

                    catch { }

                    try { res = null; }

                    catch { }


                    //URL yi Ziyaret Edilenler Arasına Ekle

                    cmd = new OleDbCommand("insert into
urllist(konu,url,tarih,saat) values('" + konu + "','" + url + "','" +
DateTime.Now.ToShortDateString() + "','" + DateTime.Now.ToShortTimeString() +
"')", OleCon);

                    cmd.ExecuteNonQuery();

                    //Sadece Konuya Ait Yazıların Bulunduğu Bölümü Çek

                    //Kelime Listesini Al

                    String[] kelimeler;

                    kelimeler = urlstr.Split(Convert.ToChar(" "));

                    //Zembereki Başlat

                    Zemberek zemberek = new Zemberek(new TurkiyeTurkcesi());

                      //Kelimeleri Tek Tek İncele

                    foreach (string kelime in kelimeler)

                    {

                        string KELIME;

                        //Noktalama İşaretlerini Bi Uçur

                        KELIME = kelime.Replace(",", "").Replace(".",
"").Replace("-", "").Replace("?", "").Replace(":", "").Replace("\"",
"").Replace("=", "").Replace("_", "").Replace("(", "").Replace(")",
"").Replace("+", "").Replace("^", "").Replace("!", "").Replace("/",
"").Replace("|", "").Replace("*", "").Replace("\\", "").Replace("/",
"").Trim();

                        if ((KELIME != "") && (IsNumber(KELIME)==false))
```

```
{
                                //Kesme İşareti
                                int kesme = 0;
                                kesme = KELIME.IndexOf("'"[0]);
                                if (kesme > 0)
                                {
                                    KELIME = KELIME.Substring(0, kesme);
                                }
                                try
                                {
                                    //Zemberekten Kelimenin Kökünü Bul
                                    Kelime[] cozumler =
zemberek.kelimeCozumle(KELIME);
                                    if (cozumler.Length > 0)
                                    {
                                        String cozum =
cozumler.GetValue(0).ToString();
                                        Boolean isim = cozum.Contains("ISIM}");
                                        Boolean fiil = cozum.Contains("FIIL}");
                                        if ((isim) || (fiil))
                                        {
                                            string tur = "";
                                            if (isim) { tur = "İSİM"; };
                                            if (fiil) { tur = "FİİL"; };
                                            int bas = 0;
                                            int bit = 0;
                                            bas = cozum.IndexOf("Kok:");
                                            bit= cozum.IndexOf("tip:");
                                            cozum = cozum.Substring(bas + 4, bit -
bas - 4).Trim();
                                           if (cozum.Length > 0)
                                            {
                                                try
                                                {
```

```
                                                cmd = new OleDbCommand("insert
into kelimelist(url,orjinal,kok,tur,konu) values('" + url + "','" + KELIME +
"','" + cozum + "','" + tur + "','" + konu + "')", OleCon);

                                                 cmd.ExecuteNonQuery();

                                            }

                                            catch

                                            {

                                            }

                                        }

                                    }

                                }

                                catch

                                {

                                }

                            }

                            //Zemberek Bitir

                            zemberek = null;

                        }

                        OleCon.Close();

                        OleCon.Dispose();

                        OleCon = null;


                    }

                    catch

                    { }

                }
```