T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AJAX TABANLI WEB SAYFALARINDAN VERİ ÇIKARIMINA BİR YAKLAŞIM

OĞUZ KIRAT

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Dr. Öğr. Üyesi. Tarık YERLİKAYA

EDİRNE-2022

OĞUZ KIRAT'ın hazırladığı **"AJAX TABANLI WEB SAYFALARINDAN VERİ ÇIKARIMINA BİR YAKLAŞIM"** başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından **Bilgisayar Mühendisliği** Anabilim Dalında bir **Yüksek lisans tezi** olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Doç. Dr. Erdinç UZUN	……
Dr. Öğr. Üyesi Emir ÖZTÜRK	……
Dr. Öğr. Üyesi Tarık YERLİKAYA	……

Tez Savunma Tarihi: 04/07/2022

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

İmza

Dr. Öğr. Üyesi Tarık YERLİKAYA

Tez Danışmanı ……

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

……

Prof. Dr. Hüseyin Rıza Ferhat KARABULUT

Fen Bilimleri Enstitüsü Müdürü

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ YÜKSEK LİSANS PROGRAMI

DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

04 / 07 / 2022

Oğuz KIRAT

İmza

Yüksek Lisans Tezi

AJAX Tabanlı Web Sayfalarından Veri Çıkarımına Bir Yaklaşım

T.Ü. Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

İnternetin yaygınlaşmasıyla birlikte sanal ortama yüklenen veri miktarı oldukça artmıştır. Bu verinin önemli bir bölümü web sayfaları aracılığıyla sunulmaktadır. Ancak web sayfaları çoğu zaman sadece önemli olabilecek veriyi içermemekte ve gün geçtikçe karmaşıklaşmaktadır.

Web veri çıkarımı (Web data extraction), değerli ve ilgi çekici bilginin web sayfalarından çıkarılması işlemidir. Veri çıkarımı işlemi, makine öğrenmesi, doğal dil işleme, arama motorları ve büyük veri seti gerektiren işlemler için önem arz etmektedir. Çünkü birçok web sayfası, verilerin bilgisayar programları kullanılarak çekilmesini sağlayan bir arayüz (örneğin; API – Application Programming Interface – Uygulama Programlama Arayüzü) sunmamaktadır.

Web veri çıkarımı konusunda birçok mevcut çalışma düzenli ifade (regex) kullanımı ya da DOM (Document Object Model) ağacının üretilmesi ve bu yönde algoritmaları ortaya koymaktadır. Ancak ilerleyen teknolojiyle birlikte birçok web sayfasının içeriği Javascript ile güncellenebilmektedir. Genel olarak AJAX olarak adlandırılan bu teknolojiyle birlikte DOM ağacı sayfanın görüntülenmesi ve scriptlerin işlenmesi sonucunda dinamik olarak değişebilmekte, hatta web sitelerinin tamamı bu yöntemle (SPA – Single Page Application, Tek Sayfa Uygulaması) oluşturulabilmektedir.

Bu yüksek lisans tezinin amacı, AJAX veya benzeri istemci tabanlı dinamik içerik teknolojileri kullanan web sayfalarında da belirli kural setleriyle daha hızlı veri çıkarımı yapabilecek bir yaklaşım aramaktır.

Yıl : 2022

Sayfa Sayısı : 66

Anahtar Kelimeler : web veri çıkarımı, veri toplama, AJAX, Javascript, veri seti oluşturma

Master's Thesis

An Approach To Data Extraction From AJAX-Based Web Pages

Trakya University Institute of Natural Sciences

Department of Computer Engineering

ABSTRACT

With the widespread use of the Internet, the amount of data loaded into the virtual environment has increased considerably. An important part of this data is provided through web pages. However, web pages often do not contain only the important data and are getting more complicated day by day.

Web data extraction is the process of extracting valuable and interesting information from web pages. The data extraction process is important for machine learning, natural language processing, search engines and processes that require a large data set. Many web pages do not offer an interface (for example; API - Application Programming Interface - Application Programming Interface) that allows data to be retrieved using computer programs, therefore extraction is needed.

Many current studies on the web data extraction topic discuss the use of regular expressions (regex) or the generation of DOM (Document Object Model) trees and algorithms. However, with the advancing technology, the content of many web pages can be updated with JavaScript. With this technology, which is generally called AJAX, the DOM tree can change dynamically as a result of displaying the page and processing scripts. Even all of the web site can be created with this method (SPA - Single Page Application).

This master thesis aims to seek an approach that can extract data faster with specific rule sets on web pages using AJAX or similar client-based dynamic content technologies.

Year 2022

Number of Pages : 66

Keywords : web data extraction, data collection, AJAX, web scraping, data scraping, dataset creation

TEŞEKKÜR

Yüksek lisans tez çalışmamın her aşamasında değerli görüşleriyle beni yönlendiren, ilgisini hiç esirgemeyen, tecrübeleri ile her daim destek olan danışman hocam Sayın Dr. Öğr. Üyesi Tarık YERLİKAYA ve çalışmalarımın her aşamasında özverili bir şekilde bana destek olan ve yol gösteren Doç. Dr. Erdinç UZUN'a teşekkürlerimi sunarım.

Hayatım boyunca desteklerini hissettiğim ve her zaman yanımda olan sevgili aileme teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

AJAX:	Asyncronious Javascript and XML (Asenkron Javascript ve XML)
API:	Application Programming Interface (Uygulama Programlama Arayüzü)
CAPTCHA:	Completely Automated Public Turing Test To Tell Computers And Humans Apart (Bilgisayarları ve İnsanları Ayırmak İçin Tamamen Otomatik Genel Turing Testi)
CSS:	Cascading Style Sheets (Basamaklı Biçim Sayfaları)
DOM:	Document Object Model (Belge Nesne Modeli)
HTML:	Hypertext Markup Language (Hipermetin Betimleme Dili)
HTTP:	Hypertext Transfer Protocol (Hipermetin Aktarım Protokolü)
IP:	Internet Protocol (Internet Protokolü)
JS:	JavaScript
JSON:	JavaScript Object Notation (JavaScript Nesne Gösterimi)
REST:	Representational State Transfer (Temsili Durum Aktarımı)
SAAS:	Software as a Service (Hizmet Olarak Yazılım)
SPA:	Single Page Web Application (Tek Sayfa Web Uygulaması)
XML:	Extensible Markup Language (Genişletilebilir Betimleme Dili)

BÖLÜM 1

GİRİŞ

1.1. Çalışmanın Amacı ve Kapsamı

Web (ve genel anlamıyla internet) çok büyük miktarda veri içerir. Ancak, bir veri tabanı gibi sorgulaması kolay yapıda değildir. Genelde veriler yapısal olmayan bir şekilde tutulur ve genellikle metin araması ile sorgulanır. Yapısal olmayan bir veri yığınından sorgulama yapılabilmesi zor olduğunda, web içeriğinden bilgi çıkarılması için verinin yapısal bir biçime dönüştürülmesi gerekir. Kurum, kuruluş ve kişilerin bu veri ihtiyaçlarını karşılayabilmek için web veri çıkarma yazılımları ortaya çıkmıştır.

Web kazıma ya da web veri çıkarma, web sitelerinden anlamları veri çıkarılmasını amaçlayan bir tekniktir. Bu işlem için genellikle bilgisayar programları, kullanıcı davranışlarını "taklit ederek" web sayfalarına erişir sayfayı indirirler, ardından önemli görülen veriyi sayfadan ayıklarlar.

Birçok web kazıma programı HTTP isteklerini bir web tarayıcıya benzer şekilde gönderip geri dönüş alarak çalışır. Bu tarz programlar basit web sayfaları ve API'ler için oldukça uygundurlar.

HTTP isteğinin döndürdüğü sonuç içerisinde kazıma programları, gerekli olabilecek veriyi bulmak için çeşitli yöntemler kullanılırlar. Bu yöntemlere genel anlamıyla ayrıştırma (parsing) denir.

Ayrıştırma işlemi veri türüne göre (örn. HTML, XML, JSON ya da binary) değişebilir. HTML formatındaki veriler için genellikle DOM çözümlemesi yapılmalıdır. Ardından çözümlenen ağaç içerisinde işe yarayacak verinin konumu bulunarak veri çıkarımı yapılır. XML ve JSON gibi veriler daha anlamsal (semantik) olduğundan daha kolay ayrıştırılabilir.

Ayrıştırma işlemi öncesinde, sırasında ya da sonrasında veriler yeniden biçimlendirilebilir, içerilerinde arama yapılabilir veri tabanları gibi farklı ortama kaydedilebilir. Daha sonra veriler, çeşitli amaçlar için işlenebilir.

Fakat, WEB 2.0 teknolojilerinin gelişmesi ve web sayfalarının istemci tarafında ziyaretten sonra da güncellenebilir olması, başka bir deyişle DOM ağacının sürekli değişebilir oluşu veri çıkarma ve ayrıştırma işlemini karmaşıklaştırmaktır.

Bu tez çalışmasında hem güncel web teknolojileriyle birlikte veri kazıma teknolojilerinden bahsedilmiş, hem de dinamik içerikli web sitelerinde kullanılabilecek temel bir bilgisayar yazılımı oluşturulması amaçlanmıştır.

Bu tez içerisinde yer alan ve bilgisayar kodları içerisinde İngilizce olarak kullanılması gereken sözde sınıflar, kütüphane isimleri vb. terimler ve özel isimler Türkçe'ye çevrilmemiştir. Ayrıca teknik olarak küçük ya da büyük harfle kullanılması gereken ya da oluşturucusu tarafından büyük/küçük harf kullanım şekli özellikle tanımlanmış teknoloji, kütüphane adları vb. ifadeler olduğu şekilde (Örneğin: macOS, JavaScript gibi) biçimlendirilmiştir.

1.2. Mevcut Çalışmalar

Literatür incelendiğinde genellikle çalışmaların webden anlamlı içerik çıkarmaya yönelik olduğu görülmüştür. Çalışmaların birçoğu reklamlar, gezinti bölmeleri gibi nispeten gereksiz "gürültü" içerisinden "değerli" sayılabilecek veriyi ayrıştırma üzerinedir.

Mevcut yaklaşımlar DOM ağacı üzerine çeşitli algoritmalar üretmiş, bazıları doğal dil işleme, sayfa karşılaştırma gibi yöntemler kullanmışlardır. Bununla birlikte yapay öğrenme teknikleriyle de çıkarım başarısını arttırabilen çalışmalar mevcuttur.

Pan, Qiu ve Yin (2008), metin/bağlantı oranlarını kullanarak içerik filtreleyen bir algoritma önermiştir.

Sun ve Liao (2011), gürültülü içeriğin genellikle sayfanın kısa kısımlarından olabileceğini gözeterek metin yoğunluğunu dikkate alıp önemli içerik çıkarım başarımını arttırmıştır.

Mehta ve Narvekar (2015), DOM ağacında istenmeyen içerikleri filtreleyen bir uygulama geliştirmiştir.

Zhou ve Mashuq (2014)., makine öğrenmesi tabanlı bir yaklaşım kullanmış, metin bloklarını sınıflandırmışlardır. Kümeleme yöntemleriyle veri noktalarını otomatik etiketleyen bir sistem geliştirmişlerdir.

Howard ve Manic (2016) paragraf etiketlerinden yola çıkarak ParEx isimli bir çalışma yapmışlardır.

Yahoo! araştırma ekibi ve Waterloo Üniversitesi üyelerinden Dalvi, Kumar ve Soliman (2011), gürültüye dayanıklı algoritmalar önermiş ve gözetimsiz çalışabilen web ölçeğinde genel çözümler sunmuşlardır

İtalyan Üniversitelerinden bir ekip olan Crescenzi, Mecca ve Marialdo (2001), Roadrunner adını verdikleri otomatik bir sarmalayıcı (wrapper) mimarisi geliştirmişlerdir. İki sayfa arasındaki benzerlik ve farklılıkları gözeten bu sistem, veri çıkarımı konusunda en çok atıf alan makalelerden biridir.

AJAX ile çalışan web sayfalarında ise gömülü tarayıcı yaklaşımları mevcuttur. Xia (2019), gömülü bir tarayıcı kullanarak sayfaya JavaScript kodu ekleyip fare tıklamalarını yakalamış, tekli ve çoklu veri çıkarımı için metotlar önermiştir.

Mesbah, Deursen ve Lenselimk (2012), kullanıcı arayüz durumlarının dinamik analizinden yola çıkarak CRAWLAJAX isimli açık kaynak kodlu bir yazılım geliştirmişlerdir.

Uzun, Yerlikaya ve Kırat (2018) çalışmasında Python kütüphanelerinin veri çıkarımındaki performansları karşılaştırılmıştır.

Uzun, Agun ve Yerlikaya (2012), makine öğrenmesi ve kural tabanlı yaklaşımı birleştirerek hibrit bir yaklaşım ile veri çıkarım başarımını arttırmışlardır.

Uzun (2020), UzunExt isimli bir algoritma geliştirmiş ve DOM ağacını oluşturmadan web sitelerindeki ek bilgileri kullanmış ve string tabanlı yaklaşımlarla veri çıkarım hızını 60 kata kadar arttırmıştır.

Otomatik test amaçlı bazı yazılımlar, kendi dahili tarayıcılarıyla ya da başka bir tarayıcıyla haberleşerek veri çekiminde kullanılabilirler. Buna benzer olarak çalışan Selenium, Puppeteer gibi araçlara tezin ilerleyen bölümlerinde değinilmiştir.

Veri çıkarımında yarı otomatik görsel araçlar üzerine de çalışmış ve kullanıcıların görsel tıkla ve işaretle mantığıyla veri çıkarıcılar tanımlamasının önü açılmıştır (Fayzrakhmanov, Sallinger, Furche & Gottlob, 2018).

STALKER (Muslea, Minton & Knoblock, 1999) ve Lixto (Baumgartner, Frölich & Gottlob, 2007) bu araçlardan bazılarıdır. Bazı araçlar ise JavaScript isteklerini kaydedip yeniden oynatarak veri çıkarımı yapmışlardır. Jalangi (Sen, Swaroop & Gibbs, 2013) bu tarz yaklaşıma örnek verilebilir.

Web veri çıkarımı konusunda kurumlara yönelik hizmetler ve ücretli/ticari çözümler de geliştirilmiştir. Bunlardan bazıları e-ticaret verisi gibi (Import.io, 2022) özel alanlara yönelmiş, bazıları da genele yönelik çözümler üretmişlerdir (Mozenda, 2022), (FMiner, 2015).

Hizmet olarak yazılım (SaaS) çözümler, hem kuruluşlar, haberler, perakende gibi alanlarda hazır yapısal verileri sunabilmekte ve çekebilmekte Diffbot (2022) ve Octoparse (2022), hem de web tabanlı ara yüzlerle tıkla ve işaretle mantığının kullanımını kolaylaştırabilmektedirler

OXPath gibi bazı yaklaşımlar, veri çıkarımı için XPath gibi standart dilleri geliştirmişlerdir (Furche, Gottlob, Fiovann, Schallhart & Seller, 2012). Bu dil üzerine görsel olarak çalışan bir uygulama da mevcuttur (Kranzdorf, Sellers, Grasso, Schallhart & Furche, 2012).

BÖLÜM 2

WEB VERİ KAZIMASI / WEB VERİ ÇIKARIMI

2.1. Web Kazıması / Web Veri Çıkarımı Nedir?

Giriş bölümünde de belirtildiği gibi, web sayfalarındaki önemli verinin çıkarılması yöntemlerine genel olarak web kazıması ya da web veri çıkarımı denir.

Web sayfalarının birçoğu tarayıcı üzerinde görüntülenmesini sağlayan etiketler ve stil dosyaları dahil olmak üzere, görüntülendikleri zaman bile genellikle araştırmacıyı ilgilendirmeyen birçok bilgi barındırmaktadır. Örneğin web sitelerindeki reklamlar, gezinti çubukları ve simgeler çoğu zaman önemli veriler değildir. Örnek olarak haber arşivi yapmak isteyen araştırma için, bir haber sitesinde haberin başlığı, içeriği ve tarihi önem arz edebilir. Aynı şekilde doğal dil analizi ile ürün başarımı ölçmek isteyen bir araştırmacı için ürünlere yapılan yorumlar ve derecelendirmeleri dışında birçok bilgi gereksiz olacaktır.

Web veri çıkarımı işlemi elle yapılabileceği gibi genellikle "crawler" da denen yazılımlar ya da botlar kullanılır. Veri çıkarımı programları genel anlamda bilgisayarın başında bir insan varmış gibi davranarak bu işlemi otomatize etmek için hazırlanırlar.

2.1.1. Web Kazıması Ne İçin Kullanılır?

Web kazıması ya da veri çıkarımı, webi anlamlandırmak gereken her yerde kullanılabilir.

Arama motorlarının hemen hepsi otomatikleştirilmiş yöntemlerle webi sınıflandırmak için veri çıkarımı yöntemlerini kullanırlar. Bir web sayfasındaki önemli sayılabilecek (arama sonuçlarına eklenebilecek) içeriğin tespiti,

izlenebilecek/gidilebilecek diğer bağlantılar gibi öğeler arama motorları tarafından tespit edilerek işlenir. Arama motorları genellikle sayfa içerisindeki bağlantıları ayrıştırarak daha önce ziyaret etmediği ya da tekrar ziyaret etmesi gerektiğini düşündüğü sayfaları listeler ve veri çıkarımı için listelerine eklerler. Arama motorlarının veri çıkarımı/veri kazıması işlemini gerçekleştiren programlarına "örümcek" ("spider") denir.

Web kazıması her zaman uzun metinler ya da paragraflar için kullanılmaz. Aynı zamanda sürekli değişebilen bilgilerin takibi için tekrarlı olarak da kullanılabilir. Örneğin güncel döviz kurlarının elde edilmesi, son haberlerin otomatik takibi, bir yazılımın yeni sürümünün çıkıp çıkmadığının tespit edilmesi ya da ürün güncel fiyatlarının eldesi gibi. Buna rağmen sürekli değişen veriler için API, web çengeli (web hook), web soketleri ya da benzeri teknolojilerin kullanılması tavsiye edilir.

Web kazıması, verilerin daha kolay işlenebilir bir biçime dönüştürülmesi için de kullanılır. Örneğin üzerinde kolaylıkla sorgulama yapılabilecek bir veri tabanına kaydetmek için SQL dosyaları ya da HTML içerinden JSON ya da CSV dosyaları oluşturulması mümkündür.

Web kazımasının ikili (binary) dosyalar için de kullanılması mümkündür. Örneğin; bir web sitesindeki resim ve videoların otomatik olarak indirilmesi, bilgisayarla görme üzerine çalışan bir araştırmacının kolaylıkla veri seti oluşturmasını sağlar.

Bunlarla birlikte, arşivleme amaçlı olarak bir web sitesinin içeriğinin tamamen kaydedilmesi içinde veri kazıması kullanılabilir.

Bazı reklam servisleri, web sitesinin içeriğini kategorileyip ilgili reklamlar göstermek için web robotlarını ve veri çıkarma tekniklerini kullanabilir (Google Dev Docs, 2022).

2.1.2. Web Kazımasının Aşamaları

Web kazıması, en azından bu çalışmada kullanıldığı üzere, bir başlangıç sayfası ile başlar. Bu sayfanın, ilgili internet sitesinin ana sayfası olması gerekmemektedir.

Başlangıç sayfası indirilir ve içerisinde kazıma yapılacak veri varsa çıkarılır. Bu veriyi çıkarmak için kullanılabilecek yöntemlere çalışmanın devamında detaylıca yer verilmiştir.

Veri çıkarmaya benzer yöntemler kullanılarak bir kuyruk oluşturulur. Bu kuyrukta tarama işlemine devam etmek için gerekli olabilecek diğer bağlantılar yer alır. Bu bağlantılar, örneğin alt sayfalar, ziyaret edilmek üzere sıraya konulur.

Program, çıkarılan tüm bağlantıları yeniden ziyaret ederek işe yarayabilecek veriyi aramaya devam eder. Bu işlem bir döngü şeklinde istenildiği kadar veri toplanana kadar devam eder.

Her ne kadar bu işlem basit gibi gözükse de web teknolojilerinin her geçen gün gelişmesi ve AJAX gibi teknolojilerle web sayfalarının dinamik olarak güncelleniyor olması, veri kazıma işlemlerini zorlaştırmaktadır.

2.2. Web Teknolojileri ve Veri Kazıması

Bu bölümde veri kazıması işleminden önce bilinmesi gereken bazı temel web teknolojilerine değinilmiştir.

2.2.1. HTTP

HTTP (Hipermetin Aktarım Protokolü - Hypertext Transfer Protocol), internet iletişim kuralları dizisi (IP) modelinin uygulama katmanında çalışan bir protokoldür (Fielding vd, 1999).

HTTP, istek-yanıt biçiminde çalışan bir protokoldür. İstemci, bir sunucuya HTTP isteği gönderir ve sunucu yanıt mesajı olarak istek sonuç bilgisini ve varsa istenilen içeriği (örneğin HTML döndürür).

HTTP, tutarlı bir aktarım taşıma protokolü üzerinde çalışacak şekilde tasarlanmıştır (Fielding vd, 1999, s. 12). Bu nedenle genelde TCP (Transmission Control Protocol) kullanılsa da UDP (User Datagram Protocol) üzerinde de kullanıma uyarlanmıştır.

2.2.2. HTTP Başlıkları

HTTP başlık alanları, HTTP istekleri yapılırken ve yanıt alınırken sunucu ve istemci arasında gönderilip alınan karakter dizisi (string) ifadelerdir. Bu ifadeler, çoğu zaman son kullanıcı tarafından görülmese de sunucu ve istemci uygulamalar tarafından işlenebilirler.

İletişimin nasıl yapılacağı, hangi karakter veya içerik kodlamasının yapılacağı (örneğin UTF-8), istemcinin türü ve özellikleri, içeriğin önbelleklenmesi ya da önbelleklenmemesi, çerezler ve iletişim yapılacak içeriğin/ortamın türü vb. birçok konu istemci-sunucu arasında başlıklar aracılığıyla taşınır.

Bu çalışma için önemli olabilecek ve sık kullanılan HTTP başlıkları ve açıklamaları aşağıdaki tabloda verilmiştir (MDN, 2022).

Çizelge 2.1. Bazı HTTP Başlıkları ve Açıklamaları

HTTP Başlığı	**Açıklama**
Authorization	Kullanıcı aracısını sunucuya yetkilendirmek için gerekli bilgileri içerir.
Last-Modified	Kaynağın son değiştirilme zamanını içerir. *If-Modified-Since* (’den beri değiştirildiyse) ve *If-Unmodified-Since* (’den beri değiştirilmediyse) şeklinde de kullanılabilir.
Etag	Kaynağın sürümünü tanımlar, *If-Match* ya da *If-None Match* bu kaynağın değerini değiştirmek için kullanılabilir.
Keep-Alive	Sürekli bağlantının ne kadar süre açık kalacağını tanımlar.
Accept	Hangi tür veri kabul edileceği konusunda sunucuya bilgi içerir.
Accept-Encoding	Hangi kodlama veya sıkıştırma algoritması kabul edilebileceği konusunda bilgi içerir.
Accept-Language	Hangi doğal dillerde geri dönüş beklendiği konusunda bilgi içerir.
Cookies ve Set-Cookie	İstemcideki çerezleri sunucuya göndermek ve aynı zamanda istemcide yeni çerez oluşturmak için kullanılırlar.
Content-Length	Kaynağın boyutu (bayt) olarak bildirilir.
Content-Type	Kaynağın ortam türünü (MIME) belirtir.
Content-Language	Kaynağın doğal dilini belirtir.
Host	Sunucunun alan adını ve isteğe bağlı olarak TCP portunu içerir.
Referer	Bu sayfaya gönderen bağlantının adresini içerir.

User-Agent	Kullanıcı aracısı hakkında bilgi içerir. Bu bilgi genellikle uygulama türü, işletim sistemi ve mimarisi, kullanıcı aracısının sürümü içermektedir.

2.2.3. HTML

HTML (HyperText Markup Language - Hipermetin İşaretleme Dili), bir web tarayıcı tarafından gösterilmek üzere dokümanları biçimlendiren bir metin tabanlı işaretleme dilidir. Genellikle HTTP üzerinden aktarılır.

HTML, sayfanın biçimlendirmesi işlemini çeşitli etiketler kullanarak tanımlar. Bu etiketler küçüktür (<) ve büyüktür (>) karakterleri arasına yazılır ve tarayıcı tarafından yorumlanırlar.

HTML etiketinin yapısı şu şekilde tanımlanabilir (WHATWG, 2022):

```
<etiket öznitelik1="değer1" öznitelik2="değer2">İçerik</etiket>
```

Örnek olarak bir bağlantı vermek için a etiketi aşağıdaki şekilde kullanılabilir:

```
<a href="https://www.trakya.edu.tr">Trakya Üniversitesi</a>
```

HTML'e genellikle JavaScript programlama dili ve CSS (Basamaklı Tasarım Sayfaları) eşlik eder.

2.2.4. CSS

CSS, bir web sayfasının tasarımını tanımlayan standartlaşmış bir yapıdır ve günümüz web teknolojilerinin (HTML ve Javascript ile birlikte) temelini oluşturmaktadır.

CSS genel anlamda seçici ve açıklayıcı ifadelerden oluşur (W3C, 2021).

Seçici ifadeler, HTML (veya benzeri dokümanların) etiketleri ve elemanları, bu elemanların tanımlayıcıları, sınıfları veya sözde sınıflarından oluşabilirler. Seçici ifadeler standartta tanımlanmış olup, bir kısmına bu çalışmanın devamında da yer verilmiştir.

Tanımlayıcı ifadeler ise seçilmiş olan elemanlara uygulanacak stili tanımlar. Tanımlayıcı ifadeler konum ve hizalama, boyut, renk, yazı tipi, kalınlık, gölgelendirme vb. birçok özniteliği belirleyebilirler (W3C, 2021).

2.2.5. JavaScript

JavaScript, neredeyse tüm internet tarayıcıları tarafından desteklenen, sayfadaki diğer öğeler ve tarayıcının API'leri ile iletişim kurabilen, JavaScript aynı zamanda masaüstü ve mobil uygulama geliştirmede, sunucu tarafında uygulama geliştirmede de kullanılabilir. JIT (Just-In-Time – Tam Zamanında/Dinamik) derlenen bir yapısı vardır.

JavaScript, ECMAScript standartına uygun, üst düzey bir programlama dilidir (ECMA International, 2021).

2.2.6. JSON

Javascript nesne gösterimi, açık standartlı bir dosya biçimi olup veri aktarımı ve değişiminde kullanılır. İnsan tarafından rahat okunabilir. Anahtar-değer mimarisini ve dizi benzeri veri yapılarını kullanır.

JavaScript'ten doğmuş olsa da birçok programlama dili, veri tabanı uygulaması ve araç tarafından okunup yazılabilirler. Şekil 2.1'de bir JSON dosya örneği verilmiştir.

Şekil 2.1. Örnek bir JSON dosyası içeriği

2.2.7. XML

XML, bir betimleme dilidir ve veriyi hem makinelerin hem de insanların okuyabileceği bir formatta saklamak ve iletmek için tasarlanmıştır. XML etiketleri < ve > ifadeleri içerisine yer alır.

XML, yazım olarak HTML'e oldukça benzemekle beraber HTML'de etiket isimler ve öznitelikler standartta belirlenmişken XML'de serbest bırakılmıştır. Bu benzerlik, XML için kullanılabilen birçok aracın, aynı zamanda HTML için de kullanılabilmesi kolaylığını getirir. Şekil 2.2'de bir XML dosyası örneği gösterilmiştir.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <ad>Ali</ad>
  <soyad>Yılmaz</soyad>
  <hayatta>true</hayatta>
  <yas>27</yas>
  <adres>
    <sokak>Özcan Sk.</sokak>
    <sehir>Edirne</sehir>
    <ulke>TR</ulke>
    <postaKodu>22030</postaKodu>
  </adres>
  <telNolari>
    <tur>ev</tur>
    <numara>2840000000</numara>
  </telNolari>
  <telNolari>
    <tur>cep</tur>
    <numara>500000000</numara>
  </telNolari>
  <cocuklar/>
  <es/>
</root>
```

Şekil 2.2. Örnek bir XML dosyası içeriği

2.2.8. AJAX

AJAX, Asenkron JavaScript ve XML'in kısaltmasıdır. Günümüzde, asenkron web uygulamaları oluşturmak için kullanılan istemci tarafı teknolojilerin tamamı için AJAX terimi kullanılmaktadır. Günümüzde, XML çoğunlukla yerini JSON formatını bırakmıştır.

Web sayfaları, AJAX sayesinde sayfanın tamamını yenilemeye ihtiyaç duymadan arka planda veri alışverişi yapabilirler.

Web tarayıcıları, XMLHttpRequest (XHR) isimli Javascript API'si sunarlar ve web sayfaları bunu kullanarak AJAX istekleri gönderip alabilirler. WHATWG ve W3C, standardı tanımlamakta ve sürdürmektedir.

XHR, modern web tarayıcılarının tümü tarafından uzun zamandır desteklenmektedir.

Çizelge 2.2. Sık kullanılan tarayıcıların XHR destekleme sürümleri

Internet Explorer	10.0 üstü (JSON desteklememektedir)
Firefox	31 ve üstü
Edge (Chromium tabanlı dahil)	12 ve üstü
Chrome	31 ve üstü
Safari (macOS)	7.1 ve üstü
Safari (iOS ve iPadOS)	8 ve üstü

Yüksek uyumluluk ve kullanım kolaylığı jQuery, React, Vue.js, Angular gibi uygulama çatılarının gelişmesine sebep olmuş ve tek sayfalı web uygulamaları kavramını ortaya çıkarmıştır.

2.3. Tek Sayfalı Web Uygulamaları

Tek sayfalı web uygulamaları, web sitesinin tamamının AJAX teknolojisi ile çalıştığı web siteleridir.

Genelde temel bir web şablonu ve yönlendirme amaçlı bir JavaScript kodu içerirler. Herhangi başka bir sayfa, tamamen yüklenmek yerine yönlendirici kod, şablona mevcut sayfadan değişiklik olacak kısmı AJAX ile getirip yükler. Bu sayede web sunucuya yapılan istek sayısı, aktarım yapılacak veri miktarı azaltılmış olur.

Tek sayfalı web uygulamaları, JavaScript ve ilgili yeni teknolojileri kullandıklarından mobil uygulamalara veya masaüstü uygulamalara yakın deneyim sunarlar.

2.4. Semantik Web

2.4.1. DOM (Belge Nesne Modeli - Document Object Model)

HTML ve XML belgelerini bir ağaç veri yapısı modelinde gösteren dil ve platform bağımsız bir ara yüzdür. Her bir düğüm, dokümandaki bir nesneyi temsil etmektedir. Her bir HTML etiketi, bir düğüm ile ifade edilebilir.

DOM ağacı, bir HTML belgesi gösterilirken tarayıcılar tarafından oluşturulur ve çalışma zamanında JavaScript ile değiştirilebilir. Her bir düğüm, olay tetikleyicileri ve çözümleyicileri, tasarım öğeleri ve sayfa içeriği içerebilir.

DOM ağacında belirli bir öğenin yerinin tespiti için seçiciler kullanılır.

2.4.2. CSS Seçicileri

CSS seçicileri, normal koşullarda bir web tarayıcının belirli bir biçimi sayfanın hangi elemanına uygulayacağını belirtmek için kullanılır ve standartta tanımlanmıştır (W3C, 2021). CSS seçicileri, DOM ağacından bir elemanı göstermek için çoğu zaman yeterli olduklarından veri çıkarımı yapılacak elemanı göstermek için de kullanılabilirler.

CSS seçicilerin yazılma kurallarıyla ilgili bilgiler, Çizelgeler 2.3-2.7'de örnekleriyle birlikte açıklanmıştır.

Çizelge 2.3. Sık kullanılan CSS seçicilerine örnekler ve açıklamaları (W3Schools,2022)

Seçici	**Örnek Gösterim**	**Açıklama**
*	*	Tüm elemanları seçer.
.sinif	.baslik	class="baslik" olan tüm elemanları seçer.
#id	#aciklamakutusu1	id="aciklamakutusu1" olan tüm elemanları seçer.
etiket	div	Tüm etiketli elemanları seçer.
etiket.sinif	div.buyuk	buyuk sınıfına sahip tüm div etiketleri seçer. Örnek:
,	div, p, a	Tüm div, p **ve** a etiketlerini seçer.

Çizelge 2.4. CSS öznitelik seçicileri (MDN, 2022)

[ozntlk]	*ozntlk* özniteleğine sahip tüm elemanları seçer.
[ozntlk=deger]	*ozntlk* isimli bir özniteliği olup bu özniteliği değeri tam olarak *deger* olan elemanları seçer.
[ozntlk~=deger]	*ozntlk* isimli bir özniteliği olup bu özniteliğin değeri boşlukla ayrılmış kelimelerden *deger* kelimesini içeren elemanları seçer.

[ozntlk^=deger]	*ozntlk* isimli bir özniteliği olup bu özniteliğin değeri *deger* ile başlayan elemanları seçer. Örnek: a[href^="https://"] seçicisi, https:// ile başlayan linkleri seçer.
[ozntlk$=deger]	*ozntlk* isimli bir özniteliği olup bu özniteliğin değeri *deger* ile biten elemanları seçer.
[ozntlk$=deger]	*ozntlk* isimli bir özniteliği olup bu özniteliğin değeri *deger* ifadesini en az bir kez içeren elemanları seçer. Örnek: a[href*="kategori"] seçicisi, içerisinde kategori geçen linkleri seçer.
[ozntlk|=deger]	*ozntlk* isimli bir özniteliği olup bu özniteliğin değeri tam olarak *deger* olan ya da *deger* ile başlayıp – ile devam eden elemanları seçer. Örnek: <etiket ozntlk="deger-ikinci"></etiket> <etiket ozntlk="deger"></etiket>

Çizelge 2.5. CSS seçilerde büyük/küçük harf duyarlılığı

[ozntlk (işlem) DeGeR i]	Kapama köşeli parantezinden önce i yada I eklemek, değerlerin büyük küçük harfe duyarsız olarak karşılaştırılmasını sağlar. (ASCII tablosunu temel alır)
[ozntlk (işlem) DeGeR s]	Kapama köşeli parantezinden önce s yada S eklemek, değerlerin büyük küçük harfe duyarlı olarak karşılaştırılmasını sağlar. (ASCII tablosunu temel alır) *Deneysel bir özelliktir.*

Birleştiriciler

İki veya daha fazla CSS seçicisinin kullanılması gereken durumlarda birleştiriciler kullanılabilir.

Çizelge 2.6. CSS seçicilerde birleştiriciler (MDN, 2022)

İsim	**Tanımlama**	**Örnek**	**Açıklama**
Mirasçı birleştiricisi	(boşluk)	div p	div etiketi **içerisinde yer alan** p etiketli elemanı seçer.
Komşu kardeş birleştiricisi	+	div+img	Ebeveyni aynı olan bir div etiketinden hemen sonra yer alan ilk img etiketli elemanı çeker.
Genel kardeş birleştiricisi	~	p~img	Ebeveyni aynı olup, bir p etiketini izleyen tüm img etiketlerini seçer.
Çocuk birleştiricisi	>	div > span	İlk seçiciye eşleşen elemanların doğrudan çocuğu olup ikinci seçiciye de eşleşen elemanları seçer. Bu birleştirici, mirasçıya göre daha katı olup, torun elemanları seçmemektedir.

Sözde Sınıflar ve Sözde Elemanlar

Sözde sınıflar bir seçicinin sonuna eklenerek özel durum tanımlarlar. Sözde sınıflar bir elemana belge ağacından bağımsız olarak erişmeyi de sağlarlar.

Sözde elemanlar ise bir elamanın belirli bir bölümüne erişmek için kullanılabilirler.

Sözde sınıf isimleri İngilizce dilinde tanımlanmıştır. Örneğin, *nth* ifadesi, İngilizce'de sayı sıralamasını belirtmek için kullanılır ve Türkçe'deki *...ıncı* ekinin

karşılığıdır. Fakat İngilizce belirtildiği şekilde kullanılması gerektiğinden sınıf isimleri Türkçe'ye çevrilmemiştir.

Çizelge 2.7. Tez konusu için faydalı olabilecek bazı CSS sözde sınıfları

Sözde Sınıf/Eleman	**Tür**	**Açıklama**
:disabled	Sözde Sınıf	Mevcut durumda devre dışı bırakılmış elemanları temsil eder.
:read-only	Sözde Sınıf	Kullanıcı tarafından değiştirilemeyecek elemanları temsil eder.
:checked	Sözde Sınıf	İşaretli seçme kutuları ve radyo butonlarını temsil eder.
:root	Sözde Sınıf	Dokümanın kökünü temsil eder. Bu, HTML için genellikle <html> etiketidir.
:empty	Sözde Sınıf	Boşluk dışında çocuğu olmayan elemanları temsil eder.
:nth-child	Sözde Sınıf	An+B gösteriminde kardeş elemanlar arasında seçmek için kullanılır.
:nth-last-child	Sözde Sınıf	An+B gösteriminde kardeş elemanlar arasında seçmek için kullanılır. Sıralama terstendir.
:nth-of-type	Sözde Sınıf	Kardeşler arasında aynı etiket tipinde olanlardı An+B konumundakini seçer.
:first-child	Sözde Sınıf	Kardeşler arasında ilk elemanı seçer.
:last-child	Sözde Sınıf	Kardeşler arasında son elemanı seçer.

::after	Sözde Eleman	Seçili elemanın son çocuğu olacak bir sözde eleman oluşturur.
::before	Sözde Eleman	Seçili elemanın ilk çocuğu olacak bir sözde eleman oluşturur.
::first-letter	Sözde Eleman	Blok seviyesi bir elemanın ilk harfini seçmek için kullanılır. Örnek: p::first-letter ifadesi bir paragrafın ilk harfini seçmek için kullanılabilir.
::first-line	Sözde Eleman	Blok seviyesi bir elemanın ilk satırını seçmek için kullanılır.

An+B Gösterimi

CSS seçiciler için faydalı olan bu konum gösterimi, ismi *:nth* ile başlayan sözde sınıflar için kullanılır.

n sayısı, 0'dan başlayarak tüm tam sayıları ifade etmektedir. A ve B ise seçici ile tanımlanır. İfadenin An veya B kısmından sadece bir kısmı yazılabilir.

Bu gösterim yerine ayrıca *odd* ve *even* anahtar kelimeleri de kullanılabilir. Türkçe anlamıyla *odd* tek, *even* ise çift sayıları temsil etmektedir.

Örneğin;

td:nth-child(odd) ve *td:nth-child(2n+1)* aynı anlama gelmekte ve bir tablonun tek sayılı sütunlarını seçmektedir.

2n+1 ifadesi n=0 için 1, n=1 için 3, n=2 için 5 şeklinde artacak ve tüm tek sayıları kapsayacaktır.

p:nth-child(5) gibi bir gösterim ile beşinci paragrafın seçimi de mümkündür.

2.4.3. XPath

XPath (XML Path Language – XML Yol Dili), bir XML belgesi içerisinde istenen elemanı bulmayı ve yerini belirtmeyi sağlayan bir W3C standardıdır. (W3C, 2017)

XPath'te kullanılan temel ifadeler ve açıklamaları Çizelge 2.8'de belirtilmiştir.

Çizelge 2.8. XPath dilinde kullanılan ifadeler.

İfade	**Açıklama**
dugumadi	"dugumadi" isimli tüm düğümleri seçer.
/	Kök düğümden seçim yapar.
//	Sonraki seçime uyan tüm düğümleri seçer. Yeri önemsemez.
.	Şimdiki düğümü seçer.
..	Şimdiki düğümün ebeveynini seçer
@	Öznitelikleri seçer.
*	Herhangi bir düğümü seçer.
|	İki düğüm setini birleştirir
or	Veya mantıksal işlemi
and	Ve mantıksal işlemi
div	Bölme işlemi

2.4.4. JSONPath

JSONPath, JSON belgeler üzerinde XPath'e benzer şekilde istenen elemana ulaşmayı sağlayan bir yapıdır. Henüz öneri aşamasında olup birden fazla uygulaması bulunsa da kullanıldığı çalışmalar mevcuttur.

2.4.5. Düzenli İfadeler

Düzenli ifadeler, bir metin içerisinde arama yapmak için kullanılan karakter dizileridir. Arama ifadesi, bazı özel karakterler kullanılarak tanımlanır. Düzenli ifadelerin yazımı, programlama dilleri arasında farklılık gösterse de genel geçer olarak Perl gösterimi ya da POSIX standardı kullanılır.

Düzenli ifadeler, veri ayrıştırma, doğrulama ve kazıma işlemlerinde de kullanılırlar.

HTML verisi için düzgün tanımlandıklarında düzenli ifadeler, diğer seçicilere göre çok daha hızlı çalışırlar (Uzun, Yerlikaya & Kırat, 2017).

Çizelge 2.9'da çalışmamızda da kullanılan Perl gösteriminde bazı önemli düzenli ifade özel karakterleri veya kullanımları açıklanmıştır.

Çizelge 2.9. Perl gösteriminde düzenli ifadelerin kullanımı

Karakter veya Örnek Kullanımı	**Açıklama**
.	Herhangi bir karakter (Yeni satır hariç)
.*	Herhangi karakterler
\	Kaçış karakteri (Örneğin \. ifadesi . anlamına gelir.)
+	1 veya daha fazla
*	0 veya daha fazla
?	Bir kez ya da hiç
{m}	Tam olarak m kez
{m,n}	m ile n arasında sayıda kez (her iki uç da dahil)
{m,}	m ya da daha fazla
[^…]	… DEĞİL (Örneğin, [^\n] ile . aynı anlamdadır.)
[…]	… içerisindeki karakterlerden biri
\d	0-9 arası herhangi bir rakam
\w	Karakter, sayı, alt çizgi ya da ideogram
\s	Boşluk karakterleri (boşluk, sekme karakteri (TAB), yeni satır vb.)
\D, \W ve \S	\d, \w veya \s ifadelerinin dışarısında kalan karakterler Örneğin \D rakam dışı karakterleri ifade eder.
\n	Yeni satır
\r\n	Windows® işletim sisteminde satır ayıracı
(…)	Gruplama
\1, \2 …	1., 2., … grubun içeriği Örnek: *i(\w)f\1de*, ifade içeriğini eşler

^	Bir satır ya da karakter dizisi başlangıcı
$	Bir satır ya da karakter dizisinin bitişi
(?i)	Büyük/küçük harf duyarlılığını etkisizleştirir.

2.5. Statik Sayfalarda Veri Çıkarımında Kullanılan Popüler Uygulama Çatıları, Kütüphaneler ve Araçlar

Bu bölümde, Python programlama dilinde, statik web sayfalarında veri çıkarımı için kullanılabilecek bazı popüler kütüphane ve uygulama çatıları incelenmiştir.

2.5.1. Requests

"İnsanlar için HTTP" sloganıyla ortaya çıkan requests kütüphanesi, HTTP/1.1 üzerinden haberleşmeyi daha kolay ve anlaşılabilir hale getirmeyi amaçlamıştır. Kütüphanenin temelinde urllib3 isimli farklı bir Python kütüphanesi bulunmaktadır.

Requests, GET VE POST gibi temel HTTP eylemleriyle isteklerinin gönderimi dışında, bağlantıyı açık tutma (keep-alive), uluslararası alan adları ve URL'ler, oturum ve çerez yönetimi, HTTPS/SSL doğrulama, çoklu parçalı dosya yükleme ve parçalı veya akış halinde dosya indirmeyi de desteklemektedir.

Kütüphane aynı zamanda HTTP başlıklarının yönetimine de imkan sağlamakta, bu sayede hem program kodu içerisinde istek ve cevapların kontrolünün önünü açmakta hem de bir tarayıcıya yakın davranışlar sergilemektedir.

Request kütüphanesi, çalışmamızda veri çıkarımının veri çekme aşamasında, indirme ve gezinme amaçlı kullanılmıştır.

2.5.2. Python html.parser Kütüphanesi

Bu kütüphane HTMLParse isimli bir sınıf tanımlar. Bu sınıf kullanılarak HTML biçimindeki veriler ayrıştırılabilir.

2.5.3. lxml

lxml, C kütüphaneleri libxml2 ve libxslt için Python bağlantıları sunan bir araç takımı kütüphanesidir.

2.5.4. html5lib

Tamamen Python dilinde geliştirilmiş bir HTML ayrıştırıcısıdır, WHATWG (Web Hypertext Application Technology Working Group) HTML standardına göre geliştirilmiştir. Bu standart, başlıca web tarayıcıları tarafından kullanılmakta olduğundan html5lib, gerçek bir tarayıcıya yakın çözümlemeler yapabilir.

2.5.5. re

Python dilinde Perl diline yakın düzenli ifade çıkarımları yapmak için kullanılır. İstenen verinin çıkarımı için düzenli ifade yazımı gerektirse de en hızlı çözümlerden biridir. (Uzun vd.)

2.5.6. HTML ve XML Çözümleyicilerinin Karşılaştırılması

Şimdiye kadar değinilen çözümleyicilerin kısa bir karşılaştırmasına Çizelge 2.10'da verilmiştir.

Yumuşak/katı tavırlılık, standart dışı ya da geçersiz dokümanlardaki çözümleme başarımını belirtmektedir.)

Çizelge 2.10. HTML ve XML çözümleyici karşılaştırması

Çözümleyici	**Avantajları/Dezavantajları**
html.parser	İyi hız Yumuşak tavırlı
lxml HTML parser	Çok iyi hız Yumuşak tavırlı Ek C bağımlılığı
lxml XML parser	Çok iyi hız XML dokümanları için kullanılabilir. Ek C bağımlılığı
html5lib	Çok yumuşak tavırlı Web tarayıcıya en yakın çözümleyici Çok yavaş Ek Python bağımlılığı
parsel	Xpath ya da CSS seçici kullanılabilir. İyi hız

regex	Mükemmel hız Yazımı zor Çok katı tavırlı.

2.5.7. BeautifulSoup4 (bs4)

Beautiful Soup, web üzerinden veri kazımayı kolaylaştırmak amacıyla tasarlanmış bir kütüphanedir. HTML ve XML çözümleyicileri üzerine kurulu olan kütüphane, çözümlenmiş ağaç üzerinde arama, düzenleme ve dolaşma için çözümler sunar. (Hajba, 2018)

Beautiful soup ile Python'un html.parser'ı, lxml,lxml-xml ve html5lib kütüphanelerinin ayrıştırıcıları kullanılabilmektedir.

2.5.8. Mechanize

Mechanize, programlanabilir bir web gezginidir. Linklere tıklama, form doldurma gibi işlemleri kod aracılığıyla yapmayı kolaylaştırır.

2.5.9. Scrapy

Açık kaynak kodlu bir veri çıkarım platformu olan scrapy, web örümceği oluşturmak amacıyla kullanılabilir. Bir başlangıç adresinden ayrıştırma yaparak farklı bir adrese gidebilecek şekilde programlanabilir.

Scrapy sınıfları komut satırı üzerinden de çalıştırılabilir. Ayrıştırıcı olarak parsel kütüphanesini kullanmaktadır.

2.6. JavaScript ve AJAX Teknolojilerini Kullanan Web Sayfaları ve Bu Sayfalarda Veri Çıkarımı

Gmail, Google Translate, YouTube, Outlook, Bing, Twitter, Reddit, iCloud web gibi çok bilinen web sitelerinde sıklıkla AJAX ve dolayısıyla JavaScript kullanılmaktadır. Teknolojinin ana avantajı olan sayfayı yenilemeden sayfa içeriğini güncelleme, arama metnin tamamlanması, yeni gelen bir e-postanın hemen gelen kutusunda görülmesi, beğen butonuna basıldığında sayfada mevcut olunan konumdan ayrılmadan beğeni işleminin yapılması gibi işlemlerde sıklıkla kullanılmaktadır.

Bu bağlamda teknoloji, kullanıcı deneyimini arttırmakta önemli bir rol oynamaktayken, örneğin beğenme işlemi için tüm tweet iletilerinin yeniden yüklenmesi yerine sadece çerez/oturum bilgisi ve tweet numarası gibi görece kısa bilgilerin sunucuya gönderilmesiyle web trafiğini önemli ölçüde azaltmaktadır.

Her ne kadar faydalı olsa da AJAX teknolojisinin veri çıkarımı ve indekslenmesi açısından olumsuz yönleri de bulunmaktadır.

Bunlardan en önemlisi JavaScript kodunun işlenmek zorunda olmasıdır. Birçok tek sayfa web uygulaması, sadece basit HTML etiketleri ve birkaç JavaScript dosyası içermekte, uygulamanın tüm içeriği AJAX ile çekilebilmektir. Bu durum, JavaScript kodunun çalıştırılmadığı durumlarda, örneğin bir arama motoru ya da basit bir HTML ayrıştırıcısı için web sitesinin boş görünmesine sebebiyet verebilir. JavaScript kodunun çalıştırılması ise, tüm sayfaların indekslenmesi açısından zorlu bir süreçtir.

Çalıştırılacak bir JavaScript kodu, ekstra zaman gerektirmektedir. Ayrıca her bir JavaScript olayı, sayfada değişikliğe sebep olabileceğinden bu da neredeyse sonsuz bir olasılık doğurmaktadır. Bu olasılıkların her biri farklı bir sayfa adresiyle tanımlanmamış da olabilir. Bu da sayfanın mevcut haline aynı web adresi ile erişilemeyeceği anlamına gelebilir.

2.6.1. Web Tarayıcısıyla Veri Çıkarımı Uygulamasının Bağlanması

Web sürücüsü, bir ya da birden fazla web tarayıcı için geliştirilmiş, web uygulamalarını test etmeye yarayan araçlardır. Web sayfalarında gezinti, (sanal) kullanıcı girişi, JavaScript işletme gibi özellikler sunarlar.

Web sürücüleri W3C WebDriver Standard (W3C Web Sürücüsü Standardı) 'na uygun olarak geliştirilirler. Popüler tarayıcılar için uyumlu web sürücüleri ve desteklendikleri platformlara Çizelge 2.11'de yer verilmiştir.

Çizelge 2.11. Sık kullanılan web sürücüleri ve desteklendikleri tarayıcı ve platformlar

ChromeDriver	Chrome Masaüstü (ChromeOS, Linux, macOS, Windows) Chrome Mobil (Android)
geckodriver	Firefox Masaüstü (Linux, macOS, Windows) Firefox Mobile (Android)
InternetExplorerDriver	Internet Explorer 11 (Windows 10)
Microsoft Edge WebDriver (EdgeDriver)	Microsoft Edge (Windows 7+)
WebDriver for Safari (SafariDriver)	Safari (macOS)
OperaChromiumDriver	Opera (Linux, macOS, Windows)

2.6.2. Sıklıkla Kullanılan Veri Çıkarımı Araçları, Kütüphaneleri ve Uygulama Çerçeveleri

2.6.2.1. Selenium

Selenium, tarayıcı tabanlı test için geliştirilmiş C#, Ruby, Java, Python ve JavaScript dillerinde desteği bulunan bir uygulama çatısıdır.

2.6.2.2. Requests-HTML

requests-html, HTML ayrıştırma için bir Python kütüphanesidir. Daha önce bahsedilen requests Python kütüphanesini temel alır. Ayrıldığı en önemli nokta JavaScript desteğidir. Kütüphane arkaplanda Chromium tarayıcısını kullanarak JavaScript kodunu çalıştırır.

Kütüphane ayrıca, CSS seçileri, XPath seçicileri ve jQuery benzeri (bir Javascript çatısı) seçicilere destek verir. Requests kütüphanesinde olduğu gibi yönlendirme, header değişikliği gibi temel HTTP işlemlerini kolaylaştırır. Asenkron çalışma özelliği de bulunmaktadır.

2.6.2.3.Puppeteer

Pupeeter, DevTools protokolü üzerinden Chromium tabanlı web tarayıcılarını yönetmeye yarayan yüksek seviyeli uygulama geliştirme arayüzü sunar.

2.6.2.4.Diğer çözümler

Ranorex, Rapise, Katalon, TestCafe gibi test uygulama çatıları, AJAX tabanlı web sayfalarında kullanılabilmektedir.

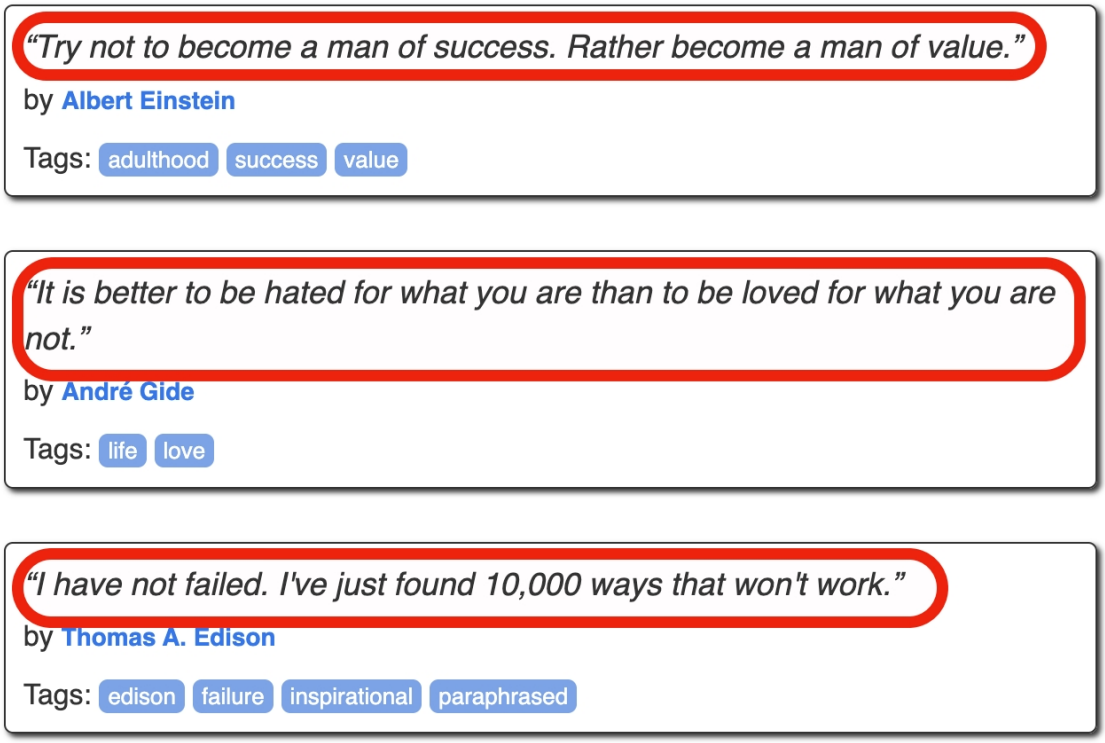
BÖLÜM 3

WEB VERİ ÇIKARIMINA BİR YAKLAŞIM VE UYGULAMASI

3.1. AJAX'tan Kural Tabanlıya

Tek sayfalı web uygulamalarında kullanılan birçok JavaScript tabanlı ön yüz uygulama çatısı (İng. "frontend framework") temsili durum aktarımı uygulama programlama arayüzlerini (İng. "RESTful API") tüketmek üzere kullanılabilirler. Bu da bir tarayıcı için sayfayı oluşturmak zaman alsa bile arkaplanda basit HTTP eylemlerinin (GET, POST, PUT, DELETE) kullanıldığı AJAX tabanlı istekler olabileceği anlamına gelir. REST tabanlı olsun olmasın, bu isteklerin "taklit edilmesi" JavaScript kodu çalıştırmadan veri çıkarımı yapılabileceği anlamına gelmektedir. REST tabanlı olsun olmasın AJAX ile oluşturulan istekler eğer belli bir örüntüyü izliyorlarsa bunları belli kurallar dahilinde tanımlayacak bir şablon tasarlamak mümkündür.

Örnek olarak, veri kazıma işlemleri için oluşturulmuş ve bazı özlü sözlerin listelendiği https://quotes.toscrape.com/scroll web sayfasından önemli veri olarak özlü sözlerin çekilmesi istendiğini varsayalım. Çekilmesi yapılacak verini sayfadaki yeri Şekil 3.1'de gösterilmiştir.

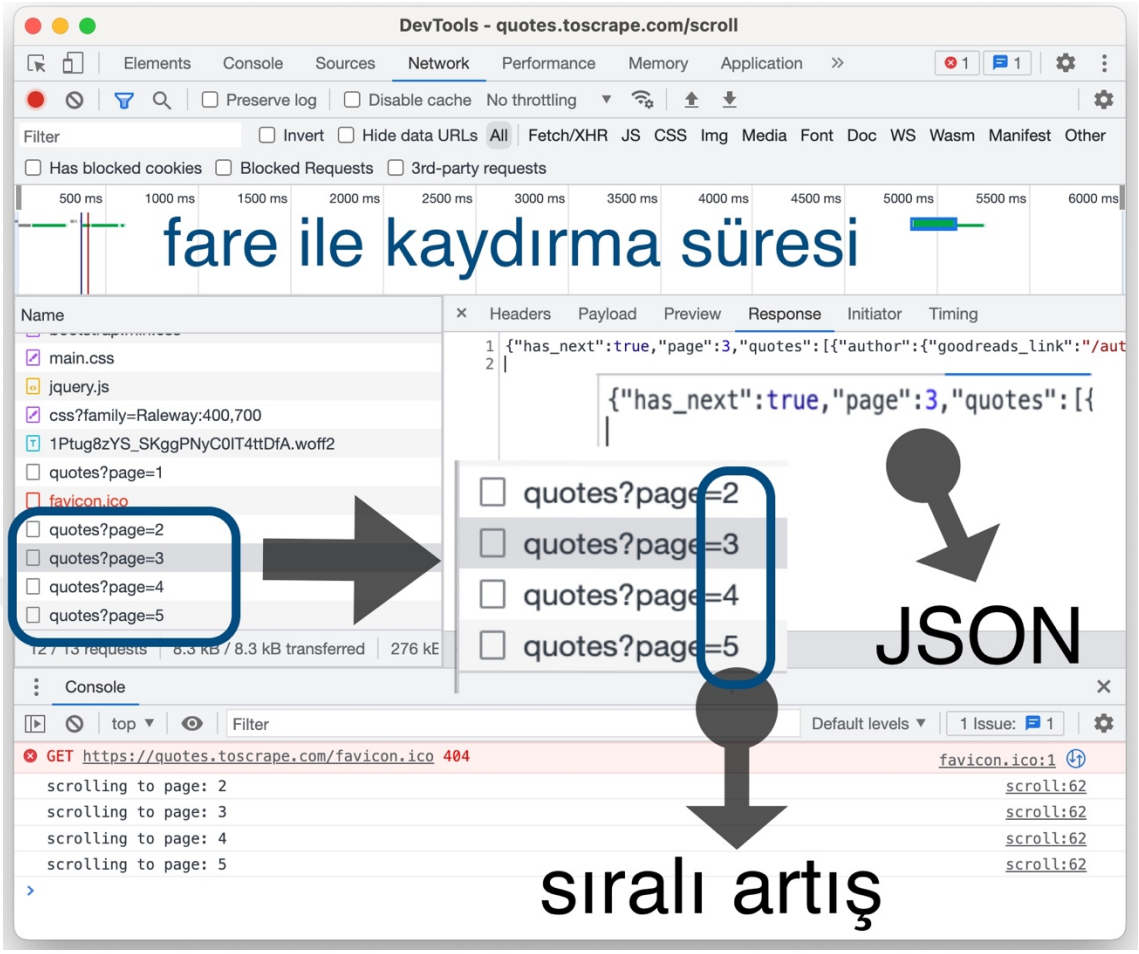


Şekil 3.1. Çekilmesi amaçlanan veri gösterimi

Tarayıcılarda bulunan geliştirici araçlarıyla incelendiğinde sayfanın aşağı kaydırılmasının bir JavaScript olayını tetiklediği ve bu olay sonucunda bir AJAX tabanlı bir HTTP isteği oluşturulduğu görülmüştür.

İsteklerin 1, 2, 3… şeklinde sıralı sayfa numaraları ilerlediği, dönen cevabın ise JSON dosyası olduğu tespit edilmiştir. Bu durumun tarayıcı geliştirici konsolundaki gösterimi Şekil 3.2’deki gibidir.

Veri çıkarımı için istenen veri bu JSON dosyası içinde yer aldığından, JSONPath kullanılarak çıkarımı yapılması yeterli olacaktır.



Şekil 3.2. Sayfanın yüklenmesi sırasındaki trafik ve gelen/giden istekler

Bu çalışmanın odaklandığı nokta ise buna benzer işlemleri kurallı hale getirmektir. Bu sayede farklı web siteleri içinde kurallar oluşturularak kod yazmadan veri çıkarımı yapılabilir. Örneğin, dönüş verisi JSON yerine HTML olsa bile XPath seçicileri kullanımıyla veri çıkarımı yapılabilir.

Çalışmada kullanılan kurallar JSONPath, XPath, CSS ve Soup seçicilerini desteklemektedirler. Katmanlı bir şekilde çıkarım yapılabilmekte ve çıkarım yapılmış bir veriden URL çıkarımı yapılıp kuyruğa eklenebilmektedir.

Veri çıkarımı yapmak isteyen bir kullanıcının ihtiyacı olabileceği düşünülerek çıkarım yapılmış verinin bir metin dosyasına ya da basit bir veri tabanına kaydedilmesi, ikili dosya ise indirilmesi ya da üçüncü parti bir komut ile işlenmesi mümkündür.

3.2. Kuralların Oluşturulması

Çalışmada tasarlanan bilgisayar yazılımı, her bir veri çıkarımı süreci için projeler kullanır. Projeler, veri çıkarımı yapılacak web sitesi ve çıkarım kurallarını tanımlayacak şekilde kural dosyalarına yazılırlar. Kural dosyaları JSON veri tipinde metin dosyaları olarak saklanır.

Bilgisayar yazılımı çalıştırılırken, veri çıkarımı işi için oluşturulan kural dosyası girdi olarak verilir.

Kural dosyalarının ileride açıklanan şablonda hazırlanması gerekse de bilgisayar yazılımına sahip farklı bir kullanıcı, aynı kural dosyasını çalıştırarak kod yazmadan güncel bir web sitesinde veri çıkarımı işlemi yapabilir.

3.3. Projenin Tanımlanması

Her bir kural dosyası ifade ettiği projenin ne yaptığını anlatan bir açıklama ve kendisine ait bir URL tanımlamalıdır. URL'nin güncel bir açıklama içeren web adresi ya da bir sürüm kontrol (örneğin Git) deposu olması tercih edilir. Kullanılabilecek değişkenler, veri tipi ve açıklamalarıyla Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1. Proje tanımlamasında ana değişkenler

Değişken	**Açıklama**	**Veri Tipi**
project-name	Proje Adı	Karakter dizisi
project-url	Projenin web sayfası	Karakter dizisi
project-description	Projenin Açıklaması	Karakter dizisi

3.3.1. Proje Geneli Ayarlar

Proje çalışırken genel olarak kullanılacak ayarlar için bu bölümde bahsedilen değişkenler kullanılmalıdır. Kullanıcı aracısı, çerez bilgileri gibi bir tarayıcı ya da örümcekten beklenen bilgiler bu bölümde tanımlanır.

project-settings sözlük veri tipinde tanımlanır. Çizelge 3.2'de bu sözlük içerisinde yer alabilecek değişkenl, açıklamaları ve veri tipleri gösterilmiştir.

Çizelge 3.2. Proje geneli ayarların açıklamaları

Değişken	**Açıklama**	**Veri Tipi**
user-agent	Proje geneli varsayılan kullanıcı tanımlayıcısı	Karakter dizisi
wait-between-requests	Her bir istek arası bekleme süresi (ms)	Pozitif tam sayı
http-headers	HTTP ön bilgilerini tutan sözlük	Sözlük
cookies	Çerez bilgisini tutar	Sözlük
cookies-file	Çerez bilgisini içeren dosyanın yolu	Karakter dizisi
export-dir	Çıktı klasörü yolu	Karakter dizisi
export-db	Çıktı dosyası yolu	Karakter dizisi

3.3.2. do komutu

Çıkarılmış veriyle ne yapılacağı burada belirttirilir. Temel anlamda çıkarılmış verinin "kullanışlı" olması beklendiğinden bu verinin ya yeni bir veri çıkaracağımız başka bir sayfaya işaret etmesi ya da bir şekilde kaydedilmesi gerektiği varsayılmaktadır. Çizelge 3.5'te burada kullanılabilecek anahtar kelimelere ve açıklamalarına yer verilmiştir.

Çizelge 3.5. do komutu açıklaması

Anahtar kelime	**Açıklama**
to_link_queue	Yeni bir linki kuyruğa ekle
to_txt	Metin dosyasına kaydeder
to_db	JSON tabanlı bir veritabanına kaydeder

download	İkili bir dosyayı indirir
download_extra	İkili bir dosyayı indirmek için farklı bir araç kullanır.

3.3.3. lookfor komutu

Veri çıkarımının nasıl yapılacağının belirlendiği bölümdür. Bu tezin ikinci bölümünde anlatılan seçicilerden biri ya da birkaçı kullanılarak esnek olarak veri çıkarımı kuralı tanımlanabilir. JSONPath, XPath, CSS seçicileri, düzenli ifadeler ya da BeautifulSoup4 tabanlı seçiciler kullanılabilir. Seçicilerle ilgili ayarlamalara Çizelge 3.4'te yer verilmiştir.

Çizelge 3.3. lookfor komutu açıklamaları

type	Düğüm seçici yöntemi. Şunlardan biri olabilir: jpath, xpath, css, regex, soup	Karakter dizisi
selector	Seçilen seçici yönteme göre seçim komutu	Karakter dizisi
href	href	Mantıksal doğruluk

3.3.4. Değişkenler

URL'de değişen/artan sayfa numarası gibi durumlarda programlama dili kullanılmadan değişken tanımlamak için kullanılır.

"variables" listesi içinde yer almalıdırlar . Değişken geleceği yerlerde {{degiskenadi}} şeklinde tanımlanırlar. Değişken oluşturma komutları Çizelge 3.6'da gösterilmiştir.

Çizelge 3.6. Değişken tanımlamaları

Değişken	**Değişken adı**	**Karakter dizisi**
type	Şunlardan biri olabilir:	static, increment,decrement
start	Başlangıç değeri	Tam sayı
end	Bitiş değeri	Tam sayı
steps	Adım sayısı	Tam sayı

3.4. Örnek Kural Dosyası

```
{
  "project-name": "Quotes To Scrape",
  "project-url": "https://quotes.toscrape.com/scroll",
  "project-description": "Javascript Scrolling Example",
  "project-settings": {
    "user-agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:85.0)
Gecko/20100101 Firefox/85.0",
    "wait-between-requests": 1000,
    "http-headers": {
      "Accept-Language": "en-US,en;q=0.5",
      "referer": "https://quotes.toscrape.com/scroll"
    },
    "export-dir": "quotes-export",
    "filename": "quotes-export.json"
  },
  "seeds": [
    {
      "request": {
        "url": "https://quotes.toscrape.com/api/quotes?page={{i}}",
        "variables": [
          {
            "var": "i",
            "type": "increment",
            "start": 1,
            "end": 5,
            "steps": 1
          }
        ],
        "look_for": {
          "type": "json",
          "path": "quotes[*][text]",
          "do": "save_to_db"
        }
      }
    }
  ]
}
```

BÖLÜM 4

DENEYLER

Hazırlanan tarayıcısız kural tabanlı web veri çekme uygulaması, tarayıcıya bağlanarak veri çeken Selenium test uygulama çerçevese ile karşılaştırılmıştır.

Karşılaştırma yapmak için Google Chrome web tarayıcısının 103.0 sürümü ve uyumlu ChromeDriver kullanılmış, tarayıcıya etkileşim için Selenium 4.3.0 (Python) kütüphanesi kullanılmıştır.

Test makinesinin ve test ortamının özellikleri Çizelge 4.1'de gösterilmiştir.

Çizelge 4.1. Test ortamı özellikleri

İşlemci	Intel Core i7 1068NG7 (2.3 Ghz, 4 çekirdek, 8 iş parçacığı)
Bellek	16GB 3733 Mhz LPDDR4X
Grafik İşlem Birimi	Intel Iris Plus Graphics (Ice Lake)
İşletim Sistemi	macOS Monterey 12.4 (İnşa 21F79)
Python Sürümü	3.9.13
İnternet Bağlantı Hızı	~100 megabit ethernet

Geliştirilen uygulama AJAX tabanlı olmayan sayfalarda çalışabiliyor olsa da test için AJAX tabanlı siteler seçilmiştir.

Özlü Sözler: https://quotes.toscrape.com/scroll adresinde yer alan özlü sözlerin çıkarımı yapılmıştır. Selenium testlerinde kaydırma için siteye JavaScript kodu entegre

edilmiştir. Kaydırma, sayfanın tutarlı çalışabileceği ~600 ms'de bir yapılmıştır. Aynı bekleme süresi tez çalışmasında uygulamada iki istek arası bekleme süresi olarak belirlenmiştir.

Fotoğraf: Grafik ağırlıklı bir fotoğraf paylaşım sitesinin ana sayfasından en son yayınlanan 20 adet fotoğraf indirilmiştir. Fotoğrafları indirmede geçen süre ve istek sayısı değerlendirmeye alınmış, ancak aktarılan veri miktarı hesaplanırken fotoğrafların ham veri boyutu dikkate alınmamıştır.

E-Ticaret Yorum: Popüler bir alışveriş sitesinden bir ürüne ait ~500 adet yorum çekilmiştir. Yorumların yüklenmesi için sayfanın aşağı kaydırılması ya da ileri butonuna basılması gereklidir. Selenium tabanlı sistemde, bu işlem sayfaya JavaScript kodu entegre edilerek yapılmıştır.

Blog (Resim): Popüler bir blog sitesinden 50 adet yazının ilk resmi (tanımlayıcı resim) çekilmiştir. Yazıların listesi AJAX ile yüklenmektedir ve resimler de kaydırıldıkça yüklenmektedir.

Blog (Yazı): Popüler bir blog sitesinden 50 adet makalenin içeriği çekilmiştir. AJAX ile yüklenen yazı listesinden her bir yazının URL adresi kuyruğa alınıp daha sonra tek tek ziyaret edilmiştir. Ziyaret edilen sayfalardaki makalenin sadece metin kısmı çıkarılmış ve veritabanına kaydedilmiştir.

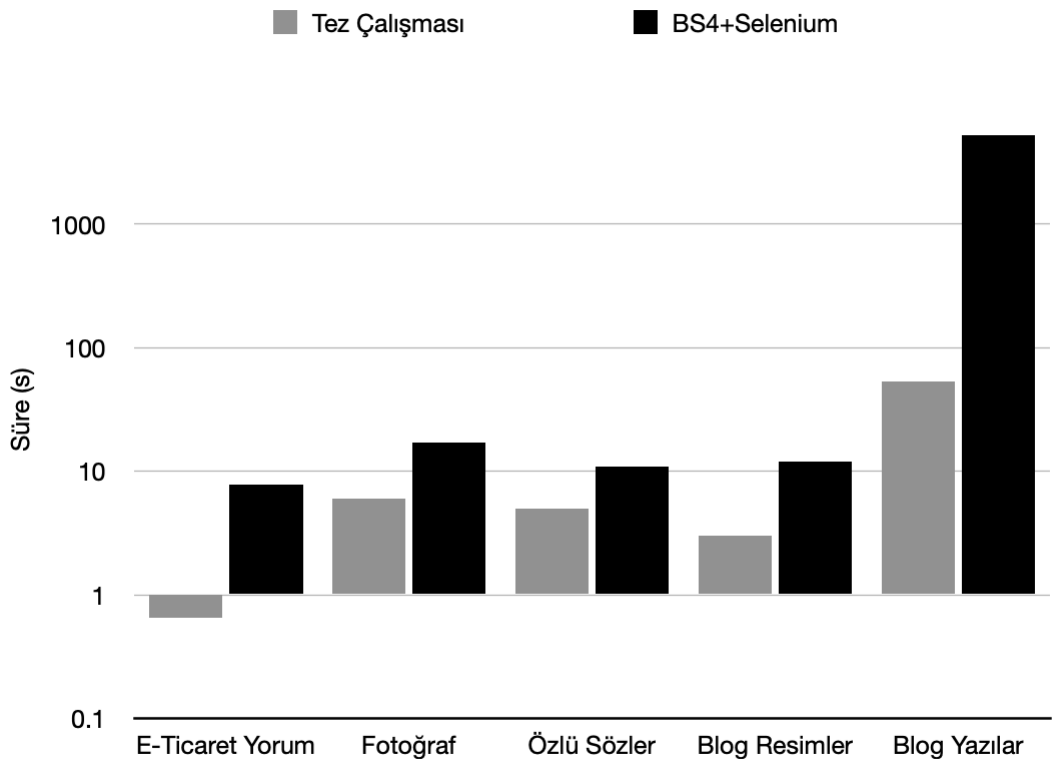
Farklı kategorilerdeki bu sitelerde hem süre göz önüne alınmış, hem de gelen/giden istek sayısı ve isteklerin toplam boyutu hesaplanmıştır.

4.1. Süre Karşılaştırması

Şekil 4.1.'de süre bakımında geliştirilen yazılım ve Selenium tabanlı çözümün çıkarım süresi karşılaştırılmıştır. Süre saniye olarak hesaplanmış, grafik logaritmik olarak çizilmiştir.

Sürelerin karşılaştırması için her iki sistem de aynı sitede 3 kez çalıştırılarak sürelerin aritmetik ortalaması alınmıştır.

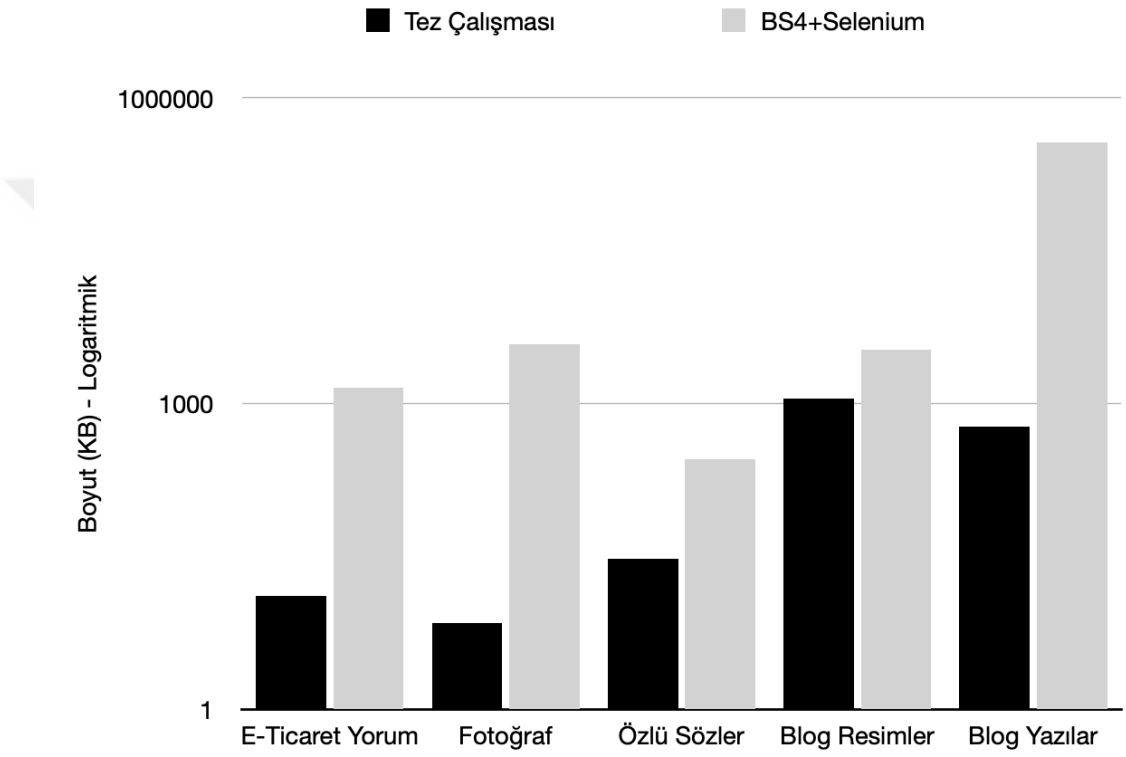
Selenium ile çalışan sistemlerde süre hesaplamasında tarayıcının açılması beklenmiş ve sistem o şekilde başlatılmıştır. Bu sayede tarayıcının başlangıcı için geçen süre hesaba katılmamıştır. Ancak normal kullanımda tarayıcı açılışının da bir zaman kaybı oluşturulacağı göz önüne alınmalıdır.



Şekil 4.1. Kural tabanlı ve tarayıcı tabanlı sistemlerin süre karşılaştırması (logaritmik)

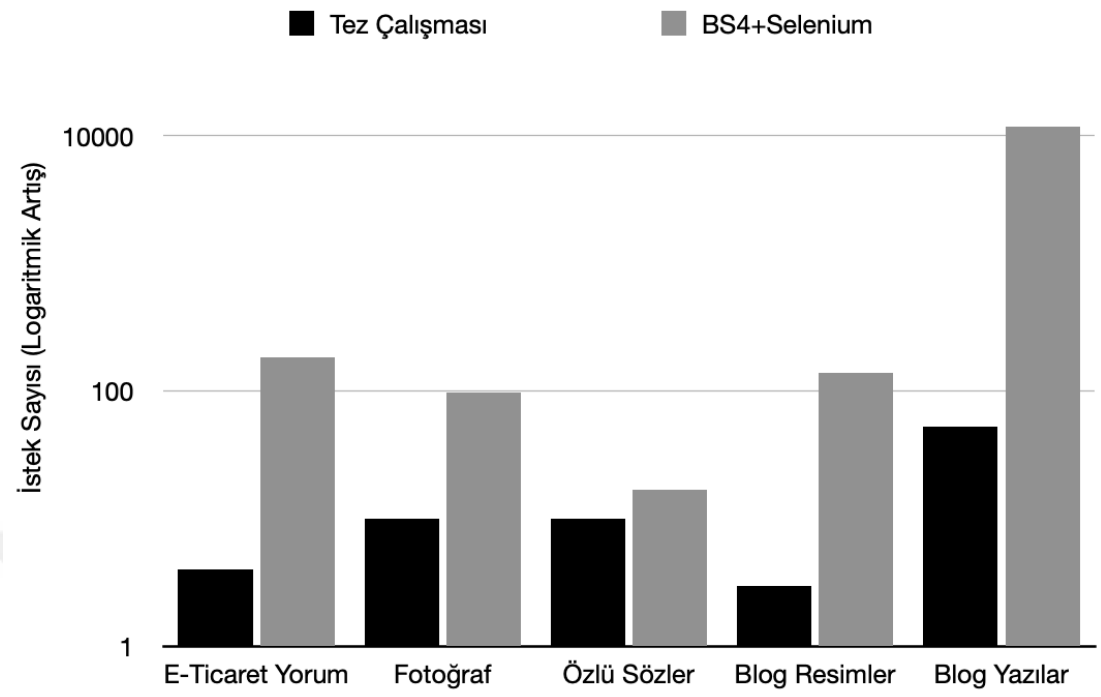
4.2. İstek Sayısı ve Aktarılan Veri Miktarı Karşılaştırması

Şekil 4.2.'de kural tabanlı yaklaşım ile geliştirilen yazılımın ve Selenium tabanlı çözümün alıp gönderdiği HTTP(S) istek sayıları karşılaştırılmıştır. Selenium tabanlı testlerde veri çekme işlemi durdurularak tarayıcı içinde geliştirici araçlarından istek bilgisi alınmıştır.



Şekil 4.2. Toplam aktarılan veri miktarı karşılaştırması (logaritmik)

Şekil 4.3.'te ise tüm isteklerin toplam boyutu karşılaştırılmıştır. Çalışmada geliştirilen kural tabanlı çözümün boyutu kod içerisinde hesaplanırken Selenium tabanlı çözümde tarayıcının geliştirici araçları kullanılmıştır.



Şekil 4.3. Gelen/Giden HTTP(S) istek sayısı (logaritmik)

4.3. Veri Çıkarımında Karşılaşılan Zorluklar ve Çözümler

Bu bölümde, veri çekme ve veri çıkarma robotlarıyla birlikte web örümceklerinin veriye erişimde karşılaşabileceği sorun ve zorluklara değinilmiş ve bazılarında çözümler önerilmiştir.

Bir web sitesinin taranması ve indirilmesi aşamasında, ilgili web sitesinin sahibi kişi, kurum ya da kuruluştan yazılı izin alınması, işlem yapılacak IP adreslerinin bildirilmesi ve karşı taraftan istek sıklığı, uygun tarih ve saat gibi bilgilerin alınması tercih edilmelidir.

Burada karşılaşılabilecek her bir zorluğun, ilgili web sitesinin sahibinin indekslenmesini ya da sitesinden veri çekilmesini istemediği anlamına gelebileceği unutulmamalıdır.

4.3.1. HTTP 429 Hata Kodu ve Sıklık Sınırlaması

HTTP 429 "Çok Fazla İstek" hata kodu belirli bir sunucuya belirli zaman aralığında yine belirli bir sayıdan fazla istek gönderilmesi sonucunda sunucunun isteklere cevap göndermek yerine gönderdiği bir HTTP yanıtıdır.

HTTP 429 hatasının oluşturulacağı zaman sınırlaması, sınırlamanın ne zaman kaldırılacağı, sınırlamanın IP adresi, çerez ya da kullanıcı aracısına (tarayıcı vb.) göre olup olmayacağı, sunucuyu işleten kişi tarafından belirlenir.

Genelde veri çekme aşamasında çok fazla sayfaya aynı anda paralel olarak ya da üst üste çok fazla istek gönderilebileceğinden karşılaşılma olasılığı oldukça yüksek olan bir sorundur.

Bu sorunun en makul çözümü aynı sunucuya gönderilecek istekler arasına daha veri çekme sürecinin başında belirli bir zamana aralığı koymaktır. Bu sayede hem karşı sunucu gereksiz yorulmamış olur hem de benzeri hatalarla karşılaşılmamış olur.

```
HTTP/1.1 429 Too Many Requests
Content-Type: text/html
Retry-After: 3600
```

Web sunucuları tekrarlı olarak istek yollayan istemcileri kısa süreli ya da kalıcı olarak engelleyebilirler. Geçici engellemeler için *Retry-After* başlığı ile kullanıcı aracısına beklemesi gereken süre bildirilebilir. Böyle durumlarda bu başlık bilgisine uymak kalıcı engellemeleri önlemek için önem arz etmektedir.

Retry-After başlığı, HTTP tarih ifadesi ya da saniye cinsinden süre belirtebilir.

```
Retry-After: Wed, 21 Nov 2022 10:02:20 GMT
Retry-After: 80
```

4.3.2. JavaScript Robot Kontrolü ve CAPTCHA

Genelde basit olarak tasarlanmış veri çekme veya veri çıkarma algoritmaları HTML kodlarına yöneldiğinden, JavaScript ile oluşturulan içeriği dikkate almazlar. Bu da tek sayfalı web uygulamaları gibi tamamen JavaScript teknolojisini kullanan web

sitelerinde sorun oluşturmaktadır. JavaScript kullanımı, genelde robotları engellemek için kullanılan bir teknoloji olmasa da robotlar için bir sorundur.

JavaScript sorununu çözen robotlar için bazı web sayfaları otomatik programları (robotlar) engellemek için JavaScript tabanlı bazı yöntemler geliştirilmişlerdir. Bu yöntemler, genellikle insan dışı ziyaretlerin ve etkileşimlerin istenmediği bölümlere konur. Bu yöntemleri başında CAPTCHA adı verilen doğrulamalar yer alır.

CAPTCHA **(**Completely Automated Public Turing test to tell Computers and Humans Apart**),** bilgisayarları ve insanları ayırmak için tamamen otomatik halka açık Turing testi anlamına gelen İngilizce kelimelerin kısaltmasıdır. CAPTCHA testleri bilgisayarın çözmekte zorlanacağı, ancak insanların kolaylıkla çözeceği konulardan seçilir. Örneğin, eğri büğrü karakterleri okuyabilme, belirli resimleri sınıflandırabilme, tam resimde belirli objeleri bulabilme gibi konular bir bulmaca şeklinde kullanıcıya sunulur.

Yapay sinir ağları ve derin öğrenme, özellikle resimden karakter tanıma gibi CAPTCHA problemlerinin artık bilgisayar tarafından da çözülebildiğini göstermiştir. Bunun üzerine genellikle CAPTCHA problemleri farklı alanlara kaymıştır. Şekil 4.4 ve 4.5'te CAPTCHA örneklerine yer verilmiştir.

Şekil 4.4. Eski nesil bir CAPTCHA örneği

Şekil 4.5. Yeni nesil bir CAPTCHA örneği (hCaptcha)

CAPTCHA problemlerinin üretimi, çözümlerin verimli kullanımı ve risk analizi ayrı bir çalışma alanı olsa da, CAPTCHA ile korunan bir sayfada veri çekimi veya işlenmesini istenmediği düşünülerek, bu çalışmada CAPTCHA problemleri için bir çözüm önerisi getirilmemiştir.

4.3.3. Robot Dışlama Standardı (robots.txt)

Robot dışlama standardı ya da kısa adıyla robots.txt, bir web sitesinin, otomatik programlarla haberleşmesi için oluşturulmuştur. Bir web sitesi, arama motorları, veri indirici ve veri çıkarıcı programlarla robots.txt adı verilen bir programla haberleşir.

Hangi sayfaların indekslenebileceği ya da gezileceği, hangi sayfaların ise otomatik programlarla ziyaret edilmesinin istenmediği bu dosyada belirttirilir. Ayrıca hangi kullanıcı aracısına (ki robotlar kendilerini kullanıcı aracısı HTTP başlığı ile tanımlar) izin verildiği, ziyaret ve istek sıklığı ve site haritası (sitenin tüm sayfalarının adreslerini içeren XML dokümanı, bu doküman sayesinde link çıkarımı yapmaya gerek kalmadan sitenin taranması mümkün olabilir) gibi bilgiler de bu dosyada yer alır.

Örnek bir robots.txt örneği aşağıdaki gibidir:

```
User-agent: googlebot                 # Tüm Google servisleri
Disallow: /googledangizliklasor/     # Bu klasöre Google bakmasın

User-agent: googlebot-news          # Google Haberler servisi
Disallow: /                         # Hiçbir sayfaya bakmasın

User-agent: *                       # Herhangi bir robot
Disallow: /gizli/                   # Bu klasöre bakmasın
```

Tüm robotların benzersiz bir kullanıcı aracısı karakter dizisi seçmesi ve robots.txt dosyalarına uyması beklenir. Programların bu kuralları dikkate almaması mümkün olsa da bu kurallara uymayan robotlar için IP bazlı engellemeler getirilmesi olasıdır.

4.3.4. HTTP Başlığı ve Tarayıcı Kontrolleri

4.3.4.1. Kullanıcı Aracısı Kontrolü

Bazı web sunucuları, sayfa içeriğini kullanıcı aracısına göre göndermektedir. Bu durum, örneğin yazılım indirme sayfalarında kullanıcının işletim sistemine uygun yazılımın otomatik seçimi gibi işlevselliklere olanak sağlamaktadır.

Bununla birlikte sıklıkla kullanılan tarayıcılar ve bu tarayıcıların sürümleri, kullanıcı aracısı karakter dizisi içerisinden tespit edilip, olağan dışı tarayıcılar için sayfa içeriğini göstermeme ya da eksik gösterme gibi önlemler alınabilmekte, tarayıcılar dışındaki kullanıcı aracıları engellenebilmektedir.

HTTP başlıklarında yer alan *User-Agent* karakter dizisi değiştirilerek kullanıcı aracısı kolaylıkla değiştirilebilmektedir. Ancak web sunucusuna programın kendini doğru tanıtması açısından benzersiz bir kullanıcı aracısı tanımlayıcı seçmesi önemlidir.

Örnek bir tarayıcı kullanıcı aracısı bilgisi aşağıdaki gibidir. Tarayıcının adı ve sürümü, kullanıldığı işletim sistemi adına bilgiler içermektedir.

```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0
Safari/537.36
```

4.3.4.2. Öneren Sayfa Kontrolü

Tarayıcılar, varsayılan ayarlarda ziyaret edilen sayfaya, hangi sayfadan geldiklerini (öneren) otomatik olarak Referer HTTP başlığı ile bildirmektedir.

Önerici sayfa, bir web sitesine hangi arama motorlarından hangi anahtar kelimelerle gelindiği gibi bilgileri öğrenmek adına önemlidir. Ancak site içerisinde dolaşımda belirli sayfalara belirli bir gezinti örüntüsü içerisinde ulaşılması bekleniyorsa, önerici sayfanın ilgisiz bir sayfa olması, sunucunun bu isteğe cevap vermemesi ya da yanlış cevap vermesi olasılığını da doğurmaktadır.

Düzgün çalışan bir web robotu, bir sayfayı nereden bulduğu bir bağlantı ile ziyaret ettiği bilgisini depolayıp öneren HTTP başlığına eklemelidir.

4.3.4.3. Diğer Kontroller

Ziyaret geçmişinin oturum ya da çerezler ile takip edildiği, HTTP etikete ve JavaScript kodlarının tutarlı çalışıp çalışmadığı gibi kontroller ile bir web sunucusu robotları engelleyebilir.

AJAX istekleri için bazı özel HTTP başlıkları oluşturulup tutarlılığı kontrol edilebilir. Her istek bir sonrakiyle ilgili bir anahtar veri taşıyor olabilir.

4.3.5. Çerez Kontrolleri

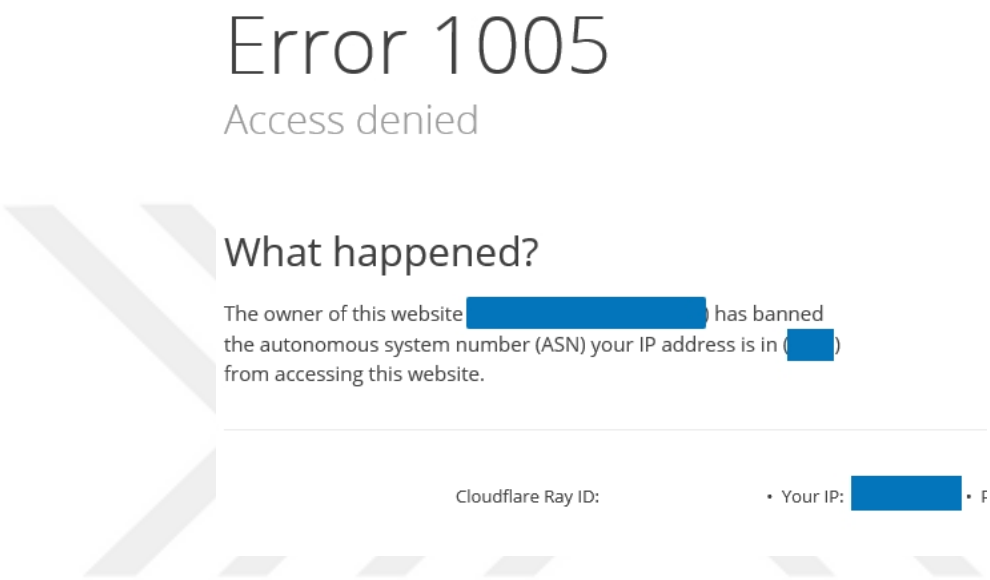
Standart bir web tarayıcının, çerez ve oturum bilgilerini düzgün bir şekilde tutması beklenir. Belirli sayfalara erişim ise yalnızca oturum açılarak mümkündür. Oturum bilgisinin web veri çekme programı tarafından da tutulması, gerekli durumlarda oturum açma ya da oturum çerezlerinin saklanması gerekebilir.

4.3.6. Veri Merkezi / İkamet Alanı IP Adresi Tespiti

Bir web sitesinin ziyaretçilerinin genellikle ev ya da mobil kişisel ev bağlantısı kullanılması beklenebilir. Web örümcekleri, veri çekme robotları gibi programlar ise daha çok veri merkezleri, akademik kurumlar ve iş yerlerinde çalışmaktadır. Bu nedenle, bazı sunucular özellikle veri merkezi IP adreslerinden gelen isteklere karşı “daha temkinli” yaklaşmakta ve bu IP adreslerine daha sert sıklık sınırlandırma algoritmaları uygulayabilmekte ya da istekleri reddetmektedir.

Bu amaçla, istekleri gönderen IP adresi, bilinen otonom sistem numaraları (ASN) IP aralıklarında aranıp ilgili IP adresinin ikamet alanı, veri merkezi, eğitim, VPN sağlayıcı vb. olup olmadığı tespit edilebilir. Örneğin 8517 otonom sistem numarası ULAKBİM'e ait olup, bu sisteme dahil bir IP adresinin bir üniversiteden gelme olasılığı çok yüksektir.

Şekil 4.6'da ASN kaynaklı erişim engelleme sonucu karşılaşılan bir hata kodu gösterilmektedir.



Şekil 4.6. Örnek bir ASN engelleme hatasının ekran görüntüsü

IP adresinin VPN, vekil sunucu vb. kullanımı ile değiştirilmesi mümkün olsa da bu çalışmada bu konuyu ele almamaktadır.

BÖLÜM 5

SONUÇLAR

5.1. Çalışmanın Sonuçları

Tez çalışması kapsamında oluşturulan bilgisayar yazılımıyla yapılan deneyler, hem AJAX tabanlı sayfalarda hem de statik sayfalarda tarayıcısız veri çıkarımı için kural tabanlı sistemlerin kullanılabileceğini göstermiştir.

Oluşturulan kural dosyalarının kolaylıkla paylaşılabilir olması, dağıtık sistemlerde kullanımı kolaylaştırmakta ve programlama dili kullanılmadan veri çıkarımı yapılmasına olanak sağlamaktadır. Ancak, kural dosyalarının doğru oluşturulması başarımı önemli ölçüde etkilemektedir.

Yapılan deneyler göstermiştir ki, her ne kadar çıkarılacak verinin sayfanın büyük bölümünü oluşturduğu web sitelerinde (toscrape.com gibi) performans artışı görülmüş olsa da, sayfa karmaşıklığının arttığı, sayfada yoğun reklam, navigasyon içeriği gibi hedef dışı içeriğin olduğu durumlarda, kural tabanlı yaklaşım "hedefe yönelik" bir çözüm sunabildiğinden performans (süre) tarayıcının çalıştığı durumlara göre 10 kattan fazla artmıştır.

Özellikle birden fazla sayfanın gezilerek metin veriye odaklanan deneylerde ise istek sayısında 200 kattan fazla azalma olmuştur. Aynı şekilde indirilen veri miktarı 600'de birine düşmüştür.

İstek ve indirilen boyutun azaltılmasının veri çekimi sırasında sunucuya binecek olan yükü de önemli ölçüde azalttığı gözlenmiştir.

Tüm testler yapılırken tarayıcı ile çalışan yaklaşımda tarayıcının açılması için geçen süre ve test başlayana kadarki bellek kullanımı dikkate alınmamıştır. Bunlar dikkate alındığında, zaman ve bellek başarımının daha da yüksek olacağı söylenebilir.

5.2. Gelecek Çalışmalar

Bu tez kapsamında yapılan çalışmada, statik ve AJAX tabanlı web sayfalarından çıkarım yapmak için kural tabanlı yaklaşım ile çalışan bir bilgisayar yazılımı hazırlanmıştır.

Gelecek çalışmalarda, kural dosyaların otomatik oluşturulması adına web sayfası içeriğinin klasik algoritmik yaklaşımlarla işlenmesi, doğal dil işleme algoritmaları kullanılması ya da yapay öğrenme yöntemlerine başvurulması düşünülebilir.

KAYNAKLAR

Baumgartner, R., Frölich, O., & Gottlob, G. (Haziran 2007). The Lixto systems applications in business intelligence and semantic Web. In European Semantic Web Conference (sayfa. 16-26). Springer, Berlin, Heidelberg.

Carey H. J. & Manic M. (2016). "HTML web content extraction using paragraph tags," 2016 IEEE 25th International Symposium on Industrial Electronics (ISIE), 2016, pp. 1099-1105, doi: 10.1109/ISIE.2016.7745047.

Crescenzi, V., Mecca, G.& Merialdo, P. (2001). Roadrunner: Towards automatic data extraction from large web sites. In VLDB (Vol. 1, pp. 109-118).

D. Pan, S. Qiu & D. Yin, "Web Page Content Extraction Method Based on Link Density and Statistic," 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008, pp. 1-4, doi: 10.1109/WiCom.2008.2664.

Dalvi, N., Kumar, R. & Soliman, M. (2011). Automatic wrappers for large scale web extraction. arXiv preprint arXiv:1103.2406.

Diffbot (2022). The Knowledge Graph of the Public Web. 22 Haziran 2022 tarihinde https://diffbot.com adresinden erişildi.

ECMA International (Haziran 2021). ECMA-262 – EcmaScript language specification. 9 Mayıs 2022 tarihinde https://www.ecma-international.org/publications-and-standards/standards/ecma-262/ adresinden erişildi.

Fayzrakhmanov, R.R., Sallienger, E., Spencer B., Furche T. & Gottlob G. (2018). Browserless Web Data Extraction: Challenges and Opportunities. WWW (International World Wide Web Conference), 2018, Nisan 23-27, 2018, Lyon, Fransa.

Ferrara, E., de Meo, P., Fiumara, G. & Baumgartner, R. (2014). Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems*, *70*, 301–323. https://doi.org/10.1016/j.knosys.2014.07.007

Fielding, R. Irvine, U.C., Gettys J., Mogul J. & Frystyk H (Haziran 1999). Hypertext Transfer Protocol. HTTP/1.1. IETF. doi:10.17487/RFC2616. RFC 2616, New York, NY, USA, 245–254. https://doi.org/10.1145/2009916.2009952

FMiner (2015). Visual web scraping software with macro recorder and diagram designer . 22 Haziran 2022 tarihinde http://www.fminer.com/ adresinden erişildi.

Furche, T., Gottlob, G., Giovanni, G., Schallhart, C. & Seller, A (2012). The VLDB Journal. DOI: 10.1007/s00778-012-0286-6

Google Dev Docs , Google Inc. (2022). Google tarayıcılarına genel bakış (kullanıcı aracıları). 22 Haziran 2022 tarihinde

https://developers.google.com/search/docs/advanced/crawling/overview-google-crawlers?hl=tr adresinden erişildi.

Hajba, G. L. (2018). "Using Beautiful Soup", Website Scraping with Python: Using BeautifulSoup and Scrapy, Apress, pp. 41–96, doi:10.1007/978-1-4842-3925-4_3, ISBN 978-1-4842-3925-4

Import.io (2022). Enterprise scale eCommerce data to drive growth. 22 Haziran 2022 tarihinde https://import.io adresinden erişildi.

Kranzdorf, J., Sellers, A., Grasso, G., Schallhart, C. & Furche, T. (Nisan 2012). WWW (International World Wide Web Conference) 2012, Lyon, Fransa

MDN (Mozilla Developer Network). (2022). CSS Selectors. 9 Mayıs 2022 tarihinde https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors adresinden erişildi.

MDN (Mozilla Developer Network). (2022). HTTP Headers. 9 Mayıs 2022 tarihinde https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers adresinden erişildi.

Mehta B. & Narvekar M., "DOM tree based approach for Web content extraction," 2015 International Conference on Communication, Information & Computing Technology (ICCICT), 2015, pp. 1-6, doi: 10.1109/ICCICT.2015.7045706.

Mesbah, A., Deursen A. & Lenselimk S. (2012). Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes. ACM Transactions on the Web Volume 6 Issue 1. https://doi.org/10.1145/2109205.2109208

Mozenda (2022). Mozenda. 23 Haziran 2022 tarihinde http://www. mozenda.com adresinden erişildi.

Muslea, I., Minton, S., & Knoblock, C. (Nisan 1999). A hierarchical approach to wrapper induction. In Proceedings of the third annual conference on Autonomous Agents (sayfa. 190-197).

Octoparse (2022). Easy Web Scraping for Anyone. 22 Haziran 2022 tarihinde https://octoparse.com adresinden erişildi.

Rex Egg, Quick-Start: Regex Cheat Sheet.10 Mayıs 2022 tarihinde https://www.rexegg.com/regex-quickstart.html adresinden erişildi.

Sen, K., Swaroop, K., Brutch, T. & Gibbs, S. (Ağustos 2013). Jalangi: a selective record-replay and dynamic analysis framework for JavaScript. ESEC/FSE 2013: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (sayfa 488-498). DOI: 10.1145/2491411.2491447

Sun F., Song D. & Liao L. (2011). DOM based content extraction via text density. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR '11). Association for Computing Machinery.

Uzun, E., Yerlikaya, T. & Kırat, O. (2018). Comparison of Python libraries used for web data extraction. Fundamental Sciences And Applications, 87.

Uzun, E., Agun, H. V., & Yerlikaya, T. (2013). A hybrid approach for extracting informative content from web pages. *Information Processing & Management*, *49*(4), 928-944.

Uzun, E. (2020). A novel web scraping approach using the additional information obtained from web pages. *IEEE Access*, *8*, 61726-61740.

W3Schools. (2022). CSS Tutorial.9 Mayıs 2022 tarihinde https://www.w3schools.com/css/default.asp adresinden erişildi.

WHATWG (Apple, Google, Microsoft ve Mozilla) (Mayıs 2022). HTML: Living Standard. 9 Mayıs 2022 tarihinde https://html.spec.whatwg.org/multipage/ adresinden erişildi.

World Wide Web Consortium (W3C) (2021). Cascading Style Sheets. Standart (W3C Önerisi). 9 Mayıs 2022 tarihinde https://www.w3.org/Style/CSS/#specs adresinden erişildi.

World Wide Web Consortium (W3C) (2017). Cascading XML Path Language (XPath) 3.1. Standart (W3C Önerisi). 9 Mayıs 2022 tarihinde https://www.w3.org/TR/xpath-31/adresinden erişildi.

Xia, T. (2009, April). Extracting structured data from Ajax site. In 2009 First International Workshop on Database Technology and Applications (pp. 259-262). IEEE.

Zhou, Z. & Mashuq, M. (2014). Web content extraction through machine learning. Standford Univ, 1-5.