European Journal of
Engineering and Applied Sciences

ISSN: 2651-3412 **Journal homepage:** dergipark.gov.tr/EJEAS

Published by Çorlu Faculty of Engineering, Namık Kemal University

Original Research Article

Web Veri Çıkarımında Çıkarım Kurallarının İncelenmesi

Erdinç Uzun[1,*], Tarık Yerlikaya[2], Oğuz Kırat[2]

[1] Bilgisayar Mühendisliği Bölümü, Çorlu Mühendislik Fakültesi, Tekirdağ Namık Kemal Üniversitesi, Tekirdağ, Türkiye
[2] Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Trakya Üniversitesi, Edirne, Türkiye

Geliş: 21.11.2018 *Kabul: 27.12.2018*

Özet: Gerekli veriyi web sayfasından çıkarmak veri madenciliği ve bilgi erişimi alanındaki uygulamalar için önemlidir. Web sayfasından veriyi çıkarmak için DOM tabanlı yöntemler veya düzenli ifadeler kullanılabilir. Bu çıkarım işlemi için hem DOM tabanlı yöntemler hem de düzenli ifadeler için birden fazla çıkarım kuralı hazırlanabilir. Bu çalışmada, çıkarım kuralları ile birden fazla veriyi elde etmenin çıkarım işlemi üzerindeki etkinliği incelenmiştir. Veri seti olarak haber, film ve alış/veriş alanlarında olmak üzere on beş web sitesi seçilmiştir. Bu web siteleri için farklı çıkarım teknikleri ile veri çıkarımı için çıkarım kural dosyaları oluşturulmuştur. Web sitelerinde özellikle yorum gibi tekrarlayan veriler üzerinde odaklanmıştır. Deneyler, oluşturulması daha zahmetli ve zaman alıcı düzenli ifadelerin DOM tabanlı yöntemlere göre çok daha iyi sonuçlar verdiğini göstermiştir. DOM tabanlı yöntemler arasında beklenildiği gibi lxml ayrıştırıcı kütüphanesi en iyi sonuçları vermiştir. Deneyler, bir geliştirici tarafından hazırlanan çıkarım kurallarının çıkarım süresini etkilediği göstermektedir. Sonuç olarak, iyi hazırlanmış çıkarım düzenli ifadeleri ile web sayfalarında çok daha hızlı bir şekilde istenilen veriye erişmek mümkündür.

Anahtar kelimeler: *Çıkarım yöntemleri, Web veri çıkarımı, DOM, Düzenli ifadeler.*

Examination of Extraction Rules in Web Data Extraction

Abstract: Extracting the desired data from a web page is an important issue for applications in the fields of data mining and information retrieval. DOM-based methods or regular expressions can be used to extract data from a web page. For this extraction process, multiple extraction rules can be prepared for both DOM-based methods and regular expressions. In this study, the effectiveness of obtaining repetitive data using extraction rules is investigated. As a data set, fifteen websites including in the fields of news, films, and shopping have been selected. Extraction rule files have been created for data extraction with different extraction techniques for these websites. Websites are mainly focused on repetitive data such as reviews. Experiments have shown that regular expressions, the creation process is more laborious and time-consuming, give better results than DOM-based methods. Among the DOM-based methods, the lxml parser library provided the best results as expected. Experiments indicate that the extraction rules prepared by a developer affect the extraction time. As a result, it is possible to extract the desired data much faster in web pages with the well-prepared regular expressions.

Keywords: *Extraction methods, Web data extraction, DOM, Regular expressions.*

Received: 22.11.2018 *Accepted: 27.12.2018*

* Sorumlu yazar.
E-posta adresi: erdincuzun@nku.edu.tr (E. Uzun)

1. Giriş

İnternetin kullanımın artması ile birlikte web sayfalarındaki içerik miktarı ve çeşitliği de artmıştır. Web veri çıkarımı[1] (web kazıma, web içerik çıkarımı olarak da bilinen) web sayfaları üzerinden geliştiricinin ihtiyaç duyduğu verileri elde etme işlemidir. Günümüzde, web ortamında çok fazla miktarda bilgi üretilmekte ve paylaşılmaktadır. Web veri çıkarımı[2] işlemi sayesinde çok zor toplanacak veriler daha kısa sürede ve daha az çaba harcanarak toplanabilir. Bu veriler toplanırken bir uzman tarafından belirlenen çıkarım kuralları kullanılır. Bu çalışmada, üretilebilecek farklı çıkarım kurallarının çıkarım performansı üzerindeki etkisi incelenmiştir.

Web içerik çıkarımı çalışmaları Wrapper[3]–[8], DOM (Document Object Model)[9], [10] ve makine öğrenmesi[11]–[13] tabanlı çalışmalar olmak üzere üç gruba ayrılabilir. Wrapper, web sayfalarından belirli bir içeriğin ayıklayan bir programdır. DOM tabanlı yöntemler, HTML ile işaretlenmiş etiketlerin yapısını, etiketlerini ve özelliklerini kullanır. Makine öğrenimine dayalı teknikler, işaretlenmiş veriler üzerinden makine öğrenimi algoritması kullanarak tahmin üretmeyi amaçlar. Genelde bu üç farklı çalışmada, DOM üzerinden elde edilen etiketli verileri kullanılması gerekir. Bu makale, çalışma Wrapper işlemi için kullanılabilecek farklı tekniklere odaklanmıştır. Çalışmanın amacı performans açısından en uygun teknik ve metotların belirlenmesidir.

Performans testleri için Python programlama dilinde üç farklı ayrıştırıcı desteği olan Beautiful Soup Kütüphanesi ve düzenli ifadeler kullanılmıştır. Daha önceki çalışmamızda içerik, yazar bilgisi, yayın tarihi, konu başlığı, resim gibi tek etiketten oluşan veriler üzerine odaklanmıştık[14]. Bu çalışmada benzer çıkarım yöntemleri kullanılmıştır. Ancak, bu çalışma benzer yöntemleri kapsamakla birlikte yanı sıra farklı çıkarım kurallarının etkisi ve tekrarlayan verinin elde edilmesi üzerinde durulmuştur. On beş farklı web sitesinden 50 farklı web sayfası indirilmiş ve on beş web sitesi için üç farklı çıkarım kuralı düzenlenmiştir.

İkinci bölümde web çıkarım işleminin nasıl yapıldığı anlatılmıştır. Üçüncü bölüm oluşturulan veri seti hakkında bilgileri kapsar. Dördüncü bölüm deneylere ayrılmıştır. Son bölüm sonuç ve tartışmayı içerir.

2. Bir Web Sayfasının Yapısı ve Çıkarım İşlemi

Bir web sayfası HTML (Hypertext Markup Language) etiketleri ve HTML etiketleri arasındaki veriden oluşur. HTML etiketleriyle birlikte görselleştirmeyi iyileştirmek için CSS (Cascade Style Sheets) ve Javascript kodları gibi ek veriler kullanılabilir Web tasarımcısı kendisine ait CSS etiketleri tanımlayabilir ve içeriği kendisi belirler. Bu gibi özellikler HTML etiketlerini kullanarak veriye ulaşmamıza olanak tanır. Şekil 1'de çok basit bir web sayfası etiket içeriği vardır.

Bir web sayfası aslında bir metin dosyasıdır. Bu metin dosyası HTML kurallarına uygun olarak oluşturulur. HTML hiyerarşik bir düzene sahiptir. Örneğin, bir div etiketi başka div etiketleri içerebilir. Şekil 1'de id ve class bilgileri ayrı CSS dosyasında tanımlanır. Örneğin bu tanım dosyasında etiketine ulaşmak için div#reviews şeklinde bir tanımlama yapılabilir. Bu işlem CSS seçim işlemi olarak adlandırılır.

```
<html>
<head>
  <title> Bir web sayfası örneği </title>
  …
</head>
<body>
  <div id="menu"> … </div>
  <div id="content"> … </div>
  <div id="reviews">
    <div class="review"> … </div>
    <div class="review"> … </div>
    <div class="review"> … </div>
    <div class="review"> … </div>
    <div id="pages">
      <a id="first" class="page" href=…>…</a>
      <a class="page" href=…> … </a>
      <a class="page" href=…> … </a>
      <a class="page" href=…> … </a>
      <a id="last" class="page" href=…> … </a>
    </div>
  </div>
</body>
</html>
```

Şekil 1. Bir web sayfası örneği

Şekil 1'de "menu" kısmı web sayfasındaki diğer web sayfalarını bağlantı içerir. "menu" isimlendirmesi tamamen web tasarımcısına kalmış seçimlik bir işlemdir. Web tasarımcısı kendisine ait "id" veya "class" özellikleri tanımlayabilir. Ayrıca, web tasarımcısı tamamen kendine özel bir tasarım yapabilir. "menu" bölümünün içeriğinde genelde diğer sayfalara bağlantı vermeyi sağlayan "a" etiketi kullanılır. "a" etiketinin "href" özelliği diğer bir web sayfasına bağlantı için kullanılır. Şekil 1'de "content" bölümü web sitesinin içeriğine ayrılmıştır. "reviews" bölümü ise kullanıcı yorumlarının görüldüğü bölümdür. "reviews" dört adet "review" içermektedir. Ayrıca, "review" bölümü yazar ismi, tarihi, puanı ve yorum metni gibi alt bölümlerden oluşabilir. "reviews" bölümü içindeki "pages" bölümü ise diğer yorumlara erişmek için bağlantıları içerir. Bu işleme sayfalama işlemi denir. Bir web sayfasında sayfalamanın amacı sunucu tarafından istemci tarafına daha az veri yollamak ve kullanıcının çok fazla metin içinde kaybolmasını engellemektir. Yorumların ilk sayfası id özelliği "first" değerine sahipken son sayfası ise "last" değerine sahiptir. Ayrıca örnek, üç adet alt bağlantı sayfası da içermektedir.

2.1. DOM yapısı

Bir web tarayıcı bir web sayfasını istekte bulunduğu sunucudan metin olarak indirir ve sayfanın yorumlanması işlemi başlar. Sayfa yorumlanması sırasında ilk yapılan işlem DOM ağacının oluşturulması işlemidir. DOM ağacı oluşturulduktan sonra Javascript betimleri çalıştırılır ve web sayfasının yüklemesi tamamlanır. DOM, HTML elementlerini yönetmek için bir ağaç yapısı şeklinde düğümleri sunan bir ara yüzdür. DOM içeriği Javascript, C#, Java, Python ve benzeri programlama dilleri ile değiştirilebilir veya bu elementleri içindeki veriye ulaşılabilir. Bir web sayfasındaki DOM elementlerini yönetmek için Javascript kullanılır. DOM web çıkarım işleminde genelde tercih edilmektedir. Birçok web çıkarım işlemi aracı DOM üzerinden çıkarımı tercih etmektedir. Örneğin, en popüler kütüphanelerden biri olan

Scrapy[1] çıkarım işlemini DOM üzerinden yapmaktadır. Python dilinde geliştirilen bu kütüphane DOM işlemi için lxml kütüphanesi tercih etmektedir. Ayrıca, birçok programlama dilinde DOM yaratmak ve üzerinde işlem yapmak için çok farklı kütüphaneler geliştirilmiştir. Örneğin,

- .Net: HAP, AngleSharp, Microsoft HtmlDocument [2]
- Java: jsoup, Lagarto, HtmlCleaner [3]
- Python: html5lib, html-parser, lxml[4]
- Node.js: Cheerio, Jsdom, Htmlparser2[5]

2.2. DOM ayrıştırma kütüphaneleri ve düzenli ifadeler

Bu çalışmada, Python için geliştirilmiş popüler üç kütüphane performans açısından karşılaştırılacaktır. Ayrıca, DOM yapısı oluşturmadan düzenli ifadeler (regular expressions) ile çıkarım işlemi yapılacaktır. Aslında düzenli ifadeler metin üzerinde bir desen yakalamak için kullanılan iyi bilinen bir tekniktir. Ancak, iç içe etiket yapısına sahip web sayfaları için problem oluşturabilir. Bu çalışmada, doğru düzenli ifadelerin çıkarım işlemine katkısı da incelenecektir.

html5lib, html-parser ve lxml kütüphaneleri incelemek için Beautiful Soup adında kütüphaneden yararlanılmıştır. Algoritma 1, bu kütüphane kullanılarak bir DOM oluşturma ve DOM üzerinden çıkarım işlemi örneği verilmiştir.

```
import bs4
soup = bs4.BeautifulSoup(html, "lxml")
icerik = soup.select(CSS_seçimi)
```

Algoritma 1. Beautiful Soup kullanımı

Beautiful Soup Kütüphanesi kullanmak için öncelikle Python kütüphaneleri arasına kurulmalıdır. Daha sonra bs4 olarak koda eklenir. Algoritma 1'de öncelikle Beautiful Soup Kütüphanesi sınıfından bir DOM nesnesi oluşturulmuştur. Burada, yapıcı metodu bir web sayfasından elde edilmiş html metnini ve ayrıştırma kütüphanesi parametre olarak alır. Algoritma 1'de ayrıştırma kütüphanesi olarak lxml seçilmiştir. Bu kısım html-parser ve html5lib olarak değiştirilebilir. Bu nesnenin "select" metodu ile DOM ağacı üzerinde tüm HTML elementlerine ulaşılabilir. Bu metot, CSS seçimi destekler. Örneğin, etiketi için "div#reviews" yazılabiliriz. Bu durumda metot tek değer döndürür. Ancak, etiketi için "div.review" kullanılabiliriz. Bu durumda metot Şekil 1'deki örnek için dört adet sonuç ve başka bir deyişle liste sonucu döndürür. Dikkat edildiyse "id" için "#" ve "class" için "." kullanılmaktadır.

Düzenli ifadelerle çıkarım için Python ile birlikte gelen "re" kütüphanesi kullanılır. Örneğin, etiketi için "<div class=\"review\">(.*?) " düzenli ifadesi kullanılabilir. Ancak, üretilen benzer bir ifade sorun çıkarabilir. Çünkü bulduğu ilk kapanış "div" etiketinde çıkarım işlemini bitirir. Başka bir deyişle açılış ve bitiş etiketlerinin "div" sayısına bakmaz. Bu durumda, "<div class=\"reviews\">(.*?)</body>" gibi bir ifade ile çıkarım işlemi yapılabilir. Bu tür ifadeleri yazmak zor ve zahmetli bir süreçtir. Bu sebepten dolayı DOM ayrıştırıcıları tercih edilmektir. Algoritma 2, bir düzenli ifade kaba kodudur.

```
import re
icerik = re.findall(düzenli_ifade, html, re.DOTALL)
```

Algoritma 2. Beautiful Soup kullanımı

"findall" metodu sonuçları liste olarak döndürür. Düzenli ifade, html metin verisi ve düzenli ifade parametresini alır.

2.3. Web çıkarım teknikleri

Bu çalışma da, "div.review" gibi tekrarlayan içeriklerin elde edilmesi incelenmiştir. Bir çıkarım işlemi çok farklı şekillerde yapılabilir. Bu çalışma üç farklı çıkarım tekniğini kapsamaktadır. Birinci teknik, tüm web sayfası dikkate alınır. Tablo 1, birinci tekniğe ait çıkarım kurallarını içerir.

Tablo 1. Birinci çıkarım tekniği

İçerik	CSS seçimi ve düzenli ifade (Regex)
Yorum	CSS: div.review Regex: <div class=\"review\">(.*?)
Diğer sayfa bağlantısı	CSS: a Regex: <a.*?href=\"(.*?)\".*?>

Diğer sayfa bağlantısı, bundan sonraki web sayfalarına ulaşmak için kullanılabilir. CSS olarak çok basit bir ifade kullanılırken düzenli ifade karşılığı biraz daha karmaşıktır. Bu çıkarım yönteminde tüm HTML dokümanı dikkate alınmıştır. Eğer web sayfası içinde istenilmeyen "div.review" elementleri var ise bu yöntem sıkıntı oluşturabilir. Başka bir deyişle sadece "reviews" bölümündeki "review" bölümlerine ulaşmak için kurallar Tablo 2'deki gibi düzenlenebilir.

Tablo 2. İkinci çıkarım tekniği

İçerik	CSS seçimi ve düzenli ifade (Regex)
Yorum	**Ebeveyn:** CSS: div.reviews Regex: <div class=\"reviews\">(.*?)</body> **Çocuklar:** CSS: div.review Regex: <div class=\"review\">(.*?)
Diğer sayfa bağlantısı	**Ebeveyn:** CSS: div.pages Regex: <div id=\" pages\">(.*?) **Çocuklar:** CSS: a Regex: <a.*?href=\"(.*?)\".*?>

Tablo 2'de ebeveyn üst element iken çocuklar alt elementleri kapsamaktadır. Bu durumda çıkarım sayısı artmaktadır ancak çıkarım işleminin yoğunluğun olduğu bölüm azalmaktadır. Başka bir deyişle, tekrarlayan elementleri elde etmek için sadece web sayfasının bir kısmı kullanılmaktadır. Bu çıkarım tekniğinde de hata oluşabilir. Örneğin, Şekil 2'deki gibi "div.review" elementi içinde yazar, tarih ve metin bilgisi olsun.

[1] https://scrapy.org/ açık kaynak kodlu istediğiniz veriye kolay ve hızlı bir şekilde ulaşmanızı sağlayan bir kütüphanedir.
[2] Sırasıyla https://html-agility-pack.net/, https://github.com/AngleSharp/AngleSharp, https://docs.microsoft.com/tr-tr/dotnet/api/system.windows.forms.htmldocument?view=netframework-4.7.2
[3]Sırasıyla https://jsoup.org/, https://jodd.org/lagarto/lagarto-parser.html, https://html-cleaner.com/
[4] Sırasıyla https://html5lib.readthedocs.io/en/latest/, https://docs.python.org/3/library/html.parser.html, https://lxml.de/html5parser.html
[5]Sırasıyla https://github.com/cheeriojs/cheerio, https://github.com/jsdom/jsdom, https://github.com/fb55/htmlparser2

```
<div class="review">
<p class="author">…</p>
<span class="date">…</span>
<p class="text">…</p>
</div>
<div class="review">
<p class="author">…</p>
<p class="text">…</p>
</div>
```

Şekil 2. “div.review” genişletilmiş hali

Şekil 2’de tarih bilgisi birinci “review” elementi içinde var iken ikincisinde kullanıcı tarih bilgisini girilmediği için görülmemektedir. Birinci ve ikinci teknikte hangi “review” bölümüne ait tarih bilgisine ait olduğu tespit etmek zordur. Bu durumda ebeveyn olarak “div.review” elementi seçilebilir.

Tablo 3. Üçüncü çıkarım tekniği

İçerik	CSS seçimi ve düzenli ifade (Regex)
Yorum	**Ebeveyn:** CSS: div.review Regex: <div class=\"review\">(.*?) **Çocuklar:** CSS: p.author, span.date, p.text Regex: <p class=\"author\">(.*?)</p>, <span class=\"date\">(.*?), <p class=\"author\">(.*?)</p>
Diğer sayfa bağlantısı	**Ebeveyn:** CSS: div.pages Regex: <div id=\" pages\">(.*?) **Çocuklar:** CSS: a#last Regex: <a.*?id=\"last\"href=\"(.*?)\".*?>

Üçüncü çıkarım tekniği sayesinde “review” elementleri elde edildikten sonra o “review” elementine ait alt çıkarım işlemleri yapılacaktır. Bu sayede daha doğru sonuçlara ulaşılabilir. Diğer taraftan bu işlemde çıkarım işlemi sayısı çok daha fazla artmıştır. Tablo 2’de diğer sayfa bağlantısının çocuk bölümünde ufak bir değişiklik yapılmıştır. “a#last” ve “.*?id=\"last\".*?” sayesinde sadece bir bağlantı bilgisi döndürülür. Başka bir deyişle, bir sonraki sayfa bilgisi döndürülür. Bir arama örümceği (crawler)[15], [16], link bilgileri toplayıp bir listeye atar ve her web sayfası elde ettiğinde tüm bağlantıları alır ve listesi ile karşılaştırır. Bu teknik sayesinde sadece bir sonraki sayfa alınmış olur ve karşılaştırma sayısı sadece bire indirgenmiş olur. Başka bir deyişle, bağlantı kontrol işlemi hızlandırılmıştır. Bu çıkarım teknikleri 15 farklı site için oluşturulmuştur.

3. Veri Seti ve Hazırlanması

On beş farklı web sitesi için 4-5 çıkarım kuralı belirlenmiştir. Bu web siteleri film, alış-veriş ve haber sitelerinin tekrarlı veri içeren bölümleri kullanılmıştır. Özellikle yorum etiketleri üzerinde durulmuştur. Kullanılan web siteleri Tablo 4’te verilmiştir.

Web sitelerinden çıkarım işlemi için kuralları içeren bir JSON dosya hazırlanmış ve her site için ikinci bölümde anlatılan üç farklı çıkarım tekniği hem DOM çıkarımı için hem de düzenli ifade çıkarımı için hazırlanmıştır. Örnek bir JSON dosyası Şekil 3’teki gibidir.

Tablo 4. Web siteleri hakkında genel bilgi

Web sitesi	Kategori	Ort. (KB)
allocine.fr	Film	339.68
amazon.com	Alış-Veriş	311.24
apple.com	Alış-Veriş	179.10
beyazperde.com	Film	337.17
hepsiburada.com	Alış-Veriş	196.09
imbd.com	Film	236.61
mediamarkt.com.tr	Alış-Veriş	66.94
moviepilot.de	Film	506.71
rottentomatoes.com	Film	77.15
senscritique.com	Film	58.55
shiftdelete.net	Haber	79.18
steamcommunity.com	Haber	26.28
teknosa.com	Alış-Veriş	182.71
quotes.toscrape.com	Test sitesi	10.56
vatanbilgisayar.com	Alış-Veriş	11.50
Ortalama		183.88

Şekil 3’teki dosyada, “seeds” dolaşılacak olan başlangıç sayfalarını belirler. “parent_rule” ebeveyn kuralını belirlerken “rules” ebeveyn kuralından sonraki çıkarım işlemini tanımlar. “parent_rule” içindeki “tagname”in * içermesi tüm web sayfasına bakılacağı anlamına gelir. “parent_nextPages” bağlantıların çıkarımlarının yapılacağı bölümü belirtirken “nextPages” bağlantı kuralının çıkarımı tanımlar. “parent_nextPages” “*” karakterinin kullanılması da tüm web sayfasının dikkate alınacağını gösterir. Bir web sitesi için bu dosyadan 3 çıkarım teknikleri ve DOM / Düzenli ifade için toplam 6 farklı JSON dosyası hazırlanmıştır. Toplamda 90 farklı JSON dosyası hazırlanmıştır.

```
{
   "project_name" : "toscrape",
   "web_site" : "http://quotes.toscrape.com",
   "seeds":[ "http://quotes.toscrape.com/"],
   "parent_rule": { "tag_name": "*" },
   "rules" :[{
         "name": "description",
         "tag_name": "span.text"},
         {  "name": "author",
         "tag_name": "small.author"}],
   "parent_nextPages": {"tag_name": "*"},
   "nextPages" : ["a"]
}
```

Şekil 3. Örnek bir JSON kural dosyası

Tüm deneyler, i7-8550u 1.8 GHz, 16 GB RAM ve Windows 10 işletim sistemi üzerinde yapılmıştır. Zaman ölçümleri için Python içindeki time kütüphanesi ve time.time() metodu kullanılmıştır. Deneyler ortalama sonuçları milisaniye cinsinden içermektedir.

4. Deneyler

Deneyler üç bölümden oluşmaktadır. Birinci deneyde DOM yaratma süreleri incelenmiştir. İkinci deney çıkarım teknikleri ve yöntemlerinin incelenmesine ayrılmıştır. Üçüncü bölüm ise bağlantı bilgisinin çıkarılma sürelerini kapsar.

4.1. DOM yaratma işlemi

DOM yaratmak için html5lib, html-parser ve lxml kütüphaneleri test edilmiştir. Tablo 5 DOM yaratma ortalama sonuçları verir.

Tablo 5. Web sitelerinin DOM yaratma süreleri

	Milisaniye		
Web sitesi	**html5lib**	**html-parser**	**lxml**
allocine.fr	453.83	162.44	120.65
amazon.com	704.33	350.93	114.83
apple.com	464.55	141.48	102.03
beyazperde.com	493.83	201.54	147.64
hepsiburada.com	497.90	157.28	116.59
imbd.com	540.60	171.49	127.65
mediamarkt.com.tr	290.61	97.22	69.69
moviepilot.de	947.18	220.25	144.16
rottentomatoes.com	301.54	86.48	66.11
senscritique.com	113.45	38.45	27.40
shiftdelete.net	128.59	41.01	28.53
steamcommunity.com	103.47	35.62	27.27
teknosa.com	472.35	146.51	112.89
quotes.toscrape.com	45.65	15.86	12.37
vatanbilgisayar.com	38.26	13.88	9.83
Ortalama	**391.42**	**109.30**	**85.73**

Web çıkarımı işleminde sıklıkla kullanılan lxml, diğer ayrıştırma kütüphaneleri göre çok daha iyi sonuçlar vermiştir. Bir DOM ağacını yaratma süresini dosya boyutu ve etiket sayısının etkilediği görülmektedir. vatanbilgisayar.com ve quotes.toscrape.com web sitelerinin DOM ağacı dosya boyutuna bağlı olarak daha kısa sürede oluşturulur. En büyük dosya boyutuna sahip moviepilot.de sitesi yaklaşık 1 saniye sürede oluşturulmaktadır.

4.2. DOM ve düzenli ifadeler üzerinden veri çıkarımı

Her web sitesi için üç farklı çıkarım tekniği hazırladığımızı ikinci bölümde belirtmiştik. Tablo 5, hazırlanan DOM ağacı üzerinde üç farklı ayrıştırıcıyı ve düzenli ifadeleri kullanarak tekrarlayan içeriği elde etmede elde edilen zaman sonuçları içermektedir.

Birinci çıkarım tekniğinde tüm web sayfasını dikkate alırken diğer çıkarım tekniklerinde daha spesifik alanlar belirlenip çıkarım işlem sayısı artmaktadır. Her ne kadar ikinci çıkarım tekniği daha fazla çıkarım işlemi içerse de zaman sonuçlarının çok yakın olduğu ve hatta iyileştiği görülmektedir. Üçüncü teknik doğruluğu arttıracak daha fazla çıkarım işlemi barındırmaktadır. Bunun zaman sonuçlarına da yansıdığı görülmektedir.

Tablo 6. Veriye erişim ve çıkarım süreleri

	Çıkarım teknikleri (milisaniye)		
	1.	**2.**	**3.**
html5lib	14.03	14.18	14.99
html-parser	14.23	13.98	15.96
lxml	15.53	15.89	17.30
Düzenli ifade	4.71	2.32	14.50
Toplam çıkarım sayısı	2017	9704	27678

Zaman sonuçları açısından DOM tabanlı yöntemlerin birbirine yakın sonuçlar verdiği görülmektedir. Düzenli ifadelerin özellikle ilk iki çıkarım tekniğinde çok daha iyi sonuçlar vermiştir. Üçüncü teknik ile birlikte çıkarım sayısının çok artması düzenli ifade sonuçlarını da etkilemiştir.

4.3. DOM ve düzenli ifadeler ile bağlantı çıkarımı

Özellikle bir web arama örümceğinde bağlantı bilgilerinin çıkarılması önemli bir işlemdir. Başka bir deyişle “a” etiketinin “href” özelliğini çıkarılması işlemidir. Bağlantı Tablo 7, bağlantı bilgileri toplamada zaman sonuçlarını sunmaktadır.

Tablo 7. Bağlantı çıkarım süreleri

	Çıkarım teknikleri (milisaniye)		
	1.	**2.**	**3.**
html5lib	4.62	4.48	4.38
html-parser	4.71	4.54	4.74
lxml	5.04	4.95	5.12
Düzenli ifade	2.29	0.31	0.28

İlk çıkarım tekniğinde web sayfasındaki tüm bağlantılar çıkarılır. İkinci ve üçüncü yöntemlerde sadece belirli bir

bölümdeki bağlantıların çıkarımı söz konusudur. DOM tabanlı algoritmaların bu çıkarım tekniklerinden çok fazla etkilenmedikleri görülmektedir. Düzenli ifadelerde ise yaklaşık 0.30 ms çıkarım süresi elde edilmiştir. Diğer yöntemlere göre çok daha iyi ve hızlı bir çıkarım sunmaktadır.

5. Sonuç ve Tartışma

Web sayfaları geliştiricilere çok fazla miktarda ve kullanışlı veriler sağlar. Günümüzde, bu verilerin işlenmesi ve gerekli verinin elde edilmesi önemli bir işlem haline gelmiştir. Bu çalışmada, tekrarlayan verilere etkin erişim yöntemleri ve kuralları araştırılmıştır.

Bir web sayfasındaki veriye DOM üzerinden CSS seçim yöntemiyle veya düzenli ifadeler üzerinden ulaşmak mümkündür. Ancak, düzenli ifadelerin hazırlanması CSS seçimi yapmaya göre çok daha zor ve zahmetli bir süreçtir. Bu sebepten birçok hazır kütüphane daha sade ve basit olan CSS veya XPATH[6] gibi çıkarım yöntemlerini kullanmaktadır. Diğer taraftan DOM ağacı üretilmeden de düzenli ifadelerle istenen veriye çok daha kısa sürelerde ulaşmak mümkündür.

Bir çıkarım işleminde diğer bir konu ise çıkarım kurallarının hazırlanmasıdır. Bu çalışmada, üç farklı çıkarım kuralı üretilmiş ve bu kurallar üzerinden veriye ulaşım amaçlanmıştır. Yapılan bu çalışma, çıkarım kurallarının iyi hazırlanmasının da performans üzerinde çok etkili olduğu göstermektedir.

Bu çalışmada, HTML tabanlı web sayfalarından çıkarım işlemi üzerine odaklanılmıştır. Ancak, günümüzde web sayfalarında AJAX[7] tabanlı çözümler kullanması yaygınlaşmaktadır. Bu çözümlerde sayfa öncelikle yüklenir ve veriler sayfa yüklendikten sonra etiketlerin arasına gönderilir. Bu durumda DOM üretilmesinin yanı sıra Javascript kodlarının çalıştırılması gerekmektedir. Bu sorunun Python programlama dilinde en çok bilinen çözümlerinden biri Selenium[8] kütüphanesidir. Sonraki çalışmalarda bu kütüphaneyi kullanmayı ve iyileştirmeyi amaçlıyoruz.

Düşündüğümüz gelecek çalışmalarından bir diğeri düzenli ifadelerin otomatik olarak bir CSS seçim işlemi üzerinden hazırlanmasıdır. Böyle bir kütüphane geliştirdiğimizde web çıkarım işlemiyle ilgilenen geliştiriciler basit CSS kuralları yazarak düzenli ifadeler üzerinden verilerine ulaşma imkânına kavuşacaktır.

Referanslar

[1] A. F. R. Rahman, H. Alam, and R. Hartono, “Content extraction from html documents,” in *1st Int. Workshop on Web Document Analysis (WDA2001)*, 2001, pp. 1–4.

[2] E. Ferrara, P. De Meo, G. Fiumara, and R. Baumgartner, “Web data extraction, applications and techniques: A survey,” *Knowledge-Based Syst.*, vol. 70, pp. 301–323, Nov. 2014.

[3] S. Flesca, G. Manco, E. Masciari, E. Rende, and A. Tagarelli, “Web Wrapper Induction: A Brief Survey,” *AI Commun.*, vol. 17, no. 2, pp. 57–61, 2004.

[4] N. Kushmerick, “Wrapper Induction for Information Extraction,” PhD Thesis, University of Washington, 1997.

[5] L. Liu, C. Pu, and W. Han, “XWRAP : An XML - enabled Wrapper Construction System for Web Information Sources,” *Proc. 16th Int. Conf. Data Eng.*, 2000.

[6] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer, 2011.

[7] B. Fazzinga, S. Flesca, and A. Tagarelli, “Schema-based Web wrapping,” *Knowl. Inf. Syst.*, vol. 26, no. 1, pp. 127–173, 2011.

[8] E. Uzun, T. Yerlikaya, and M. Kurt, “A lightweight parser for extracting useful contents from web pages,” in *2nd International Symposium on Computing in Science & Engineering-ISCSE 2011, Kusadasi, Aydin, Turkey*, 2011, pp. 67–73.

[9] L. M. Álvarez-Sabucedo, L. E. Anido-Rifón, and J. M. Santos-Gago, “Reusing web contents: a DOM approach,” *Softw. Pract. Exp.*, vol. 39, no. 3, pp. 299–314, Mar. 2009.

[10] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm, “DOM-based content extraction of HTML documents,” in *Proceedings of the twelfth international conference on World Wide Web - WWW ’03*, 2003, p. 207.

[11] L. Fu, Y. Meng, Y. Xia, and H. Yu, “Web content extraction based on webpage layout analysis,” in *Proceedings - 2nd International Conference on Information Technology and Computer Science, ITCS 2010*, 2010, pp. 40–43.

[12] E. Uçar, E. Uzun, and P. Tüfekci, “A novel algorithm for extracting the user reviews from web pages,” *J. Inf. Sci.*, vol. 43, no. 5, pp. 696–712, Sep. 2016.

[13] E. Uzun, H. V. Agun, and T. Yerlikaya, “A hybrid approach for extracting informative content from web pages,” *Inf. Process. Manag.*, vol. 49, no. 4, pp. 928–944, 2013.

[14] E. Uzun, T. Yerlikaya, and O. Kırat, “Comparison of Python Libraries used for Web Data Extraction,” in *7th International Scientific Conference “TechSys 2018” – Engineering, Technologies and Systems, Technical University of Sofia, Plovdiv Branch May 17-19*, 2018, pp. 108–113.

[15] M. Kobayashi and K. Takeda, “Information retrieval on the web,” *ACM Comput. Surv.*, vol. 32, no. 2, pp. 144–173, Jun. 2000.

[16] M. Kumar, R. Bhatia, and D. Rattan, “A survey of Web crawlers for information retrieval,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 7, no. 6, p. e1218, 2017.

[6] XPath, HTML ve XML etiketleri içerisinde yer alan verilere ulaşmak için kullanılır. W3C tarafından geliştirilmiş ve geliştirilmeye devam eden bir standarttır. https://www.w3.org/TR/1999/REC-xpath-19991116/

[7] AJAX (Asynchronous JavaScript and XML) amacı tüm sayfayı tekrar yükletmeden, sadece gerekli olan veriyi dinamik olarak istemciye (tarayıcıya) getirmek veya sunucuya veri göndermektir.

[8] Bir web tarayıcı uygulaması gibi kodlama yapmanıza olanak sağlayan ve birçok programlama diline desteği olan bir kütüphanedir. https://www.seleniumhq.org/