Avrupa Bilim ve Teknoloji Dergisi
Sayı 17, S. 1313-1324, Aralık 2019

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 17, pp. 1313-1324, December 2019

Research Article

Dijital Dokümanlar Üzerinde Otomatik Biçim Tanıma ve Farklı İçeriklere Uyarlama: Özgeçmişler Üzerinde Durum Çalışması

Alper Kantarci[1*], Süleyman Eken[2], Ahmet Sayar[3]

[1] Kocaeli Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kocaeli, Türkiye (ORCID: 0000-0003-2456-8648)
[2] Kocaeli Üniversitesi, Teknoloji Fakültesi, Bilişim Sistemleri Mühendisliği Bölümü, Kocaeli, Türkiye (ORCID: 0000-0001-9488-908X)
[3] Kocaeli Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kocaeli, Türkiye (ORCID: 0000-0002-6335-459X)

(İlk Geliş Tarihi 1 Aralık 2019 ve Kabul Tarihi 31 Aralık 2019)

(**DOI:** 10.31590/ejosat.661562)

ATIF/REFERENCE: Kantarci, A., Eken, S. & Sayar, A. (2019). Dijital Dokümanlar Üzerinde Otomatik Biçim Tanıma ve Farklı İçeriklere Uyarlama: Özgeçmişler Üzerinde Durum Çalışması. *Avrupa Bilim ve Teknoloji Dergisi*, (17), 1313-1324.

Öz

Çoğu bilgisayar işleminin merkezinde yer alan toplu kategorizasyona ilişkin olarak bilgi geri çağırmayı etkileyen iki tür ilgili veri vardır: yapısal veriler ve yapılandırılmamış veriler. Yapılandırılmış veriler, ilişkisel bir veritabanına dahil edilmesi gibi yüksek derecede organizasyona sahip bilgileri ifade eder. Bununla birlikte, yapılandırılmamış veriler kendi iç yapısına sahip olabilir, ancak bir e-tabloya veya veritabanına tam olarak karşılık gelmezler. Özgeçmişler bu tür verilerdir. Genelde PDF (Portable Document Format, Taşınabilir Belge Formatı) formatında sunulan özgeçmişler, PDF etiketleme özelliği kullanılarak yapısal hale getirilebilir; fakat çoğu PDF verisi etiketlenmemiş ve yapısal olmayan haldedir. Teknik olmayan iş dünyası kullanıcıları ve veri analistlerinin bu tür kapalı kutularla başa çıkmaları çok zordur.

Bu çalışma kapsamında, kişilerin özgeçmiş hazırlayarak zamanlarını kaybetmemek ve farklı kabul görmüş formatlarda kişilerin kendi bilgilerine göre kendilerine has özgeçmişler hazırlayabilmesine imkân verecek web tabanlı zeki özgeçmiş tasarımcısı geliştirildi. PDF dokümanlarının içerik yapısı, metin verisi ve bu verinin yazı tipi ve dokümandaki lokasyon bilgileri çıkartıldı ve elde edilen bu bilgiler okuma sırasına göre belirli yapılara dönüştürülerek önceden tanımlanmış olan XML (Extensible Markup Language, Genişletilebilir İşaretleme Dili) tabanlı özgeçmiş tasarımı oluşturuldu. Elde edilen bu tasarımlar kullanılarak kişisel PDF dökümanları oluşturuldu. PDF analizi ve PDF oluşturma işlemi, Java iText-pdf kütüphanesi yardımıyla gerçekleştirildi. Tasarım verileri arayüz aracılığıyla kullanıcıya sunularak kullanıcı istediği tasarımı kendi dökümanını oluştururken seç ve uygula yaklaşımıyla aktarabilmektedir.

PDF dokümanından elde edilen şablonun XML formatında kaydedilmesi ve farklı içeriklere uyarlama aşamasında, kaydedilmiş hazır XML formatındaki şablonların kullanılması öngörüldü. XML formatındaki şablonların otomatik oluşturulabilmesi ve sonradan doğruluğunun test edilebilmesi için XSD (XML Schema Definition, XML Şeması Tanımı) tanımlandı. Geliştirilen uygulama ile özgeçmişlerin otomatik biçimlerinin tanınması ve farklı içeriklerin adaptasyonu sağlandı.

Anahtar Kelimeler: Doküman analiz ve tanıma, PDF, Bilgi çıkarımı, XML, XSD

[*] Sorumlu Yazar: Kocaeli Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kocaeli, Türkiye (ORCID: 0000-0000-0000-0000), xxxx@xxx.xx.xx

Automatic Structure Recognition on Digital Documents and Adapting to Different Contents: Case Study on Resumes

Abstract

With respect to the mass categorization that is central to most computer operations, there are two types of relevant data which affect speed of assimilation as well as information recall: structured data and unstructured data. Structured data refers to information with a high degree of organization, such that inclusion in a relational database. However, unstructured data may have its own internal structure, but does not conform neatly into a spreadsheet or database. CVs (Curriculum vitae) are this kind of data. Typically, CVs presented in PDF format can be structured using the PDF tagging feature, however most PDF data is untagged and unstructured. It is very difficult for non-technical business users and data analysts to deal with such closed boxes.

Within the scope of this study, a web based smart resume designer was developed which will allow people gain time while creating their own resumes according to their own information in different accepted formats. The content structure of the PDF documents, the text data and the font and location information of this data were extracted and the information obtained was converted into certain structures in the order of reading and a predefined XML based resume template was created. Personal PDF documents are created using this template. PDF analysis and PDF creation was done directly by accessing the content stream of the PDF document with the help of the iText-pdf library, which is the Java library. Presentation templates is served to end-user on a desktop applicaiton with a GUI and users can select any metadata to create own document with select-and-apply approach.

It is predicted that the template obtained from the PDF document will be saved in XML format and the templates in the ready-made XML format will be used for adaptation to different contents. The XML schema (XSD-xml schema definition) is defined for the automatic creation of templates in XML format and subsequent testing of their accuracy. With the application developed, automatic forms of resumes were recognized and different contents were adapted.

Keywords: Document analysis and recognition, PDF, Information extraction, XML, XSD

1. Giriş

İş bulmak için atılacak ilk adım profesyonelce yazılmış bir özgeçmiştir. Özgeçmişinin içeriğini, isim, soy isim (kişisel bilgiler), iletişim bilgileri, eğitim durumu, iş tecrübesi deneyimler, referanslar, özel zevkler gibi konuların özetleri oluşturur. Özgeçmiş doğru, anlaşılır, açık ve kısa olmalıdır ki dikkat çekilebilsin. Etkili bir özgeçmiş hazırlamak için uzmanlar bazı noktalara dikkat edilmesi gerektiğini vurgulamaktalar (Tunçer, 2013). Bunlardan bazıları:

- Özgeçmiş en fazla 2 sayfa olmalıdır.
- Uzun paragraflardan kaçınılmalıdır.
- Genellikle "Times New Roman" veya "Arial" gibi kolay okunabilen karakterler kullanılmalı, 11 ya da 12 punto ile yazılmalıdır.
- Kelime ve cümlelerin altı çizilmemelidir.
- Ters kronolojik sıra takip edilmelidir.
- Gereksiz bilgiler eklemekten kaçınılmalıdır.
- Takım elbiseli çekilmiş fotoğraflar eklenmelidir.
- Silik, bulanık, arka planı karışık fotoğraflar eklenmemelidir.

Genel olarak bakıldığında özgeçmiş hazırlamak ince işçilik isteyen bir süreçtir. Bu yüzden hemen özgeçmiş hazırlayıp başvurularda bulunmak yerine zamana yayıp genel geçer kurallara uygun olarak ilgi çekici bir özgeçmiş hazırlamak yapılan başvurulara geri dönüşler almayı epey kolaylaştırmaktadır. Yukarıda listelenen veya uzmanlar tarafından söylenen kriterlerde özgeçmişler hazırlamak özellikle Office programlarını kullanmayı tam bilmeyen kimseler tarafından hem çok sıkıcı hem de çok zaman alan bir süreç olmaktadır. Biz de bu çalışma kapsamında, kişilerin özgeçmiş hazırlayarak zamanlarını kaybetmemek ve farklı kabul görmüş formatlarda kişilerin kendi bilgilerine göre kendilerine has özgeçmişler hazırlayabilmesine imkân verecek web tabanlı zeki özgeçmiş tasarımcısı geliştirdik. Yaptığımız araştırmalara göre literatürde PDF özgeçmiş dokümanlarını kullanarak kullanıcılara özgü yeni özgeçmişlerin hazırlanmasını sağlayan bir çalışmaya rastlamadık.

İşletim sisteminden bağımsız olması nedeniyle, PDF formatı elektronik belgelerin paylaşılması, arşivlenmesi, alınması ve yazdırılması için yaygın olarak kullanılır (Mohemad, Hamdan, Othman, & Mohamad, 2011; Hassan, 2009). Yaygın olarak kullanıldığı alanlar ve faydaları yanısıra, içerik ve yapı analizi açısından dezavantajlara da sahiptir (Mohemad, Hamdan, Othman, & Mohamad, 2011). Sonuç olarak PDF formatlı dokümanlarda sunulan bilgiler, otomatik olarak okunması ve yeniden kullanılması gibi işlem gerektiren, özgeçmiş gibi karar alma uygulamaları için elverişli değildir (Mohemad, Hamdan, Othman, & Mohamad, 2011). Günümüzde yaygın olarak kullanılmasına rağmen PDF belge içeriği iç yapısı hakkında da bilgi eksikliği var (Liu, Bai, Mitra, & Giles, 2009; Jiang & Yang, 2009). PDF belge yapısını otomatik olarak analiz etme ve tanıma, ilgili bilgilerin çıkarılması ve bu bilgilerin hem yapı hem de anlamsal

biçimlerde ayrıştırılması, paylaşım, bilgi arama, karar verme ve diğer çeşitli amaçlar için büyük önem taşımaktadır (Mohemad, Hamdan, Othman, & Mohamad, 2011). Çoğu PDF belgesi etiketlenmemişti ve temel üst düzey doküman mantıksal yapısal bilgisine sahip olmadığı için, belgelerin yeniden kullanılması veya değiştirilmesini zorlaştırmaktadır (Chao & Fan, 2004). PDF formatlı özgeçmiş belgelerinin yapı analizi en basit yapıya sahip olanları için bile zorlu bir işlemdir. PDF belgelerini, resim veya HTML'e dönüştürüp sonra OCR (Optical Character Recognition, Optik Karakter Tanıma) vs. gibi tekniklerle işlemek yerine, içeriği doğrudan PDF`ten çıkarmak ve analiz etmek daha kolay ve kesindir (Liu, Bai, Mitra, & Giles, 2009). Bu çalışmada PDF formatındaki CV dokümanlarının tasarım şablonlarının otomatik olarak tanınması, bu şablonların daha sonra kullanılabilirliğini sağlamak amaçlı XML formatına dönüştürülüp kaydedilmesi ve farklı içeriklere uyarlanması gerçekleştirildi. XML formatındaki şablonların otomatik oluşturulabilmesi ve sonradan doğruluğunun test edilebilmesi için önceden XML şeması tasarlandı.

Literatürde direk önerilen konu ile ilgili olmasa da PDF dosyalarının otomatik doküman yapısı çıkarımı, matematiksel dokümanların analizi, nesne seviyesinde analiz, mantıksal yapı keşfi, taranmış PDF dokümanlarının düzen analizi, taranmış sağlık hizmeti belgelerinden bilgi çıkarma ile ilgili farklı çalışmalar vardır. Araştırmacılar tarafından PDF içindeki nesneler (metinler, görseller vs.) otomatik olarak çıkartılmış elde edilen veriler satır, paragraf ve daha üst seviye mantıksal yapılara gruplandırılmıştır. Bu şekilde yapısal olmayan verilerden mantıksal temsillerin saptanması genel olarak "doküman anlama/yorumlama" şeklinde literatürde yer bulmuştur.

Doküman anlama genellikle taranmış dokümanlar/görüntüler üzerinde yapılmaktadır (Aiello, Monz, Todoran, & Worring, 2002; Altamura, Esposito, & Malerba, 2000; Eken, Atay, Sönmez & Sayar, 2018; Eken, Karabaş, Sarı & Sayar, 2018; Eken ve Sayar, 2013). Proje kapsamında, yapılan çalışmalardaki gibi biz de özgeçmiş PDF dokümanlarından isim, soy isim (kişisel bilgiler), iletişim bilgileri, eğitim durumu, iş tecrübesi deneyimler, referanslar, özel zevkler gibi metinsel nesneler ile kişi görüntüsü gibi görsel nesnelerin doküman içindeki konumlarıyla (düzen) tespit edilmesi ve XML formatında ilgili özgeçmişin ifade edilmesi gerçekleştirildi. Mohemad ve arkadaşları (Mohemad, Hamdan, Othman, & Mohamad, 2011) Microsoft Word 2007 ortamında oluşturulmuş PDF belgelerinin otomatik doküman yapısı çıkarımını, paragraf ve tablo (ya da çizelge halinde olan) yapılara odaklanarak, Java tabanlı ortamda, JPEDAL isimli java kütüphanesini kullanarak, sezgisel, kural tabanlı ve önceden tanımlanmış göstergeler şeklinde 3 farklı strateji izleyerek gerçekleştirmişlerdir. Hassan (Hassan, 2009) ise görsel prensiplere göre aşağıdan yukarıya (bottom-up) bir kümeleme algoritması önermiştir. Chao ve arkadaşı (Chao & Fan, 2004) PDF belgesindeki mantıksal bileşenleri tanımlayan teknikler geliştirmiş ve bu mantıksal bileşenlerden anahatlar, stil özellikleri ve içerik çıkartıp sonucu XML formatında ifade etmişlerdir. Liu ve arkadaşları (Liu, Bai, Mitra, & Giles, 2009) PDF dokümanlarından metin çıkarma araçlarının ortak karşılaştığı metin sırası probleminin çözümü için iki algoritma önermiş ve algoritmaların performansını karşılaştırmışlardır. Gabdulkhakova ve arkadaşı (Gabdulkhakova & Tamir, 2012) grafik operatörler tarafından dijital PDF belgelerinde çizilen içeriği analiz etmek için nesne tabanlı bir yöntem sunmuşlardır. Baker ve arkadaşları (Baker, Sexton, Sorge, & Suzuki, 2011) matematiksel metinlerin belge analizi için biri yapısal tanıma için sanal bir bağlantı ağı ile birlikte karakter tanıma (OCR) yaklaşımı ve diğeri düzlem, ifade yapılarını çıkarmak için iki aşamalı ayrıştırıcı ile doğrudan PDF dosyasının sembol bilgisini kullanan iki aşama önermişlerdir.

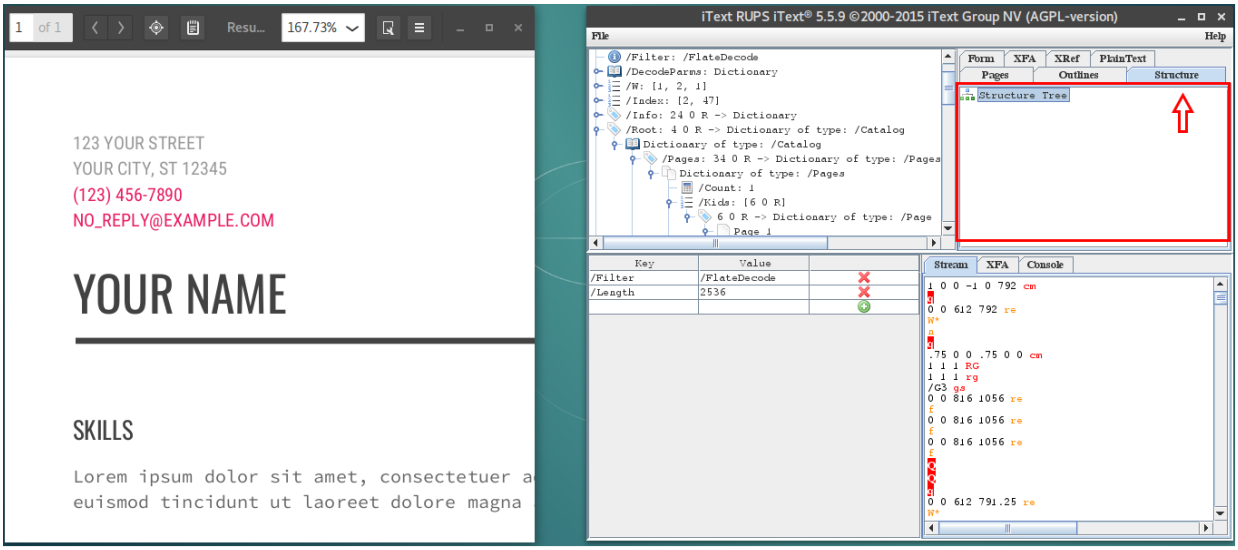
İlgilenilen bir diğer konu da PDF ve XML'in birbirlerine karşılıklı olarak dönüştürülebilmesidir. PDF dokümanlarından XML dokümanlarının elde edilmesindeki amaç indeksleme ve geri getirim yoluyla dokümanlar üzerinde yapılacak bir arama için arama uzayını (search space) daraltmaktır (Eken, Ekinci & Sayar, 2014). Bu tür çalışmalar literatürde "belge özetleme-söylem çıkarımı" olarak geçmektedir. Constantin ve arkadaşları (Constantin, Pettifer, & Voronkov, 2013) bilimsel PDF dokümanları içinde yer alan başlık, yazarlar, özet, yazar dipnot bilgisi, ana metin, alt başlıklar ve metinleri, resim, tablo, referanslar, bibliyografik kısım gibi 18 tane mantıksal elementi çıkarıp XML tag'ları şekline dönüştürebilen PDFX mimarisini önermişlerdir. Daha sonra ground-truth bilgileri var olan veri setleri ile yaptıkları çalışmayı test etmişlerdir. Biz de özgeçmişlerde yer alan yukarıda bahsedilen alanları çıkarıp XML tag'lerine dönüştüren, elde edilen XML şablonları üzerinde edit işlemlerine izin veren daha sonra kullanıcı bilgilerine göre yeniden özgeçmiş PDF dosyalarının oluşturmasını sağlayan bir özgeçmiş tasarımcısı geliştirdik.

2. Materyal ve Metot

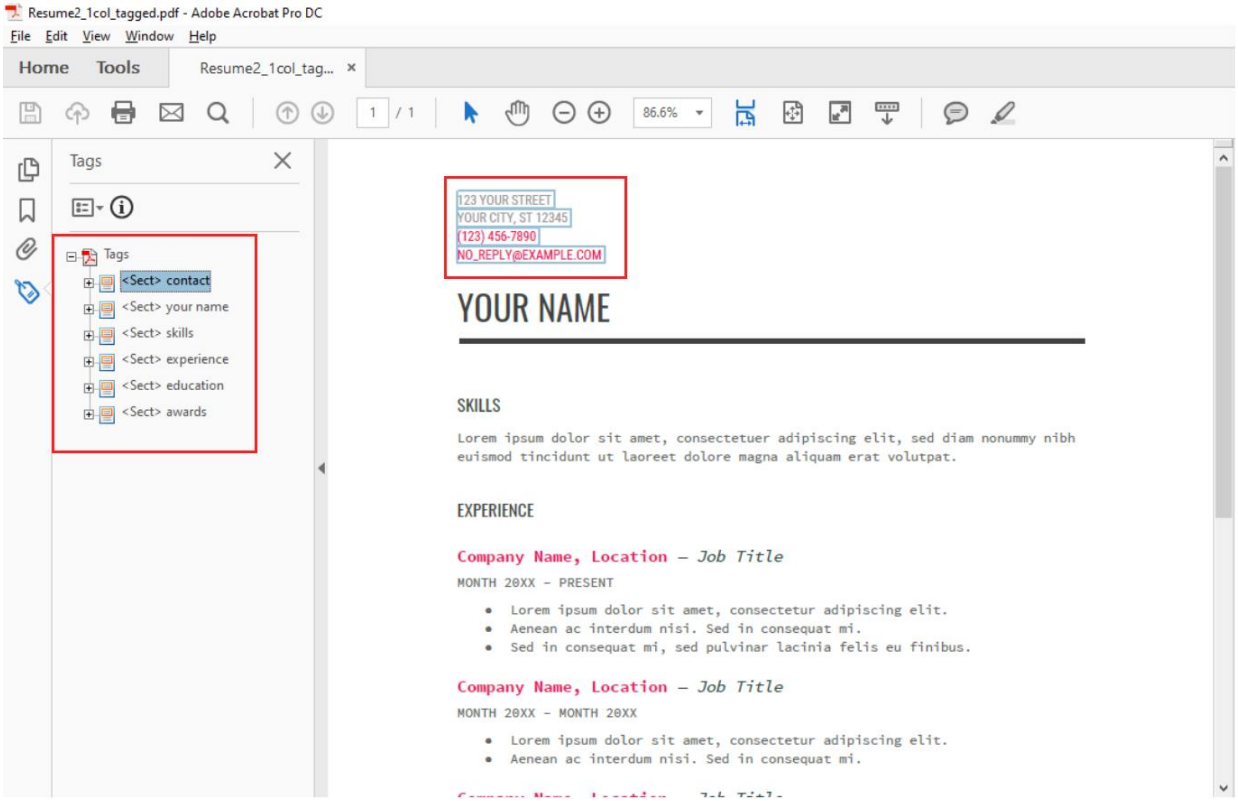
PDF formatındaki CV dosyalarının tasarım şablonlarının otomatik olarak tanınması ve tasarımın farklı içeriklere uyarlanması gerçekleştirildi. Java kütüphanesi olan iText-pdf kullanılarak pdf içeriği tarandı ve şablon oluşturuldu.

Şablonun XML formatında kaydedilmesi ve farklı içeriklere uyarlama aşamasında, kaydedilmiş hazır XML formatındaki şablonların kullanılması öngörüldü. XML formatındaki şablonların otomatik oluşturulabilmesi ve sonradan doğruluğunun test edilebilmesi için XML tanımlandı.

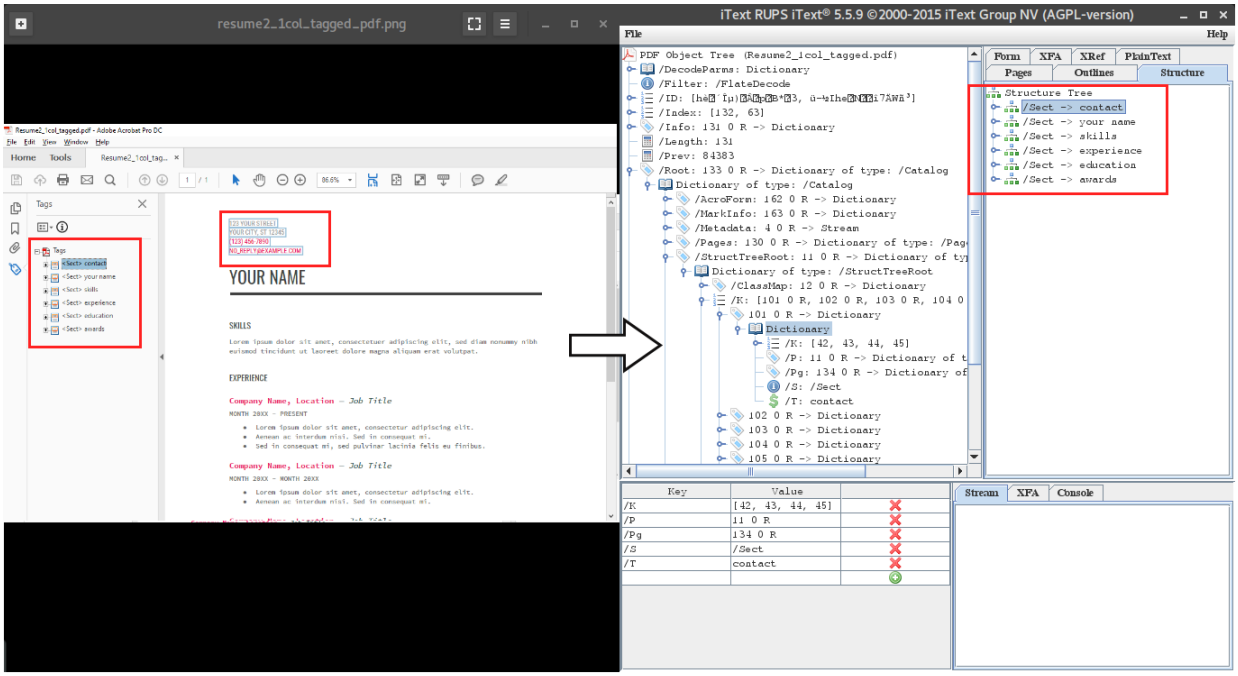
PDF verileri yapısal olmayan verilerdir buna gore bu verilerin tanımlanabilir bir yapısı yoktur. Bu yapıları gereği pdf verilerinin tasarım şablonlarının çıkartılması projede çözulmeye çalışılan problemlerden biridir. PDF verileri etiketleme özelliği kullanarak yapısal hale getirilebilir; fakat genelde çoğu pdf verisi etiketlenmemiş ve yapısal olmayan haldedir. Aşağıda kırmızı işaretli alanda bir PDF inceleme aracı olan iText RUPS uygulaması ile PDF dokümanının yapısı ağaç görünümü ile gösterilmektedir:



Şekil 1. Yapısal olmayan/Taglenmemiş PDF (iText RUPS uygulaması)



Şekil 2. Bir kısmı yapısal hale getirilmiş/Taglenmiş PDF(Adobe Acrobat)



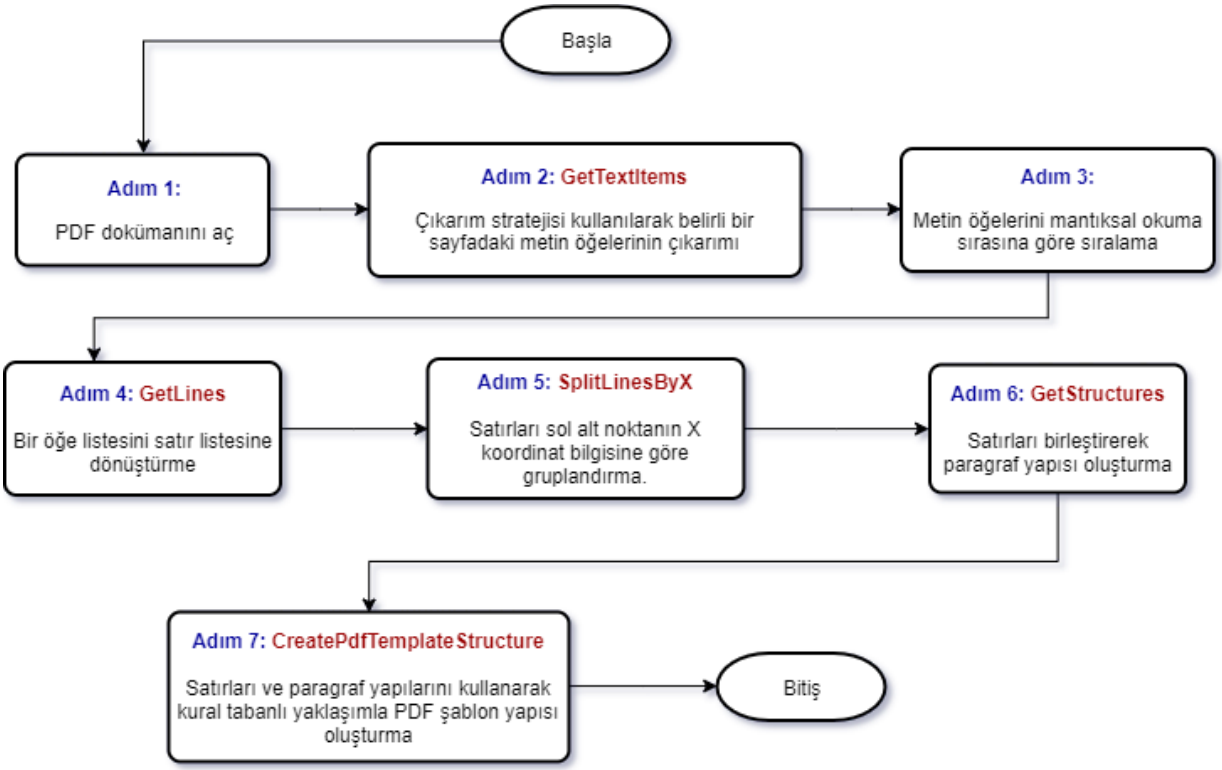
Şekil 3. Bir kısmı yapısal hale getirilmiş/Taglenmiş PDF (iText RUPS uygulaması)

Şekil 1'de görüldüğü gibi yapısal olmayan PDF dokümanlarında bir yapı tanımlanmadığı için iText RUPS uygulamasının Structure sekmesindeki Structure Tree listesi boştur. Bu tür PDF dokümanları bir yapıya sahip olmadığı için, PDF içeriği baştan sona taranarak, doküman içerisindeki ögeler ve ögeler arasındaki ilişkiler analiz edilerek PDF yapısı oluşturulabilir.

Şekil 2 ve Şekil 3'te ise Tagged (taglenmiş, yapısal) PDF yapısı görülmektedir. Bu tür PDF dokümanlarında yapı önceden tanımlandığı için, içerik okuma, değiştirme, tanımlanan yapı yardımıyla çok daha kolay bir şekilde gerçekleştirilebilir.

2.1 PDF Şablon Çıkarımı

Şekil 4'te bu çalışmanın genel algoritma akış diagramı gösterildi. Algoritmadaki 1. Adım normal herhangi bir text dosyasını okuma işlemi gibi genel ve basit bir işlem olan PDF dokümanını açma işlemini kapsamaktadır. Algoritmanın asıl önemli, detaylı ve geniş adımlarını kapsayan diğer adımlar yazının devamında alt başlıklar açılarak ve açıklayıcı şekiller ile desteklenerek anlatıldı. Özet olarak ilk önce iText-pdf kütüphanesi yardımıyla PDF dokümanı açıldı. Açılan PDF dokümanının içerisindeki metin öğeleri (bileşenleri) okundu. Okunan metin öğeleri rastgele bir sırada okunabildiği için ve bu bizim işimize yaramadığı için mantıksal okuma sırasına göre sıralandı. Elde edilen karakter ya da sözcük öğeleri kullanılarak satır yapıları oluşturuldu. Satır yapıları birden fazla sütun yapısına sahip PDF dokümanlarında sütun bazında gruplandı. Elde edilen satır yapıları birleştirilerek paragraf yapıları oluşturuldu. Son olarak elde edilen satır ve paragraf yapıları kullanılarak kural tabanlı yaklaşımla PDF şablon yapısı çıkartıldı (oluşturuldu).

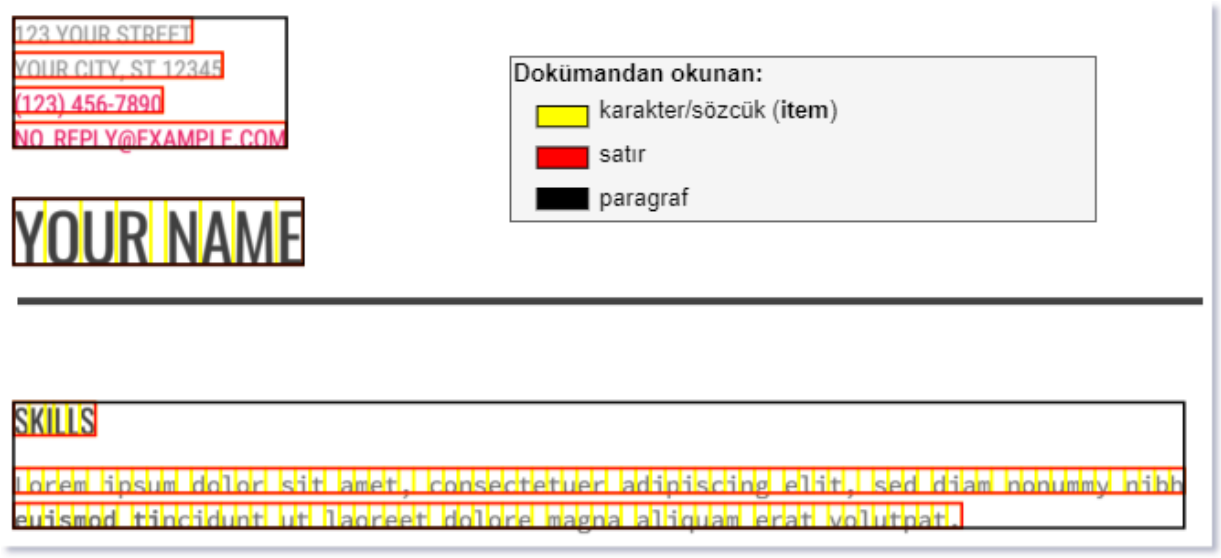


Şekil 4. PDF şablon çıkarım algoritması

2.1.1 Metin öğelerinin çıkarımı (Get text items)

Bu aşamada iText-pdf kütüphanesi kullanılarak PDF dokümanı metni tarandı. PDF şablonu metin içerisindeki item (öğe) ler ve bunların arasındaki ilişki baz alınarak oluşturulacağı için tüm item ler özellikleriyle beraber taranıp TextItem nesne tipinde oluşturuldu ve List<TextItem> items listesinde tutularak Şekil 4'te gösterilen Adım 3'e girdi olarak gönderildi. Aşağıda Şekil 5'te TextItem nesne yapısı gösterildi.

Şekil 5. TextItem nesne yapısı

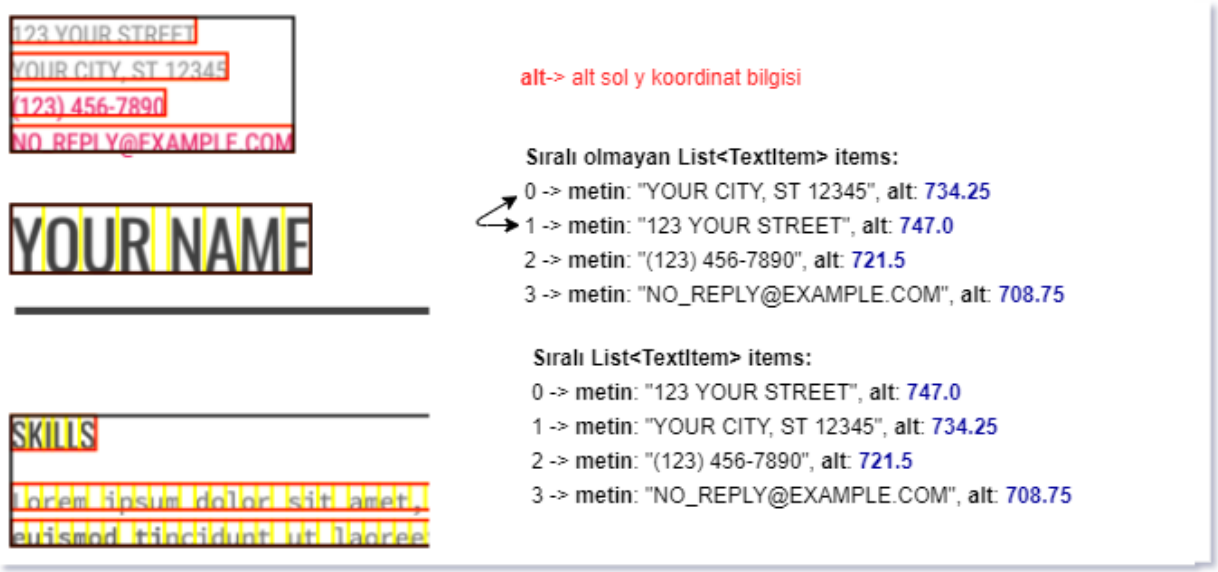


Şekil 6. Dokümandan okunan item ve oluşturulan satır ile paragraf bileşenleri

Şekil 6'da gösterildiği gibi item olarak adlandırdığımız bileşen PDF yapısına bağlı olarak bazen karakter bazen de bir bütün sözcük halinde, iText-pdf kütüphanesinin hazır bir fonksiyonu kullanılarak okundu. Satır ve paragraf bileşenleri ise, okunan item bileşenlerine Şekil 4'teki Adım 4 ve Adım 6'da gösterilen algoritmaların uygulanması sonucu oluşturuldu.

2.1.2 Mantıksal okuma sırasına göre sırlama (Sort text items)

iText-pdf kütüphanesi ile PDF içeriği okunurken, PDF dokümanında tanımlı olan Xref tablosundaki bilgilere ve tabloda belirlenen sıraya göre içerik okunuyor. Bu okuma, bir insanın okuması gibi yukarıdan aşağıya doğru bir yönde olmayabiliyor. PDF şablonunu oluşturabilmek için dokümandaki item lerin mantıksal okuma sırasına göre sıralanması önemli. Bu sorunu çözebilmek icin Şekil 5'te gosterilen compareTo(TextItem) fonksiyonu tanımlandı. Bu fonksiyon Java'daki Collections.sort() fonksiyonunun TextItem tipindeki bir listeyi sıralarken izleyeceği karşılaştırma ve sıraya dizme kurallarını barındırmaktadır. Aşağıdaki görselde, liste sıralaması yapılırken uygulanan compareTo() karşılaştırması detaylı bir şekilde gösterildi:

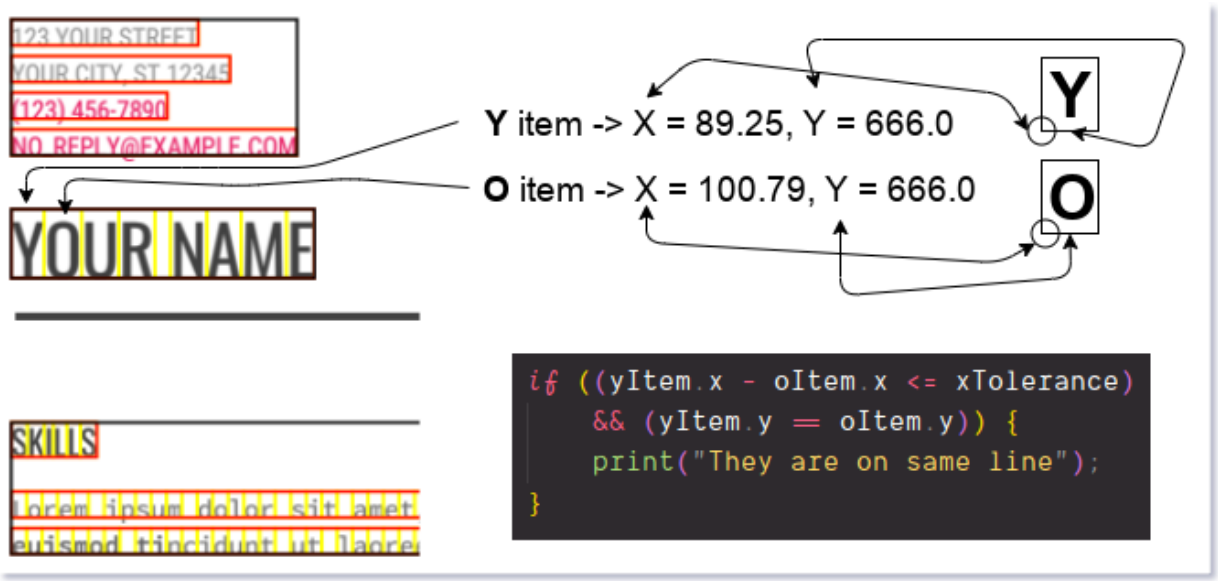


Şekil 7. List<TextItem> items, listesinin sıralanması

Şekil 7'de gösterildiği gibi sıralanmamış items listesinde 0. Eleman'ın sırası, okuma sırasına göre yanlış yerde bulunmaktadır. Sıralama yaparken listenin başından itibaren, her elemanın bottom/alt (elemanı kaplayan dörtgenin alanının alt y koordinatı) değeri bir sonraki ile karşılaştırılarak, bottom değeri yüksek olanın üstte kalacağı şekilde sırlama yapıldı. Görselden görülebileceği gibi sıralanmamış listedeki 0. Eleman, sıralanmış listede 1. Eleman olarak yer değiştirdi ve böylece PDF teki gibi mantıksal okuma sırasına göre bileşenler sıralandı.

2.1.3 Öğe listesini satır listesine dönüştürme (Get lines)

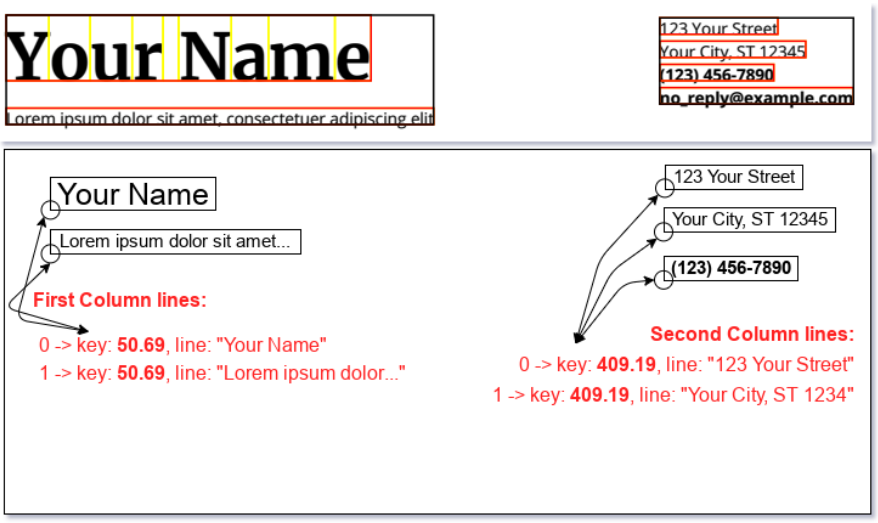
Şekil 6'da kırmızı renkle gösterilen satır bileşenlerini oluşturabilmek için sıralamada olduğu gibi items listesinden sırasıyla ikişer eleman alınarak, bu elemanların aynı Y koordinatı uzerinde ve belirli bir X toleransı aralığında olup olmadığı kontrolü yapıldı. Şekil 8'de, aynı satırda olan item ların belirlenmesi ve line (satır) yapısının oluşturulması, detaylı bir şekilde gösterildi:



Şekil 8. Öğeler aynı satırda mı kontrolünün yapıldığı areOnSameLine() fonksiyonu

2.1.4 Birden çok sütunlu PDF dokümanlarında satırları sütun bazında gruplandırmak (Split lines by X coordinate)

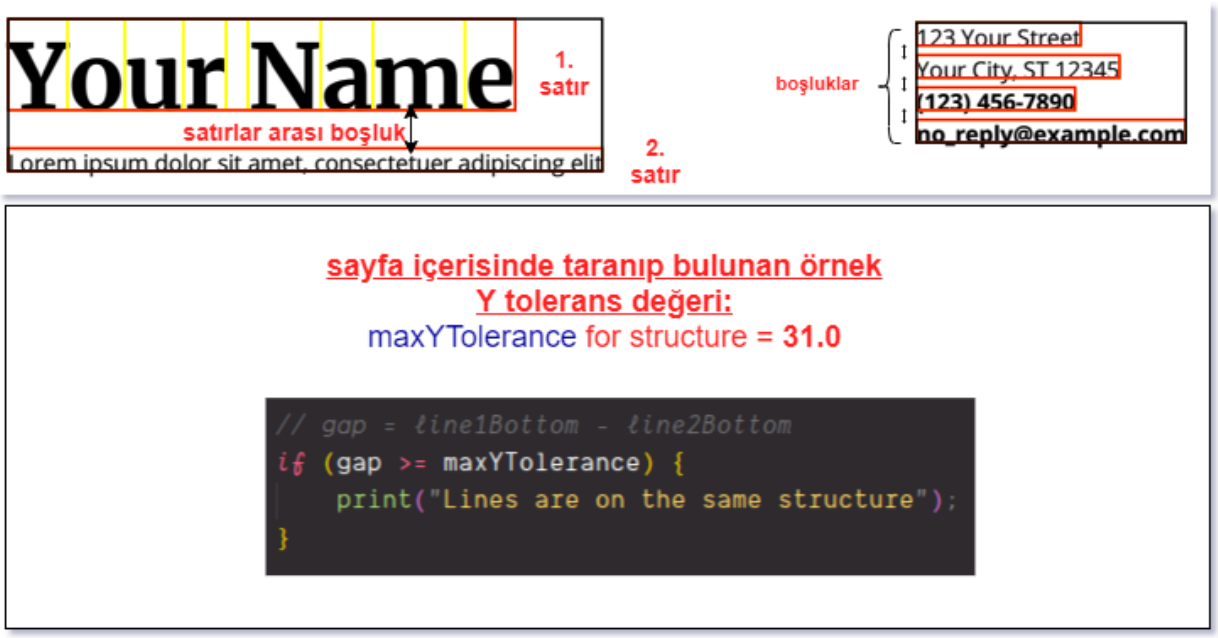
Birden fazla sütunlu PDF dokümanlarında Şekil 4'te gösterilen Adım 4'te oluşturulan satır yapılarının X koordinat değerine göre gruplandırılması gerçekleştirildi ve birden fazla sütuna sahip PDF dokümalarındaki sütun sayısından kaynaklanan anlamsız veri oluşma problemi çözüldü. Şekil 9'da gruplandırma detaylı bir şekilde gösterildi:



Şekil 9. Satırların sütunlara göre gruplandırılması

2.1.5 Satırları birleştirerek paragraf yapılarını oluşturmak (Get structures)

Şekil 6'da siyah renkle gösterilen paragraf bileşenlerini oluşturabilmek için sıralamada olduğu gibi lines listesinden sırasıyla ikişer eleman alınarak, bu elemanların belirli bir Y toleransı aralığında olup olmadığı kontrolü yapıldı. Şekil 10'da, aynı paragrafta (structure da) olan satırların belirlenmesi, detaylı bir şekilde gösterildi:



Şekil 10. Satırlar aynı paragrafa mı ait kontrolünün yapıldığı areInSameStructure() fonksiyonu

2.1.6 Önceki adımlarda oluşturulan yapılar kullanılarak PDF şablon yapısının oluşturulması

PDF şablonunu oluşturmak için kural tabanlı yaklaşım izlendi. Bu yaklaşımda önceden belirlenen kurallara göre, PDF dokümanından okunan verilerin içeriklerinin karşılaştırılması ve yapının oluşturulması sağlandı. PDF şemasının kaydedilip farklı içeriklere uyarlanabilmesi için XML veri yapısı kullanıldı. XML dosyasının kolayca oluşturulabilmesi ve doğrulama işlemlerinin gerçekleştirilebilmesi için XML şeması tanımlandı. Kural tabanlı karşılaştırma yapıldıktan sonra XML yapısında PDF şeması oluşturuldu. Ileri ki aşamalarda kural veritabanının grafik arayüz ve kullanıcı yardımıyla güncellenmesi ve bu sayede daha geniş bir şablon farklılığına sahip PDF dokümanlarının şablon çıkarımının mümkün kılınması amaçlanmaktadır. Şekil 11'de önceden tanımlanan XML şeması gösterildi:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Template">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="TemplateItem" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="TagName" type="xs:string"/>
                            <xs:element name="TagContent" type="xs:string"/>
                            <xs:element name="Font">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="FontSize" type="xs:double"/>
                                        <xs:element name="FontName" type="xs:string"/>
                                        <xs:element name="FontColor" type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="Rectangle">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="X" type="xs:double"/>
                                        <xs:element name="Y" type="xs:double"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Şekil 11. Tanımlanan PDF şablonu XML şeması (XSD - xml schema definition)

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Template>
    <TemplateItem>
        <TagName>phone</TagName>
        <TagContent>(123) 456-7890</TagContent>
        <Font>
            <FontSize>9.0</FontSize>
            <FontName>RobotoCondensed-Regular</FontName>
            <FontColor>(DeviceRgb) r: 233, g: 29, b: 99, hex: #e91d63</FontColor>
        </Font>
        <Rectangle>
            <X>89.25</X>
            <Y>721.5</Y>
        </Rectangle>
    </TemplateItem>
    <TemplateItem>…
    <TemplateItem>…
    <TemplateItem>…
    <TemplateItem>…
    <TemplateItem>…
    <TemplateItem>…
    <TemplateItem>…
    <TemplateItem>…
</Template>
```

Şekil 12. Oluşturulan PDF şeması

Şekil 12'de <Template> listesi içerisinde birden fazla, kural tabanlı karşılaştırma sonucu seçilen <TemplateItem> bileşenleri gösterildi. Görselde sadece ilk <TemplateItem> elemanı detaylı bir şekilde sunuldu. Kişiye özel CV dokümanı oluşturulma aşamasında Şekil 12'de gösterilen XML yapılı PDF şeması kullanıldı.

Bu XML dosyasındaki <TemplateItem> elementleri CV dokümanında belirli metinlerin, PDF dokümanının neresinde ve hangi yazı tipi, yazı boyutu ve yazı rengi yapısında oluşturulması gerektiği bilgisini içermektedir.

<TagName> elementi PDF dokümanında belirli metinlerin ayrımını yapabilmek için ayarlanan etiket ismi ya da etiket id si olarak tanımlandı. Örneğin kişisel CV dokümanı oluştururken 'telefon numarası' metnine ait bilgileri XML dosyasından çekebilmek için XML dosyasında TagName'i 'phone' olan TemplateItem'ı bulmak ve yazı tipi, lokasyon bilgileri gibi bilgilere ulaşıp ordaki bilgileri kullanarak telefon numarasının PDF dokümanında oluşturulması yolu izlendi.

<TagContent> elementi, orjinal CV'ye ait PDF şablonu çıkartma aşamasında oluşturulan herhangi bir yapının metin içeriğini temsil etmektedir. Örneğin orjinal CV dokümanından şablon yapısı çıkartılırken ki 'telefon numarası' kısmındaki metnin içeriği olarak özetlenebilir.

<Font> elementi, yine belirli bir metne (yapıya) ait yazı tipi bilgilerini içinde bulundurmaktadır. Aşağıda Font elementi içerisinde tanımlanan diğer elementler ve bigileri sunuldu.

- <FontSize> elementi, yazı boyutu bilgisini
- <FontName> elementi, yazı tipi bilgisini
- <FontColor> elementi, yazı rengi bilgisini içermektedir.

<Rectangle> elementi, isminin verilme sebebi kullanılan java kütüphanesi PDF dokümanından okunan verilerin lokasyon bilgisini Rectangle isimli nesne yapısında sunmasından kaynaklanmaktadır. Bu element, yine belirli bir metnin (yapının) PDF dokümanının xy düzlemi içerisinde, tam olarak hangi koordinatlarda bulunduğu bilgisini içermektedir. Aşağıda Font elementi içerisinde tanımlanan diğer elementler ve bigileri sunuldu.

- <X> elementi, bileşenin xy düzlemindeki X koordinatını
- <Y> elementi, bileşenin xy düzlemindeki Y koordinatını teşkil etmektedir.

3. Araştırma Sonuçları ve Tartışma

Yapılan çalışmada PDF içeriği kütüphane yardımıyla, direk doğal PDF yapısı içerisinden okunduğu için, PDF belgelerini, resim veya HTML`e dönüştürüp sonra OCR vs. gibi tekniklerle işlemek yerine, içeriği doğrudan PDF`ten çıkarmak ve analiz etmenin daha kolay ve kesin yöntem olduğu sonucu doğrulandı. PDF içerisinden okunup tag lenen öğeler önceden tanımlı kurallar doğrultusunda gerçekleştirildi ki kural veri tabanının grafik bir arayüz yardımıyla kullanıcı tarafından güncellenmesi ile daha geniş çapta PDF şablon türlerinin tanınmasına olanak sağlayabileceği sonucuna varıldı. Ayrıca okunan öğelerin yazı tipi stili meta datası da PDF dokümanından okunabildiği için daha tutarlı bir şekilde kişiye özel yeni CV dokümanlarının oluşturulması gerçekleştirildi. Kullanılan Google Font API (Application Programming Interface, Yazılım Programlama Arayüzü)'si yardımıyla orjinal yazı tipi gerçek zamanlı indirilip, yazı tipinde bile bir farklılık oluşturmamaya özen gösterildi.

4. Sonuç

Ileriki çalışmalarda kural tabanlı yaklaşımın, literatürdeki farklı yaklaşımlar ile birlikte kullanılması, görüntü işleme ve NLP (Doğal dil işleme) teknolojilerinin çalışmaya dahil edilmesi ile çok daha tutarlı şablonların elde edilmesi öngörülmektedir. Yazı tiplerinin çeşitliliğinden dolayı farklı font indirme API'leri dahil edilerek, daha geniş çapta yazı tiplerinin desteklenmesi de gerçekleştirilebilir.

5. Teşekkür

Bu çalışma Kocaeli Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi (BAP) tarafından 2018/136 nolu proje kapmasında desteklenmektedir.

Kaynakça

Aiello, M., Monz, C., Todoran, L., & Worring, M. (2002). Document Understanding for a Broad Class of Documents. *International Journal on Document Analysis and Recognition, 5(1)*, 1-16.

Altamura, O., Esposito, F., & Malerba, D. (2000). Transforming paper documents into XML format with WISDOM++. *International Journal on Document Analysis and Recognition, 4(1)*, 2-17.

Baker, J. B., Sexton, A. P., Sorge, V., & Suzuki, M. (2011). Comparing Approaches to Mathematical Document Analysis from PDF. *2011 International Conference on Document Analysis and Recognition* (s. 463-467). Beijing: IEEE. doi:10.1109/ICDAR.2011.99

Chao, H., & Fan, J. (2004). *Layout and Content Extraction for PDF Documents.* doi:10.1007/978-3-540-28640-0_20

Constantin, A., Pettifer, S., & Voronkov, A. (2013). PDFX: fully-automated PDF-to-XML conversion of scientific literature. *2013 ACM symposium on Document engineering* (s. 177-180). New York: ACM.

Eken, S., Atay, B., Sönmez, B. C., & Sayar, A. (2018). DocDig: Dijitalleştirilmiş Dokümanlarda İçerik Tabanlı Figür Arama. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, 6(1), 68-78.

Eken, S., Ekinci, E., & Sayar, A. (2014). XML Anahtar Kelimeleri Yardımıyla Türkçe Aritmetik Problemlerin Anlaşılması ve Çözülmesi. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, 2(1), 48-55.

Eken, S., Karabas, A., Sarı, H., & Sayar, A. (2018). A framework for recognition and animation of chess moves printed on a chess book. *Int. Arab J. Inf. Technol.*, 15(1), 29-36.

Eken, S., & Sayar, A. (2013). Animating Chess Moves Recorded on Chess Informant. *In Proceedings of the 3rd International Symposium on Computing in Science and Engineering* (pp. 35-40).

Gabdulkhakova, A., & Tamir, H. (2012). Document understanding of graphical content in natively digital PDF documents. *2012 ACM symposium on Document engineering*, (s. 137-140). New York. doi:https://doi.org/10.1145/2361354.2361385

Hassan, T. (2009). Object-Level Document Analysis of PDF Files. *ACM DL*, 47-55.

Jiang, D., & Yang, X. (2009). Converting PDF to HTML approach based on text detection. *In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS '09)* (s. 982-985). New York: ACM. doi:https://doi.org/10.1145/1655925.1656103

Liu, Y., Bai, K., Mitra, P., & Giles, C. L. (2009). Improving the Table Boundary Detection in PDFs by Fixing the Sequence Error of the Sparse Lines. *10th International Conference on Document Analysis and Recognition* (s. 1006-1010). Barcelona: IEEE.

Mohemad, R., Hamdan, A. R., Othman, Z. A., & Mohamad, N. M. (2011). Automatic Document Structure Analysis of Structured PDF Files. *IJNCAA*, 404-411.

Tunçer, M. (2013, April 9). *Özgeçmiş Hazırlama Tüyoları ve CV Örneği*. 12 18, 2019 tarihinde Kariyer.net: https://www.kariyer.net/kariyer-rehberi/ozgecmis-hazirlama-tuyolari-ve-cv-ornegi/ adresinden alındı