T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

RESİM FORMATINDAKİ DİJİTAL DÖKÜMANLARIN BİLGİSAYAR GÖRÜSÜ ve MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK LATEX FORMATINA ÇEVRİLMESİ

Murat KAZANÇ

DANIŞMAN
Dr. Öğr. Üyesi Mustafa DAĞTEKİN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Haziran, 2022

TEZ KABUL VE ONAYI

Murat KAZANÇ tarafından, **Dr. Öğr. Üyesi Mustafa DAĞTEKİN** danışmanlığında hazırlanan **"RESİM FORMATINDAKİ DİJİTAL DÖKÜMANLARIN BİLGİSAYAR GÖRÜSÜ ve MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK LATEX FORMATINA ÇEVRİLMESİ"** başlıklı bu çalışma, jürimiz tarafından **07/06/2022** tarihinde yapılan sınav sonucunda **oy birliği** ile başarılı bulunarak **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Tez Jürisi

		İmza	**Sonuç**
DANIŞMAN	Dr. Öğr. Üyesi Mustafa DAĞTEKİN İstanbul Üniversitesi-Cerrahpaşa Bilgisayar Mühendisliği Anabilim Dalı		☐ Kabul ☐ Ret
ÜYE	Doç. Dr. Pelin GÖRGEL İstanbul Üniversitesi-Cerrahpaşa Elektronik Mühendisliği Anabilim Dalı		☐ Kabul ☐ Ret
ÜYE	Dr. Öğr. Üyesi Ertuğrul SAATÇİ Kültür Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı		☐ Kabul ☐ Ret
ÜYE	Prof. Dr. Xxxx XXXX ..... Üniversitesi ..... Anabilim Dalı		☐ Kabul ☐ Ret
ÜYE	Prof. Dr. Xxxx XXXX ..... Üniversitesi ..... Anabilim Dalı		☐ Kabul ☐ Ret

Canım kızım İpek’e ithaf ediyorum…

BÜTÇE DESTEKLERİ

RESİM FORMATINDAKİ DİJİTAL DÖKÜMANLARIN BİLGİSAYAR GÖRÜSÜ ve MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK LATEX FORMATINA ÇEVRİLMESİ

Bu tez çalışması için herhangi bir kurumdan bütçe desteği alınmamıştır.

TEŞEKKÜR

Bu çalışma boyunca gösterdiği her türlü destek ve yardımdan dolayı çok değerli danışman hocam Sayın Dr. Öğr. Üyesi Mustafa DAĞTEKİN'e ve katkılarından dolayı değerli hocam Sayın Dr. Öğr. Üyesi Tolga ENSARİ'ye en içten dileklerimle teşekkür ediyorum.

Ayrıca tüm tez yazımı sürecinde bana destek olan aileme, motivasyonumun her zaman yüksek olmasını sağlayan arkadaşlarıma, desteklerini esirgemeyen meslektaşlarıma en içten dileklerimle teşekkürlerimi sunuyorum.

Haziran 2022 Murat KAZANÇ

İÇİNDEKİLER

Sayfa No

TEZ KABUL VE ONAYI ..... **ii**
BÜTÇE DESTEKLERİ ..... **iv**
TEŞEKKÜR ..... **v**
İÇİNDEKİLER ..... **vi**
ŞEKİL LİSTESİ ..... **viii**
TABLO LİSTESİ ..... **ix**
SİMGE VE KISALTMA LİSTESİ ..... **xi**
ÖZET ..... **xiii**
ABSTRACT ..... **xv**
1. GİRİŞ **1**
2. KAVRAMSAL ÇERÇEVE **4**
2.1. LİTERATÜR İNCELEMESİ 4
2.2. GENEL KISIMLAR 7
2.2.1. OpenCV 7
2.2.2. Tesseract OCR Motoru 8
2.2.3. Makine Öğrenmesi 8
2.2.4. Yapay Sinir Ağları ve Derin Öğrenme 10
2.2.5. Doğal Dil İşleme 11
2.2.6. Metin Sınıflama İşlemi 13
2.2.7. Metin Sınıflama İçin Metrikler 14
2.2.8. LaTex Kodu Yazmak 16
3. YÖNTEM **18**
3.1. VERİ KÜMESİ 18
3.1.1. Veri Kümesinin Elde Edilmesi 18
3.1.2. Veri Ön İşleme 19
3.2. MODEL EĞİTİMLERİ 25
3.2.1. Metin / Şekil Sınıflaması Modelinin Eğitilmesi 25
3.2.2. Sayfa Tipinin Sınıflaması Modelinin Eğitilmesi 26

3.2.3. Ara Sayfa Metin Sınıflaması Modelinin Eğitilmesi 27
3.2.4. İlk Sayfa Metin Sınıflaması Modelinin Eğitilmesi 32
3.2.5. Son Sayfa veya Referans Sayfası Metin Sınıflama Modelinin Eğitilmesi 33
3.3. SEGMENTASYON İŞLEMİ 34
3.4. OKUMA SIRASININ BULUNMASI 37
3.5. METİN / ŞEKİL SINIFLAMASININ YAPILMASI 40
3.6. SAYFA TİPİ SINIFLAMASININ YAPILMASI 40
3.7. SAYFA TİPİNE GÖRE METİN SINIFLAMA YAPILMASI 40
3.7.1. İlk Sayfa 40
3.7.2. Ara Sayfa 41
3.7.3. Son veya Referans Sayfası 41
3.7.4. Metin Sınıflama Sonuçlarının Görüntülenmesi 41
3.8. LATEX KODUNUN ÜRETİLMESİ 42
3.9. UYGULAMANIN GELİŞTİRİLMESİ 43
4. BULGULAR 45
4.1. SEGMENTASYON 46
4.2. METİN / ŞEKİL SINIFLANDIRMASI 48
4.3. OKUMA SIRASININ BULUNMASI 48
4.4. SAYFA TİPİNİN BULUNMASI 48
4.5. SAYFA TİPİNE ÖZEL METİN SINIFLARININ BULUNMASI 48
4.5.1. İlk Sayfa 48
4.5.2. Ara Sayfa 50
4.5.3. Son Sayfa 50
4.6. LATEX KODUNUN ÜRETİLMESİ 50
4.7. UYGULAMANIN GENEL DEĞERLENDİRİLMESİ 51
5. TARTIŞMA 52
6. SONUÇ VE ÖNERİLER 55
KAYNAKLAR 56
İNTİHAL RAPORU İLK SAYFASI 60
KURUM İZNİ YAZILARI 61

ŞEKİL LİSTESİ

Sayfa No

Şekil 1 : Örnek YSA [2] 11
Şekil 2 : Metin / Şekil Veri Kümesi için Korelasyon Matrisi 21
Şekil 3 : Sayfa Tipi Belirleme Veri Kümesi Sınıf Dağılımı 22
Şekil 4 : Ara Sayfa Veri Kümesi Sınıf Dağılımı 23
Şekil 5 : İlk Sayfa Veri Kümesi Sınıf Dağılımı 24
Şekil 6 : Son Sayfa veya Referans Sayfası Veri Kümesi Sınıf Dağılımı 24
Şekil 7 : Orijinal Resim (a), 4x4 Genişletme işlemi (b) ve 12x12 Genişletme işlemi (c) 35
Şekil 8 : Okuma Sırası Bulma Algoritması 37
Şekil 9 : Metin Sınıflaması Sonuçlarının Kaynak Doküman Üzerinde Gösterilmesi 42
Şekil 10 : Uygulama Geliştirme Algoritması 43
Şekil 11 : Kaynak Dijital Doküman (a) ve Uygulamanın Oluşturduğu PDF Çıktısı (b) 45
Şekil 12 : Kaynak Dijital Doküman (a) ve Uygulamanın Başarısız Olduğu PDF Çıktısı (b) .. 46
Şekil 13 : Segmentasyon Sonucu Oluşan Blokların Gösterimi 47
Şekil 14 : Segmantasyonda Başarısız Olunan Örnekler 47
Şekil 15 : İlk Sayfa için Segmentasyon ve Metin Sınıflama Sonuçları 49
Şekil 16 : Ara Sayfa için Segmentasyon ve Metin Sınıflama Sonuçları 49
Şekil 17 : Son Sayfa için Segmentasyon ve Metin Sınıflama Sonuçları 50
Şekil 18 : Örnek Metin Blokları 51
Şekil 19 : Geliştirilen Uygulama Sonucu (a) ve LayoutParser Sonucu (b) Karşılaştırması 53
Şekil 20 : Geliştirilen Uygulama Sonucu (a) ve LayoutParser Sonucu (b) Karşılaştırması 53

TABLO LİSTESİ

Sayfa No

Tablo 1 : Karmaşıklık Matrisi 15

Tablo 2 : Veri Kümesi Örneği 19

Tablo 3 : Metin / Şekil Sınıflaması için Veri Kümesi Analizi 20

Tablo 4 : Metin / Şekil Veri Kümesi 21

Tablo 5 : Metin / Şekil Sınıflaması Model Eğitim Sonuçları 25

Tablo 6 : Sayfa Tipi Sınıflaması için Eğitilen Modellerin Sonuçlarının Karşılaştırması 26

Tablo 7 : Sayfa Tipi Sınıflaması için Eğitilen SGDClassifier Değerlendirmesi 27

Tablo 8 : Sayfa Tipi Sınıflaması için Eğitilen SGDClassifier Karmaşıklık Matrisi 27

Tablo 9 : Ara Sayfa Metin Sınıflaması için Eğitilen Modellerin Karşılaştırması 28

Tablo 10 : Ara Sayfa Metin Sınıflaması için Eğitilen SVM Değerlendirmesi 28

Tablo 11 : Ara Sayfa Metin Sınıflaması için Eğitilen SVM Karmaşıklık Matrisi 28

Tablo 12 : PyTorch Kütüphanesi ve Ön Eğitimli BERT Modelin Değerlendirmesi 29

Tablo 13 : PyTorch Kütüphanesi ve Ön Eğitimli BERT Modelin Karmaşıklık Matrisi 29

Tablo 14 : Tensorflow Keras ile LSTM Model Değerlendirmesi 30

Tablo 15 : Tensorflow Keras ile LSTM Model Karmaşıklık Matrisi 30

Tablo 16 : Tensorflow Keras LSTM Model ve GloVe Değerlendirmesi 31

Tablo 17 : Tensorflow Keras LSTM Model ve GloVe Karmaşıklık Matrisi 31

Tablo 18 :Eğitilen Modellerin Doğruluk ve Model Boyutu Karşılaştırılması 31

Tablo 19 : İlk Sayfa Metin Sınıflaması için Eğitilen Modellerin Karşılaştırması 32

Tablo 20 : İlk Sayfa Metin Sınıflaması için Eğitilen SGDClassifier Değerlendirmesi 32

Tablo 21 : İlk Sayfa Metin Sınıflaması için Eğitilen SGDClassifier Karmaşıklık Matrisi 33

Tablo 22 : Son Sayfa Metin Sınıflaması için Eğitilen Modellerin Karşılaştırması 33

Tablo 23 : Son Sayfa Metin Sınıflaması için Eğitilen SVM Değerlendirmesi.....34
Tablo 24 : Son Sayfa Metin Sınıflaması için Eğitilen SVM Karmaşıklık Matrisi.....34
Tablo 25 : Kaynak Dijital Resimden Segmantasyon Sonucu Çıkarılan Veri Kümesi 36

SİMGE VE KISALTMA LİSTESİ

Simgeler **Açıklama**

q : Kümülatif sınıf toplamı
P : Olasılık
μ : Ortalama
σ : Varyans
t : Eşik değeri

Kısaltmalar **Açıklama**

PDF : Taşınabilir Belge Formatı (Portable Document Format)
NLP : Doğal Dil İşleme (Natural Language Processing)
GPU : Grafik İşlemci Birimi (Graphics Processor Unit)
TPU : Tensör İşleme Birimi (Tensor Processing Unit)
YSA : Yapay Sinir Ağı (Artificial Neural Network)
OCR : Optik Karakter Tanıma (Optical Character Recognition)
NLTK : Doğal Dil Araçları (Natural Language Toolkit)
CNN : Evrişimli Sinir Ağı (Convolutional Neural Network)
FPN : Özellik Piramit Ağı (Feature Pyramid Networks)
CRF : Koşullu Rastgele Alanlar (Conditional Random Field)
LSTM : Uzun Kısa Süreli Bellek (Long Short Term Memory)
KNN : K En Yakın Komşu (K-Nearest Neighbors)
SVM : Destek Vektör Makinesi (Support Vector Machine)
SGD : Stokastik Gradyan İniş (Stochastic Gradient Descent)
RNN : Tekrarlayan Sinir Ağı (Recurrent Neural Network)
BERT : Bidirectional Encoder Representations from Transformers
R-CNN : Region Based Convolutional Neural Networks
GLCM : Gri Düzey Birlikte Oluşum Matrisi (Gray-Level Co-Occurrence Matrix)
XML : Genişletilebilir İşaretleme Dili (Extensible Markup Language)
CSV : Virgülle Ayrılmış Değerler (Comma Separated Values)

HTML : Hiper Metin İşaretleme Dili (Hypertext Markup Language)

JSON : JavaScript Nesne Gösterimi (JavaScript Object Notation)

ÖZET

YÜKSEK LİSANS TEZİ

RESİM FORMATINDAKİ DİJİTAL DÖKÜMANLARIN BİLGİSAYAR GÖRÜSÜ ve MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK LATEX FORMATINA ÇEVRİLMESİ

Murat KAZANÇ

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman : Dr. Öğr. Üyesi Mustafa DAĞTEKİN

Günümüzde kâğıda basılı belgelerin yerini dijital dokümanlar almıştır. Bunlar resim biçiminde veya özel dosya formatları şeklinde olabilmektedir. Yaygın olarak kullanılan kelime işlemci programları genel kitleye hitap eder ve bu programların kullanımları en basit düzeyde tutulmuştur. LaTex dosya formatı kullanılarak hazırlanan dokümanlar profesyonel düzeyde detaylı içerikler hazırlanmasını sağlamaktadır. LaTex dosya formatı özellikle kitap ve akademik dokümanların yazımında tercih edilmektedir. Akademik çevrelerce tercih edilen bu dosya formatının bir dezavantajı vardır. Doküman oluşturulurken kod yazılması gerekmektedir. Bu işlem çoğu kullanıcı için sorun olmaktadır.

Resim formatındaki bir dokümandan LaTex kodunun üretilmesi bilgisayarlı görü ve doğal dil işleme alanlarının birlikte kullanılmasını gerektirmektedir. Bu çalışmada öncelikle resim üzerinde metin, tablo ve şekillerin bulunduğu yerlerin sınırları (bloklar) tespit edilmiştir. Sonrasında bulunan bu blokların doğal dil işleme metotları ile metinlerin sınıfları tespit

edilmiştir. LaTex kodu oluşturulacak dokümanın akışının bozulmaması için okuma sırası tespit edilmiştir. Son aşama olarak tüm bilgiler kullanılarak LaTex kodu üretilmiştir.

Haziran 2022 , 81 sayfa.

Anahtar kelimeler: Makine Öğrenmesi, Bilgisayarlı Görü, Doğal Dil İşleme, Metin Sınıflama, Okuma Sırası, LaTex

ABSTRACT

M.Sc. THESIS

CONVERTING DIGITAL DOCUMENTS IN IMAGE FORMAT TO LATEX FORMAT USING COMPUTER VISION AND MACHINE LEARNING METHODS

Murat KAZANÇ

İstanbul University-Cerrahpaşa

Institute of Graduate Studies

Department of Computer Engineering

Computer Engineering

Supervisor : Assist. Prof. Dr. Mustafa DAĞTEKİN

Nowadays, digital documents have steadily replacing printed documents. These documents can be in dijital images or in custom file format. Widely used word processing programs appeal to the general audience and their use has been developed to be as a simple to use as possible for the end user. Documents prepared using LaTex file format provide a more professional-level content. The LaTex file format is especially preferred for the writing books and academic documents. This file format, which is preferred by academic circles, has a drawback. Code must be written when creating the document. This process is a problem for most users.

Generating LaTex code from an image-formatted document requires both the use of computer vision and natural language processing areas. In this study, the boundaries (blocks) of the places where text, tables and figures are located on the picture were found out. The natural language processing methods and classes of the texts were detected in these blocks afterwards. The reading order has been determined so that the flow of the document to be created in LaTex code is not disrupted. As a final stage, LaTex code is generated using all the information available.

July 2022, 81 pages.

Keywords: Machine Learning, Computer Vision, Natural Language Processing, Text Classification, Reading Order, LaTex

1. GİRİŞ

Dijitalleşen dünya ile kâğıda basılı belgelerin yerini dijital dokümanlar almıştır. Bunlar resim biçiminde veya özel dosya formatları şeklinde olabilmektedir. Yaygın olarak kullanılan kelime işlemci programları genel kitleye hitap eder ve bu programların kullanımları en basit düzeyde tutulmuştur. LaTex dosya formatı kullanılarak hazırlanan dokümanlar profesyonel düzeyde detaylı içerikler hazırlanmasını sağlamaktadır. LaTex dosya formatı özellikle kitap ve akademik dokümanların yazımında tercih edilmektedir. Akademik çevrelerce tercih edilen bu dosya formatının bir dezavantajı vardır. Doküman oluşturulurken kod yazılması gerekmektedir. Bu işlem çoğu kullanıcı için sorun olmaktadır.

Resim formatındaki bir dokümandan LaTex kodunun üretilmesi bilgisayarlı görü ve doğal dil işleme alanlarının birlikte kullanılmasını gerektirmektedir.

Bilgisayarlı görü [1], dijital görüntülerden veya video görüntülerinden bir bilgi veya anlam çıkartmaya çalışan disiplinler arası bir alandır.

Bilgisayarlı görü ile metin işleme basit olduğu düşünülen bir konu olmuştur. Ancak çalışmalar yapıldıkça sorunun bu kadar basit olmadığı görülmüştür. 1980'li yıllarda özellikle yapay sinir ağlarında yaşanan gelişmeler ile örneğin el yazısı tanıma gibi çalışmalar bu alana ilgiyi arttırmıştır [2]. Günümüzde bilgisayarlı görü derin öğrenme metotları ağırlıklı olarak kullanılmaktadır. Ancak eğitimlerin uzun sürmesi sebebiyle daha verimli algoritma arayışları sürmektedir.

Ayrıca 2000'li yıllardan itibaren internetin dünya genelinde yaygınlaşması ile üretilen dijital içerik miktarı her geçen yıl artarak katlanmaktadır. Bu içeriklerden olan dijital dokümanlardan anlam ve bilgi çıkarılması bu işlemin otomatikleştirilmesi önemli bir konu olmuştur. İnternet ortamının büyük bir dijital kütüphane haline gelmiştir. Bu durum bilgisayarlı görü alanında ihtiyaç duyulan büyük miktarda veri ihtiyacını karşılamakta yardımcı olmaktadır.

Doğal dil işleme (NLP) 1950'lerde yapay zekâ ve dil bilimin keşişimi ile başlamıştır [3]. Büyük hacimli metinleri verimli bir şekilde indekslemek ve arama yapmak için istatistik tabanlı bir yöntem olmuştur. Günümüzde işler biraz daha karışık hale gelmiştir. Metin sınıflandırma, özet

çıkartmak gibi çok daha karmaşık işler NLP ile gerçekleştirilebilmektedir. Başlangıçta kural tabanlı çalışan sistemler kullanılmıştır. Sonrasında Naive Bayes, K-En Yakın Komşular, Gizli Markov Modelleri, Koşullu Rastgele Alanlar, Karar Ağaçları, Rastgele Ormanlar ve Destek Vektör Makineleri gibi makine öğrenme yaklaşımları yaygın olarak kullanılmıştır.

Hesaplama gücü ve paralelleştirmede grafik işlem birimleri (GPU) kullanılmasıyla birlikte geleneksel yöntemlere göre kıyas kabul etmeyecek büyüklükte eğitilebilir parametreler ile çalışan Yapay Sinir Ağlarının (YSA) kullanımı giderek popülerleşmiştir. Son birkaç yılda bu yaklaşımlar yerine ağırlıklı olarak YSA modellere geçiş yapılmıştır.

Bir NLP modelin değerlendirmesinde verilerdeki yüksek değişkenlik ve her bir görevdeki performansı değerlendirmek için kullanılan çok sayıda metrik vardır [4]. Çoğu zaman benzer modelleri karşılaştırmak bile zordur çünkü her biri için farklı metrikler kullanılarak rapor edilmektedirler.

Bu çalışmada geliştirilecek uygulamada akademik yayınlar üzerine yoğunlaşılmıştır. Akademik yayınlar yapısı gereği ilk sayfasında başlık, yazar bilgisi, özet ve anahtar kelimeler gibi özel metin grupları içermektedir. Yayının ara sayfaları tek sütun veya çift sütun olacak şekilde başlık, paragraf, tablo, şekil ve liste gibi içerik grupları içermektedir. Yayının son sayfası ya da sayfaları özel olarak referans listesini içermektedir. Bir dijital akademik yayının anlaşılabilmesi için bahsedilen bu içerik gruplarının öncelikle dijital dokümanda içerik bloğu olarak bulunması ardından içeriğin metin veya şekil olarak ayrıştırılması gerekmektedir. Metin olarak sınıflandırılan blokların içerikleri sayfanın tipine (ilk, ara veya son) göre sınıflandırılması gerekmektedir.

Ayrıca dijital yayının içerik akışının bozulmaması için okuma sırasının da bulunması gerekmektedir. Tüm bunlar dikkate alındıktan sonra elde edilen bilgiler doğrultusunda LaTex kodunun oluşturulması mümkündür.

Bu çalışmanın kapsamının belirlenmesinde bazı konular çalışma kapsamına dahil edilmemiştir. Dijital bir dokümanın temsilinde olması gereken yazı tipi, yazı büyüklüğü gibi konulara bu çalışma kapsamında yer verilmemiştir. Ayrıca dijital doküman içerisinde yer alan denklem ve tablo içerik gruplarının LaTex koduna dönüştürülmesi çok kapsamlı çalışmalar gerektirmektedir. Hatta sadece tablo ve denklemlerden içerik çıkarılması ile ilgili yarışmalar

mevcuttur. Bu çalışmada yapılan işlemler gelecek çalışmalar için bir başlangıç noktası ve genişletilebilir bir çalışma ortamı sunacağı düşünülmektedir.

Bu tez çalışmasının organizasyonu şu şekildedir:

Kavramsal çerçeve bölümünde geçmişten günümüze konu ile ilgili literatür incelenerek yapılan çalışmalardan bahsedilmiştir. Sonrasında Çalışma sırasında kullanılacak teknolojiler olan OpenCV kütüphanesi (Open Computer Vision) [5], Optik karakter tanıma (OCR), Okuma sırası bulma, denetimli makine öğrenmesi modelleri, derin öğrenme modelleri, sınıflamada kullanılan metrikler ve LaTex hakkında genel bilgiler verilmiştir.

Yöntem bölümünde uygulamanın geliştirilme aşamaları verilmiştir. Bu aşamalar veri kümesinin elde edilmesi, veri ön işleme yapılması, gerekli makine öğrenmesi ve derin öğrenme modellerinin eğitilmesi, metin / şekil sınıflamasının yapılması, okuma sırasının bulunması, sayfa tipinin bulunması, metin sınıflaması yapılması ve LaTex kodunun üretilmesidir.

Bulgular bölümünde geliştirilen uygulama değerlendirilmiştir. Tartışma bölümünde geliştirilen uygulama mevcut uygulamalar ile karşılaştırılmıştır. Sonuç ve öneriler bölümünde gelecekte neler yapılabileceğinden bahsedilmiştir.

2. KAVRAMSAL ÇERÇEVE

Bu bölümde çalışma alanı ile ilgili literatürde yapılan çalışmalar incelenmiştir. Ayrıca çalışma kapsamında geliştirilecek uygulamaya temel olacak teknik konular incelenecektir.

2.1. LİTERATÜR İNCELEMESİ

LaTex (LAY-tek) bir belge hazırlama sistemidir. Çokça tercih edilen Microsoft Word, Libre Office gibi kelime işlemci programlarının çalışma stili şu şekilde ifade edilir "What You See Is What You Get" (Ne Görürsen Onu Alırsın). Bu programlarda belge yazılırken biçimlendirilmiş hali görünmektedir. LaTex ile bu farklıdır. Belgenin içeriği düz yazı olarak yazılır. LaTex kodları ile istenilen biçimlendirmeler yapılır ancak derleme sonrası sonuç görüntülenebilir. Bu açıdan web sayfaları geliştirmek için kullanılan "HTML" metin işaretleme diline benzetilebilir.

LaTex 1978 yılında Donald E. Knuth tarafından geliştirilmeye başlanmıştır [6]. Çıkış noktası ise baskıya verilen dökümanların yetersiz baskı kalitesi olmuştur. Yıllar içerisinde LaTex özellikle akademik çevreler tarafından kitap ve makale yazımında benimsenmiş ve günümüzde açık kaynak kodlu olarak geliştirilmeye devam edilmektedir.

Günümüzde LaTex'in tercih edilmesi sebepleri arasında şunlar gösterilebilmektedir. Profesyonel kalitede baskı alınabilmesi, standart olması, her yerde çalışabilmesi ve açık kaynak olmasıdır.

Resim formatındaki bir dokümanı LaTex koduna dönüştürebilmek için ilk yapılması gereken işlem resim üzerinde metin ya da şekil gibi blokların bulunmasıdır. Bu işlem için geliştirilen kütüphanelerden bir tanesi "LayoutParser" dir. Derin öğrenme metodunu kullanan bu kütüphane dijital doküman analizini kolaylaştıran açık kaynak kütüphane olmayı hedeflemektedir. LayoutParser'ı öne çıkaran bir diğer özelliği ise ön eğitimli yapay sinir ağı modellerine sahip olmasıdır. Bunlar; PubLayNet (Akademik dokümanlar için) [7], PRImA (dergi ve akademik raporlar için) [8], Newspaper (gazeteler için) [9] ve TableBank (iş ve akademik dokümanlar için) [10]. Sayılan ön eğitimli modeller ortak sınıflara sahip olduğu gibi kendilerine has sınıfları da vardır. Örneğin TableBank ön eğitimli modeli sadece tabloları

bulmaktadır. LayoutParser ayrıca “Tesseract” kütüphanesini kullanan OCR özelliğine de sahiptir [11].

Resimdeki blokların bulunması ile alakalı bir diğer çalışma “LayoutLM” dir [12]. Bu çalışmada resimde bulunan blokları sınıflandırmak için şekilsel özelliklerinin yanı sıra blokların içerisindeki metinlerinde dikkate alınması gerektiği söylenmektedir. Metinlerin anlaşılması için ön eğitimli doğal dil işleme modellerinden örneğin BERT (Bidirectional Encoder Representations for Transformers) kullanmaktadır.

Blokların bulunması ile ilgili bir diğer çalışmada segmentasyon için iki derin öğrenme modeli kullanılmıştır [13]. İlk derin öğrenme modeli olarak Mask R-CNN (Region Based Convolutional Neural Networks) ile blokların yerleri tespit edilmiştir. İkinci derin öğrenme modeli ile farklı ölçekteki nesneleri algılamakta kullanılan FPN (Feature Pyramid Networks) kullanılmıştır. Bu iki derin öğrenme modelinin birlikte çalışması sonucu bloklar sınıflandırılmaya çalışılmıştır.

Başka bir çalışmada farklı bir odak noktası ile akademik yayınlar içerisinde şekillerin önemli olduğunu iddia ederek resim ve altındaki açıklama yazısını tespit etmek ana hedefleri olmuştur [14]. Geliştirme aşamasında derin öğrenme modeli kullanmak yerine OCR ve kural tabanlı tespit yöntemi geliştirmişlerdir.

Dijital belge içerisinde metin ve metin olmayan blokların tespit edilmesi için yapılan bir diğer çalışmada algoritma temelli bir sistem önerilmiştir [15]. Yatay ve dikey çizgiler tespit edilerek kaldırılır. Sonrasında genişletme (dilation) yöntemi ile metin veya metin olmayan içerik gruplanarak bloklar ortaya çıkarılır. Resimler metinlerden farklı doku özelliklerine sahiptirler. Bulunan bloklar içerisinde GLCM (Gray-Level Co-Occurrence Matrix) ile doku özellikleri bulunmaktadır. GLCM gri tonlamalı resimler üzerinde çalışabilmektedir. GLCM ile resmin tamamı ya da parçalar halinde her bir piksel için dikey, yatay ve çapraz yönlerdeki piksellerin yan yana gelme sayıları kullanılarak oluşturulan bir matristir. Bir sonraki aşamada K-Ortalama Kümeleme algoritması ile bloklar metin veya metin değil olarak sınıflandırılır.

Evrişimli Sinir Ağları (CNN) kullanılarak tablo veya grafikler dijital dokümanlarda tespit edilebilmektedir [16]. Günümüzde hızla ve hızlanmaya devam ederek artan bilgi üretimi gerçekleşmektedir. Gelinen noktada bir insanın veya insan grubunun bu bilgi akışını takip etmesi mümkün değildir. Bu soruna çözüm olarak doğal dil işleme metotları ile dijital belgeler

içerisinde bilgi almaya çalışmak yerine akademik yayınlar içerisinde bulunan tablo ve grafikler metinlere göre daha fazla bilgi vermektedirler.

Dokümanlarda tabloların tespiti için yapılmış bir diğer çalışmada Kodlayıcı-Kod Çözücü (Encoder-Decoder) Modeli kullanılmıştır [17]. Kodlayıcı Model'de CNN kullanılarak resmin özellik haritası çıkarılmıştır. Kod Çözücü Modelde ise dikkat mekanizması ile standart uzun kısa süreli bellek (LSTM) modeli kullanılarak tablo hücreleri tespit edilmiştir.

OCR kullanılarak matematik denklemlerinin resimlerden elde edilmesi başarılı sonuçlar vermemektedir [18]. Temel sorun matematik denklemlerinin 2 boyutlu olmasıdır. Bu soruna çözüm olarak denklemlerin tek boyutlu olarak ifade edilmesi düşünülmüştür. Bu işlemi gerçekleştirebilmek için LaTex kodu dönüşümü yapılması önerilmektedir [18].

LaTex yazılımının en güçlü olduğu noktalardan bir tanesi denklemlerin yazılmasıdır. Bir resimden denklemlerin bulunup LaTex kodu olarak ifade edilmesi kolaylık sağlamaktadır [19]. Denklemlerde bulunan üst simge, alt simge veya özel sembollerin ifadesi bir vektör dizi olarak tanımlanabilmektedir. Bu işlemi gerçekleştirmek için yapılan bir çalışmada yine Kodlayıcı-Kod Çözücü Modeli kullanılmıştır. Kodlayıcı olarak CNN kullanılarak özellik haritası çıkarılmıştır. Kod Çözücü olarak yumuşak dikkat mekanizması kullanan bir LSTM ile LaTex kod bloğu üretilmiştir.

Denklemlerin LaTex koduna dönüştürülmesi ile ilgili bir diğer çalışmada çoğu araştırmacının kullandığı Kodlayıcı – Kod Çözücü Modeli yerine Transformer modeli kullanılmıştır [20]. Buradaki temel sebep Kodlayıcı – Kod Çözücü modelin tüm bağlamı ve sembollerin hiyerarşik konumlarını bilememesidir. Yapılan çalışma üç aşamadan oluşmaktadır bunlar: tüm bağlamın çıkarılması, konum bilgisi ve sembollerin birbiri ile bağımlılıklarını ortaya koyan bir Transformer kullanılması ve LaTex kodunun üretilmesi için Maske Tabanlı Dikkat Mekanizması kullanan bir Kod Çözücü kullanılmasıdır.

Dijital dokümanlardan bilgi çıkarmada yapılan bir diğer işlem de metinlerin fontlarının tespit edilmesidir. Bu özellik içerikten ziyade bize doküman ile ilgili yapısal bilgi vermektedir. Yapılan bir araştırmada CNN özel bir türü olan Yığınlı Evrişimli Oto Kodlayıcı (SCAE) kullanılmıştır [21].

Dijital bir metindeki fontu bulmak çok büyük bir sınıflama problemidir [22]. On binlerce farklı font mevcuttur. Ayrıca sadece metnin yazı fontunu bulmak dışında yazı ağırlığı ve eğik olma gibi durumlarda göz önünde bulundurulursa aşması zor bir durum ortaya çıkmaktadır. Araştırmacılar geliştirdikleri modelin özelliği doğrusal olmayan görsel temsillerini doğrudan öğrenebilmektedir.

OCR sistemleri üzerine yapılan bir çalışmada CNN ve LSTM kullanımı ile OCR yapan modellerin yüksek doğruluk oranı yakaladığı belirtilmiştir. Ancak modelin diskte büyük boyutlar kapladığı ve yüksek hesaplama maliyeti olduğu söylenmiştir. Önerdikleri yöntem ise modelin özellik çıkarma işleminde değişiklik yapmadan LSTM tarafını küçültmektir. Son olarak modelin boyutunu küçültmek için bir sıkıştırma yöntemi kullanılması olmuştur [23].

Dijital doküman dosya formatı PDF üzerinde yapılan bir çalışmada [24] LaTex kodu ile üretilmiş PDF dosyaları üzerinde tersine mühendislik ile LaTex kodu elde edilmeye çalışılmıştır. Kaynak PDF dosyasının meta verilerindeki ilişkileri anlama ve buna uygun kodların üretilmesini sağlayan bir derin öğrenme modeli kullanılmıştır. Model çıktısı Dijital dokümanın LaTex kodu yeniden oluşturulmuştur.

Yapılan literatür çalışmasının dışında gerçek dünya uygulamaları da incelenmiştir. Kelime işlemci belge formatı olan DOCX'i LaTex koduna dönüştüren web uygulamaları mevcuttur. Bu uygulamalar kaynak dosya içerisinde XML formatında metin bilgisi ve metin ile ilgili font, font size ve tüm görsel özellikleri barındırmaktadır. Resim formatındaki bir dökümanı LaTex koduna dönüştürmede asıl zorluk bu özelliklerin bulunmasıdır. Yine farklı uygulamalarda verilen dijital dokümanda OCR yapılarak tüm bulunan karakterler sayfa üzerindeki konumlarına sabit koordinatlar ile yerleştirilmiştir. Bu çalışmalarda bir insan tarafından yazılmış LaTex koduna benzer ya da yakın bir kod üretememektedirler.

2.2. GENEL KISIMLAR

2.2.1. OpenCV

OpenCV günümüzde açık kaynak kodlu bilgisayarlı görü kütüphanesidir. Resim ve videolar üzerinde analizler ve manipülasyonlar yapmamızı sağlamaktadır. Kullanımı çok yaygın olan bu kütüphane neredeyse bilgisayarlı görü uygulamalarının tamamında kullanılmaktadır. OpenCV C++ programlama dili ile yazılmıştır. Geliştiricileri tarafından 2009 yılında yapılan büyük bir güncelleme ile günümüzdeki halini almıştır. İçerisinde 2500'den fazla optimize edilmiş

algoritma barındırmaktadır. OpenCV resim üzerinde çalışabilmek için resmin her pikseli 3 ana renk kanalı (Blue-Green-Red) olacak şekilde 0 ile 255 arasında değerler alan çok boyutlu matrise dönüştürülür. Bütün işlemler resmi temsil eden bu matris üzerinden gerçekleştirilir. Örneğin bir filtre uygulanacağı zaman filtre matrisi bu matris ile işleme sokularak aslında bir matris operasyonu yapılmış olur. OpenCV ile yapılabilecek işlemlere örnek olarak resimdeki nesne kenarlarının bulunması, renk tonlaması değişimi, resim üzerine yazı yazılması veya geometrik şekiller eklenmesi verilebilir [25].

2.2.2. Tesseract OCR Motoru

OCR dijital resimler üzerinde bulunan metinlerin tespit edilip düzenlenebilir metinler haline getirilmesi için kullanılan bir metottur. Tesseract OCR Motoru bu görevi yerine getirmek amacıyla oluşturulmuş bir projedir. Başlangıçta ticari olarak kullanılsa da günümüzde açık kaynak kodlu olarak geliştirilmeye devam edilmektedir. Tesseract OCR Motoru kendi sayfa düzeni analiz aracına sahip değildir. Bu sebeple bizim ona OCR işlemi yapacağı resim bölümlerini hazırlayıp vermemiz gerektiği anlamına gelmektedir. Yine metinler arasındaki bağlantıları da bizim kontrol etmemiz gerektiği anlamına gelmektedir [26]. Tesseract OCR Motoru karakterlerin bulunması ve kelimelerin oluşturulması için kaynak vektörler ve algoritmalar kullanmaktadır.

Okuma sırası görsel zengin dokümanları anlamada kritik bir öneme sahiptir. Microsoft firmasının popüler kelime işlemci programı okuma sırası bilgisini belgenin diğer nitelikleri ile bir XML dosyası içerisinde barındırmaktadır. Bu format kolaylıkla PDF veya resim dosyasına dönüştürülmektedir. Dönüşüm işlemi sonucunda dokümanın anlaşılmasında kritik önemi olan okuma sırası bilgisi kaybolmaktadır. Güncel metotlar OCR sonuçları ile çalışmaktadır. Ancak çoğu OCR sistemi karakter veya kelimeleri belge üzerinde yukarıdan aşağıya ve soldan sağa doğru taramaktadır. Bu şekilde yapılan OCR işleminde birden fazla sütunu bulunan bir belgede elde edilen sonuç belgenin doğal akışından farklı olmaktadır. Metin bloklarının yerleşimleri ve doğal dil işleme metotları kullanılması OCR sistemlere göre daha doğru sonuçlar elde edilmesini sağlayacaktır [11].

2.2.3. Makine Öğrenmesi

Makine öğrenmesi mevcut verilere, matematik ve istatistik bilimi kullanılarak geliştirilen algoritmalar ile yeni verilere yönelik çıkarım ve sınıflama işlemlerinin gerçekleştirilmesidir. Genel olarak makine öğrenmesi işlem basamakları şu şekildedir;

- Veri kümesinin oluşturulması
- Veri kümesinin ön işlem yapılması (eksik verilerin giderilmesi veya kaldırılması, aykırı değerlerin veri kümesinden çıkarılması gibi)
- Beklenen işe uygun makine öğrenmesi modellerinin belirlenip uygulanması
- Veri kümesine uygulanan modellerin değerlendirilmesi
- Sonuçlar istenilen başarı değerlerine ulaşana kadar gerekirse parametre değişiklikleri ile yeniden model eğitim işlemlerinin gerçekleştirilmesi.

Makine öğrenmesi çatısı altında 3 temel grup bulunmaktadır. Bunlar denetimli öğrenme, denetimsiz öğrenme ve yarı denetimli öğrenmedir. Bu çalışma kapsamında denetimle öğrenme metotları kullanılacağı için diğer gruplara değinilmeyecektir.

Denetimli öğrenme için etiketlenmiş veri kümesine ihtiyaç vardır. Burada temel amaç model eğitiminde verilen girdiye karşılık oluşacak çıktıyı modelin karar vermeyi öğrenmesidir. Hedef modelin daha önce görmediği yeni girdilere karşılık doğru çıktıyı tahmin edebilmesidir. Bu sebeple mümkün olan en büyük etiketlenmiş veri kümesi kullanılması ve modelin fazla durumla karşılaşması modelin doğruluğunu ve kararlığını yükseltecektir [27].

Denetimli öğrenmede modelin çıktısı sürekli değerler alıyorsa regresyon model olarak adlandırılır. Eğer çıktı sonlu ve ayrık değerler kümesi ise sınıflama modeli olduğunu söyleyebilir.

Denetimli öğrenme modelleri için en büyük zorluğun veri kümesini oluşturmak olduğu söylenebilir. Bir diğer zorluk ise gerçek hayatta etiketler için netlik olmayan muğlak durumlar olmasıdır. Ancak model çıktısının insanlar için anlamlı bir sonuç vermesi ise bir avantajdır.

2.2.3.1. K-En Yakın Komşu Algoritması

K-En Yakın-Komşu (KNN) algoritması, örnek tabanlı bir öğrenme yöntemidir. KNN, her örneğin n-boyutlu bir uzayda bir nokta ile ilgili olduğunu ve bir nitelikler dizisi olarak tanımlanabileceğini varsaymaktadır. Sınıflama işlemi tüm veri noktalarını kullanarak yeni girişi bir uzaklık işleviyle ölçerek yeni veri noktasına en yakın olan veri noktalarını temel alarak sınıflandırır. KNN ile ilgili sorun olabilecek bir durum vardır. Bu her yeni veri girişini sınıflamak için eğitim süreci dışında büyük veriler ile çalışmanın maliyetli olmasıdır [28].

2.2.3.2. Karar Ağacı Algoritması

Büyük ve çok boyutlu veri kümelerinde kurallar uygulayarak oluşturulan bir ağaç yapısıdır. Ağaç yapılarında bulunan kök ve yaprak düğümler vardır. Veriler fark yaratan önemli öğeyi temel alarak bölümlenir. Bu algoritma yüksek sınıflandırma hızına, güçlü öğrenme yeteneğine ve basit yapıya sahiptir. Ancak veri kümesinde sınıflar arasında dengesizlik olması durumunda daha fazla değere sahip sınıfı seçme eğilimindedir [29].

2.2.3.3. Destek Vektör Makinesi

Destek Vektör Makinesi (SVM) veri kümelerinde hiper düzlem tabanlı bir sınıflandırıcıdır.

SVM örüntü sınıflandırma görevleri için kullanılabilmektedir. SVM temel amacı veri kümesindeki sınıflar arasında maksimum mesafeyi temsil eden çizgiyi bulmaktadır. Bu çizginin doğrusal olmama durumu için kullanılan fonksiyonlara kernel (çekirdek) denir. Kernel fonksiyonları şunlar olabilirler; Polynomial, Gaussian, RBF (Radial basis function – radyal tabanlı işlev) ve Sigmoid. SVM'nin diğer makine öğrenmesi modellerine göre avantajlı tarafı veriyi ezberleme durumuna karşı dirençli olmasıdır [30].

2.2.3.4. Stokastik Gradyan İniş Sınıflayıcı

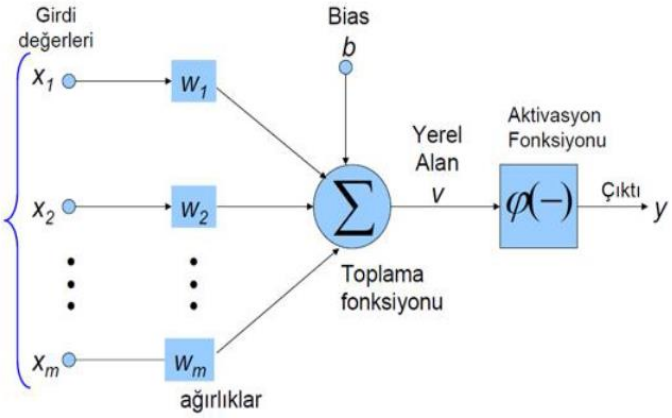
Stokastik gradyan iniş (SGD) dışbükey kayıp fonksiyonları için verimli ve güçlü bir optimizasyon yöntemdir. SGD, OvA (one-vs-all) yönteminde birden çok ikili sınıflandırıcıyı birleştirdiği için çok sınıflı sınıflandırmayı destekler [31]. Bu tahmin edici, düzenli doğrusal modellere uygular ve kayıp gradyanları her bir örnek için ayrı ayrı tahmin eder. Parametre olarak kare Öklid normu L2, mutlak norm L1 veya her ikisinin bir kombinasyonunu kullanarak model parametrelerini sıfır vektörüne doğru küçülten kayıp fonksiyonuna ekler.

SGD metin sınıflandırma ve doğal dil işlemede sıklıkla karşılaşılan büyük ölçekli ve seyrek makine öğrenimi problemlerine başarıyla uygulanmaktadır. SGD verimlidir ve uygulaması kolaydır. Ancak SGD bir dizi hiper parametre belirlemek gerekmektedir.

2.2.4. Yapay Sinir Ağları ve Derin Öğrenme

YSA insan beynindeki sinir hücrelerinden esinlenilerek geliştirilmiştir [2]. Beyin sinir hücrelerinin birbirleri ile kurdukları bağlantılar modellenerek ağ yapısı oluşturulmuştur. Sinir hücreleri aktivasyon fonksiyonları olarak adlandırılan ve sinir hücresinin tetiklenmesini belirleyen özel fonksiyonlara sahiptirler. YSA ağa girdi olarak verilen bilgilere karşılık bir çıktı üretmektedir. YSA temsili Şekil 1' de görülmektedir. Bu işlemi gerçekleştirebilmek için önce

ağın bir veri kümesi ile eğitilmesi gerekir. Eğitim sonucunda ağ yapısı içerisindeki sinir hücreleri veri kümesine uygun ağırlıkları kazanmaktadır. Sinir hücrelerinin öğrenmesi geri yayılım algoritması ile gerçekleştirilmektedir.



Şekil 1 : Örnek YSA [2]

Derin öğrenme, birden çok işlem katmanından oluşan hesaplama modellerinin, birden çok soyutlama düzeyiyle verilerin temsillerini öğrenmesine olanak tanır [32]. Bu yöntemler, konuşma tanıma, görsel nesne tanıma, nesne algılama, ilaç keşfi, genom ve diğer birçok alanda teknolojinin büyük ölçüde gelişmesini sağlamıştır. Derin öğrenme bir önceki katmandaki temsilden her katmandaki temsili hesaplamak için kullanılan dahili parametrelerini nasıl değiştirmesi gerektiğini bulmak için geri yayılım algoritmasını kullanarak büyük veri kümelerinde karmaşık yapıyı keşfeder.

Derin ağlar görüntü, video, konuşma ve ses işlemede çığır açarken, tekrarlayan ağlar metin ve konuşma gibi sıralı verilere ışık tutmuştur.

2.2.5. Doğal Dil İşleme

Doğal Dil İşleme (NLP) insan dillerinin hesaplamalı işlenmesini ve anlaşılmasını içeren konuları kapsayan bir bilim dalıdır.

NLP alanındaki çalışmalar temel ve uygulama alanı olarak iki alt bölüme ayırmak mümkündür. Temel alanlar morfolojik işleme, kelimelerin anlamlı bölümlere ayrılması ve kullanılan kelimelerin konuşmanın gerçek kısımlarının belirlenmesidir [4]. Ayrıca anlamsal işleme için cümle diyagramları oluşturan söz dizimsel işleme ve ayrıştırma yine bu alt bölümde yer almaktadır. Uygulama alanında ise diller arası çeviri, metin özetleme, soru cevap, metin sınıflaması gibi konuları içermektedir.

1980’li yıllardan günümüze bu alanda istatistik, olasılık ve makine öğrenmesini de içeren veriye dayalı hesaplamalar kullanılmıştır. Hesaplama gücü ve paralelleştirmede grafik işlem birimleri (GPU) kullanılmasıyla birlikte geleneksel yöntemlere göre kıyas kabul etmeyecek büyüklükte eğitilebilir parametreler ile çalışan YSA kullanımı giderek popülerleşmiştir.

Naive Bayes, K-En Yakın Komşular, Gizli Markov Modelleri, Koşullu Rastgele Alanlar (CRF), Karar Ağaçları, Rastgele Ormanlar ve Destek Vektör Makineleri gibi makine öğrenme yaklaşımları yaygın olarak kullanılmıştır. Son birkaç yılda bu yaklaşımlar yerine ağırlıklı olarak YSA modellere geçiş yapılmıştır [4].

YSA, her biri belirli sayıda girdi alan ve bir çıktı sağlayan birbirine bağlı düğümlerden veya nöronlardan oluşur. Çıktı katmanlarındaki düğümlerin her biri, girdi düğümlerinden aldıkları değerler üzerinde ağırlıklı toplam hesaplaması yapar ve daha sonra bu toplamlar üzerinde basit doğrusal olmayan dönüşüm fonksiyonlarını kullanarak çıktılar üretir. Ağırlıklardaki güncellemeler, ağların çıkış düğümlerinde sergilediği hatalara veya kayıplara yanıt olarak gerçekleştirilir. Bu tür güncellemeler geri yayılım olarak adlandırılan bir yaklaşım olan düğümlerdeki hataların türevleri hesaplanarak bulunan stokastik gradyan inişi kullanılarak yapılır. Farklı ağ türlerini birbirinden ayıran temel faktörler, düğümlerin nasıl bağlandığı ve katman sayısıdır. Bir derin sinir ağı çok sayıda gizli katmana sahip ağlar derin olarak kabul edilir ve çok katmanlı ağlar çok derin olarak kabul edilir.

Yoğun olarak kullanılan bir tekrarlamalı sinir ağı türü, Tekrarlayan Sinir Ağıdır (RNN). NLP'nin çoğu kelimelerin sırasına veya fonemler veya cümleler gibi diğer öğelere bağlı olduğundan, yenilerini işlerken önceki öğelerin hafızasına sahip olması gerekmektedir. Bu nedenle, iki RNN katmanını kullanarak ve çıktılarını birleştirerek cümlelere ileriye ve geriye doğru bakılmaktadır. Birden fazla RNN katmanı varsa daha iyi bir temsile yol açmaktadır. Bu durum bir girdinin etkisinin tek bir RNN katmanından daha uzun süre kalmasını sağlayarak daha uzun vadeli etkilere izin vermektedir. Yüksek düzeyde hafıza ile tasarlanmış bir RNN, uzun kısa süreli bellek (LSTM) olarak adlandırılmaktadır. LSTM'lerde, özyinelemeli düğümler, belirli bilgileri saklamak, unutmak veya açığa çıkarmak için tasarlanmış bir şekilde bağlanan nörondan oluşmaktadır [4]. Genellikle uzak geçmişten gelen bilgileri hatırlamak önemlidir ama aynı zamanda diğer yeni bilgiler önemli olmayabilir. LSTM blokları kullanılarak bu önemli bilgiler çok daha uzun süre tutulabilirken alakasız bilgiler unutulabilmektedir.

Bir NLP modelin değerlendirmesinde verilerdeki yüksek değişkenlik ve her bir görevdeki performansı değerlendirmek için kullanılan çok sayıda metrik vardır. Çoğu zaman benzer modelleri karşılaştırmak bile zordur çünkü her biri için farklı metrikler kullanılarak rapor edilmektedirler [4]. Ortak bir metrik seti üzerinde mutabık kalınması ve sahada net karşılaştırmalar sağlanması şimdilik mümkün görünmemektedir.

Ayrıca, metrikler genellikle yalnızca en iyi sonuçlar için rapor edilmektedir. Ortalama durumlar ve değişkenlik gösteren veya en kötü durumlardan bahsedilmemektedir [4].

2.2.6. Metin Sınıflama İşlemi

Makine öğrenmesi veya derin öğrenme metotları fark etmeksizin geliştirilecek NLP modeli için veri kümesinin oluşturulması gerekmektedir. Doğal Dil İşleme'de (NLP), metin ve belgelerin çoğu, durak sözcükleri, yanlış yazımlar, argolar vb. gibi metin sınıflandırması için geliştirilen uygulamada kullanılmayacak olan birçok sözcük içerir. Bu sebeple veri kümesi içerisinde temizlik ve ön işleme yapılması gerekmektedir. Bu aşamalar şu şekildedir

Tokenizasyon, bir metni kelimelere, sembollere veya diğer anlamlı öğelere ayırma işlemidir. Bu adımın temel amacı, bir cümledeki kelimeleri tek tek çıkarmaktır.

Durak Kelimeler (Stop Words), cümle içerisinde olan ama bazı dillerde bir anlam katmayan kelimelerdir ve veri kümesinden çıkarılmalıdır. İngilizce dili için ‘the’, ‘a’, ‘is’ gibi kelimeler Durak Kelimelere örnek gösterilebilir.

Küçük harfe dönüştürme, veri kümesindeki karmaşıklığı azaltmak için yapılması gereken bir diğer işlemde veri kümesindeki tüm metinleri küçük harfe dönüştürmektir. Bizim için anlamı olan büyük küçük harf model eğitimi için bir önem arz etmemektedir.

Gürültü giderme (Noise Removal), veri kümesinde noktalama işaretleri ve özel karakterler içerebilir. Bunlar metin sınıflama için bir anlam ifade etmemektedir ve çıkarılmaları gerekmektedir.

Kelime Gömme (Word Embedding) ve Kelime Çantası (Bag of Words), yukarıda yapılan işlemler sonrasında işlenmiş her bir elemana unigram denir. Unigramların makine öğrenimi algoritmaları için kullanılabilir girdilere dönüştürmek için farklı kelime temsil prosedürleri önerilmiştir [33].

- Term Frequency-Inverse Document Frequency (TF-IDF) bir kelimenin bulunduğu metni ne kadar temsil ettiğini göstermektedir. TF ile kelimenin metinde geçme sıklığıdır. DF tüm veri kümesindeki metin sayısına o kelimenin geçtiği metin sayısının bölünmesi ile bulunur. DF değerinin logaritması alınarak IDF değeri bulunur. TF ve IDF değerleri çarpılarak hedef değer elde edilmiş olur.

- GloVe (Global Vectors for Word Representation) kelimeler için vektör temsilleri elde etmek için denetimsiz bir öğrenme algoritmasıdır [33]. Eğitim veri kümesinde bir metinden toplanan küresel kelime-kelime (bigram) birlikte olma istatistikleri üzerinde gerçekleştirilir. Projenin kendi web sitesi üzerinden ön eğitimli model indirilebileceği gibi başka bir veri kümesi üzerinde eğitim gerçekleştirilmesi mümkündür.

Bahsedilen iki kelime temsil vektörleri dışında başka kelime çalışmaları da mevcuttur ancak bu çalışmada kullanılmadıkları için bahsedilmemiştir.

2.2.7. Metin Sınıflama İçin Metrikler

Makine öğrenmesinde modelin değerlendirilmesi sistemin başarımını görmek ve iyileştirmek için anahtar rol oynamaktadır. Sınıflandırma sürecinde eğitim aşaması, doğrulama aşaması ve test aşaması vardır. Veri kümesinin belli bir bölümü modelin eğitim aşamasında kullanılarak model eğitilir. Eğitim aşamasında oluşan hatalar modelin eğitim verilerine ne kadar iyi uyduğunu ölçer. Model eğitim verilerine göre ayarlandığı için başarı doğrulama değerlendirmesine göre yüksek çıkabilmektedir. Doğrulama aşamasında model eğitildikten sonra modelin veri kümesinde görmediği veriler ile değerlendirme yapılır. Test aşaması veri kümesinin dışında veriler kullanılarak gerçekleştirilir. Bu aşamanın değerlendirilmesi daha muğlaktır çünkü bu veriler sınıf etiketleri veya çıktıları bilinmemektedir. Bir sınıflandırıcı için değerlendirmede kullanılan en yaygın metot karmaşıklık matrisidir.

Karmaşıklık matrisi veri kümesindeki çıktılar ile modelin tahmin ettiği çıktılardan oluşmaktadır. Örnek olarak ikili bir sınıflandırıcı için 2x2'lik bir karmaşıklık matrisi ile 4 olası çıktı gösterilmektedir. Veri kümesinden ki veri doğru ve tahminde pozitif ise doğru pozitif (TP) olarak ifade edilir. Eğer tahmin yanlış sınıflandırılmış ise yanlış negatif (FN) olarak ifade edilir ve bu durum tip II hata olarak adlandırılır. Veri kümesinde çıktı yanlış ve tahmin pozitifi ise doğru negatif (TN) olarak ifade edilir. Eğer tahmin pozitif sınıflandırılmış ise yanlış pozitif (FP) olarak ifade edilir ve tip I hata olarak adlandırılır. Karmaşıklık matrisi birçok yaygın sınıflandırma metriğini hesaplamak için kullanılmaktadır [34].

Tablo 1 : Karmaşıklık Matrisi

		Doğru Sınıflar	
		Pozitif (P)	Negatif (N)
Tahmin Edilen Sınıflar	Doğru (T)	Doğru Pozitif (TP)	Yanlış Pozitif (FP)
	Yanlış (F)	Yanlış Negatif (FN)	Doğru Negatif (TN)

Doğruluk (Acc), sınıflandırma performansı için en sık kullanılan ölçülerden biridir ve doğru sınıflandırılmış çıktının veri kümesindeki toplam çıktı sayısına oranıdır.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Duyarlılık ya da hassasiyet (Recall), doğru pozitif sınıflandırılmış çıktıların toplam pozitif çıktı sayısına karşılık gelir.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Kesinlik (Precision) doğru pozitif sınıflandırılmış çıktıların, doğru pozitif ve yanlış pozitif toplam çıktı sayısına oranına karşılık gelir.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

F1 skoru ve kesinlik ve duyarlılığın harmonik ortalamasıdır. F1 skoru değeri sıfırdan bire değişir ve F skoru yüksek değerleri, yüksek sınıflandırma performansını gösterir.

$$F1\ skoru = 2\ x\ \frac{Kesinlik\ x\ Hassasiyet}{Kesinlik + Hassasiyet} \tag{2.4}$$

Bir metrik olarak değerlendirilmese de insanların karbon ayak izi dünyamızı korumak için önemsenen bir konudur. Geçmişten günümüze NLP gelişim sürecinde çok yol kat edilmiştir. Zaman içerisinde kural tabanlı sistemlerden makine öğrenmesi kullanan sistemlere geçilmiştir. Günümüzde daha çok derin öğrenme modelleri kullanılmaktadır. Ancak bu işin hem işlem yükü hem de karbon ayak izi nedeniyle maliyetleri artmıştır. Derin öğrenme için kullanılan modellerin eğitiminde GPU ya da TPU (Tensor Processing Unit) fark etmeksizin yüksek işlem

gücü olan donanımlar kullanılmaktadır. Bu donanımların enerji tüketimleri de fazladır. İster yerel olarak ister bulut bilişim olarak modelin eğitilmesi ve geliştirilmesinde bu durum değişmemektedir. Ayrıca derin öğrenme ile ilgili her yenilikte kullanılan işlem gücü, haliyle karbon ayak izi ve de model boyutu artmaktadır. Yapılan bir deney çalışmasında Google tarafından geliştirilen BERT modelinin eğitimi kıtalar arası bir uçuş kadar emisyon üretmektedir [35].

Büyük boyutlu modellerin yerel olarak kullanılması bir problemdir. Bu sebeple bulut hizmet olarak hizmet verilmektedir. Özellikle mobil cihazların disk kapasiteleri ve işlem güçleri düşünüldüğünde uzun vadede daha basit çözümlerin üretilmesi gerektiği görülmektedir.

2.2.8. LaTex Kodu Yazmak

LaTex ile çalışmaya başlarken LaTex ortamına ihtiyaç vardır [36]. Çoğunlukla LaTex kodu yazmaya başlamak için sisteme eklenecek editör ortam ihtiyacını sağlamaktadır. Her işletim sistemi için bir LaTex editörü mevcuttur. Bu editörlerin kendilerine göre avantajları vardır. Örneğin bazı editörler yazılan kodların çıktısını gerçek zamanlı olarak gösterebilirken bazıları gösterememektedir. Bazı editörler kod yazdırmak yerine Microsoft'un kelime işlemci programı Word gibi görsel olarak yazım işleminin gerçekleştirilmesini sağlamaktadır. Bu çalışmada editörlerin bize sağlayacağı kod yazma kolaylıkları gibi özellikleri önemsizdir. Sadece yazılan LaTex kodunun sonucunun görülmesi ve PDF çıktısının verilmesi yeterlidir. Farklı bir çözüm olarak yazılan LaTex kodlarına ortak geliştirme ortamı sağlayan bulut sistemler giderek yaygınlaşmaktadır.

LaTex açık kaynak yapısı gereği geniş bir geliştirici topluluğuna sahiptir. Paketler sayesinde birçok yeni özellik sisteme entegre edilebilmektedir. Farklı yazı karakterleri yine paketler olarak geliştiriciler tarafından web sitesi [6] üzerinden yayınlanmaktadır. LaTex'in kullanımındaki bir diğer rahatlık ise hazır stil dosyaları kullanarak yazdığımız yazının ön tanımlı biçimlendirme ile otomatik olarak çıktısını alabiliyor olmamızdır. Bu avantajı formatın önemli olduğu akademik çevreler ve profesyonel iş ortamlarında kendisine yer bulmasını sağlamıştır.

LaTex kod yapısı HTML gibi bir işaretleme diline benzetilebilir. HTML kodları düz metin olarak yazılırlar ve internet tarayıcılarında yorumlanarak komutların sonuçları çıktı olarak görüntülenmektedir. LaTex'in kendine özgü bir kod yapısı vardır. Bu komutlara ait bazı özellikler şu şekildedir:

- Komutlar büyük küçük harf duyarlıdırlar.

- #, $, %, ^, &, _, {, }, ~, \ karakterlerinin özel anlamları olduğu için içerikte direkt olarak kullanılamazlar. "\" karakteri başlarına eklenerek kullanılmaları gerekmektedir. Bu karaktere kaçış karakteri denilmektedir.

- Her komut "\" karakteri ile başlar. Sonrasında komut adı yazılır. Eğer komut parametre veya parametreler alıyorsa süslü parantezler içerisinde virgülle ayrılarak yazılmaktadır.

- Komutların bir kısmının içerikte birden fazla satırda geçerli olması için begin-end blokları kullanılması gerekmektedir.

Yapısal olarak LaTex kodu iki ana kısımdan oluşuyor diyebiliriz.

- Preamble: burada belge tipi, belge sütun sayısı, kenar boşlukları gibi tüm belgeye uygulanacak ayarlamalar için LaTex kodları yazılır. Ayrıca gerekliyse yazar ve ana başlık bilgileri için kodları burada yazılmaktadır. Yine belge içerisinde kullanılacak paketler de burada tanımlanır.

- Body: belgemizin içeriği burada yazılır. Burada LaTex komutları ile belgemizi inşa ederiz. Şekil eklemek, liste oluşturmak, tablo eklemek ve denklem yazmak gibi işlemler için LaTex projesinin resmî web sitesi veya konu ile ilgili yazılmış kaynak kitaplardan faydalanmak mümkündür.

3. YÖNTEM

Bu bölümde uygulamanın geliştirilme aşamaları anlatılacaktır. Bu aşamalar veri kümesinin elde edilmesi, veri ön işleme yapılması, gerekli makine öğrenmesi ve derin öğrenme modellerinin eğitilmesi, metin / şekil sınıflamasının yapılması, okuma sırasının bulunması, sayfa tipinin bulunması, metin sınıflaması yapılması ve LaTex kodunun üretilmesidir.

3.1. VERİ KÜMESİ

3.1.1. Veri Kümesinin Elde Edilmesi

Çalışmada kullanılacak veri kümesi için yapılan internet taramalarında yapılacak çalışma için uygun bir veri kümesi bulunamamıştır. Bu sebeple konu ile ilgili yapılan literatür taraması esnasında web ortamında ulaşılan farklı kaynaklardan (IEEE xplore [37], Science Direct [38], ArXiv [39], Springer [40] ve ACM DL [41]) 103 tane PDF formatında yayından faydalanılmıştır.

Geliştirilecek sistemin eğitimi için toplanan 103 PDF öncelikle her sayfası ayrı bir resim olacak şekilde resim formatına dönüştürülmüştür. Dönüşüm sırasında resim dosyaları isimlendirilirken kaynak PDF'in adının arkasına tarih ve en sonda kaynak dokümanın kaçıncı sayfa olduğu bilgisi eklenmiştir.

Resim dosyaları ile makine öğrenmesi ve NLP'de kullanılacak derin öğrenme modelleri diret olarak beslenemeyecekleri için içerisindeki bilgilerin çıkarılması gerekmektedir. Bu bilgiler metin, başlık, liste, tablo veya resim bloklarının konumları ve blokların içerdikleri metinlerdir. Blokların sınıflarının bulunması için tek tek etiketlemek yerine yine derin öğrenme metodu kullanılan "LayoutParser" kütüphanesinden faydalanılmıştır. Bu kütüphane ile her bir resmin bloklarının yeri ve sınıfları bulunmuş final işlemi olarak OCR kullanılarak içerdikleri metinlerde bulunarak CSV formatında kaydedilmiştir. Veri kümesinin kaydedilmesi için CSV dosya formatının kullanılması veri ön işleme için kullanılacak kütüphaneler ile tamamen uyumlu olması ve az yer kaplamasıdır.

CSV formatında veriler sütunlara virgül, noktalı virgül veya farklı özel karakterler ile ayrılabilmektedir. Bir diğer özelliği CSV dosyaları herhangi bir font büyüklüğü, font ağırlığı veya farklı bir özellik depolamadığı için diskte az yer kaplamaktadır.

Veri kümesi oluşturulurken "LayoutParser" kullanılmasının bir diğer sebebi tüm blokları yakalayamadığı halde bulduğu blokların yüksek doğrulukta olmasıdır. Veri kümesi için kullanılan 1141 resimden 11965 blok çıkartılmıştır. Google Colabratory [42] kullanılarak yapılan çalışma yaklaşık olarak 10 saat kadar sürmüştür.

CSV dosyasına kaydedilen bilgiler "LayoutParser" standart formatı ile değil daha sonra geliştirilecek olan segmentasyon sistemine uygun olarak ve kaynak resim adı, sayfa boyutu gibi gerekli bilgiler eklenmiştir. Aşağıda örnek çıktı görülmektedir.

Tablo 2 : Veri Kümesi Örneği

İndeks	Kutu İndeksi	Kutu X1	Kutu X2	Kutu Y1	Kutu Y2	Kutu Metni	Kutu Tipi	Sayfa Boyutu	Sayfa Numarası
11952	0	138	839	1223	1615	In the first sessio...	Text	2200x1700	2
11953	1	135	836	1632	1840	The first session...	Text	2200x1700	2
11954	2	133	836	648	1140	2020, 4TIgX was...	Text	2200x1700	2
11955	3	134	313	1171	1201	A. First lecture\n	Title	2200x1700	2
11956	4	118	1598	539	592	Fig. 1. (a) Use of ...	Text	2200x1700	2
11957	5	136	838	1988	2042	Fig. 2. Use of the...	Text	2200x1700	2
11958	6	123	841	1989	2013	Fig. 2. Use of the...	Text	2200x1700	2
11959	7	168	849	330	474	\begin{figure}\n\...	List	2200x1700	2
11960	8	913	1513	444	473	Fig. 1. Logo of...	Text	2200x1700	2
11961	9	879	1590	649	1251	then introduced...	Text	2200x1700	2
11962	10	885	1090	1312	1342	B. Second lecture\n	Title	2200x1700	2
11963	11	882	1591	1365	1516	A brief review of...	Text	2200x1700	2
11964	12	881	1588	1537	1813	By using the...	Text	2200x1700	2
11965	13	886	1591	1835	2049	Naturally, the ...	Text	2200x1700	2

3.1.2. Veri Ön İşleme

Veri kümesini ihtiyacımıza hazır hale getirmek için ön işlem yapılması gerekmektedir.

3.1.2.1. Blok Alanı ve Sayfa Alanının Bulunması

Daha sonra geliştirilecek sistemde şekil / metin ayrımını yapabilmek için veri kümesinde bloğun kapladığı alanın resmin tüm alanına oranı ve bloğun içerdiği kelime sayısına ihtiyaç vardır.

Öncelikle dikdörtgen olarak koordinatları ifade eden bloğun gerekli sütunlarından genişlik ve yükseklik hesaplanarak veri kümesi tablosuna yeni bir sütün olarak eklenmiştir.

Sonra veri kümesinden sayfa genişlik ve yüksekliği kullanılarak sayfa alanı hesaplanmıştır.

Son olarak bloğun yüzdesel olarak sayfada kapladığı alan bulunmuştur ve veri kümesine eklenmiştir. Geliştirme aşamasında blokların konumları ve diğer gereksiz bilgiler atılarak veri kümesi elde edilmiştir.

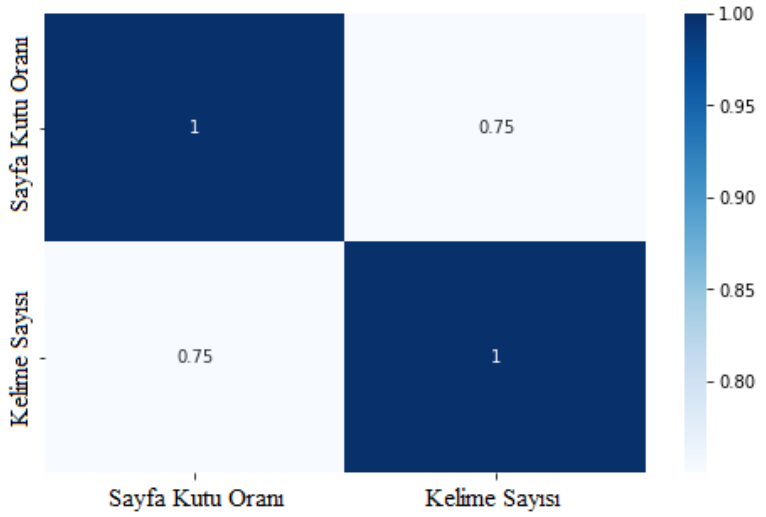
Elde edilen veri kümesine temel veri analizi yapıldığında aşağıdaki tablo elde edilmiştir.

Tablo 3 : Metin / Şekil Sınıflaması için Veri Kümesi Analizi

Blok Tipi	Sayı	Sayfa Blok Oranı (%)	Blok Kelime Sayısı
Metin	8405	4,11	55,36
Başlık	1873	0,51	3,8
Şekil	795	12,47	31,3
Liste	680	12,91	149,54
Tablo	213	9,96	59,35

Aykırı değer aranmış ama bulunmamıştır. Kayıp veri aranmış ama bulunmamıştır. Blok tipleri şekil ve metin (liste, tablo, metin ve başlık birleştirildi) olacak şekilde düzenlenmiştir. Çalışma sırasında tablo ayrı bir sınıf olarak yapılan deneyler başarısız olmuştur.

Veri kümesinde uygun olan özelliklere normalizasyon yapılarak ön işleme tamamlanmıştır. Ön işleme sonunda elde edilen veri kümesinin korelasyon matrisi görüntüsü Şekil 2'de verilmiştir. Şekil 2'de korelasyon matrisine bakıldığında kutunun sayfada kapladığı alan ile kelime sayısı arasında pozitif korelasyon olduğu görülmektedir.



Şekil 2 : Metin / Şekil Veri Kümesi için Korelasyon Matrisi

3.1.2.2. Metin Bloklarının Kaç Kelime Olduğunun Bulunması

Bloklar içerisindeki metinler OCR işleminden kalan anlamsız karakterler ve noktalama işaretleri gibi kelime sayısı için önemli olmayan karakterler içermektedir. Doğru bir kelime sayısına ulaşabilmek için bunların temizlenmesi gerekmektedir. Ardından metnin kelimelere ayrılması işlemi için NLTK kütüphanesi kullanılmıştır.

Öncelikle metin içerisinde A'dan Z'ye harfler ve 0 ile 9 arasındaki rakamlar dışında kalan karakterler bir fonksiyon ile temizlenmiştir. Sonrasında NLTK [43] kütüphanesi ile metinlerin kelime sayıları bulunmuştur. Son olarak bloğun sayfa alanına oranı ve kelime sayısı kullanılarak bir parametre elde edilerek veri kümesine eklenmiştir.

Tablo 4 : Metin / Şekil Veri Kümesi

İndeks	Kutu Metni	Kutu Tipi	Sayfa Numarası	Sayfa Kutu Oranı	Kelime Sayısı	Kutu Alanı Kelime Oranı
95	Table 1 summarizes...	Text	12	5,80	82	0.07073
96	\n\n \n\n \n\nfigure\n...	Figure	12	7,43	15	0.49533
97	We demonstrate the...	Text	12	2,51	35	0.07171
98	Fig. 5: Examples of...	Text	12	2,58	32	0.08062
99	4.4 Quantitative R…	Title	12	0,34	3	0.11333

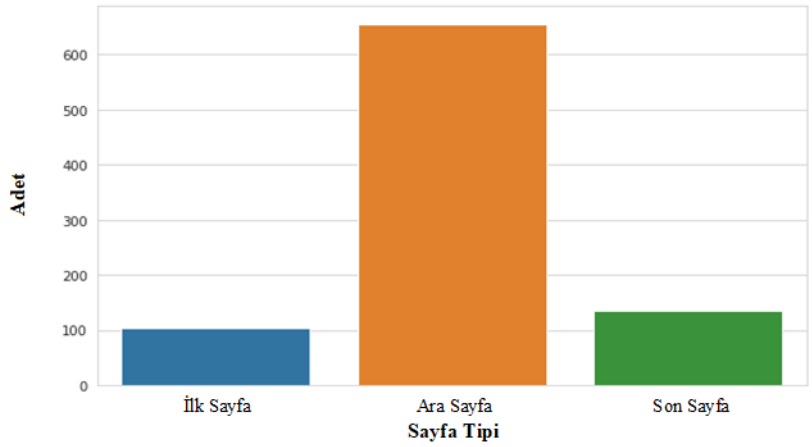
Bu işlem sonrası veri kümesi kaydedilerek segmentasyon için hazır hale getirilmiştir.

3.1.2.3. Sayfa Tipi Belirlenmesi için Kullanılacak Veri Kümesi Ön İşleme

Akademik yayınların yapısal olarak tam bir standardı yoktur ama belli ortak yapılar mevcuttur. Örneğin akademik bir yayının ilk sayfası bir ana başlık ve özet bölümü içermelidir. Ancak anahtar kelimeler her yayın formatında bulunmamaktadır. Bir diğer örnek olarak son sayfa referans bilgisini içermelidir. Bu farklılıklardan dolayı dijital dokümanın her sayfası için aynı metin sınıflama modelini kullanmak yerine dijital dokümanın sayfaları için üç farklı sınıf belirlenmiştir. Bunlar ilk sayfa, ara sayfa ve son (referans) sayfa olmuştur. Sayfa tipi belirlendikten sonra o sayfa tipine uygun metin sınıfları için geliştirilecek modeller kullanılacaktır. Bu sebeple oluşturulan veri kümesinde başlangıçta elde edilen veri kümesinden seçim yapılmıştır. Sonrasında tüm sayfaya OCR işlemi uygulanarak veri kümesi oluşturulmuştur.

Veri kümesindeki metinlerin özel karakter ve sayılardan arındırılması için düzenli ifadeler kullanılmıştır. Yine Durak Kelimeler olarak adlandırılan ve cümleye anlam katmayan kelimelerin çıkarılması için NLTK kütüphanesi kullanılmıştır. Bahsedilen işlemler bir fonksiyon haline getirilmiştir. Bu fonksiyon veri kümesi içerisindeki metin sütununa uygulanarak gerekli temizleme işlemi gerçekleştirilmiştir.

Veri kümesinde toplam 893 satır bulunmaktadır ve sınıfların dağılımı Şekil 3'te görülmektedir.



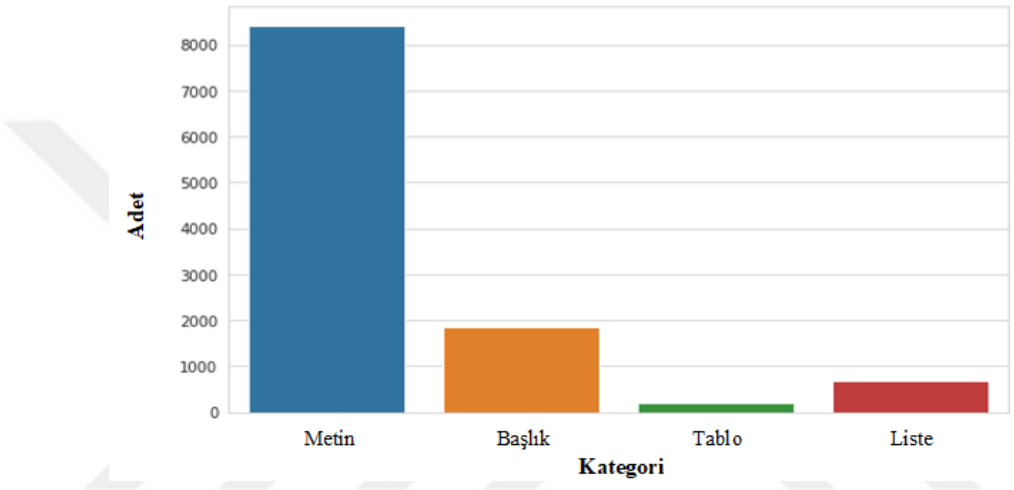
Şekil 3 : Sayfa Tipi Belirleme Veri Kümesi Sınıf Dağılımı

3.1.2.4. Ara Sayfa için Veri Kümesi Ön İşleme

Öncelikle veri kümesinden blok tipi şekil olan kayıtlar çıkarılmıştır. Bu aşamada o sınıf ile ilgili işlem yapılmayacaktır. Sonrasında blok tipleri sayısal kodlamaya dönüştürülmüştür.

Veri kümesindeki metinlerin özel karakter ve sayılardan arındırılması için düzenli ifadeler kullanılmıştır. Yine Durak Kelimeler olarak adlandırılan ve cümleye anlam katmayan kelimelerin çıkarılması için NLTK kütüphanesi kullanılmıştır. Bahsedilen işlemler bir fonksiyon haline getirilmiştir. Bu fonksiyon veri kümesi içerisindeki metin sütununa uygulanarak gerekli temizleme işlemi gerçekleştirilmiştir.

Veri kümesinde toplam 11171 satır bulunmaktadır ve sınıfların dağılımı Şekil 4'te görülmektedir.



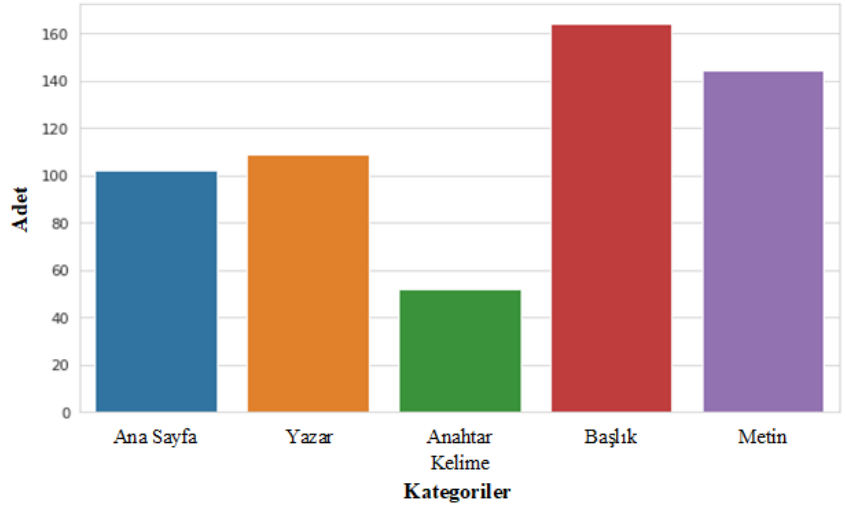
Şekil 4 : Ara Sayfa Veri Kümesi Sınıf Dağılımı

3.1.2.5. İlk Sayfa için Veri Kümesi Ön İşleme

İlk sayfa için veriler tek tek etiketlenerek elde edilmiştir.

Veri kümesindeki metinlerin özel karakter ve sayılardan arındırılması için düzenli ifadeler kullanılmıştır. Yine Durak Kelimeler olarak adlandırılan ve cümleye anlam katmayan kelimelerin çıkarılması için NLTK kütüphanesi kullanılmıştır. Bahsedilen işlemler bir fonksiyon haline getirilmiştir. Bu fonksiyon veri kümesi içerisindeki metin sütununa uygulanarak gerekli temizleme işlemi gerçekleştirilmiştir.

Veri kümesinde toplam 571 satır bulunmaktadır ve sınıfların dağılımı Şekil 5' te görülmektedir.



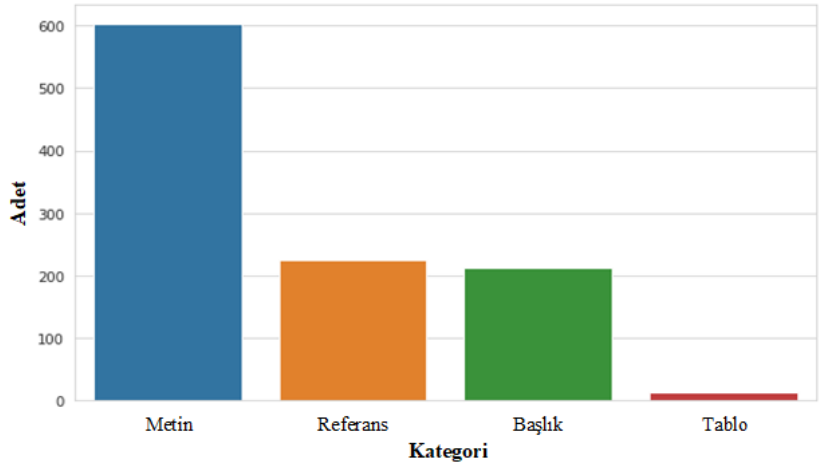
Şekil 5 : İlk Sayfa Veri Kümesi Sınıf Dağılımı

3.1.2.6. Son Sayfa veya Referans Sayfası Veri Kümesi Ön İşleme

Son sayfa veya referans sayfası için veri kümesi "LayoutParser" kütüphanesi ve manuel olarak etiketleme birlikte yapılarak oluşturulmuştur.

Veri kümesindeki metinlerin özel karakter ve sayılardan arındırılması için düzenli ifadeler kullanılmıştır. Yine Durak Kelimeler olarak adlandırılan ve cümleye anlam katmayan kelimelerin çıkarılması için NLTK kütüphanesi kullanılmıştır. Bahsedilen işlemler bir fonksiyon haline getirilmiştir. Bu fonksiyon veri kümesi içerisindeki metin sütununa uygulanarak gerekli temizleme işlemi gerçekleştirilmiştir.

Veri kümesinde toplam 1053 satır bulunmaktadır ve sınıfların dağılımı Şekil 6'da görülmektedir.



Şekil 6 : Son Sayfa veya Referans Sayfası Veri Kümesi Sınıf Dağılımı

3.2. MODEL EĞİTİMLERİ

Bu çalışma kapsamında farklı görevleri yerine getirmek için farklı model eğitimleri gerçekleştirilmiştir. Bu bölümde modellerin eğitimleri anlatılacaktır.

3.2.1. Metin / Şekil Sınıflaması Modelinin Eğitilmesi

Sayfa tipinin belirlenmesi için hazırlanan veri kümesi eğitim için %80 ve test için %20 olacak şekilde ayrılmıştır. Çalışma için denenen algoritmalar ve veri kümesi test bölümüne uygulanan tahmin sonucu oluşan karışıklık matrisleri Tablo 5'te verilmiştir.

Tablo 5 : Metin / Şekil Sınıflaması Model Eğitim Sonuçları

Sınıflama Modeli	Doğruluk		Karışıklık matrisi	
			Şekil	**Metin**
Decision Tree	0.9736	**Şekil**	133	29
		Metin	34	2198
			Şekil	**Metin**
Linear SVM	0.9690	**Şekil**	103	59
		Metin	15	2217
			Şekil	**Metin**
KNN	0.9803	**Şekil**	138	24
		Metin	23	2209
			Şekil	**Metin**
Naive Bayes	0.9294	**Şekil**	33	129
		Metin	40	2192
			Şekil	**Metin**
Quadratic Discriminant Analysis	0.9548	**Şekil**	92	70
		Metin	38	2194
			Şekil	**Metin**
Random Forest	0.9791	**Şekil**	133	29
		Metin	21	2211
			Şekil	**Metin**
Ada Boost	0.9757	**Şekil**	123	39
		Metin	19	2213

Makine öğrenmesi modelleri sonuçlarına bakıldığında en başarılı model KNN olmuştur.

Keras framework kullanılarak 354 eğitilebilir parametre, "adam" optimizer ve 100 devir (epoch) eğitilen model başarısız olmuştur. Bu sebeple yapılan bu çalışmadan bahsedilmeyecektir.

Veri kümesinde şekil sınıfı dışında belirlenen verilerin metin sınıflarının belirlenmesi gerekmektedir. Bu işlem için belirlenen sayfa tipine özel model uygulanması gerekmektedir. Üç farklı sayfa tipi düşünülerek farklı modeller geliştirilmiştir.

3.2.2. Sayfa Tipinin Sınıflaması Modelinin Eğitilmesi

Sayfa tipinin belirlenmesi için hazırlanan veri kümesi eğitim için %80 ve test için %20 olacak şekilde ayrılmıştır. Sonrasında model eğitiminde kullanabilmek için veriler kelime temsilini gösteren vektöre dönüştürülmüştür.

Model eğitimi aşamasında üç farklı çalışma yapılmıştır.

Scikit Learn kütüphanesi hızlı ve basit bir şekilde modeller eğiterek değerlendirme yapmamızı sağlayan açık kaynak bir kütüphanedir. Bu kütüphanenin "GridSearch" metodu kullanılarak önden tanımlaması yapılmış makine öğrenmesi modellerinin sırası ile eğitilmektedir. Sonuçlardan en iyi modelin bulunması ve diğer modellere göre değerlendirilmesini kolayca sağlamaktadır.

"GridSearch" metoduna parametre olarak verilecek makine öğrenmesi modelleri şunlardır; KNeighborsClassifier (K En Yakın Komşu), DecisionTreeClassifier (Karar Ağacı), MultinominalNB (Çok Terimli Naive Bayes), DecisionTreeClassifier (Doğrusal Destek Vektör Makinesi), SVM (Polinomsal Destek Vektör Makinesi), RandomForestClassifier (Rastsal Ağaç) ve SGDClassifier (Stokastik Gradyan İniş). Eğitilen modellerin doğruluk değerleri Tablo 6'da verilmiştir.

Tablo 6 : Sayfa Tipi Sınıflaması için Eğitilen Modellerin Sonuçlarının Karşılaştırması

Model	Doğruluk
KNeighborsClassifier	0.865922
DecisionTreeClassifier	0.865922
MultinominalNB	0.921788
DecisionTreeClassifier	0.955307
SVM	0.944134
RandomForestClassifier	0.910615
SGDClassifier	0.966480

Sonuçlara bakıldığında SGDClassifier en başarılı model olmuştur. Bu modele ait metrikler Tablo 7'de verilmiştir.

Tablo 7 : Sayfa Tipi Sınıflaması için Eğitilen SGDClassifier Değerlendirmesi

	Kesinlik	Duyarlılık	f1-Skoru	Örnek Sayısı
İlk Sayfa	0.94	0.84	0.88	19
Ara Sayfa	0.96	0.99	0.98	131
Son Sayfa	1.00	0.93	0.96	29
Doğruluk			0.97	179
Makro Ortalama	0.97	0.92	0.94	179
Ağırlıklı Ortalama	0.97	0.97	0.97	179

Modele ait karmaşıklık matrisi Tablo 8'de görülmektedir.

Tablo 8 : Sayfa Tipi Sınıflaması için Eğitilen SGDClassifier Karmaşıklık Matrisi

		Tahmin		
		İlk Sayfa	Ara Sayfa	Son Sayfa
Gerçek	İlk Sayfa	16	3	0
	Ara Sayfa	1	130	0
	Son Sayfa	0	2	27

Sonuçlar incelendiğinde eğitilen modelin yeterli düzeyde başarı sağladığı görülmektedir.

Yukarıda anlatılan çalışmaya ek olarak Tensorflow framework ve GloVe kullanılarak eğitilen model başarısız olmuştur. Ayrıca Keras framework ile geliştirilen modelde başarısız olmuştur. Bu sebeple yapılan bu çalışmalardan bahsedilmeyecektir.

3.2.3. Ara Sayfa Metin Sınıflaması Modelinin Eğitilmesi

3.2.3.1. Scikit Learn Kütüphanesi ile Geliştirilen Metin Sınıflama Modeli

Scikit Learn kütüphanesinin bize sunduğu metin sınıflama metotlarından 7 tanesi ile elde edilen doğruluk değerleri Tablo 9'da verilmiştir.

Tablo 9 : Ara Sayfa Metin Sınıflaması için Eğitilen Modellerin Karşılaştırması

Model	Doğruluk
KNeighborsClassifier	0.816555
DecisionTreeClassifier	0.840268
MultinominalNB	0.810738
DecisionTreeClassifier	0.868456
SVM	0.876510
RandomForestClassifier	0.854586
SGDClassifier	0.861298

Tablo 9'da görüldüğü üzere en başarılı model SVM olmuştur. Bu modele ait detaylı analiz değerleri Tablo 10'da verilmiştir.

Tablo 10 : Ara Sayfa Metin Sınıflaması için Eğitilen SVM Değerlendirmesi

	Kesinlik	Duyarlılık	f1-Skoru	Örnek Sayısı
Metin	0.88	0.98	0.92	1680
Başlık	0.88	0.66	0.75	389
Liste	0.86	0.40	0.54	124
Tablo	0.93	0.31	0.46	42

Doğruluk			0.88	2235
Makro Ortalama	0.89	0.58	0.67	2235
Ağırlıklı Ortalama	0.88	0.88	0.86	2235

Yine bu modele ait karmaşıklık matrisi Tablo 11'de görülmektedir.

Tablo 11 : Ara Sayfa Metin Sınıflaması için Eğitilen SVM Karmaşıklık Matrisi

		Tahmin			
		Metin	Başlık	Liste	Tablo
Gerçek	Metin	1642	29	8	1
	Başlık	134	255	0	0
	Liste	72	3	49	0
	Tablo	27	2	0	13

Sonuçlar incelendiğinde tablo ve liste bloklarının sınıflandırılmasında başarımın %50 altında olduğu görülmektedir.

3.2.3.2. PyTorch Kütüphanesi ve Ön Eğitimli BERT Modeli

PyTorch kütüphanesi ile son teknoloji (state of art) BERT modelinden faydalanılarak geliştirilen modele ait sonuçlar Tablo 12'de verilmiştir.

Tablo 12 : PyTorch Kütüphanesi ve Ön Eğitimli BERT Modelin Değerlendirmesi

	Kesinlik	Duyarlılık	f1-Skoru	Örnek Sayısı
Metin	0.96	0.98	0.97	1681
Başlık	0.95	0.94	0.95	375
Liste	0.85	0.69	0.76	136
Tablo	0.98	0.95	0.96	43
Doğruluk			0.96	2235
Makro Ortalama	0.94	0.89	0.91	2235
Ağırlıklı Ortalama	0.96	0.96	0.96	2235

Bu modele ait karmaşıklık matrisi Tablo 13'te verilmiştir.

Tablo 13 : PyTorch Kütüphanesi ve Ön Eğitimli BERT Modelin Karmaşıklık Matrisi

		Tahmin			
		Metin	Başlık	Liste	Tablo
Gerçek	Metin	1651	16	14	0
	Başlık	22	353	0	0
	Liste	40	1	94	1
	Tablo	0	0	2	41

Sonuçlar değerlendirildiğinde en başarısız olunan sınıfın %70 doğruluk ile liste olduğu görülmüştür. Diğer sınıflar için doğruluk %90'ın üzerinde olduğu görülmektedir.

3.2.3.3. Tensorflow Keras ile LSTM Model

Tensorflow ve Keras kütüphanesi kullanılarak geliştirilen modelin sonuçları Tablo 14'te verilmiştir.

Tablo 14 : Tensorflow Keras ile LSTM Model Değerlendirmesi

	Kesinlik	Duyarlılık	f1-Skoru	Örnek Sayısı
Metin	0.74	0.43	0.54	136
Başlık	0.86	0.48	0.62	50
Liste	0.91	0.95	0.93	1682
Tablo	0.81	0.79	0.80	367
Doğruluk			0.88	2235
Makro Ortalama	0.83	0.66	0.72	2235
Ağırlıklı Ortalama	0.88	0.88	0.88	2235

Bu modele ait karmaşıklık matrisi Tablo 15'te verilmiştir.

Tablo 15 : Tensorflow Keras ile LSTM Model Karmaşıklık Matrisi

		Tahmin			
		Metin	Başlık	Liste	Tablo
Gerçek	Metin	58	3	73	2
	Başlık	11	24	15	0
	Liste	9	1	1604	68
	Tablo	0	0	77	290

Sonuçlar değerlendirildiğinde başarısız olunan sınıfların %42 doğruluk ile liste ve %48 ile tablo olduğu görülmüştür. Diğer sınıflar için doğruluk %80 başlık ve %95 metin olduğu görülmektedir.

3.2.3.4. Tensorflow Keras LSTM Model ve GloVe

Bir önceki model eğitiminde yine tensorflow ve keras kullanılmıştır. Burada farklı olarak Stanford Üniversitesi tarafından geliştirilen GloVe (Global Vectors for Word Representation) kelime vektörü (word vector) olarak kullanılmıştır. Modelin sonuçları Tablo 16'da verilmiştir.

Tablo 16 : Tensorflow Keras LSTM Model ve GloVe Değerlendirmesi

	Kesinlik	Duyarlılık	f1-Skoru	Örnek Sayısı
Metin	0.63	0.46	0.53	134
Başlık	0.46	0.49	0.47	35
Liste	0.93	0.92	0.92	1710
Tablo	0.78	0.87	0.82	356
Doğruluk			0.88	2235
Makro Ortalama	0.70	0.68	0.69	2235
Ağırlıklı Ortalama	0.88	0.88	0.88	2235

Bu modele ait karmaşıklık matrisi Tablo 17'de verilmiştir.

Tablo 17 : Tensorflow Keras LSTM Model ve GloVe Karmaşıklık Matrisi

		Tahmin			
		Metin	Başlık	Liste	Tablo
Gerçek	Metin	40	0	74	5
	Başlık	1	18	24	2
	Liste	3	1	1600	71
	Tablo	0	0	44	352

Sonuçlar değerlendirildiğinde başarısız olunan sınıfların %34 doğruluk ile liste ve %40 ile tablo olduğu görülmüştür. Diğer sınıflar için doğruluk %89 başlık ve %95 metin olduğu görülmektedir.

3.2.3.5. Eğitilen Modellerin Doğruluk ve Boyut Olarak Karşılaştırılması

Yukarıda tek tek sonuçları verilen modelleri doğruluk ve model boyutu olarak karşılaştırılması Tablo 18'de verilmiştir.

Tablo 18 :Eğitilen Modellerin Doğruluk ve Model Boyutu Karşılaştırılması

	Doğruluk (%)	Model Boyutu (MB)
Scikit Learn (SVM)	87,65	1,8
PyTorch + BERT	96	411
Tensorflow Keras	89,8	1,4
Tensorflow Keras with GloVe	88	18,2

Sonuçlara bakıldığında PyTorch + BERT modelinin en başarılı model olduğu görülmektedir. F1 skoruna bakılması durumunda da yine bu model en başarılı modeldir. Ancak diğer modellere göre oransal olarak boyutu da çok büyüktür.

Sadece doğruluk değerine bakmak bizi yanlış yönlendireceği için F1 skorları incelendiğinde diğer modellerin genellikle tablo ve liste sınıflarında başarısız oldukları görünmektedir.

3.2.4. İlk Sayfa Metin Sınıflaması Modelinin Eğitilmesi

Scikit Learn kütüphanesinin bize sunduğu metin sınıflama metotlarından 7 tanesi ile elde edilen doğruluk değerleri Tablo 19'da verilmiştir.

Tablo 19 : İlk Sayfa Metin Sınıflaması için Eğitilen Modellerin Karşılaştırması

Model	Doğruluk
KNeighborsClassifier	0.408696
DecisionTreeClassifier	0.773913
MultinominalNB	0.782609
DecisionTreeClassifier	0.817391
SVM	0.800000
RandomForestClassifier	0.817391
SGDClassifier	0.826087

Tablo 19'da görüldüğü üzere en başarılı model SGDClassifier olmuştur. Bu modele ait detaylı analiz değerleri Tablo 20'de verilmiştir.

Tablo 20 : İlk Sayfa Metin Sınıflaması için Eğitilen SGDClassifier Değerlendirmesi

	Kesinlik	Duyarlılık	f1-Skoru	Örnek Sayısı
Ana Başlık	0.73	0.80	0.76	20
Yazar	0.71	1.00	0.83	17
Anahtar Kelimeler	1.00	0.57	0.73	7
Başlık	0.93	0.74	0.82	38
Metin	0.86	0.91	0.88	33

Doğruluk			0.83	115
Makro Ortalama	0.85	0.80	0.80	115
Ağırlıklı Ortalama	0.85	0.83	0.82	115

Yine bu modele ait karmaşıklık matrisi Tablo 21'de görülmektedir.

Tablo 21 : İlk Sayfa Metin Sınıflaması için Eğitilen SGDClassifier Karmaşıklık Matrisi

		Tahmin				
		Ana Başlık	Yazar	Anahtar Kelimeler	Başlık	Metin
Gerçek	Ana Başlık	16	0	0	1	3
	Yazar	0	17	0	0	0
	Anahtar Kelimeler	1	0	4	0	2
	Başlık	3	7	0	28	0
	Metin	2	0	0	1	30

Sonuçlar incelendiğinde en düşük Anahtar Kelime ve Ana Başlık bloklarının sınıflandırılmasında başarımın %75 civarında olduğu görülmektedir.

3.2.5. Son Sayfa veya Referans Sayfası Metin Sınıflama Modelinin Eğitilmesi

Scikit Learn kütüphanesinin bize sunduğu metin sınıflama metotlarından 7 tanesi ile elde edilen doğruluk değerleri Tablo 22' de verilmiştir.

Tablo 22 : Son Sayfa Metin Sınıflaması için Eğitilen Modellerin Karşılaştırması

Model	Doğruluk
KNeighborsClassifier	0.180095
DecisionTreeClassifier	0.772512
MultinominalNB	0.772512
DecisionTreeClassifier	0.853081
SVM	0.895735
RandomForestClassifier	0.881517
SGDClassifier	0.819905

Tablo 22'de görüldüğü üzere en başarılı model SVM olmuştur. Bu modele ait detaylı analiz değerleri Tablo 23'te verilmiştir.

Tablo 23 : Son Sayfa Metin Sınıflaması için Eğitilen SVM Değerlendirmesi

	Kesinlik	Duyarlılık	f1-Skoru	Örnek Sayısı
Metin	0.88	0.97	0.92	126
Referans	0.97	0.82	0.89	44
Başlık	0.86	0.82	0.84	38
Tablo	0.00	0.00	0.00	3
Doğruluk			0.90	211
Makro Ortalama	0.68	0.65	0.66	211
Ağırlıklı Ortalama	0.89	0.90	0.89	211

Yine bu modele ait karmaşıklık matrisi Tablo 24'te görülmektedir.

Tablo 24 : Son Sayfa Metin Sınıflaması için Eğitilen SVM Karmaşıklık Matrisi

		Tahmin			
		Metin	Referans	Başlık	Tablo
Gerçek	Metin	122	1	3	0
	Referans	7	36	1	0
	Başlık	7	0	31	0
	Tablo	2	0	1	0

Sonuçlar incelendiğinde Tablo bloklarının sınıflandırılmasında başarımında hem veri kümesinin yetersiz hem de başarısız olduğu görülmektedir.

3.3. SEGMENTASYON İŞLEMİ

Resim formatındaki dijital dokümandan LaTex kodu üretilmesinde ilk aşama resim üzerinde içerik bölgelerinin (blok) bulunmasıdır. Bu görev için OpenCV kütüphanesi kullanılmıştır.

Kaynak resim okunarak bir değişkene aktarılır. İlk olarak resmin yükseklik ve genişliği daha sonra kullanılmak üzere değişkenlere aktarılır. Ayrıca resim ile aynı isimde bir klasör oluşturulmuştur. Bu klasör içerisine uygulamanın sonuçları kaydedilecektir.

Blokların sınırlarının daha kolay bulunabilmesi için resmin renk skalası resim gri tonlamaya dönüştürülmüştür. Gri tonlama işlemi için OpenCV RGB (Red Green Blue – Kırmızı Yeşil Mavi) her bir kanal Denklem 3.1'de görüldüğü gibi özel bir kat sayı ile çarpmaktadır.

$$RGB\ to\ Gray: Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{3.1}$$

Gri tonlamalı resmin ikili tonlamaya (siyah ve beyaz) çevirmek için eşik değeri Otsu eşik belirleme metodu kullanılmıştır. Bu metotta önce resme ait histogram çıkarılmaktadır. Ardından histogramdaki her bir sınıf eşik değer kabul edilerek önceki değerler arka plan önceki değerler ön plan kabul edilerek geri kalan sınıflar için Denklem 3.2'de görüldüğü şekilde varyans değeri hesaplanmaktadır. Hedef sınıf içi varyansı en aza indirmek ve sınıflar arası varyansı maksimize etmektir.

$$\begin{aligned} q_1(t) &= \sum_{i=1}^{t} P(i) \quad \& \quad q_2(t) = \sum_{i=t+1}^{I} P(i) \\ \mu_1(t) &= \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)} \\ \sigma_1^2(t) &= \sum_{i=1}^{t} [i-\mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i-\mu_2(t)]^2 \frac{P(i)}{q_2(t)} \end{aligned} \tag{3.2}$$

Otsu metodunun özelliği eşik değerinin sabit bir değer olmaması dinamik olarak belirlenmesidir. Eşik değer üstündeki pikseller beyaz ve eşik değer altında kalan pikseller siyah olacaklardır.

Sonrasında blokları ortaya çıkarmak için ince ayar yapılabilecek bir yapısal eleman kullanılarak genişletme (dilation) işlemi ile uygun olan harf, kelime veya paragrafların birleştirilmesi sağlanmıştır. Genişletme işlemi örneği ve yapısal eleman büyüklüğü ektisi Şekil 7 (a)'da orijinal resim, Şekil 7 (b)'de 4x4 yapısal eleman ile genişleme uygulanan resim ve Şekil 7 (c)'de 12x12 yapısal eleman ile genişleme uygulanan resim görülmektedir.



(a) **(b)** **(c)**

Şekil 7 : Orijinal Resim (a), 4x4 Genişletme işlemi (b) ve 12x12 Genişletme işlemi (c)

Bir sonraki aşamada genişletme işlemi yapılmış resim örneğine konturları bulmak için hazır OpenCV komutu kullanılmıştır. Bu fonksiyon ile pikseller incelenerek birbirinden ayrı blokların koordinatları elde edilmektedir. Fonksiyon çalışırken şu parametrelere ihtiyaç duymaktadır. İlk parametre resim örneği, ikinci parametre mümkün olan en büyük şekilde ve içinde başka kontur olmayacak şekilde konturların bulunmasını sağlayacak parametre

seçilmiştir. Son bir parametre ile bulunan konturlar ile ilgili gelecek bilgilerin detayı belirlenmiştir. Bu işlem ile konturların bulunması tamamlanmıştır. Ancak bulunan konturlar üzerinde bazı filtrelemeler yapılması gerekmektedir.

Her bir kontur için *x, y, genişlik* ve *yükseklik* değerleri dönmektedir. Bulunan konturların bazıları bizim için gereksizdir. Resimde buluna sayfa numarası ya da web ortamında indirilen bir yayının üst ve alt kısmına yerleştirilen hangi akademik kurum üzerinden web sitesine bağlantı kurulduğunu yazan bilgilendirme mesajları örnek olarak verilebilir. Bu konturların dikkate alınmaması gerekmektedir. Bunları kontur listesinden çıkarabilmek için şu durumlar gözetilmiştir; konturun kapladığı alan sayfanın %10'undan küçükse veya en boy oranı %20'den küçükse veya konturun *y* noktası değeri sayfanın yüksekliğinin %3'ünden küçükse veya konturun y noktası değeri sayfanın yüksekliğinin %95'ünden büyükse bu kontur oluşturulacak blok listesine alınmamıştır.

Ayrı bir işlem olarak kontur koordinatları kullanılarak oluşturulan resim parçalarına OCR uygulanmıştır. OCR sonuçlarında özel karakterler ve noktalama işaretleri çıkarılarak NLTK kütüphanesi yardımıyla kelime sayısı bulunmuştur.

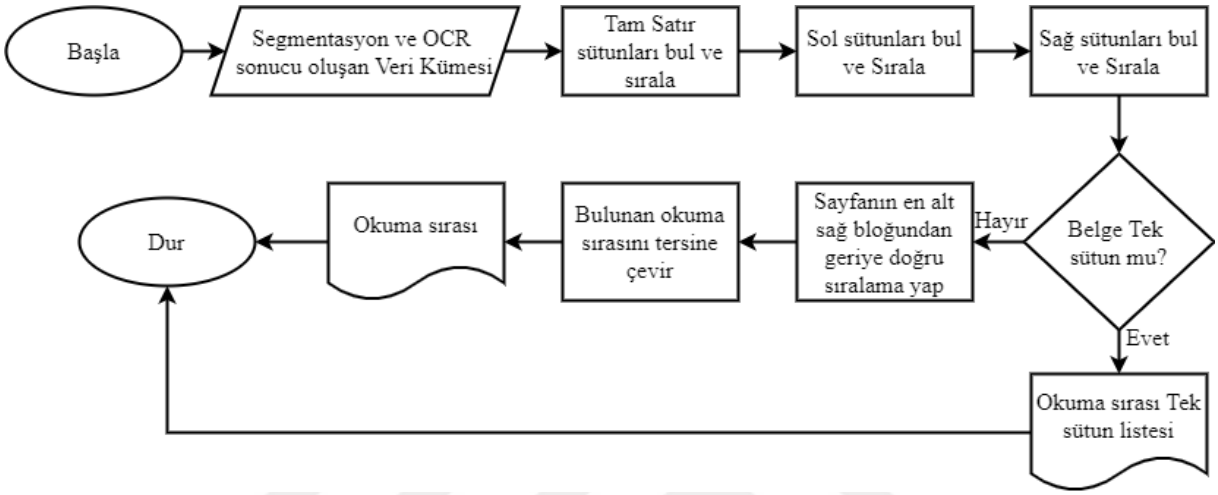
Bloklar için oluşturulan veri kümesi içerisinde kontur listesinden elde edilen *x, y, genişlik, yükseklik* değerleri, blok alanın sayfa alanına oranı, bloğun içerdiği metin ve blok kelime sayısı bulunmaktadır.

Tablo 25 : Kaynak Dijital Resimden Segmantasyon Sonucu Çıkarılan Veri Kümesi

İndeks	Blok İndeks	Blok X	Blok Y	Blok Genişlik	Blok Yükseklik	Blok Sayfa Oranı	Blok Metni	Blok Kelime Sayısı
0	0	120	2070	408	57	0.62	978-1-7281-8470-...	4
1	1	176	1940	259	73	0.51	* authors contrib...	4
2	2	154	1354	680	578	10.51	In recent years, te...	136
3	3	383	1312	222	36	0.21	I. INTRODUCTION…	2
4	4	154	649	680	617	11.22	Abstract—In the l...	173

3.4. OKUMA SIRASININ BULUNMASI

Bir dokümanın anlaşılmasında okuma sırası önemli bir rol oynamaktadır. Doküman birden fazla sütun ise sütun sayısı bulunmalıdır sonrasında içeriklerin (metin, şekil ve tablo gibi) bulunması gerekir. LaTex koduna geçiş yapılabilmesi için doğru bir okuma sırasının bulunması gerekmektedir. Bu işlemi gerçekleştirmek için uygulanan algoritma Şekil 8'de görülmektedir.



Şekil 8 : Okuma Sırası Bulma Algoritması

Şekil 8' de akış şeması verilen algoritmaya ait sözde kod:

```
GET → bloklar(), sayfaGenişliği
FOR i TO bloklar.sayısı STEP i++
        IF bloklar.genislik(i)-bloklar.x(i) > sayfaGenisliği/2
                tamSütunlar.ekle(bloklar.id(i))
        IF bloklar.genislik(i) <  sayfaGenisliği/2
                solSütunlar.ekle(bloklar.id(i))
        IF bloklar.x(i) >  sayfaGenisliği/2
                sağSütunlar.ekle(bloklar.id(i))
ENDFOR
yukarıdanAşağıSırala(tamSütunlar)
yukarıdanAşağıSırala(solSütunlar)
yukarıdanAşağıSırala(sağSütunlar)
IF bloklar.sayısı = tamSütunlar.sayısı
        okumaSırası = tamSütunlar
```

```
ELSEIF tamSütunlar.sayısı = 0
        okumaSırası = solSütunlar + sağSütunlar
ELSE
        FOR tamSütunlar.sayısı TO m STEP m—
                FOR sağSütunlar.sayısı TO n STEP n—
                        IF tamSütunlar.y(m) < sağSütunlar.y(n)
                                okumaSırası.ekle(sağSütunlar(n)
                FOR solSütunlar.sayısı TO k STEP k—
                        IF tamSütunlar.y(m) < solSütunlar.y(k)
                                okumaSırası.ekle(solSütunlar(k)
        ENDFOR
        okumaSırası = tersi(okumaSırası)
ENDIF
OUT → okumaSırası
```

Bir önceki işlem basamağı olan dokümanın segmentlere ayrılmasından sonra blokların başladığı x, y koordinatları, genişlik ve yükseklik bilgileri kullanılarak okuma sırasını bulmak işlemi gerçekleştirilmiştir. Ayrıca okuma sırası bulunurken sayfa kenar boşluk bilgileri ve varsa sütunlar arası boşluk bilgisi de bulunmuştur.

Öncelikle blokların id, x, y, genişlik ve yükseklik bilgileri ayrı listelere alındı. İşlemi gerçekleştirirken kullanılacak okuma sırası, tam sütun, sol sütun ve sağ sütun bilgileri için boş listeler oluşturuldu.

İlk olarak tüm blokların genişlik bilgisine bakılarak dokümanın genişliğinin yarısından büyük olması durumuna ve dokumanın genişliğinin orta noktası kontrol edilen bloğun sınırları içerisinde mi yer almaktadır kontrol edilmiştir. Eğer şartlar doğruysa bu blok tam sütun listesine eklendi. Bu sırada kontrol edilen bloklar için en küçük *x* değeri ve en büyük *x + genişlik* değerleri sol ve sağ kenar boşluklarını bulmak için saklandı.

Tüm bloklardan varsa tam sütunlar sonrasında yazılan bir sıralama fonksiyonu ile *y* değerlerine göre sıralamaları yapıldı.

Eğer doküman iki sütun ise sol sütunların bulunması için blokların *x* + *genişlik* değerinin dokümanın genişliğinin yarısından küçük ise sol sütun listesine kaydedildi. Sol kenar boşluğunu bulmak için blokların en küçük *x* değeri ve sütunlar arası boşluğu bulabilmek için en büyük *x* + *genişlik* değerleri saklandı.

Tüm bloklardan varsa sol sütunlar sonrasında yazılan bir sıralama fonksiyonu ile y değerlerine göre sıralamaları yapıldı.

Sağ sütunların bulunması için blokların x değerinin dokümanın genişliğinin yarısından büyük ise sağ sütun listesine kaydedildi. Sağ kenar boşluğunu bulmak için blokların en büyük x artı genişlik değeri ve sütunlar arası boşluğu bulabilmek için en küçük x değerleri saklandı.

Tüm bloklardan varsa sağ sütunlar sonrasında yazılan bir sıralama fonksiyonu ile y değerlerine göre sıralamaları yapıldı.

Bloklar sınıflandırıldıktan sonraki işlem okuma sırasını bulmak olmuştur. Bunun için öncelikle tüm blokların sayısı tam sütunların sayısına eşitse okuma sırası tam sütun sıralamasıdır.

Eğer tam sütun sayısı sıfır ise okuma sırası sol sütunların ardından sağ sütunların listesinin birleşimidir.

Eğer dokuman tek ve çift sütunlardan oluşuyor ise bloklar önce sağ sütunlar sonra sol sütunlar ve en son tam sütunlar içerisinden en büyük y değerleri gözetilerek sıralama gerçekleştirilmiştir. Burada dikkat edilecek husus okuma sırası tersten oluşturulmuştur. Bu sebeple işlem sonunda oluşan okuma listesi tersine çevrilmiştir.

Son olarak blokların sınıfları belirlenirken saklanan değerler kullanılarak kenar boşlukları ve varsa sütunlar arası boşluk değerleri hesaplanmıştır. Bu değerler piksel cinsinden oldukları için Denklem 3.3 kullanılarak yazılan bir fonksiyon ile milimetre birimine dönüştürülmüşlerdir. Fonksiyon 3 parametre almaktadır. Birincisi bulunan kenarlık değeridir. İkincisi eğer sayfanın üst veya alt kenar boşluğu bulunacaksa bir A4 kağıdının milimetre cinsinden yüksekliği 297 mm ya da sol veya sağ kenar boşluğu bulunacaksa A4 kağıdının milimetre cinsinden genişliği

210 mm bilgisi parametre olarak gönderilmelidir. Üçüncü olarak veri kümesinden elde edilen dijital dokümanın yüksek veya genişlik bilgisidir.

$$Mili\ Metre = Değer * A4\ Bilgisi/Dijital\ Döküman\ Bilgisi \quad (3.3)$$

3.5. METİN / ŞEKİL SINIFLAMASININ YAPILMASI

Segmentasyon sonucu bulunan ve okuma sırasına konulan bloklar için bir sonraki aşama hangi blokların metin hangilerinin şekil olduğunu bulmaktır. Bu işlem için modelin eğitiminde kullanılan verilerden normalizasyon için gerekli değerler hesaplanarak yeni bulunan sayfa blok alanı oranı ve kelime sayısı değerlerine normalizasyon işlemi uygulanmıştır.

Makine öğrenmesi modeli tahmin sonuçları veri kümesine yeni bir sütun olarak eklenmiştir.

Bir sonraki aşamada şekil olarak sınıflandırılan bloklar koordinatlarına göre kaynak resim üzerinden kırpılarak yeni bir resim dosyası olarak ve o bloğun ID numarası adı olacak şekilde kaydedilmiştir.

3.6. SAYFA TİPİ SINIFLAMASININ YAPILMASI

Sayfa tipini belirlemek için kaynak resmin tamamına OCR yapılmıştır. Sayfa tipini belirlemek için modelin eğitiminde kullanılan veri kümesinin vektör hali kaynak olarak kullanılarak yapılan OCR sonucu da vektöre dönüştürüldükten sonra modele tahmin yaptırılmıştır. Tahmin sonucu üç sınıf olabilmektedir bunlar: ilk sayfa, ara sayfa ve son sayfadır.

3.7. SAYFA TİPİNE GÖRE METİN SINIFLAMA YAPILMASI

Sayfa tipine uygun metin sınıflama işlemi yapılması için eğitilmiş üç farklı makine öğrenmesi modeli eğitilmiştir. Bulunan sayfa tipi bilgisine göre bu modellerden bir tanesi ile tahmin işlemi gerçekleştirilmiştir.

3.7.1. İlk Sayfa

İlk Sayfa tipinde metin sınıflarını belirlemek için eğitilen modelin eğitiminde kullanılan veri kümesinin vektör hali kaynak olarak kullanıldıktan sonra modele tahmin yaptırılmıştır. Tahmin sonuçları metin sınıfları olarak veri kümesinde yeni bir sütun olarak eklenmiştir. Tahmin sonucunda bloklar için beş sınıf olabilmektedir bunlar; ana sayfa, yazar, anahtar kelime, başlık

ve metin'tir. Veri kümesinde eğer şekil olarak sınıflandırılmış bir blok var ise bu bloğa model uygulanmadan doğrudan olarak -1 değeri verilmiştir. Bunun sebebi sonraki aşamada LaTex kodu oluşturulurken karışıklığın önlenmesidir.

3.7.2. Ara Sayfa

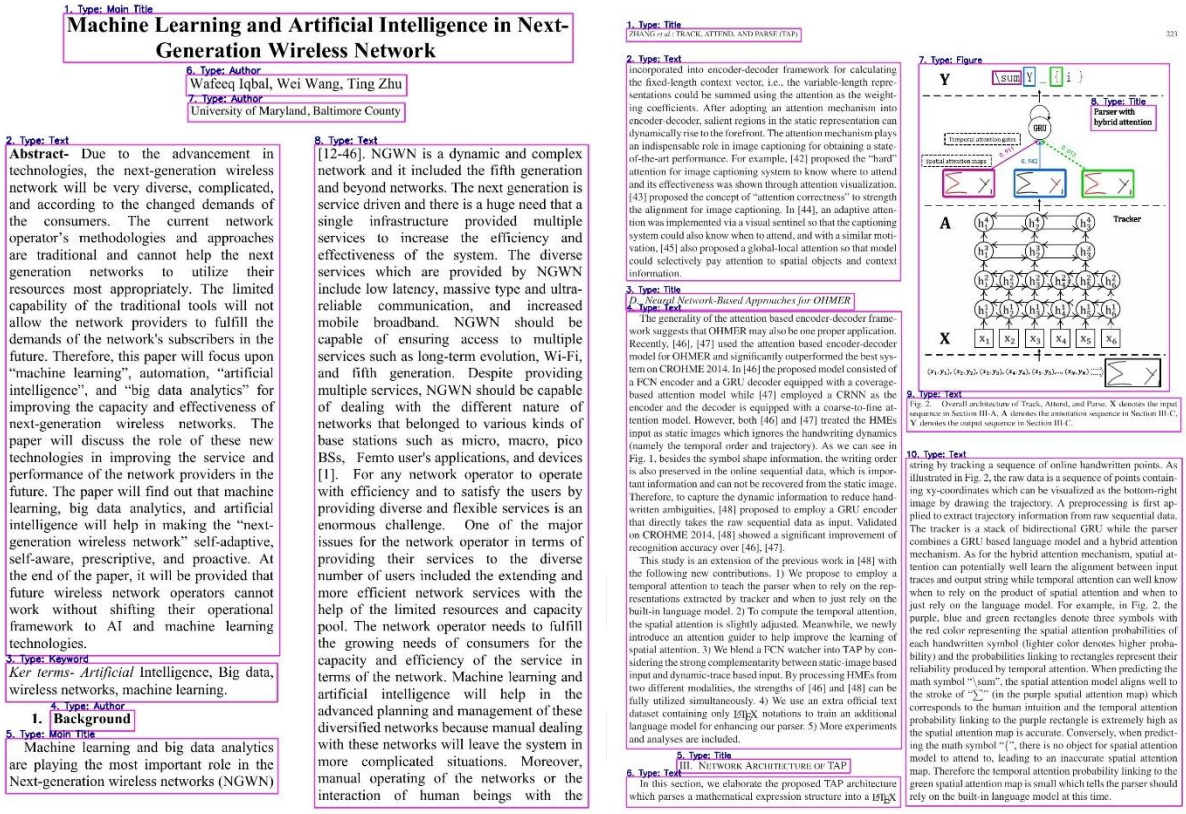
Ara Sayfa tipinde metin sınıflarını belirlemek için modelin eğitiminde kullanılan veri kümesinin vektör hali kaynak olarak kullanıldıktan sonra modele tahmin yaptırılmıştır. Tahmin sonuçları metin sınıfları olarak veri kümesinde yeni bir sütun olarak eklenmiştir. Tahmin sonucunda bloklar için beş sınıf olabilmektedir bunlar; metin, başlık, liste, tablo'dur. Veri kümesinde eğer şekil olarak sınıflandırılmış bir blok var ise bu bloğa model uygulanmadan doğrudan olarak -1 değeri verilmiştir. Bunun sebebi sonraki aşamada LaTex kodu oluşturulurken karışıklığın önlenmesidir.

3.7.3. Son veya Referans Sayfası

Son Sayfa tipinde metin sınıflarını belirlemek için eğitilen modelin eğitiminde kullanılan veri kümesinin vektör hali kaynak olarak kullanıldıktan sonra modele tahmin yaptırılmıştır. Tahmin sonuçları metin sınıfları olarak veri kümesinde yeni bir sütun olarak eklenmiştir. Tahmin sonucunda bloklar için beş sınıf olabilmektedir bunlar; metin, referans, başlık, tablo'dur. Veri kümesinde eğer şekil olarak sınıflandırılmış bir blok var ise bu bloğa model uygulanmadan doğrudan olarak -1 değeri verilmiştir. Bunun sebebi sonraki aşamada LaTex kodu oluşturulurken karışıklığın önlenmesidir.

3.7.4. Metin Sınıflama Sonuçlarının Görüntülenmesi

Kaynak resim üzerinde bulunan blokların dikdörtgenler içerisinde gösterilmesi her bir dikdörtgenin sol üst köşesinde okuma sırası ve bloğun metin sınıfı yazacak şekilde bu aşamaya kadar yapılmış işleri gösteren resim kaynak klasöre kaydedilmiştir. Örnek resimler Şekil 9 (a)' da ve Şekil 9 (b)' de görülmektedir.



(a) **(b)**

Şekil 9 : Metin Sınıflaması Sonuçlarının Kaynak Doküman Üzerinde Gösterilmesi

Veri kümesinin son hali CSV formatında kaynak klasöre kaydedilmiştir. Sayfa kenarlık bilgileri varsa sütunlar arası boşluk, kaynak resmin genişliği ve yüksekliği, varsa sütunlara ait bilgiler ve sayfa tipi bilgisi JSON formatında kaynak klasöre kaydedilmiştir. Bu şekilde yapılan çalışmada bu aşamaya kadar yapılan işlemlerin hepsi kayıt altına alınmıştır.

JSON standart bir veri kayıt formatıdır. Tüm programlama dilleri ve web ortamlarında standart olarak veri transferinde kullanılmaktadır. Sonraki çalışmalarda ihtiyaç duyulabileceği düşünülerek veri kümesi haricindeki elde edilen veriler bu format ile saklanmaktadır.

3.8. LATEX KODUNUN ÜRETİLMESİ

LaTex kodları içerisinde komutların yazılmasında kullanılan {, }, %, $, ^, & gibi özel karakterlerinden metin içerisinde bulunması derleme hatalarına sebep olacaktır. Oluşacak hataları önlemek için veri kümesi içerisindeki metin sütunu yazılan bir fonksiyon ile bu özel karakterlerin bulunması durumunda karakterden önce kaçış karakteri denilen "\" karakteri eklenmiştir.

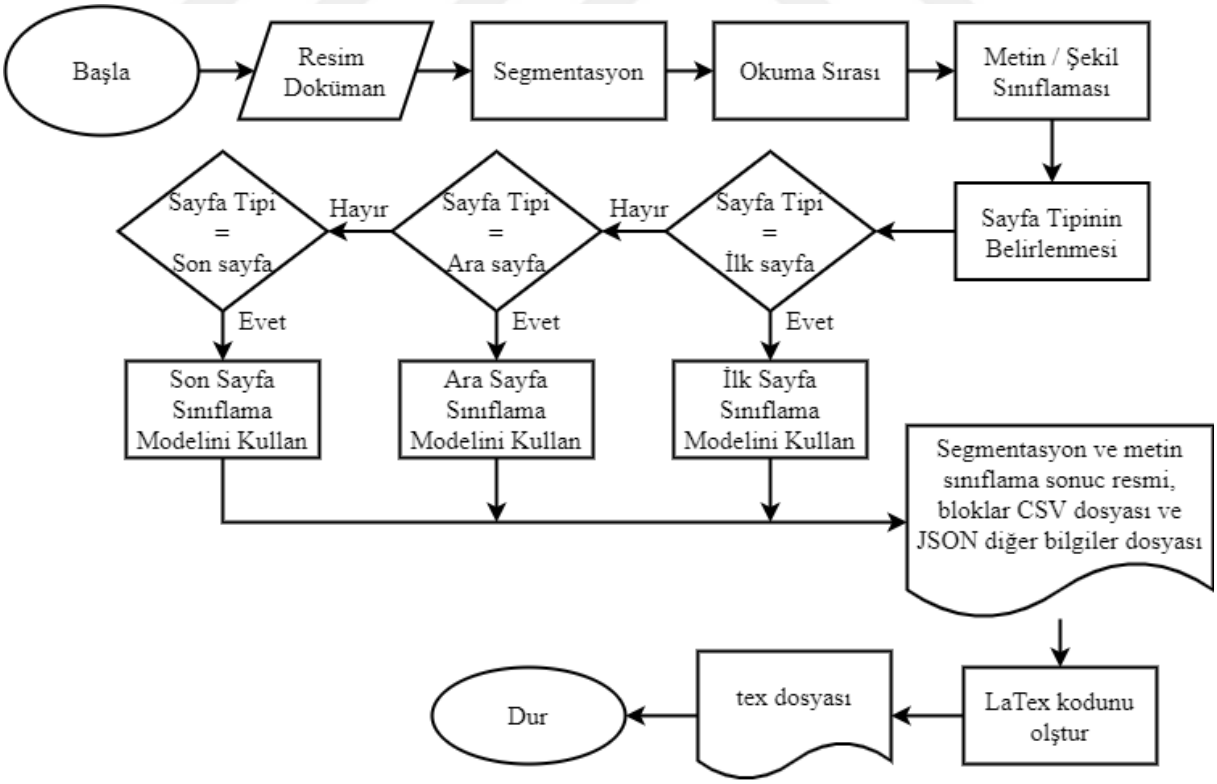
Oluşturulan LaTex kodu tex uzantılı bir dosyaya kaynak klasöre kaydedilmiştir. Sayfa tipine göre metin sınıfları bir listeden alınarak işleme başlanmıştır.

“Preamble” bölümü için JSON dosyasından gelen bilgiye göre bulunan bloklar tek sütun mu çift sütun mu tespit edilmiştir. Sonra sayfa kenarlıkları ve gerekiyorsa sütunlar arası boşluk bilgileri için gerekli kod eklenmiştir. Gerekli paketler eklenmiştir. Eğer kaynak resim ilk sayfa olarak sınıflandırılmışsa ana sayfa ve yazar bilgileri buraya eklenmiştir.

“Body” bölümü için veri kümesinde yer alan metin sınıflarına uygun olarak LaTex kodları eklenmiştir.

3.9. UYGULAMANIN GELİŞTİRİLMESİ

Çalışmada anlatılan yöntemler kullanılarak uygulamayı geliştirmek için mobilite avantajından dolayı Google Colabratory [42] tercih edilmiştir. Yaygın kütüphaneleri geliştirme aşamasını hızlandırdığı için programlama dili olarak Python kullanılmıştır. Geliştirme aşamaları detaylıca bahsedilmiştir. Çok aşamalı geliştirme sürecinin algoritması Şekil 10’da görülmektedir.



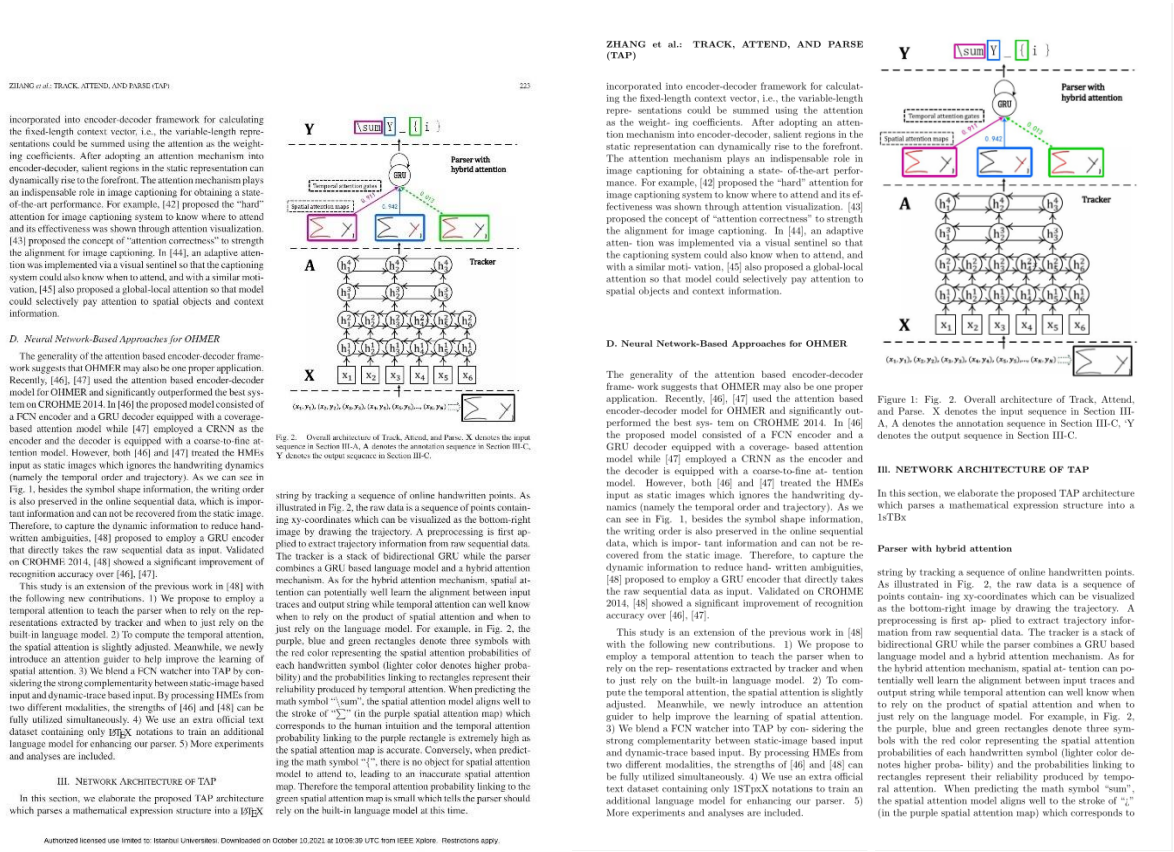
Şekil 10 : Uygulama Geliştirme Algoritması

Şekil 10’da akış şeması verilen algoritmaya ait sözde kod:

```
GET → resim
griResim = griTonlamaYap(resim)
siyahBeyazResim = otsuEsikDeğerBelirleme(griResim)
genişletmeUygulanmışResim = genişletme(siyahBeyazResim)
bloklar() = konturBul(genişletmeUygulanmışResim)
okumaSırası = okumaSırasıBul(bloklar())
bloklar() = metinŞekilBelirle(bloklar())
sayfaTipi = sayfaTipiBelirle(resim)
IF sayfaTipi = "ilkSayfa"
        bloklar() = ilkSayfaMetinSınıflama(bloklar())
ELSEIF sayfaTipi = "araSayfa"
        bloklar() = araSayfaMetinSınıflama(bloklar())
ELSEIF sayfaTipi = "sonSayfa"
        bloklar() = sonSayfaMetinSınıflama(bloklar())
ENDIF
OPEN texFile AS tex
        tex. preambleEkle(preambleBilgileri)
        IF sayfaTipi = "ilkSayfa"
                tex.preambleEkle("Ana Başlık","Yazar",)
                tex.bodyEkle("Başlık","Metin","Anahtar Kelime","Şekil")
        ELSEIF sayfaTipi = "araSayfa"
                tex.bodyEkle("Başlık","Metin","Liste","Tablo","Şekil")
        ELSEIF sayfaTipi = "sonSayfa"
                tex.bodyEkle("Başlık","Metin","Referans","Tablo","Şekil")
CLOSE texFile
OUT → texFile
```

4. BULGULAR

Geliştirilen uygulamaya girdi olarak verilen bir kaynak dijital yayın sonucunda çıktı olarak oluşan tex dosyasının derlenmiş PDF çıktı örnekleri Şekil 11'de görülmektedir.



(a) **(b)**

Şekil 11 : Kaynak Dijital Doküman (a) ve Uygulamanın Oluşturduğu PDF Çıktısı (b)

Şekil 11 (a)'da orijinal doküman Şekil 11 (b)'de geliştirilen uygulamanın PDF çıktısı görülmektedir. Bu örnekte geliştirilen uygulama şekil başarı ile bulunmuştur. İki sütunlu sayfa yapısı oluşturulmuştur. Hata yapılan blok mevcuttur. Ayrıca Font tipi ve paragraf boşlukları gibi bu uygulamada dikkate alınmayan özelliklerden dolayı bire bir eşleşme olmamaktadır.

Yapılan deneyler sonucunda geliştirilen uygulama bir dokümanda tamamen doğru sonuçlar verebiliyorken başka bir dokümanda blokların yarısını bile doğru tespit edememektedir. Örneğin Şekil 12 (a)'da ve Şekil 12 (b)'de metin sınıflarından yanlış tahmin edilenler olmuştur.

Ayrıca segmentasyon işleminde referans bloğunun numaralarının bloğa dahil edilmemesi başka bir hata olmuştur.

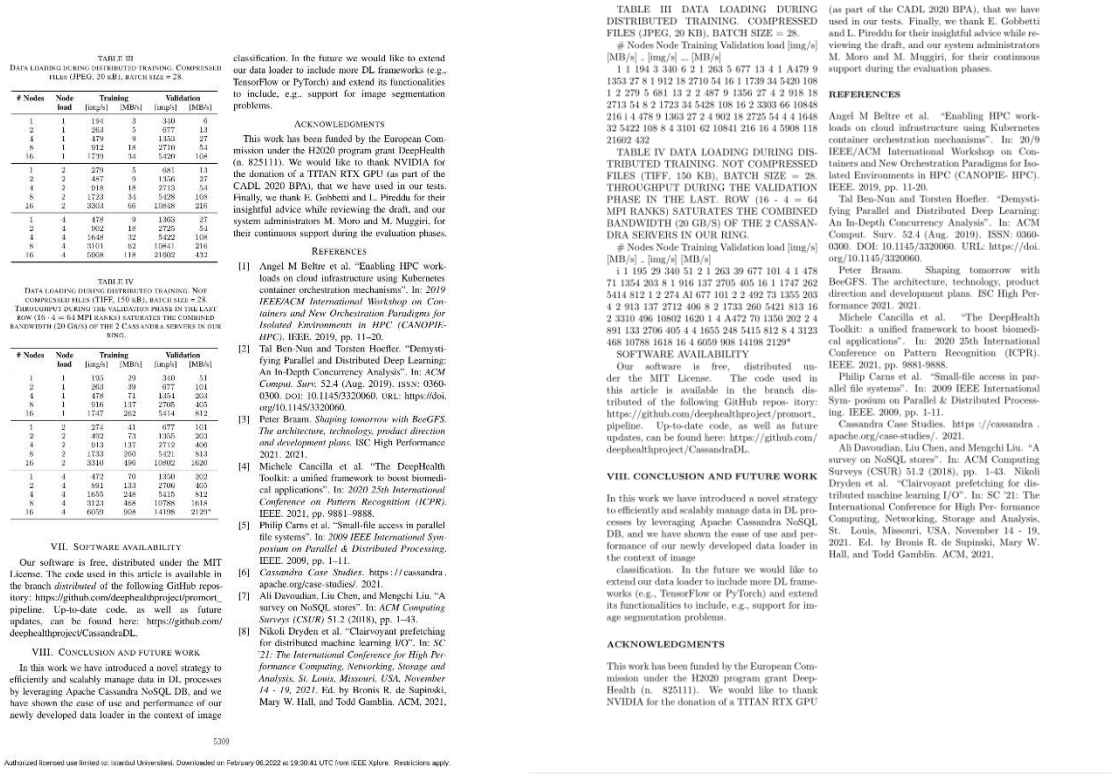


TABLE III
DATA LOADING DURING DISTRIBUTED TRAINING. COMPRESSED FILES (JPEG, 20 KB), BATCH SIZE = 28.

# Nodes	Node load	Training [img/s]	Training [MB/s]	Validation [img/s]	Validation [MB/s]
1	1	194	3	340	6
2	1	263	5	677	14
4	1	479	9	1353	27
8	1	912	18	2710	54
16	1	1739	34	5420	108
1	2	279	5	681	14
2	2	487	9	1356	27
4	2	918	18	2713	54
8	2	1723	34	5428	108
16	2	3303	66	10848	216
1	4	478	9	1363	27
2	4	902	18	2725	54
4	4	1648	32	5422	108
8	4	3101	62	10841	216
16	4	5908	118	21602	432

TABLE IV
DATA LOADING DURING DISTRIBUTED TRAINING. NOT COMPRESSED FILES (TIFF, 150 KB), BATCH SIZE = 28. THROUGHPUT DURING THE VALIDATION PHASE IN THE LAST ROW ($16 \cdot 4 = 64$ MPI RANKS) SATURATES THE COMBINED BANDWIDTH (20 GB/S) OF THE 2 CASSANDRA SERVERS IN OUR RING.

# Nodes	Node load	Training [img/s]	Training [MB/s]	Validation [img/s]	Validation [MB/s]
1	1	195	29	340	51
2	1	263	39	677	101
4	1	478	71	1354	203
8	1	936	137	2705	405
16	1	1747	262	5414	812
1	2	274	41	677	101
2	2	492	73	1355	203
4	2	913	137	2712	406
8	2	1733	260	5421	813
16	2	3310	496	10882	1620
1	4	472	70	1350	202
2	4	891	133	2706	405
4	4	1655	248	5415	812
8	4	3123	468	10788	1618
16	4	6059	908	14198	2129*

VII. Software availability

Our software is free, distributed under the MIT License. The code used in this article is available in the branch *distributed* of the following GitHub repository: https://github.com/deephealthproject/promort_pipeline. Up-to-date code, as well as future updates, can be found here: https://github.com/deephealthproject/CassandraDL.

VIII. Conclusion and future work

In this work we have introduced a novel strategy to efficiently and scalably manage data in DL processes by leveraging Apache Cassandra NoSQL DB, and we have shown the ease of use and performance of our newly developed data loader in the context of image classification. In the future we would like to extend our data loader to include more DL frameworks (e.g., TensorFlow or PyTorch) and extend its functionalities to include, e.g., support for image segmentation problems.

Acknowledgments

This work has been funded by the European Commission under the H2020 program grant DeepHealth (n. 825111). We would like to thank NVIDIA for the donation of a TITAN RTX GPU (as part of the CADL 2020 BPA), that we have used in our tests. Finally, we thank E. Gobbetti and L. Pireddu for their insightful advice while reviewing the draft, and our system administrators M. Moro and M. Muggiri, for their continuous support during the evaluation phases.

References

[1] Angel M Beltre et al. "Enabling HPC workloads on cloud infrastructure using Kubernetes container orchestration mechanisms". In: *2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*. IEEE, 2019, pp. 11–20.

[2] Tal Ben-Nun and Torsten Hoefler. "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis". In: *ACM Comput. Surv.* 52.4 (Aug. 2019). ISSN: 0360-0300. DOI: 10.1145/3320060. URL: https://doi.org/10.1145/3320060.

[3] Peter Braam. *Shaping tomorrow with BeeGFS. The architecture, technology, product direction and development plans*. ISC High Performance 2021. 2021.

[4] Michele Cancilla et al. "The DeepHealth Toolkit: a unified framework to boost biomedical applications". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 9881–9888.

[5] Philip Carns et al. "Small-file access in parallel file systems". In: *2009 IEEE International Symposium on Parallel & Distributed Processing*. IEEE. 2009, pp. 1–11.

[6] *Cassandra Case Studies*. https://cassandra.apache.org/case-studies/. 2021.

[7] Ali Davoudian, Liu Chen, and Mengchi Liu. "A survey on NoSQL stores". In: *ACM Computing Surveys (CSUR)* 51.2 (2018), pp. 1–43.

[8] Nikoli Dryden et al. "Clairvoyant prefetching for distributed machine learning I/O". In: *SC '21: The International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, Missouri, USA, November 14 - 19, 2021*. Ed. by Bronis R. de Supinski, Mary W. Hall, and Todd Gamblin. ACM, 2021,

Authorized licensed use limited to: Istanbul Universitesi. Downloaded on February 06,2022 at 19:30:41 UTC from IEEE Xplore. Restrictions apply.

TABLE III DATA LOADING DURING DISTRIBUTED TRAINING. COMPRESSED FILES (JPEG, 20 KB), BATCH SIZE = 28.

Nodes Node load Training [img/s] _ [MB/s] Validation [img/s] _ [MB/s]

1 1 194 3 340 6 2 1 263 5 677 13 4 1 A479 9 1353 27 8 1 912 18 2710 54 16 1 1739 34 5420 108 1 2 279 5 681 13 2 2 487 9 1356 27 4 2 918 18 2713 54 8 2 1723 34 5428 108 16 2 3303 66 10848 216 1 4 478 9 1363 27 2 4 902 18 2725 54 4 4 1648 32 5422 108 8 4 3101 62 10841 216 16 4 5908 118 21602 432

TABLE IV DATA LOADING DURING DISTRIBUTED TRAINING. NOT COMPRESSED FILES (TIFF, 150 KB), BATCH SIZE = 28. THROUGHPUT DURING THE VALIDATION PHASE IN THE LAST. ROW (16 - 4 = 64 MPI RANKS) SATURATES THE COMBINED BANDWIDTH (20 GB/S) OF THE 2 CASSANDRA SERVERS IN OUR RING.

Nodes Node Training Validation load [img/s] [MB/s] _ [img/s] _ [MB/s]

i 1 195 29 340 51 2 1 263 39 677 101 4 1 478 71 1354 203 8 1 916 137 2705 405 16 1 1747 262 5414 812 1 2 274 A1 677 101 2 2 492 73 1355 203 4 2 913 137 2712 406 8 2 1733 260 5421 813 16 2 3310 496 10802 1620 1 4 A472 70 1350 202 2 4 891 133 2706 405 4 4 1655 248 5415 812 8 4 3123 468 10788 1618 16 4 6059 908 14198 2129*

SOFTWARE AVAILABILITY

Our software is free, distributed under the MIT License. The code used in this article is available in the branch distributed of the following GitHub repos- itory: https://github.com/deephealthproject/promort_ pipeline. Up-to-date code, as well as future updates, can be found here: https://github.com/ deephealthproject/CassandraDL.

VIII. CONCLUSION AND FUTURE WORK

In this work we have introduced a novel strategy to efficiently and scalably manage data in DL processes by leveraging Apache Cassandra NoSQL DB, and we have shown the ease of use and performance of our newly developed data loader in the context of image classification. In the future we would like to extend our data loader to include more DL frameworks (e.g., TensorFlow or PyTorch) and extend its functionalities to include, e.g., support for image segmentation problems.

ACKNOWLEDGMENTS

This work has been funded by the European Commission under the H2020 program grant DeepHealth (n. 825111). We would like to thank NVIDIA for the donation of a TITAN RTX GPU (as part of the CADL 2020 BPA), that we have used in our tests. Finally, we thank E. Gobbetti and L. Pireddu for their insightful advice while reviewing the draft, and our system administrators M. Moro and M. Muggiri, for their continuous support during the evaluation phases.

REFERENCES

Angel M Beltre et al. "Enabling HPC workloads on cloud infrastructure using Kubernetes container orchestration mechanisms". In: 20/9 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE- HPC). IEEE. 2019, pp. 11-20.

Tal Ben-Nun and Torsten Hoefler. "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis". In: ACM Comput. Surv. 52.4 (Aug. 2019). ISSN: 0360-0300. DOI: 10.1145/3320060. URL: https://doi.org/10.1145/3320060.

Peter Braam. Shaping tomorrow with BeeGFS. The architecture, technology, product direction and development plans. ISC High Performance 2021. 2021.

Michele Cancilla et al. "The DeepHealth Toolkit: a unified framework to boost biomedical applications". In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE. 2021, pp. 9881-9888.

Philip Carns et al. "Small-file access in parallel file systems". In: 2009 IEEE International Sym- posium on Parallel & Distributed Processing. IEEE. 2009, pp. 1-11.

Cassandra Case Studies. https ://cassandra . apache.org/case-studies/. 2021.

Ali Davoudian, Liu Chen, and Mengchi Liu. "A survey on NoSQL stores". In: ACM Computing Surveys (CSUR) 51.2 (2018), pp. 1-43. Nikoli Dryden et al. "Clairvoyant prefetching for distributed machine learning I/O". In: SC '21: The International Conference for High Per- formance Computing, Networking, Storage and Analysis, St. Louis, Missouri, USA, November 14 - 19, 2021. Ed. by Bronis R. de Supinski, Mary W. Hall, and Todd Gamblin. ACM, 2021,

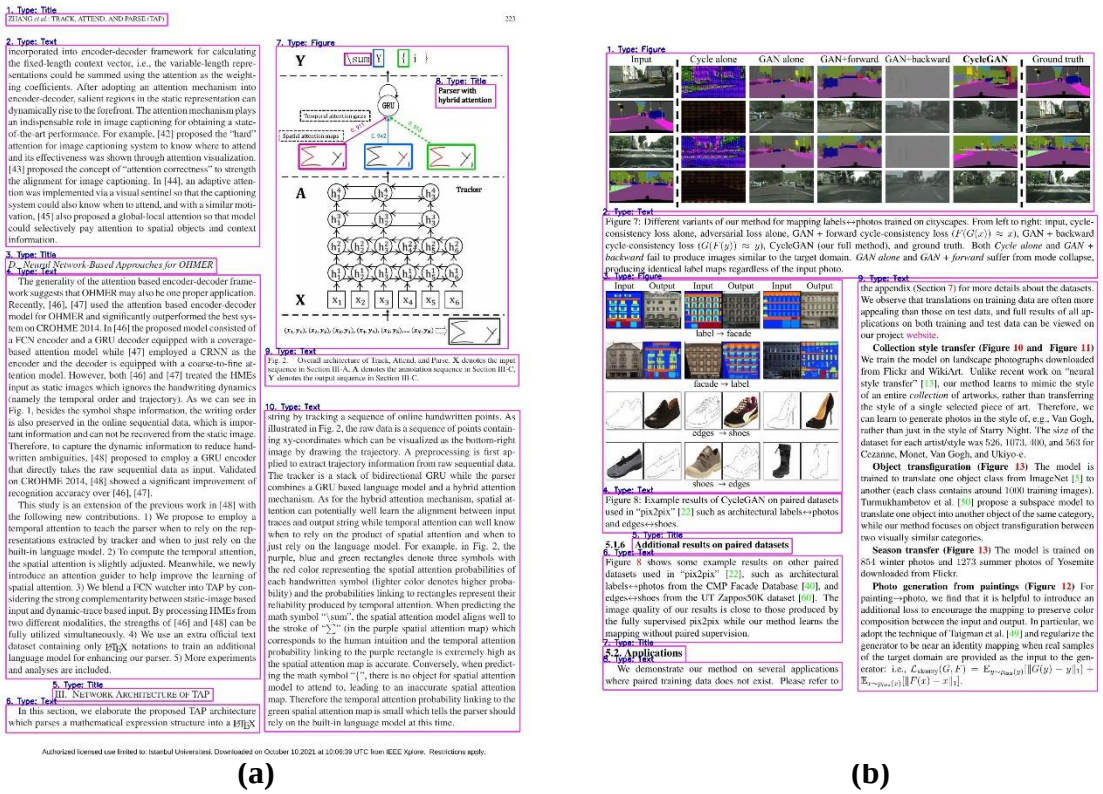
(a) **(b)**

Şekil 12 : Kaynak Dijital Doküman (a) ve Uygulamanın Başarısız Olduğu PDF Çıktısı (b)

Şekil 12 (a)'da ve Şekil 12 (b)'de verilen örnekler birçok aşama ve model tahmini sonucunda bir ürün oluşturmaktadır. Bu nedenle ara aşamalarında değerlendirilmesi gerekmektedir.

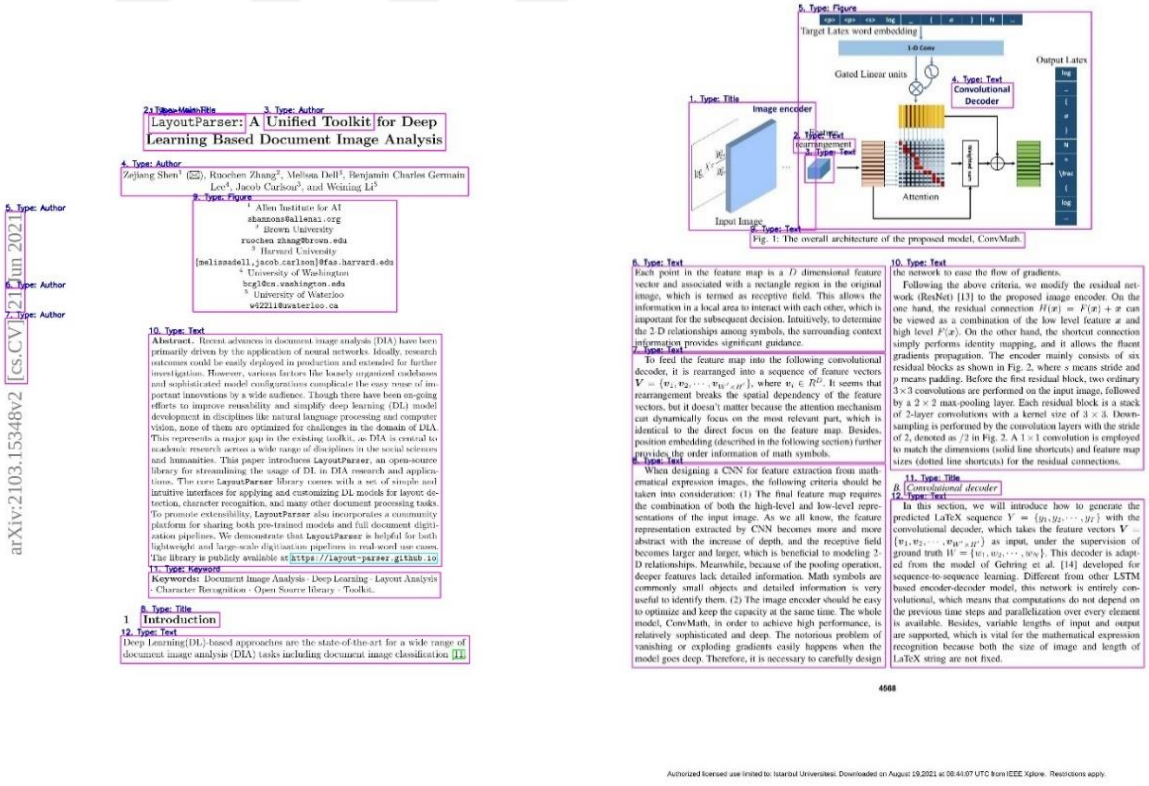
4.1. SEGMENTASYON

OpenCV'de konturların bulunması ile elde edilen bloklar değerlendirildiğinde bazı deneylerde çok başarılı sonuçlar alındığı bazı deneylerde ise başarısız olunduğu görülmüştür. Dijital doküman üzerindeki şekiller düzensiz parçalı bir yapıya sahiplerse ya da dokümanın ilk sayfalarında ana başlıkta segmentasyon başarımı düşük olduğu görülmüştür. Şekil 13 (a)'da ve Şekil 13 (b)'de başarılı örnekler gösterilmiştir.



(a) **(b)**

Şekil 13 : Segmentasyon Sonucu Oluşan Blokların Gösterimi



(a) **(b)**

Şekil 14 : Segmantasyonda Başarısız Olunan Örnekler

Şekil 13 (a)'da ve Şekil 13 (b)'de başarılı örneklere bakıldığında segmentasyon işleminin çok başarılı olduğu düşünülebilir. Ancak Şekil 14 (a)'da ve Şekil 14 (b)'de başarısız olunan örnekler verilmiştir.

Segmentasyon sistemi tekrar edilebilir bir çalışma sergilememesinin sebebi olarak dokümanların belli bir standart görünüşe sahip olmamalarıdır.

4.2. METİN / ŞEKİL SINIFLANDIRMASI

Uygulamada bir şekil bloğunun dokümanın toplam alanına oranı ve bloğun alanı ile içinde bulunan kelime sayısı oranı dikkate alınarak geliştirilmiştir. Bu teorik olarak doğrudur. Ancak gerçek örnekler üzerinde yapılan tahminlerde hatalar çıktığı görülmüştür. Örneğin segmantasyonda başarısız olarak verilen çıktılarda görülmektedir ki metin blokları şekil olarak sınıflandırılabilmektedir.

4.3. OKUMA SIRASININ BULUNMASI

Okuma sırası algoritması heuristic olarak çalıştığı için segmentasyondan gelen bilgiler doğrultusunda kural tabanlı olarak çalışmaktadır. Yapılan deneyler sonucunda segmentasyondaki hatalardan kaynaklı olarak hatalar oluşabildiği görülmüştür. Şekil 13'te algoritmanın doğru çalıştığı ve Şekil 14'te hataların olduğu örnekler verilmiştir.

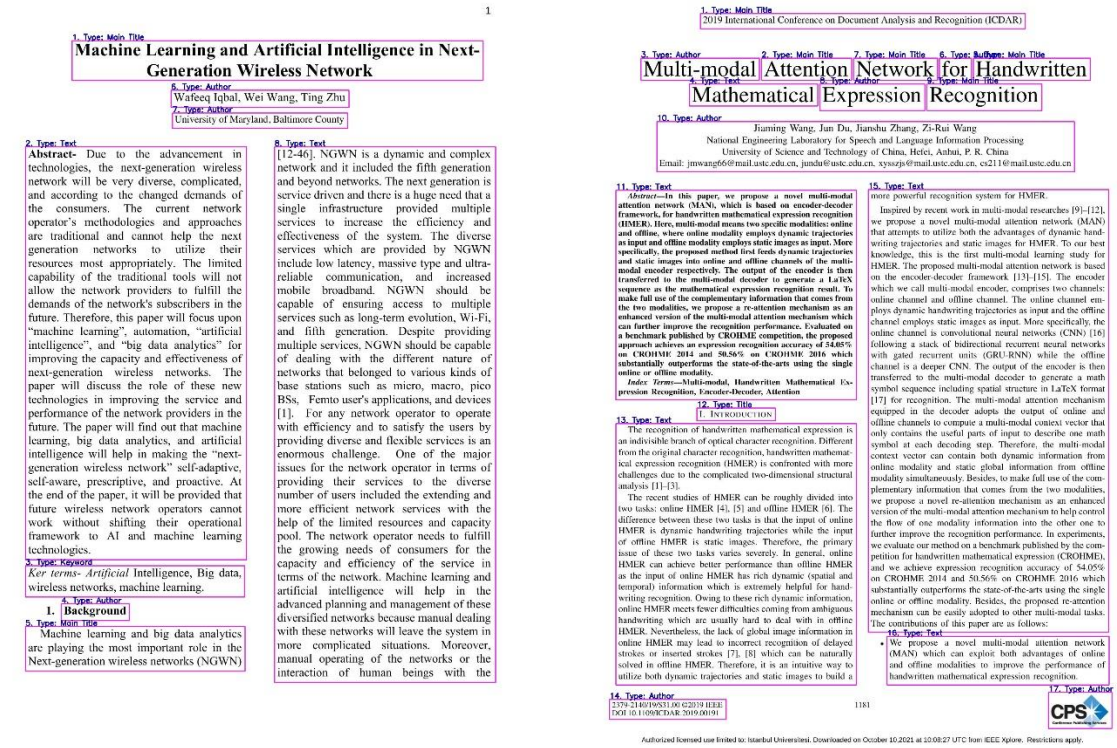
4.4. SAYFA TİPİNİN BULUNMASI

Sayfa tipinin bulunması için uygulanan makine öğrenmesi modeli yapılan deneylerde tamamen doğru sonuçlar vermiştir. Hatalı bir tahmin yapılmamıştır.

4.5. SAYFA TİPİNE ÖZEL METİN SINIFLARININ BULUNMASI

4.5.1. İlk Sayfa

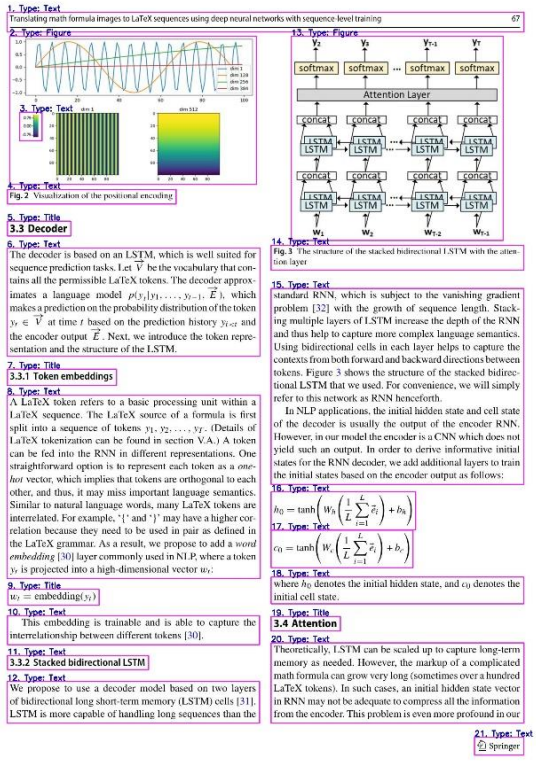
Sayfa tipi ilk sayfa olan bir kaynak dokümanlar için Şekil 15 (a)'da ve Şekil 15 (b)'de iki tane örnek sonuç görülmektedir. Ana başlık ve yazar bloklarının bulunmasında segmentasyon sorunları sınıflama işlemini olumsuz etkilemektedir. Segmentasyonda yaşanan hataların sistemin geri kalanını olumsuz etkilediği görülmüştür.



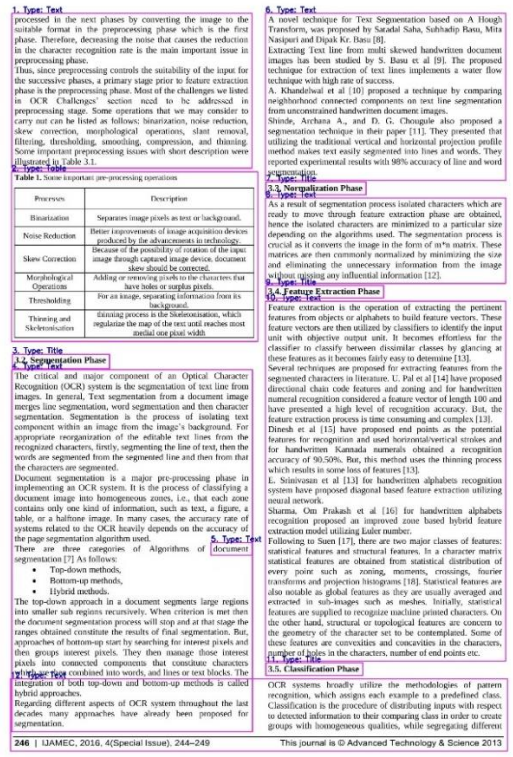
(a)

(b)

Şekil 15 : İlk Sayfa için Segmentasyon ve Metin Sınıflama Sonuçları



(a)



(b)

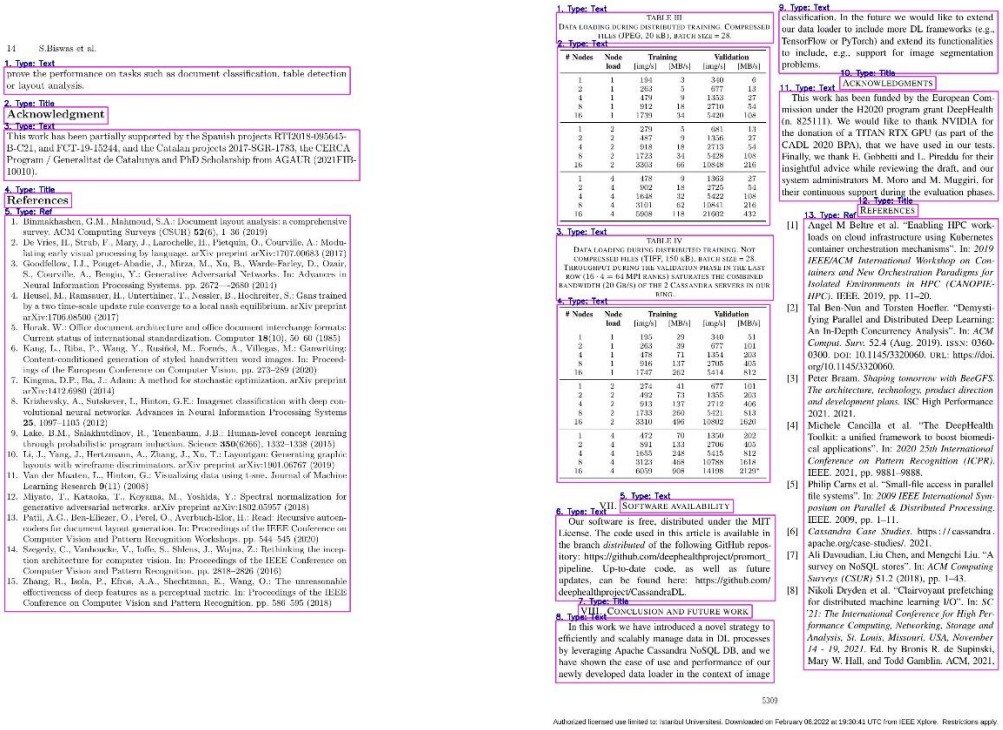
Şekil 16 : Ara Sayfa için Segmentasyon ve Metin Sınıflama Sonuçları

4.5.2. Ara Sayfa

Sayfa tipi ara sayfa olan bir kaynak dokümanlar için Şekil 16 (a)'da ve Şekil 16 (b)'de iki tane örnek sonuç görülmektedir. Deneyler göstermiştir ki liste ve tablo bloklarının tespit edilmesinde tekrar edilebilir bir başarı görülmemiştir.

4.5.3. Son Sayfa

Sayfa tipi son sayfa olan bir kaynak dokümanlar için Şekil 17 (a)'da ve Şekil 17 (b)'de iki tane örnek sonuç görülmektedir. Deneyler göstermiştir ki tablo ve referans blokları metin blokları ile karıştırılmaktadır. Bu blokların tespit edilmesinde tekrar edilebilir bir başarı görülmemiştir.



(a) **(b)**

Şekil 17 : Son Sayfa için Segmentasyon ve Metin Sınıflama Sonuçları

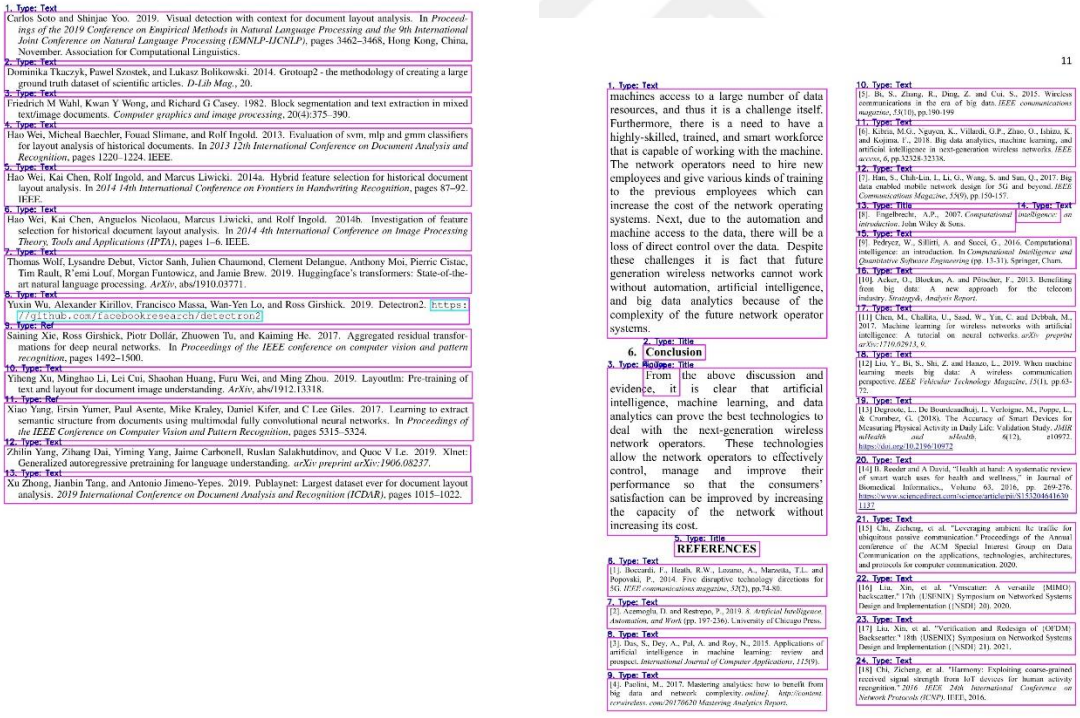
4.6. LATEX KODUNUN ÜRETİLMESİ

Uygulamada LaTex kodu heuristic olarak üretilmektedir. Bundan dolayı uygulamanın önceki aşamalarında oluşturulan çıktılar bu adımda kullanılmaktadır. Önceki adımlarda oluşmuş hatalar birikmeli bir şekilde bu adıma aktarıldığı için bu aşamayı ölçmek için doğru bir metrik bulunamamıştır.

4.7. UYGULAMANIN GENEL DEĞERLENDİRİLMESİ

Uygulama genel olarak değerlendirildiğinde sistemin bazı dokümanlarda çok başarılı olduğu ama bazı dokümanlarda başarısız olduğu görülmüştür. Bunun sebepleri arasında sistemin birçok aşamadan oluşan kompleks yapıda olmasıdır. Bu durum bir noktada oluşan hatanın birikmeli olarak sistemin geri kalanını olumsuz etkilemesine sebep olmuştur.

Akademik bir yayın doğası gereği ağırlıklı olarak metin bloklarından oluşmaktadır. Bu metin blokları kendi içinde sınıflara ayrılmaktadır. Çoğu zaman bu metin bloklarını birbirinden ayırmak oldukça zor bir iştir. Şekil 18 (a)'da ve Şekil 18 (b)'de örnek olarak referans sayfaları verilmiştir. Birbirinden farklı referans sistemleri olması işi zorlaştırmaktadır. Referans bloğunu bir listeden ya da normal bir paragraf yapısından ayırmak için NLP metotları kullanılmasına rağmen bu zor bir görevdir.



(a) **(b)**

Şekil 18 : Örnek Metin Blokları

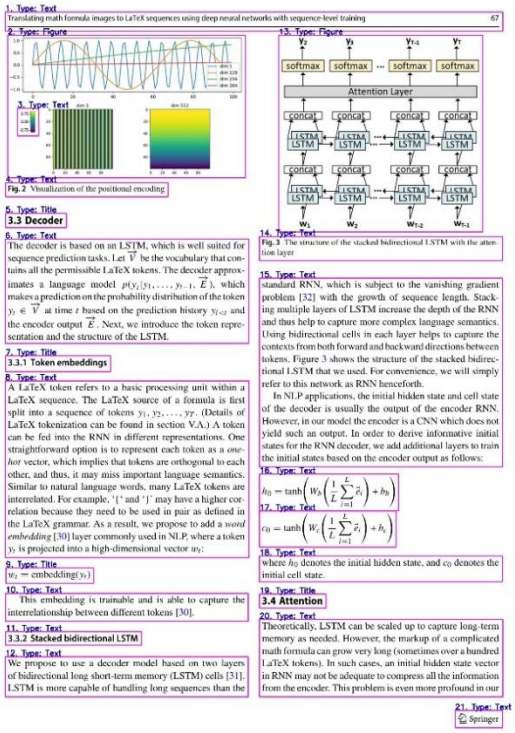
Sistemdeki NLP uygulamaları düşünüldüğünde her bir metin sınıfı için fazlasıyla opsiyon olması sistemin gerçekleştirilmesini daha da zorlaştırmaktadır.

5. TARTIŞMA

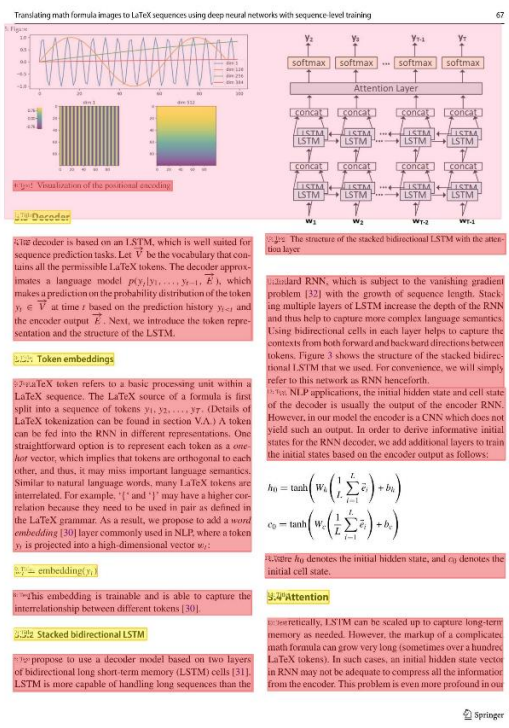
Literatür taraması ve incelenen ticari uygulamalarda hep bir konu üzerine odaklanıldığı görülmüştür. Örneğin sadece denklemlerin LaTex koduna dönüştürülmesi üzerine geliştirilmiş ticari bir çözümde tarayıcıda kullanıcı tarafından seçmek suretiyle işaretlenen denklem bloğu LaTex koduna dönüştürülmektedir. Bu şekilde başarılı sonuçlar elde edilen çalışmalar mevcuttur. Ayrıca PDF dosyalarından elde edilen meta veriler ile JSON formatında çıktı üreten çözümlerde mevcuttur. Ancak süreç bu çalışmada geliştirilen uygulamayla farklı bir çözüm tekniği uygulanmaktadır. Bunlar dışında çok basit düzeyde OCR ile elde edilen metni yine heuristic olarak LaTex koduna dönüştüren çalışmalar vardır. Ancak bunlarda yine bu çalışmada yapılanlar ile kıyaslanamayacak düzeyde basit kalmışlardır.

Bu çalışmada yapılanlar ile karşılaştırılmak üzere yüksek benzerlik olan bir çalışma bulunmamaktadır. Ancak bu çalışma için veri kümesi oluşturmada faydalanılan derin öğrenme metotu kullanan LayoutParser kütüphanesi ile segmentasyon ve blokların sınıflandırılması açısından karşılaştırma yapmak mümkün olmaktadır. LayoutParser farklı ön eğitimli modelleri kullanabilmektedir. Bizim çalışmamıza uygun olarak PublayNet modeli tercih edilmiştir. Bu model IBM laboratuvarları tarafından 1 milyondan fazla akademik yayın kullanılarak eğitilmiştir (PubLayNet: Largest Dataset Ever For Document Layout Analysis).

Karşılaştırma işlemi için ara sayfa olarak sınıflanmış bir doküman üzerinden karşılaştırma yapılması uygun olacaktır. Çünkü geliştirilen uygulamanın ilk sayfa ve son sayfa olarak sınıflandırılan dokümanlar için metin sınıfları benzersiz olmuştur. Şekil 19 (a)'da çalışmada geliştirilen sistem ve Şekil 19 (b)'de LayoutParser kullanılan modelin sonucu karşılaştırılması görülmektedir.

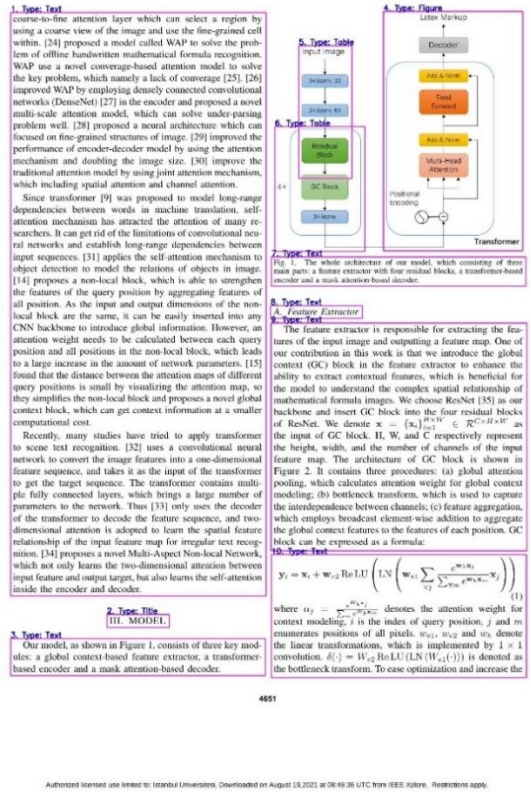


(a)

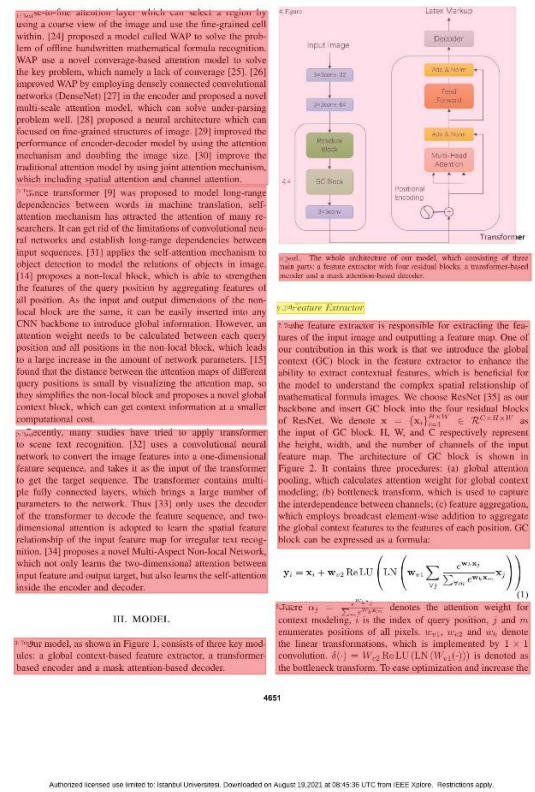


(b)

Şekil 19 : Geliştirilen Uygulama Sonucu (a) ve LayoutParser Sonucu (b) Karşılaştırması



(a)



(b)

Şekil 20 : Geliştirilen Uygulama Sonucu (a) ve LayoutParser Sonucu (b) Karşılaştırması

Şekil 19 (a)'da ve Şekil 19 (b)'de verilen sistem çıktıları incelendiğinde LayoutParser hem segmentasyonda hata yapmıştır. Hem de dikkate almadığı içerik olmuştur. Bu örnekler üzerinde bizim yaptığımız çalışma daha başarılıdır.

Şekil 20 (a)'da ve Şekil 20 (b)'de çıktılar incelendiğinde bizim çalışmamız şekle hatalı segmentasyon yapmıştır. Ancak LayoutParser çıktısı bir başlığı atlamış ve kapsama almadığı içerik olmuştur.

Genel olarak karşılaştırma yapıldığında örnekler üzerinde bazen bizim çalışmamız bazen de LayoutParser daha başarılı olmuştur. Ancak bizim geliştirdiğimiz sistem okuma sırası ve sayfa tipine özel metin sınıfları ile daha kapsamlı bir çalışma olmuştur.

6. SONUÇ VE ÖNERİLER

Bu çalışmanın ana unsurlarından birini oluşturan bilgisayarlı görü üzerine 50 yıldan fazladır sayısız çalışma yapılmıştır. Ancak hala genel bir çözüm üretilememiştir. Yaptığımız çalışmada uygulanan bilgisayarlı görü yöntemi ile belirli bir başarı yakalanmış olsa da sistem farklı örneklerde farklı başarı düzeyinde değerler elde etmiştir. Çalışmaların devam ettirilmesi durumunda daha genel bir çözüme varılabileceği düşünülmektedir.

Doğal dil işlemede makine öğrenmesi modellerinin modası geçmiş olarak değerlendirilmektedir. Bu çalışmayı geliştirme aşamasında yapılan deneylerde anahtar kelimeler barındıran veri kümelerinde makine öğrenmesi modellerinin hala kısa eğitim süresi, küçük model boyutu ve hızlı cevap süresi ile değerlendirilebilecekleri gösterilmiştir. Ayrıca popüler NLP modellerinin eğitim süresi ve model boyutları göz önüne alındığında bu çalışma için geliştirilen NLP modellerinin çok pratik olduklarını söylemek mümkündür.

Çalışmanın geliştirilme aşamasında konu ile ilgili kapsamın çok geniş olduğu görülmüştür. Genelde incelenen çalışmaların konun tek bir alanı ile ilgili oldukları görülmüştür. Bu çalışmada akademik yayınlar üzerinde detaylı olarak bilgisayarlı görü ve doğal dil işleme uygulamaları yapılmıştır. Bu sebeple bu çalışma gelecekte geliştirilmeye devam edilebileceği ya da başka çalışmalara bir ortam hazırlayabileceği düşünülmektedir.

KAYNAKLAR

[1]. T. S. Huang, "Computer Vision: Evolution and Promise," *1996 CERN School of Computing*, pp. 21–25, 1996, doi: 10.5170/CERN-1996-008.21.

[2]. M. Furkan Keskenler and E. Fahri Keskenler, "Geçmişten Günümüze Yapay Sinir Ağları ve Tarihçesi," *Takvim-i Vekayi*, vol. 2, no. 5, pp. 8–18, 2017.

[3]. P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: an introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, Sep. 2011, doi: 10.1136/amiajnl-2011-000464.

[4]. D. W. Otter, J. R. Medina, and J. K. Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, 2021, doi: 10.1109/TNNLS.2020.2979670.

[5]. "OpenCV." https://opencv.org/ [Ziyaret Tarihi 22 Mayıs 2022].

[6]. CTAN Team, "What are TEX and its friends?" https://www.ctan.org/tex [Ziyaret Tarihi 22 Mayıs 2022].

[7]. X. Zhong, J. Tang, and A. J. Yepes, "PubLayNet: largest dataset ever for document layout analysis," Aug. 2019, [Çevrimiçi]. Available: http://arxiv.org/abs/1908.07836

[8]. "PRImA." https://www.primaresearch.org/ [Ziyaret Tarihi 22 Mayıs 2022].

[9]. "Newspaper Navigator." https://news-navigator.labs.loc.gov/ [Ziyaret Tarihi 22 Mayıs 2022].

[10]. M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, and Z. Li, "TableBank: A Benchmark Dataset for Table Detection and Recognition," Mar. 2019, Accessed: Feb. 06, 2022. [Çevrimiçi]. arXiv1903.01949

[11]. Z. Wang, Y. Xu, L. Cui, J. Shang, and F. Wei, "LayoutReader: Pre-training of Text and Layout for Reading Order Detection," Aug. 2021, Accessed: Oct. 20, 2021. [Çevrimiçi]. arXiv:2108.11591

[12]. Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLM: Pre-training of Text and Layout for Document Image Understanding," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2020, vol. 20, pp. 1192–1200. doi: 10.1145/3394486.3403172.

[13]. C. Xu *et al.*, "A Page Object Detection Method Based on Mask R-CNN," *IEEE Access*, vol. 9, pp. 143448–143457, 2021, doi: 10.1109/ACCESS.2021.3121152.

[14]. C. Clark and S. Divvala, “PDFFigures 2.0,” in *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, Jun. 2016, pp. 143–152. doi: 10.1145/2910896.2910904.

[15]. S. Deivalakshmi, P. Palanisamy, and G. Vishwanathan, “A novel method for text and non-text segmentation in document images,” in *2013 International Conference on Communication and Signal Processing*, Apr. 2013, pp. 255–259. doi: 10.1109/iccsp.2013.6577054.

[16]. I. Kavasidis *et al.*, “A Saliency-Based Convolutional Neural Network for Table and Chart Detection in Digitized Documents,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11752 LNCS, pp. 292–302, Sep. 2019, doi: 10.1007/978-3-030-30645-8_27.

[17]. Y. Deng, D. Rosenberg, and G. Mann, “Challenges in End-to-End Neural Scientific Table Recognition,” in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, Sep. 2019, pp. 894–901. doi: 10.1109/ICDAR.2019.00148.

[18]. Z. Yan, X. Zhang, L. Gao, K. Yuan, and Z. Tang, “ConvMath: A Convolutional Sequence Network for Mathematical Expression Recognition,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, Jan. 2021, pp. 4566–4572. doi: 10.1109/ICPR48806.2021.9412913.

[19]. Z. Wang and J. C. Liu, “Translating math formula images to LaTeX sequences using deep neural networks with sequence-level training,” *International Journal on Document Analysis and Recognition*, vol. 24, no. 1–2, pp. 63–75, 2021, doi: 10.1007/s10032-020-00360-2.

[20]. N. Pang, C. Yang, X. Zhu, J. Li, and X.-C. Yin, “Global Context-Based Network with Transformer for Image2latex,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, Jan. 2021, pp. 4650–4656. doi: 10.1109/ICPR48806.2021.9412072.

[21]. Z. Wang *et al.*, “DeepFont: Identify Your Font from An Image,” *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 451–459, 2015, doi: 10.1145/2733373.

[22]. G. Chen *et al.*, “Large-Scale Visual Font Recognition,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 3598–3605. doi: 10.1109/CVPR.2014.460.

[23]. H. Ding, K. Chen, and Q. Huo, “Compressing CNN-DBLSTM models for OCR with teacher-student learning and Tucker decomposition,” *Pattern Recognition*, vol. 96, p. 106957, Dec. 2019, doi: 10.1016/j.patcog.2019.07.002.

[24]. B. Safnuk and G. Hu, “Reconstructing LaTeX Source Files from Generated PDFs - a Neural Network Approach,” in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, Jul. 2018, pp. 890–895. doi: 10.1109/INDIN.2018.8472050.

[25]. I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, “A brief introduction to OpenCV,” 2012.

[26]. R. Smith, “An Overview of the Tesseract OCR Engine,” in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, Sep. 2007, vol. 2, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.

[27]. Q. Liu and Y. Wu, “Supervised Learning,” in *Encyclopedia of the Sciences of Learning*, Boston, MA: Springer US, 2012, pp. 3243–3245. doi: 10.1007/978-1-4419-1428-6_451.

[28]. B. Sun, J. Du, and T. Gao, “Study on the Improvement of K-Nearest-Neighbor Algorithm,” in *2009 International Conference on Artificial Intelligence and Computational Intelligence*, 2009, vol. 4, pp. 390–393. doi: 10.1109/AICI.2009.312.

[29]. M. Brijain, R. Patel, M. Kushik, and K. Rana, “A Survey on Decision Tree Algorithm For Classification,” *International Journal of Engineering Development and Research (www.ijedr.org)*, vol. 2, p. 1, 2014, doi: 10.1.1.673.2797.

[30]. J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” *Neurocomputing*, 2020, doi: 10.1016/J.NEUCOM.2019.10.118.

[31]. S. Elbagir and J. Yang, “Sentiment Analysis of Twitter Data Using Machine Learning Techniques and Scikit-learn,” in *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence*, Dec. 2018, pp. 1–5. doi: 10.1145/3302425.3302492.

[32]. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[33]. Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, and Brown, “Text Classification Algorithms: A Survey,” *Information*, vol. 10, no. 4, p. 150, Apr. 2019, doi: 10.3390/info10040150.

[34]. A. Tharwat, “Classification assessment methods,” *Applied Computing and Informatics*, vol. 17, no. 1, pp. 168–192, Jan. 2021, doi: 10.1016/j.aci.2018.08.003.

[35]. E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Considerations for Deep Learning in NLP,” Jun. 2019, [Çevrimiçi]. arXiv:1906.02243

[36]. “LaTex.” https://www.latex-project.org/ [Ziyaret Tarihi 22 Mayıs 2022].

[37]. “IEEE xplore.” https://ieeexplore.ieee.org/Xplore/home.jsp [Ziyaret Tarihi 22 Mayıs 2022].

[38]. “Science Direct.” https://www.sciencedirect.com/ [Ziyaret Tarihi 22 Mayıs 2022].

[39]. “ArXiv.” https://arxiv.org/ [Ziyaret Tarihi 22 Mayıs 2022].

[40]. “Springer.” https://link.springer.com/ [Ziyaret Tarihi 22 Mayıs 2022].

[41]. “ACM DL.” https://dl.acm.org/ [Ziyaret Tarihi 22 Mayıs 2022].

[42]. “Google Colabratory.” https://colab.research.google.com/ [Ziyaret Tarihi 22 Mayıs 2022].

[43]. “Natural Language Toolkit.” https://www.nltk.org/ [Ziyaret Tarihi 22 Mayıs 2022].

İNTİHAL RAPORU İLK SAYFASI

UYARI: Bu bölümde intihal programında çıkan benzerlik yüzdesinin bulunduğu detaylı raporun ilk sayfasının yer alması tüm öğrenciler için zorunludur! Bu uyarıyı okuduktan sonra siliniz!

KURUM İZNİ YAZILARI

Uyarı: Canlı ve cansız deneklerle yapılan tüm çalışmalar için kurum izin belgelerinin eklenmesi zorunludur. Gizlilik ve mahremiyet içeren durumlarda kurum adı kapatılmalıdır.

☐ **Kurum izni gerekmektedir.**

☒ **Kurum izni gerekmemektedir.**

Murat KAZANÇ
(İmza)