



T.C.
ALTINBAŞ ÜNİVERSİTESİ
Lisansüstü Eğitim Enstitüsü
Bilişim Teknolojileri

**DERİN ÖĞRENME ALGORİTMALARI
KULLANILARAK KARAKTER TABANLI
ANLAMLI KELİME ÜRETİMİ**

İsa ERGİN

Yüksek Lisans Tezi

Danışman

Dr.Öğr.Üyesi Timur İNAN

İstanbul, 2023

**DERİN ÖĞRENME ALGORİTMALARI KULLANILARAK
KARAKTER TABANLI ANLAMLI KELİME ÜRETİMİ**

İsa ERGİN

Bilişim Teknolojileri

Yüksek Lisans Tezi

ALTINBAŞ ÜNİVERSİTESİ

2023

“İsa ERGİN” tarafından hazırlanmış ve 22.12.2023 tarihinde sunulmuş “DERİN ÖĞRENME ALGORİTMALARI KULLANILARAK KARAKTER TABANLI ANLAMLI KELİME ÜRETİMİ” başlıklı tez Bilişim Teknolojileri Ana Bilim Dalında Yüksek Lisans Tezi olarak **oy birliği** ile kabul edilmiştir.

Dr. Öğr. Üyesi Timur İNAN
Danışman

Tez Savunma Sınavı Jüri Üyeleri:

Dr. Öğr. Üyesi Timur İNAN

Yazılım Mühendisliği
Bölümü,

Altınbaş Üniversitesi

Doç. Dr. Oğuz ATA

Yazılım Mühendisliği
Bölümü,

Altınbaş Üniversitesi

Doç. Dr. Kazım YILDIZ

Bilgisayar Mühendisliği
Bölümü,

Marmara Üniversitesi

Bu tezin Yüksek Lisans. tezi olarak bütün şartları sağladığını beyan ederim.

Tezin Lisansüstü Eğitim Enstitüsüne teslim tarihi: ____/____/____

Bu tezdeki tüm bilgilerin akademik kurallara ve etik davranışlara uygun olarak edinildiğini ve sunulduğunu beyan ederim. Ayrıca, bu kuralların ve davranışların gerektirdiği şekilde, bu çalışmada, orijinal olmayan tüm materyalleri ve sonuçları tamamen alıntı yaptığımı ve referans gösterdiğimi de beyan ederim

İsa ERGİN



İTHAF

Tez yazım sürecimin her aşamasında anlayışlı tutumuyla, kıymetli görüş, öneri ve değerlendirmeleri ile yol göstererek teze önemli katkılarda bulunan değerli danışmanım Sayın Dr.Öğr.Üyesi Timur İNAN'a,

Lisansüstü öğrenim hayatım süresince bilgi ve tecrübeleri ile beni aydınlatan tüm hocalarıma, en içten sevgi, saygı ve teşekkürlerimi sunuyorum.



ÖZET

DERİN ÖĞRENME ALGORİTMALARI KULLANILARAK KARAKTER TABANLI KELİME ÜRETİMİ

ERGİN, İsa

Yüksek Lisans Tezi, Bilişim Teknolojileri Anabilim Dalı, Altınbaş Üniversitesi

Danışman: Dr. Öğr. Üyesi Timur İNAN

Tarih: Aralık / 2023

Sayfa: 106

Bu çalışmanın amacı derin öğrenme algoritmaları kullanılarak karakter tabanlı bir dil modeli geliştirmek ve Türkçe dil bilgisi kurallarına uygun anlamlı kelime üretmektir. Metin üretimi, bilgisayarların insanların iletişim kurma şeklini taklit ederek doğal dilde metin oluşturmak için kullanılmaktadır. Doğal dilde metin verileri yan yana olarak ve zaman sırasına göre sıralanmış birbiriyle ilişkili kelime veya kelime gruplarının oluşturduğu bütünsel bir düzen şeklinde bulunmaktadır. Metin içerisinde bulunan karakterlerin ve kelimelerin sırasının ve anlamının önemi çok büyüktür. Bu nedenle oluşturulacak kelime, cümle ve metin parçalarında karakterler veya kelimeler arasındaki ilişkilerin iyi öğrenilebilmesi için geçmiş bilgileri hatırlayarak öğrenen derin öğrenme algoritmalarının kullanılması gerekmektedir. Derin öğrenme algoritmalarından özyinelemeli yapay sinir ağları geçmiş bilgileri hatırlayarak sıralı örüntüler oluşturmada başarılı sonuçlar verdiği için bu alanda etkili bir şekilde kullanılmaktadır. Çalışmada LSTM, GRU, Kodlayıcı-Kod Çözücü ve dikkat mimarisi kullanılarak Türkçe dil bilgisi kurallarına uygun anlamlı kelimeler üretilmektedir. Çalışmada özellikle Kodlayıcı-Kod Çözücü ve dikkat modeline ağırlık verilmektedir. Nedeni ise diğer modellerin (LSTM/GRU) giriş ve çıkış verilerinin uzunluğu arttıkça karakterler arasındaki ilişkileri hatırlamakta yetersiz kalmalarıdır. Bu sorunların üstesinden gelebilmek için daha uzun girdi ve çıktı dizelerinde daha başarılı sonuçlar veren Kodlayıcı-Kod Çözücü ve Dikkat mimarisi kullanılmaktadır. Çalışmada her bir model 100, 300, 500 epoch değerlerinde sıcaklık örnek alma yönteminin farklı eşik değerlerinde

çalıştırılmaktadır. LSTM modeli 100 epoch ve sıcaklık örnek alma yönteminin 0.5 eşik değerinde 91,3% başarı oranı ile en iyi sonucu, GRU modeli 300 epoch ve sıcaklık örnek alma yönteminin 0.5 eşik değerinde 93.8% başarı oranı ile en iyi sonucu, Kodlayıcı-Kod Çözücü ve Dikkat modeli 500 epoch ve sıcaklık örnek alma yönteminin 0.2 eşik değerinde 95.7 % başarı oranı ile en iyi sonucu vermektedir. Çalışma sonunda kodlayıcı-kod çözücü ve dikkat mimarisi kullanılarak oluşturulan dil modelinin anlamlı kelime üretiminde ve daha uzun kelime gruplarının oluşturulmasında diğer modellere (LSTM,GRU) göre daha yüksek başarı elde edildiği görülmektedir. Karakter tabanlı geliştirilen modellerin Türkçe dil bilgisi kurallarına uygun kelime ve kelime grupları üretmede başarılı olduğu görülmektedir.

Anahtar Kelimeler: Yapay Zeka, Doğal Dil İşleme, Derin Öğrenme, LSTM, Metin Üretimi.

ABSTRACT

USING DEEP LEARNING ALGORITHMS CHARACTER-BASED WORD GENERATION

ERGİN, İsa

M.Sc., Computer Engineering, Altınbaş University,

Supervisor: Dr. Öğr. Üyesi Timur İNAN

Date: December / 2023

Pages: 106

The aim of this study is to develop a character-based language model using deep learning algorithms and to produce meaningful words in accordance with Turkish grammar rules. Text generation is used to create text in natural language by imitating the way computers communicate. In natural language, text data exists side by side and in the form of a holistic arrangement of related words or phrases arranged in chronological order. The order and meaning of the characters and words in the text are of great importance. For this reason, deep learning algorithms that learn by remembering past information should be used in order to learn the relationships between characters or words in words, sentences and text fragments to be created. Recursive artificial neural networks, one of the deep learning algorithms, are used effectively in this field because they give successful results in creating sequential patterns by remembering past information. In the study, meaningful words in accordance with Turkish grammar rules are produced by using LSTM, GRU, Encoder-Decoder and attention architecture. In the study, especially the Encoder-Decoder and attention model are emphasized. The reason is that other models (LSTM/GRU) fail to remember the relationships between characters as the length of the input and output data increases. In order to overcome these problems, the Encoder-Decoder and Attention architecture is used, which gives more successful results in longer input and output strings. In the study, each model is operated at different threshold values of the temperature sampling method at 100, 300, 500 epoch values. LSTM model 100 epoch and temperature sampling method best result at 0.5

threshold with 91.3% success rate, GRU model 300 epoch and temperature sampling method best result with 93.8% success rate at 0.5 threshold, Encoder-Decoder and Attention model gives the best result with a success rate of 95.7% at 500 epochs and 0.2 thresholds of the temperature sampling method. At the end of the study, it is seen that the language model created by using encoder-decoder and attention architecture has higher success in producing meaningful words and creating longer word groups than other models (LSTM, GRU). It is seen that the character-based models are successful in producing words and word groups in accordance with Turkish grammar rules.

Keywords: Artificial Intelligence, Natural Language Processing, Deep Learning, LSTM, Text Generation.



İÇİNDEKİLER

Sayfa

ÖZET	vi
ABSTRACT	viii
TABLO LİSTESİ.....	xiii
ŞEKİL LİSTESİ	xiv
KISALTMALAR.....	xvi
SEMBOL LİSTESİ	xvii
1. GİRİŞ VE AMAÇ	1
1.1 LİTERATÜR ARAŞTIRMASI	3
1.2 TEZİN AMACI.....	6
1.3 TEZİN ORGANİZASYONU.....	7
2. GENEL BİLGİLER.....	8
2.1 YAPAY ZEKA	8
2.1.1 Dar (Zayıf) Yapay Zeka	9
2.1.2 Genel Yapay Zeka	9
2.1.3 Süper Yapay Zeka	10
2.2 MAKİNE ÖĞRENMESİ	10
2.2.1 Makine Öğrenmesi Modelleri	11
2.2.1.1 Denetimli öğrenme	11
2.2.1.2 Denetimsiz öğrenme	12
2.2.1.3 Yarı denetimli öğrenme	13
2.2.1.4 Pekiştirmeli öğrenme	14
2.3 YAPAY SİNİR AĞLARI	14
2.3.1 İleri Beslemeli YSA	18
2.3.2 Geri Beslemeli YSA.....	18
2.3.3 YSA Öğrenme Kuralları.....	19
2.3.3.1 Hepp öğrenme kuralı	20

2.3.3.2	Delta kuralı	20
2.3.3.3	Hopfield kuralı	20
2.3.3.4	Kohonen kuralı	20
2.4	DERİN ÖĞRENME	20
2.5	DOĞAL DİL İŞLEME	21
2.5.1	Doğal Dil Anlama	22
2.5.2	Doğal Dil Üretimi	22
2.5.3	Kural Tabanlı Sistemler	22
2.5.4	Makine Öğrenimi Tabanlı Sistemler	22
2.5.5	Veri Ön İşleme	23
2.5.5.1	Gürültülü veri temizleme	24
2.5.5.2	Duraklama kelimelerinin kaldırılması	24
2.5.5.3	Kelime köklerinin bulunması	24
2.5.5.4	Simgeleme-Parçalama	24
2.5.5.5	N-gram özelliklerini ayıklama	25
2.5.6	Kelime Vektörleri	25
2.5.6.1	Geleneksel (frekans) tabanlı kelime gösterimi	25
2.5.6.2	YSA tabanlı kelime temsil yöntemleri	29
2.6	NLP'DE KULLANILAN DERİN ÖĞRENME ALGORİTMALARI	31
2.6.1	Tekrarlayan Sinir Ağları	31
2.6.2	Uzun Kısa Süreli Bellek	34
2.6.3	Geçitli Tekrarlayan Birim	36
2.6.4	Üretken Düşman Ağları	37
2.6.5	Kodlayıcı ve Kod Çözücü Modeller	39
2.6.6	Dikkat Mekanizması	40
2.6.7	Transformer Network	41
2.6.8	Dil Modellerinden Kelime Gömmeleri (ELMO)	41
2.6.9	Transformers'dan Çift Yönlü Kodlayıcı Temsilleri (BERT)	41

2.6.10 Üretken Ön İşlemeli Dönüştürücü Model (GPT-3).....	42
3. GEREÇ VE YÖNTEM.....	43
3.1 NLP DİL MODELLERİ	43
3.2 KULLANILACAK YÖNTEMLERİN İNCELENMESİ.....	44
3.2.1 Kodlayıcı Kod Çözücü Modeller	45
3.2.2 Dikkat Mekanizması	47
3.2.2.1 İçerik vektörünün hesaplanması	48
3.3 MODELİN OLUŞTURULMASI.....	50
3.3.1 Metin Üretimi.....	50
3.3.1.1 Kontrol edilebilir metin üretimi	50
3.3.1.2 Otomatik metin üretimi	50
3.3.1.3 Karakter tabanlı ve kelime tabanlı metin oluşturma.....	51
3.3.1.4 Dil modeli.....	52
3.3.1.5 Örnek alma (sampling) yöntemleri.....	53
3.3.2 Kodlama Yapısı ve Mimarisi	58
4. BULGULAR.....	65
5. TARTIŞMA VE SONUÇ.....	82
REFERANSLAR	84

TABLO LİSTESİ

Sayfa

Tablo 2.1: Kelime Torbası Vektör Dönüşüm Örneği.	26
Tablo 2.2: N-Gram Harf Özellik Çıkarımı.	28
Tablo 2.3: N-Gram Kelime Özellik Çıkarımı.	28
Tablo 4.1: LSTM Modelinin 100 Epoch Sonuçları.	67
Tablo 4.2: LSTM Modelinin 300 Epoch Sonuçları.	68
Tablo 4.3: LSTM Modelinin 500 Epoch Sonuçları.	69
Tablo 4.4: GRU Modelinin 100 Epoch Sonuçları.	72
Tablo 4.5: GRU Modelinin 300 Epoch Sonuçları.	73
Tablo 4.6: GRU Modelinin 500 Epoch Sonuçları.	74
Tablo 4.7: Kodlayıcı Kod Çözücü ve Dikkat Alg. Modelinin 100 Epoch Sonuçları.	77
Tablo 4.8: Kodlayıcı Kod Çözücü ve Dikkat Alg. Modelinin 300 Epoch Sonuçları.	78
Tablo 4.9: Kodlayıcı Kod Çözücü ve Dikkat Alg. Modelinin 500 Epoch Sonuçları.	79

ŞEKİL LİSTESİ

	<u>Sayfa</u>
Şekil 2.1: Turing Testi	8
Şekil 2.2: Makine Öğrenmesi Yöntemleri.....	11
Şekil 2.3: Yapay Sinir Ağı	15
Şekil 2.4: YSA Modelinin Matematiksel İşleyişi.....	16
Şekil 2.5: Aktivasyon Fonksiyonları	17
Şekil 2.6: Mimari Yapılarına Göre YSA.....	17
Şekil 2.7: İleri Beslemeli YSA	18
Şekil 2.8: Geri Beslemeli YSA.....	18
Şekil 2.9: YSA Kara Kutu Yakıştırması.....	19
Şekil 2.10: Doğal Dil İşleme Aşamaları.....	23
Şekil 2.11: RNN ve İleri Beslemeli YSA Gösterimi.....	32
Şekil 2.12: Rnn Modeli.....	32
Şekil 2.13: Rnn Model İç Yapısı.....	33
Şekil 2.14: LSTM Modeli.....	34
Şekil 2.15: GRU Modeli.....	37
Şekil 2.16: GAN Mimarisi.....	38
Şekil 3.1: Kodlayıcı Kod Çözücü Model.....	46
Şekil 3.2: Kodlayıcı Kod Çözücü Örneği.....	46
Şekil 3.3: Kodlayıcı Kod Çözücü ve Dikkat Mimarisi.....	48
Şekil 3.4: Otomatik Metin Üretimi Örneği.....	50

Şekil 3.5: Dil Modeli Örneği.	52
Şekil 3.6: Karakter Tabanlı Dil Modeli Çalışma Örneği.	53
Şekil 3.7: Aç Gözlü Arama Örneği.	54
Şekil 3.8: Sıcaklık Örnekleme Örneği.	55
Şekil 3.9: Top K Örneği.	57
Şekil 3.10: Veri Hattı.	59
Şekil 3.11: LSTM Modeli Parametreleri.	60
Şekil 3.12: GRU Modeli Parametreleri.	61
Şekil 3.13: Kodlayıcı Kod Çözücü ve Dikkat Modeli Parametreleri.	62
Şekil 3.14: Kodlayıcı Kod Çözücü ve Dikkat Modeli Parametreleri.	64
Şekil 4.1: LSTM Karakter Tabanlı Modele Ait Sonuç Grafiği.	70
Şekil 4.2: GRU Karakter Tabanlı Sonuç Grafiği.	76
Şekil 4.3: Karakter Tabanlı Dikkat Sonuç Grafiği.	80

KISALTMALAR

NLP	:	Natural Language Processing (Doğal Dil İşleme)
YSA	:	Artificial Neural Network (Yapay Sinir Ağları)
RNN	:	Recurrent Neural Network (Yinelemeli Sinir Ağı)
LSTM	:	Long-Short Term Memory (Uzun Kısa Süreli Bellek)
GRU	:	Gated Recurrent Unit (Geçitli Tekrarlayan Birim)



SEMBOL LİSTESİ

$\emptyset$	: Aktivasyon Fonksiyonu
h_t	: Hidden State – Gizli Durum Vektörü
x_t	: Girdi Vektörü
U	: Girdi Ağırlığı
y_t	: Çıktı Vektörü
i_t	: Giriş / Güncelleme Kapısının Aktivasyon Vektörü
o_t	: Çıkış Kapısının Aktivasyon Vektörü
f_t	: Unut Kapısı Vektörü
σ	: Sigmoid Aktivasyon Fonksiyonu
r_t	: Sıfırlama Kapısı Vektörü
z_t	: Güncelleme Kapısı Vektörü

1. GİRİŞ VE AMAÇ

Doğal dil işleme (Natural Language Processing-NLP) insanların yazılı ve sözlü olarak kullandığı doğal dillerin bilgisayarlar tarafından analiz edilerek çözümlenmesini sağlayan yapay zekanın bir alt uygulama alanıdır [1]. NLP bir disiplin olarak uzun yıllardır gelişmektedir. 1960 yılında yapay zeka ve dilbilimin bir alt dalı olarak, doğal dilin otomatik olarak üretilmesi ve anlaşılmasındaki sorunları incelemek amacıyla oluşturulmuştur [2]. Doğal diller, insanlar tarafından konuşulan dillerdir. Doğal dilde üretilen ve bilgisayarlar aracılığıyla erişilebilen bilgilerin geniş boyutlara ulaşmasından dolayı doğal dil işleme ihtiyacı artarak devam etmektedir. NLP yapay zeka uygulamalarında en çok zorlanılan uygulamalar olarak görülmektedir. NLP'nin zor olmasının ana nedenleri insan dilinin belirsiz ve sürekli gelişim içinde olmasından kaynaklanmaktadır [1]. Doğal dili anlamak, yalnızca sözcükleri değil, sözcüklerin anlam yaratmak için birbirleriyle nasıl bağlantılı olduklarını da anlamaktır. İnsanlar kullandıkları dillerin değişimine ve gelişimine kolayca adapte olabilmekteyken makinaların bu değişime ayak uydurması hiç de kolay değildir. Bu durum NLP'nin makinalar için zor bir uygulama alanı olmasına sebep olmaktadır. NLP çalışmalarında genel olarak karşılaşılan birtakım zorluklar bulunmaktadır. Bunlardan bazıları kurlsız ve anlaşılmaz konuşmalar ve yazılar, atasözleri ve deyimler gibi kinayeli anlatımlar, kelimelerin farklı eklerle farklı anlamlara gelmesi, kelimelerin metin içinde farklı yerde kullanılmasına bağlı olarak anlamının değişmesi gibi temel zorluklardır [3]. İnsanlar tarafından kullanılan doğal dillerin bilgisayarlar tarafından anlaşılıp kullanılabilmesi için doğal dillerin yapısının incelenip analiz edilerek kurallarının belirlenmesi ve NLP algoritmalarının kullanabileceği bir formata dönüştürebilmesi gerekmektedir. Bu dönüştürme işlemlerini yapabilmek içinde dilbilim ve bilgisayar biliminin gücünden birlikte faydalanmak gerekmektedir [4].

NLP dil bilimi, istatistik, bilgisayar bilimi, makine öğrenmesi, yapay sinir ağları ve derin öğrenme olmak üzere farklı bilim dallarından beslenmektedir [3]. Özellikle makine öğrenmesi ve derin öğrenme algoritmalarının geliştirilmesi ile doğal dil işleme alanında çok farklı uygulamaların geliştirilmesi sağlanmıştır. NLP uygulamaları yapay zekanın ilk çıktığı yıllardan beri farklı yöntemler kullanılarak oluşturulmaya çalışılmaktadır. NLP'nin ilk evrelerinde kural tabanlı yöntemler kullanılarak NLP uygulamaları geliştirilmeye çalışılmıştır. Bu yöntemlerde daha çok metin içinde geçen kelimelerin, kelime gruplarının

ya da cümlelerin arandığı ve bu kavramlar görüldüğünde belirli kararların verildiği ve bilgisayar programlarının daha çok kullanıldığı kural yöntemli yaklaşımlardır. Bu yöntemlerde kuralların her zaman manuel olarak oluşturulması ve geliştirilmesi gerekmektedir [5]. Daha sonraki süreçlerde her geçen yıl büyüyen veri hacmi ve işlemcilerin hesaplayabilme kapasitelerinin artması, makine öğrenmesi algoritmalarının geliştirilmesi ve NLP uygulamalarında kullanılan derin öğrenme yöntemlerinin ortaya çıkması ile çok daha başarılı sonuçlar elde edilmiştir. Bu yaklaşım yöntemi dili analiz ederek, anlayarak öğrenen algoritmalara dayanmaktadır. Veri setini kullanarak veriler arasındaki ilişkilerden çıkarımlar yaparak ve istatistiksel yöntemler kullanarak öğrenmektedir. En önemli avantajı kendi kendine öğrenebilmesinde yatmaktadır. Bu nedenle bu yöntemlerde yüksek beceriler gerektiren manuel kural/dilbilgisi kodlamasına da gerek yoktur [6].

Metin üretimi, bilgisayarların, insanların doğal olarak iletişim kurma şeklini taklit ederek otomatik olarak doğal dilde metin oluşturmasını sağlamaktadır. Metin verileri yan yana olarak ve zaman sırasına göre sıralanmış birbiriyle ilişkili kelime veya kelime gruplarının oluşturduğu bütünsel bir düzen şeklinde bulunmaktadır [3]. Dil bilgisi kurallarına uygun tutarlı anlamlı metinlerin üretilmesi NLP'nin temel sorunlarından biridir. Metin üretiminde NLP alanında geliştirilen derin öğrenme algoritmaları sayesinde, doğal dillerin yapısını analiz ederek öğrenen modeller oluşturularak birbiriyle tutarlı anlamlı metinler üretilmeye çalışılmaktadır [7]. Metin üretiminde karakter ve kelimelerin doğru bir şekilde sıralanması için geçmiş bilgilerin hatırlanması çok önemlidir. Bundan dolayı metin üretiminde dil modeli oluştururken özyinelemeli yapay sinir ağları kullanılmaktadır. Çünkü genel yapay sinir ağları tüm girdilere eşit derecede dikkat etmekte ve girdi ile çıktı arasındaki ilişkinin zaman, sıra ve geçmiş bilgilerle olan bağlantılarını dikkate almadığı için girdi ve çıktıyı ilişkilendirememektedir. Bundan dolayı metin üretimi uygulamasında derin öğrenme algoritmalarından olan özyineleme yapay sinir ağları tercih edilmektedir. Bu sinir ağları adım adım çalışmakta ve bir adımı bitirmeden diğer adıma geçmemekte ve geçmiş bilgileri hatırlayarak öğrenmektedir [8].

Günümüzde metin üretimi uygulamalarında genellikle özyinelemeli sinir ağlarından olan tekrarlayan sinir ağları (Recurrent Neural Network-RNN), uzun kısa süreli bellek (Long Short-Term Memory-LSTM), Geçitli Tekrarlayan Birim (Gated Recurrent Unit-GRU), üretken düşman ağları (Generative Adversarial Network- GAN) kodlayıcı kod çözücü

modelleri ile birlikte dikkat mekanizması, ve transformer modelleri kullanılmaktadır [3]. Bu tez kapsamında LSTM/GRU ve özellikle kodlayıcı-kod çözücü ve dikkat mekanizması kullanılarak karakter tabanlı kelime üretimi uygulaması gerçekleştirilmektedir. Uygulamada özellikle kodlayıcı-kod çözücü ve dikkat mekanizmasının kullanılma sebebi olarak dikkat mekanizmasının daha önce NLP alanında kullanılan derin öğrenme algoritmalarının çalışma mantığını değiştirerek uzun metin dizelerinde daha iyi sonuçlar vermesidir. RNN, LSTM ve GRU modelleri daha çok küçük veri setlerinde başarılı olmakta, büyük veri setlerinde ve uzun metin dizelerinde çalışırken hafızaları yetmemekte ve bu nedenle kötü sonuçlar vermektedirler. Kodlayıcı ve kod çözücü modeller ise yapısında LSTM/GRU yapılarının farklı parametre ve yöntemlerle kullanılmasıyla eğitime çalışılmaktadır [9]. Kodlayıcı kod çözücü modellerin yapısında bulunan LSTM/GRU modelleri tüm giriş verilerini işleyerek verinin tüm bilgilerin özetini tutan bir çıkış özet vektörü oluşturmaktadır. Bu yaklaşımın çok fazla eksikliği bulunmaktadır. Kodlayıcı kod çözücü modeller giriş verisiyle ilgili yeterli bilgiye sahip olamadıklarında, kodlayıcıdan kod çözücüye giriş verisiyle ilgili yetersiz bilgi aktarılacağı için kod çözücü metin üretimi ile ilgili doğru tahminler yapamamaktadır. Dolayısıyla kodlayıcı-kod çözücü modellerinin giriş verilerin uzunluğu arttıkça performansı yetersiz kalmaktadır. Bu sorunların üstesinden gelebilmek için kodlayıcı ve kod çözücü modellerinin yapısında dikkat mimarisinin kullanılması önerilmiştir [10]. Bu yöntemde kodlayıcı kod çözücü modellerdeki gibi giriş verisinin tüm bilgisini artık tek bir özet vektör olarak oluşturmaya çalışılmamaktadır Kod çözücü artık sadece kodlayıcı vektöre bağlı olarak çıktı üretmek yerine her adımda çıktıyı üretirken giriş verilerinin farklı bölümlerine odaklanarak çıktı verisini üretmeye çalışmaktadır. Daha da önemlisi, modelin giriş cümlesine ve şu ana kadar ne ürettiğine bağlı olarak neye dikkat etmesi gerektiğini kendi kendine öğrenmesine izin vermektedir [11].

1.1 LİTERATÜR ARAŞTIRMASI

Literatür çalışması bölümünde metin üretimi alanında karakter ve kelime tabanlı olarak derin öğrenme algoritmaları kullanılarak yapılan çalışmalara yer verilmiştir.

Poulos ve Valle'in [12] tarafından yapılan çalışmada RNN ve LSTM dil modelleri kullanılarak el yazısı veri seti üzerinde eğitilen kelime tabanlı metin üretimi uygulaması gerçekleştirilmiştir.. Çalışma sonunda LSTM modellerinin uzun metin dizeleri üretmede RNN modellerine göre daha başarılı olduğu görülmüştür. RNN ve LSTM modelleri metin

dizeleri üretirken geçmiş bilgileri hatırlayarak öğrenen modellerdir. RNN modellerin geçmiş bilgileri uzun süre hatırlamada zorluk yaşamaktadır. LSTM modelleri yapılarında kullandıkları kapılar sayesinde uzun metin dizelerinde daha başarılı olmaktadır. Araştırmacılar aynı zamanda kelime tabanlı olarak eğitilen dil modellerinin sözlüğünde oluşan veri setindeki her bir farklı kelimenin şartlı olasılık değerlerinin hesaplanmasından ve tahmin aşamasında en olası kelimelerin seçilme işlemlerinin karmaşıklığından dolayı modelin çalışması kompleks ve zor olmaktadır.

Eriguchi vd. [13] yaptıkları çalışmada LSTM modelini kullanarak kelime tabanlı cümle üretme uygulaması gerçekleştirmişlerdir. Oluşturulan sistemin amacı eğitim için verilen veri setini kullanarak kelimelerin sırasal ve anlamsal ilişkilerini öğrenen bir dil modeli geliştirerek kelime kelime anlamlı cümle üretmektir. Çalışma sonunda oluşturulan sistemin cümlelerin hem anlamsal hem de sözdizimsel kuralları eksik de olsa yakalabildiği görülmüştür.

Feng vd. [14] tarafından yapılan çalışmada n-gam tabanlı LSTM dil modelleri kullanılarak metin üretimi uygulaması yapılmaktadır. RNN ve LSTM modellerinin dil kurallarının öğrenilmesinde etkili modeller olduğundan bahsedilmiştir. Eğitilen dil modeli sayesinde dil modelinin sözlüğünü oluşturan her bir karakter grubunun şartlı olasık değerleri hesaplanarak bir skor puanı oluşturulmaktadır. Metin üretim aşamasında tahmin edilecek karakter ya da karakter gruplarının belirlenmesi için ışıın arama algoritması kullanılmaktadır. Değerlendirme için BLEU metrik kullanılmıştır. LSTM modelinde n-gram tabanlı yöntemlerin kullanılmasının başarıyı kelime tabanlı yöntemlere göre yaklaşık %3-4 puan arttırdığından söz edilmektedir.

Renduchintal vd. [15] tarafından yapılan çalışmada karakter tabanlı LSTM modeli kullanılarak oluşturulan metin üretimi çalışmasında, LSTM hücre sayısının model başarısı üzerindeki etkisini incelemişlerdir. Arttırılan LSTM hücresi sayısının başlangıçta tahmini iyileştirdiğini tespit etmişlerdir. Ancak LSTM hücre sayısının belirli bir sayının üzerine çıkarılmasının daha kötü sonuçların alınmasına neden olduğunu görmüşlerdir.

Yang vd. [16] tarafından LSTM ve GRU modellerinin metin oluşturmada hangisinin performansının daha iyi olduğu ile ilgili çalışmalar yapmışlardır. Bu geliştirilen modellerin metin oluşturma alanında hangisinin daha iyi performansa sahip olup olmadığı net belli

değildir. Bu nedenle, bu sorunu çözmek için keşfedici bir metin oluşturma deneyi yapmışlardır. LSTM ile oluşturulan metin kalitesinin, LSTM'nin farklı bir varyantı olan GRU modeliyle karşılaştırmışlardır. LSTM modelinin uzun metin alanındaki değerlendirme sonuçlarının, üç değerlendirme indeksi (karışıklık derecesi, BERT puanı ve BLEURT.) aracılığıyla GRU modelinkinden açıkça daha iyi olduğunu tespit etmişlerdir. Gelecekte, metin üretimi için LSTM'ye dayalı bir ön eğitim modeli tasarlamaktan söz etmişlerdir.

Bahdanau vd. [17] tarafından karakter-kelime tabanlı LSTM dil modeli çalışması yapmışlardır. Bu çalışmada karakter ve kelime bilgisini birleştirerek LSTM dil modelinin başarısını arttırmaya çalışmışlardır. Kelime ve alt kelime bilgisini bir LSTM dil modelinde birleştirerek dil modelini tasarlamışlardır. Çalışma farklı model boyutları, karakterler için çeşitli gömme boyutları, karakterlerin eklendiği farklı sıralar ve karakterlerin ağırlık paylaşımı için İngilizce ve Felemenkçe üzerinde test edilmiştir. Bu modelin hem aynı sayıda gizli birime sahip kelime düzeyinde dil modellerinden hem de aynı sayıda parametreye sahip kelime düzeyindeki dil modellerinden daha iyi performans gösterdiğini tespit etmişlerdir. Bu bulgular, küçük bir model boyutu ve büyük bir model boyutu için İngilizce ve Felemenkçe için doğrulanmıştır. Sözcük ve karakter yerleştirmelerini birleştirerek, benzer miktarda parametreye sahip bir temel modele kıyasla İngilizce'de %2,77'ye ve Felemenkçe'de %4,57'ye varan göreceli iyileştirme elde edilmiştir. Ayrıca, daha fazla sayıda parametreye sahip temel kelime düzeyindeki modellerden de daha iyi performans göstermektedir.

Meng vd. [18] tarafından kelime düzeyinde LSTM kullanılarak metin oluşturma çalışması yapmışlardır. Çalışmada, belirli bir veri kümesi üzerinde metin oluşturmak ve performansını analiz etmek için bir LSTM modeli tasarlanmaktadır. LSTM'ler, dil modelleme için en yaygın kullanılan ağlardır. LSTM yapıları alakalı bilgilerin geriye doğru yayılması yoluyla uzun vadeli bilgileri hatırlayabilmektedir. LSTM hücrelerinde giriş, unutma ve çıkış kapıları olmak üzere üç kapı bulunur. Kapılar, her bir zaman damgasında, saklanması gereken ve unutulması gereken geçmiş bilgilerin öneminin belirlenmesine yardımcı olur. Çalışma sonunda eğitilen model test edilerek toplamda 0,16 kayıp %96 eğitim doğruluğu ve %80 test doğruluğu elde edilmiştir. Bu doğruluk, modelin hiper parametrelerini ayarlayarak eğitim ve test kaybını azaltarak doğruluğu daha da artırılabilceği düşünülmüştür.

Johansen vd. [19] tarafından yalnızca karakter düzeyinde girdilere dayanan bir sinir ağı modeli geliştirmişlerdir. Model uzun kısa süreli bellek (LSTM) tekrarlayan sinir ağı dil modeli (RNN) ve evrişimli sinir ağı (CNN) kullanılarak tasarlanmıştır. Performans ve boyut arasındaki dengeyi değerlendirmek için modelin küçük ve büyük iki versiyonu kullanılmıştır. Sonuçlar incelendiğinde karakter tabanlı modelin daha az parametreye sahip olmasına rağmen, giriş katmanında kelime yerleştirmelerini kullanan temel modellerden daha iyi performans gösterdiğini tespit etmişlerdir. Modelin tahmin ettiği kelimeler analiz edildiğinde, modelin yalnızca karakterlerden üretilen kelimelerin zengin anlamsal ve imlasal özellikleri yakalayabildiği görülmektedir. Zengin morfolojiye sahip dillerde (Arapça, Çekçe, Fransızca, Almanca, İspanyolca, Rusça), model yine daha az parametreyle kelime düzeyinde eğitilen LSTM tabanlı modellerden daha iyi performans göstermektedir. Sonuçlar, birçok dilde karakter girdilerinin dil modelleme için yeterli olduğunu göstermektedir.

Noord vd. [18] çalışmalarında karakter tabanlı dikkat merkezli kodlayıcı ve kod çözücü kullandıkları modelde, dillerin morfolojik kalıplarını göz ardı ederek, sadece kelimeler üzerinde çalışmışlardır. Bu çalışma sonucunda karakter tabanlı dil modellerinin dil kurallarını kelime tabanlı modellere göre daha iyi yakaladığı sonucunu elde etmişlerdir. Modelin aynı zamanda tahmin aşamasında basit bir softmax katmanı kullanarak karakterler arasındaki ilişkileri yakalamada başarı elde etmesine rağmen, modelin çok fazla gizli katman kullandığını ve buna bağlı olarak çok fazla bellek tükettiğini tespit etmişlerdir. Geliştirdikleri karakter tabanlı modelin +3.37'ye varan BLEU puan kazanımları ile sözcük düzeyinde ve alt sözcük düzeyinde muadillerine göre tutarlı iyileştirmeler sağladığını görmüşlerdir.

1.2 TEZİN AMACI

Tezin amacı NLP biliminin bir alt uygulama alanı olan metin üretimi alanında derin öğrenme algoritmalarından LSTM/GRU, Kodlayıcı-Kod çözücü ve Dikkat mimarisi kullanılarak türkçe veri seti üzerinde çalışan karakter tabanlı bir dil modeli geliştirerek anlamlı türkçe kelimeler üretmek ve akademik alana katkıda bulunmaktır.

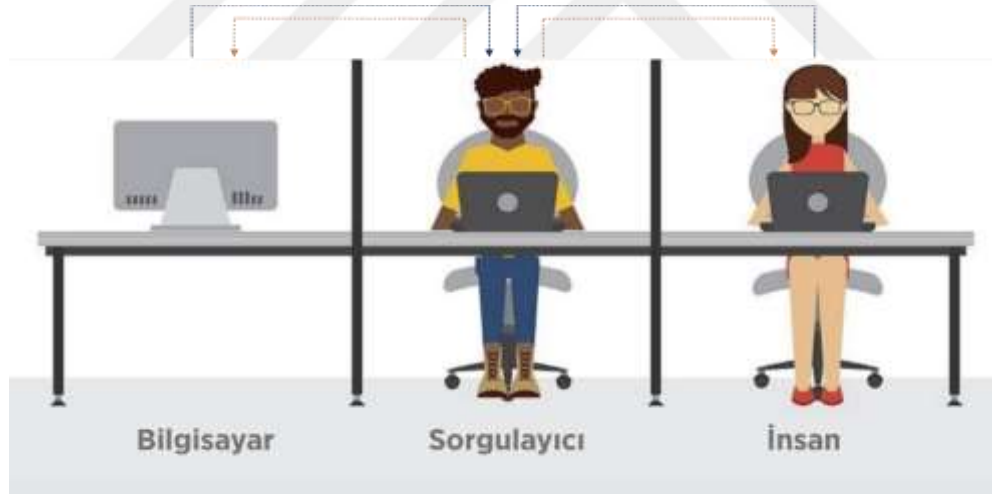
1.3 TEZİN ORGANİZASYONU

Birinci bölümde giriş ve amaç başlığı altında; doğal dil işleme, metin üretimi, derin öğrenme algoritmaları hakkında temel bilgiler , konuyla ilgili literatür çalışmaları ve tezin amacından bahsedilmiştir. İkinci bölümde genel bilgiler başlığı altında; yapay zeka, makina öğrenmesi, yapay sinir ağları, derin öğrenme, doğal dil işleme, kelime vektörleri, yinelemeli yapay sinir ağları ve önceden eğitilmiş dil modellerinden bahsedilmiştir. Üçüncü bölümde gereç ve yöntemler başlığı altında; metin üretiminde kullanılacak derin öğrenme algoritmalarından ve özellikle kodlayıcı kod çözücü ve dikkat mekanizmasının matematiksel modeli ve kodlama mimarisinden söz edilmiştir. Dördüncü bölümde bulgular başlığı altında; yapılan çalışma ile ilgili alınan sonuçlar ve değerlendirmeler yapılmıştır. Beşinci bölümde sonuç kısmında; alınan sonuçların iyileştirilmesi için öneriler belirtilmiş ve çalışmanın akademik literatür için katkısı ifade edilmiştir.

2. GENEL BİLGİLER

2.1 YAPAY ZEKA

Bilgisayarların insana özgü davranışlarından olan anlam çıkarma, akıl yürütme, geçmiş bilgilerden öğrenerek gelecekteki durumlar ile ilgili kararlar alabilme yeteneğini yapay zeka denilmektedir. Yapay zeka bunu yaparken çok büyük miktardaki veriyi eğitim verisi olarak alıp veriler arasındaki ilişkileri ve kalıpları öğrenerek insana özgü zihinsel işlemlerin yapılmasını sağlayan ve birden fazla yaklaşımı olan disiplinler arası bir bilimdir [19]. Yapay zeka konusunda ilk önemli adım 20. yüz yılın sonlarına doğru Alan Mathison Turing tarafından atılmıştır. Alan Turing'in "Bilgisayar Makineleri ve Zeka (1950)" makalesi ile ve daha sonra yaptığı Turing testi ile yapay zekanın amacını ve vizyonunu belirlemeye çalışmıştır. Turing Testi, bir bilgisayarın, sorgulayıcının sorduğu sorulara verdiği cevapların bir insan tarafından verildiğine sorgulayıcıyı inandırmaya çalışarak kandırmaya odaklanmaktadır [20].



Şekil 2.1: Turing Testi [21].

Şekil 2.1’de görüldüğü gibi Turing testi üç katılımcıyı içermektedir; bir bilgisayar, bir insan ve birde sorgulayıcı insandan oluşmaktadır. Tüm iletişim klavye ve ekran üzerinden gerçekleşmektedir. Sorgulayıcı, bilgisayar ve insan olan iki katılımcıya aynı soruları defalarca sorarak hangisinin verdiği cevabın insan tarafından verildiğini tahmin etmeye çalışmaktadır. Sorgulayıcı bilgisayarı insandan ayırt edemiyorsa o zaman bilgisayar akıllı, düşünen bir varlık olarak kabul edilmektedir

Yapay zekanın geçmişı uzun yıllar önceye dayansa da bu zaman diliminde yapay zekanın gelişimi zaman zaman kesintiye uğramıştır. Özellikle 1974-1980 arası ve 1987-1993 arası ‘yapay zekanın kışı’ denilen yapay zekaya desteğın kesildiğı ve duraklama dönemine girildiğı bir dönem olmuştur. Bu yıllarda yapay zeka alanında çok fazla bilimsel gelişme yaşanmamıştır. Ancak 1993 senesinden sonra bilgisayar mimarisindeki donanım gücünün ilerlemesi, büyük veriye erişimin kolaylaşması, yapay zeka ile ilgili yeni algoritmaların geliştirilmesi ve bu alana olan ilginin artarak devam etmesi ile günümüze kadar gelişimini arttırarak devam ettirmiştir [22]. Yapay zeka insan zekasına yakınlığına göre dar, genel ve süper yapay zeka olmak üzere 3 gruba ayrılmaktadır.

2.1.1 Dar (Zayıf) Yapay Zeka

Zayıf yapay zeka günümüzde kullanılan yapay zeka çeşididir. Bu yapay zeka çeşidi tek bir görev için kullanılmaktadır. Bu dar görevlerinden ve kendilerine verilen görevlerin sınırlı olmasından dolayı bu ismi almışlardır. Sadece bir sorunu çözmek için tasarlandıkları için kapsamı sınırlı olarak inşa edilmişlerdir. Bu makinalar akıllı olarak ifade edilse de tek bir görev için tasarlandıkları için belirli sınırlar içinde çalışmaktadır [23]. Zayıf yapay zeka insan zekası gibi hareket etmekten uzaktır sadece verilen eğitim veri seti içindeki ilişkileri ve kuralları yakalayarak insan davranışını taklit etmeye çalışmaktadır. Bu alanda son yıllarda makine öğrenimi ve derin öğrenme yöntemleri kullanılarak farklı alanlarda kullanılan çok sayıda uygulama geliştirilmiştir. Günümüzde en çok kullanılan zayıf yapay zeka uygulamalarına öneri sistemleri, spam filtreleme, google arama, makine çevirisi, görüntü tanıma, sürücüsüz arabalar, sanal asistanlar, izleme/dinleme/satın alma davranışına dayalı eğlence veya pazarlama içeriğı önerileri, tıp uygulamaları, doğal dil işleme gibi uygulamalar örnek verilebilir. Bu uygulamalarda yapay zeka sadece o görevleri için eğitilir ve öğrenme verilen veri setine bağılı olarak gerçekleştirilmektedir [24].

2.1.2 Genel Yapay Zeka

Dar yapay zeka bugünkü yapay zekanın geldiğı durumu ifade ederken genel yapay zeka gelecekte yapay zekanın nereye ulaşacağını ifade eden ve insanlar gibi düşünebilen, öğrenebilen ve karar verebilen bir yapay zeka türüdür. Günümüzde ki yapay zeka hala bu seviyeye ulaşamamıştır. Ama gelişmeler tüm hızıyla devam etmektedir. Makinaların bu seviyede zekaya ulaşabilmeleri ve insanlar gibi bilinçli davranış gösterebilmeleri için bilişsel

gelişim kazandıracak yapay zeka algoritmalarının geliştirilmesi gerekmektedir. 'Bilinci' tanımlama ihtiyacı, genel bir yapay zeka yaratmanın ayrılmaz bir parçasıdır [25]. İnsanlar beynin iç işleyişini çözmeye çalışırken, zekanın sınırlarını anlamaktan hala çok uzak kalmaktadır. Bunlar düşünüldüğünde gelecekte yapay zekanın insanların sahip olduğu zekaya benzer bir zekaya sahip olabileceği öngörülmektedir.

2.1.3 Süper Yapay Zeka

Süper yapay zeka artık makinaların insanlar gibi davranabildiği, öğrenebildiği, öz farkındalığına sahip olduğu ve insan zekası ve yeteneğinin üst seviyeye çıktığı bunun yanında bir bilince de sahip olduğu ve duygu taşıyabileceğini öngördüğü bir yapay zeka türüdür. Bu seviyeye ulaşan yapay zekanın artık ihtiyaçlarını, inançlarını ve arzularını da oluşturabileceği düşünülmektedir. Süper yapay zekanın insanların yaptığı matematik, bilim, spor, sanat, tıp, hobiler, duygusal ilişkiler gibi alanlarda insanlardan daha iyi olacakları öngörülmektedir [26]. Sonuç olarak, süper zeki makinaların karar verme ve problem çözme yeteneklerinin insanlardan çok daha üstün olacağı düşünülmektedir

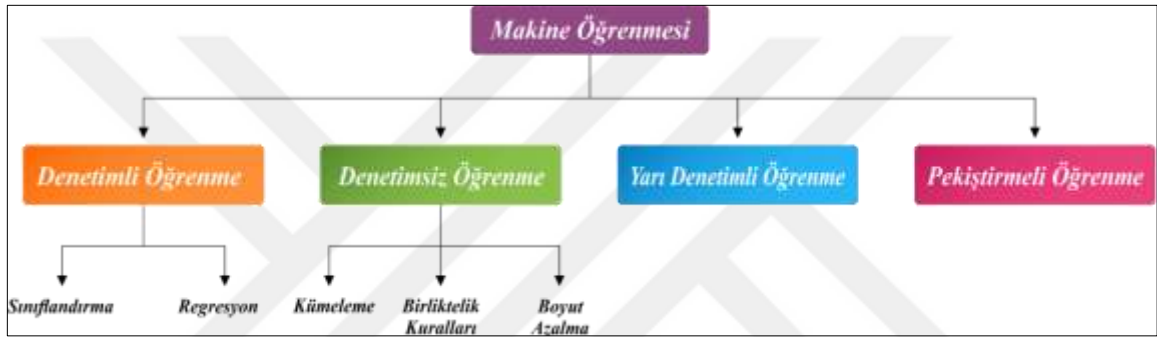
2.2 MAKİNE ÖĞRENMESİ

Makine öğrenmesinin temeli yapay zeka, matematik ve istatistik bilimine dayanan, klasik bilgisayar programlama mantığı içermeyen verilen veri setlerinden karmaşık örüntüleri yakalayabilme, çıkarımlarda bulunabilme ve verilen problemlerin genel modelini çıkararak öğrenebilen algoritmalara dayanmaktadır [27]. Geçmiş bilgileri kullanarak deneyim kazanıp kendi kendine öğrenme gerçekleştirdiği için insan zekasına benzetilmektedir. Büyük ve karmaşık bilgilerin elle işlenmesi ve analizinin yapılması mümkün olmadığından, geçmişte kullanılan verilerin makine öğrenmesi algoritmalarıyla eğitilip öğrenmesi gerçekleştirilerek gelecek ile ilgili çıkarımlar yapılabilmesi mümkün olmaktadır [28]. Nasıl ki gerçek dünyada insanlar tecrübelerinden öğrenerek alanlarında uzmanlaşabiliyorsa makinalarda geçmişteki bilgileri kullanarak gelecek için deneyimler kazanabilmekte ve insanlar gibi öğrenebilmektedir Makinalar için bu durum ona verilen veri setinin eğitim ve test verisi olarak ikiye ayrılıp eğitim verisiyle makine öğrenmesi algoritmasının öğrenmesi, test verisi ile ne kadar doğru kararlar verdiğinin testinin yapılmasıyla gerçekleştirilmesiyle olmaktadır. Daha sonra eğitim verisi ile eğitilen modelden matematiksel bir fonksiyon elde edilmektedir. Makine öğrenmesinde iyi bir öğrenmenin gerçekleştirilmesi için yeterli miktarda verinin

verilmesi, uygun algoritmaların seçilmesi ve en iyi parametrelerinin belirlenmesi ile mümkün olmaktadır [29].

2.2.1 Makine Öğrenmesi Modelleri

Makine öğrenmesi problemlerinde kullanılan çok sayıda farklı algoritma bulunmaktadır. Bu algoritmalar matematiksel ve istatistiksel hesaplamalar üzerinde çalışmaktadır. Bu algoritmaların doğru kararlar verebilmesi için, uygun veri setlerinin doğru algoritmalarla işlemlere sokulması gerekmektedir [30]. Şekil 2.2’de makina öğrenmesi algoritmaları gösterilmektedir.



Şekil 2.2: Makine Öğrenmesi Yöntemleri.

2.2.1.1 Denetimli öğrenme

Denetimli öğrenme yönteminde modele verilecek veri setinde bulunan öznelik değişkenleri ile etiketler dediğimiz hedef değişkenler birlikte verilmek durumundadır. Dolayısıyla denetimli öğrenme algoritmaları girdi verilerinin özellik değerleri ile etiket değerleri birlikte verilerek öğrenmeyi gerçekleştirmektedir. Algoritma verilen giriş değerlerine göre çıkış değerleri arasındaki ilişkileri yakalayarak fonksiyonel bir bağlantı hesaplamaktadır. Daha sonra kendi kendine öğrenme aşamasında hatayı yeterince minimize etmeye çalışarak ve kayıp fonksiyonu aracılığıyla doğruluğunu ölçerek en iyi öğrenmeyi gerçekleştirecek bağlantıyı elde etmektedir. Elde ettiği çıktı değerlerini test için ayrılan değerlerle karşılaştırarak en doğru modeli oluşturmaktadır. Kurulan bu model ile gerçek dünya verileri kullanılarak sınıflandırma ya da tahmin uygulamaları gerçekleştirilmektedir [31]. En çok kullanılan denetimli öğrenme algoritmalarına Lojistik Regresyon, Naive Bayes, Karar Ağaçları, K-En Yakın Komşu, Rastgele Orman, Destek Vektör Makineleri ve Yapay Sinir

Ağları gösterilebilir. Denetimli öğrenme algoritmaları sınıflandırma ve regresyon (tahmin) algoritmaları olarak iki kategoriye ayrılmaktadır [32].

Sınıflandırma

Sınıflandırma algoritmaları girdi verilerinin özellik verileriyle etiket verilerini kullanarak girdi değerlerinin hangi gruba ait olduğunu belirlemeye çalışan bir yöntemdir. Sınıflandırma algoritmaları verilen veri setinin etiketli eğitim veriyle çalışarak işlemekte olduğu verileri algoritmanın öğrendiği sınıf etiketleriyle gruplara ayırmaktadır. Bu şekilde öğrenen sınıflandırma algoritmaları yeni verilen veriler üzerinde çalışarak hangi verilerin hangi sınıfa ait olduğunu belirleyebilmektedir. Model eğitilirken veri setinde mutlaka etiketli veri bulunması gerektiğinden ve algoritmaların öğrenmesinin verilen bu etiket sayesinde gerçekleşmesinden dolayı bu yöntem bir denetimli öğrenme yöntemi olarak geçmektedir. Sınıflandırma modellerinin nasıl kullanıldığına ilişkin örneklerle; spam algılama, nesneleri tanımlama ve sınıflandırma, konuşma tanıma, yüz tanıma, belgelerin ve yazıların otomatik sınıflandırılması gibi örnekler verilebilmektedir. Verilere uygulanan sınıf etiketlerinin sayısına bağlı olarak sınıflandırma problemlerini ikili sınıflandırma, çoklu sınıflandırma, çok etiketli sınıflandırma olarak üçe ayrılmaktadır [33].

Regresyon

Bir diğer denetimli öğrenme algoritması tahmine dayalı olarak çalışan regresyon algoritmasıdır. Regresyon, bir makine öğrenimi modelinin sürekli sonuçları tahmin etme süreci olarak kullanılmaktadır. Regresyon algoritmaları verilen veri setinin etiketli eğitim veri setini kullanarak girdi değerlerine karşılık gelen çıkış değerleri arasındaki kalıp ve ilişkilerden öğrenerek, yeni verilen veri setinin gelecekle ilgili tahmin değerlerini hesaplayabilmektedir [34]. Regresyon modelleri kullanılarak bir evin fiyatını tahmin etmek veya belirli bir zamanda hisse senedi fiyatındaki eğilimi tahmin etmek gibi tahmine dayalı uygulamalarda kullanılmaktadır.

2.2.1.2 Denetimsiz öğrenme

Denetimsiz öğrenme denetimli öğrenmenin aksine etiketlenmemiş verileri kullanmaktadır. Algoritmaya hiçbir etiket verisi verilmez, girdisindeki yapıyı bulmak için kendi başına bırakılmaktadır. Denetimsiz öğrenme başlı başına bir amaç doğrultusunda verilerdeki gizli

kalıpları keşfederek bir amaca yönelik özellikleri öğrenmeye çalışmaktadır [35]. Denetimsiz öğrenme yöntemlerine temel olarak kümeleme, birliktelik kuralları ve boyut azaltma verilmektedir

Kümeleme

Denetimsiz öğrenme yöntemlerinden biri olan kümeleme algoritmaları veri setinin etiketsiz verileri üzerinde çalışarak benzer verileri aynı grupta toplamayı amaçlayan bir algoritma ailesidir. Kümeleme de aynı grupta bulunan verilerin birbirine çok benzer olması beklenirken farklı gruptaki verilen birbirine benzerliklerinin çok uzak olması beklenmektedir. Kümeleme de veriler benzerliklerine göre gruplara ayrılırken uzaklık hesaplamaları kullanılmaktadır [36]. En bilinen kümeleme algoritmalarına K-means algoritması ve Hiyerarşik kümeleme algoritmaları örnek verilmektedir.

Birliktelik Kuralları

Birliktelik kuralları olayların birlikte geçme sıklıklarının analizin yapıldığı ve veri setinde bulunan verilerin birbiriyle ne kadar ilişkili olduğunu istatistiksel olarak kendi kendine öğrenerek bunu çözümleyen denetimsiz bir makine öğrenmesi yöntemidir [37]. Genellikle marketlerde ürünlerin hangilerinin birlikte satılabileceği ilişkisinin kurulduğu market sepeti uygulamalarında kullanılmaktadır.

Boyut Azaltma

Boyut azaltma (Principal Component Analysis-PCA) bir veri kümesi için girdi değişkenlerinin sayısının azaltılmasını sağlayan denetimsiz bir makine öğrenmesi tekniğidir. PCA, boyutların sayısını azaltırken, orijinal veri kümesinin maksimum bilgilerinin azaltılmış daha az özellik ile veri kümesinde tutulmasını sağlamaktadır [38].

2.2.1.3 Yarı denetimli öğrenme

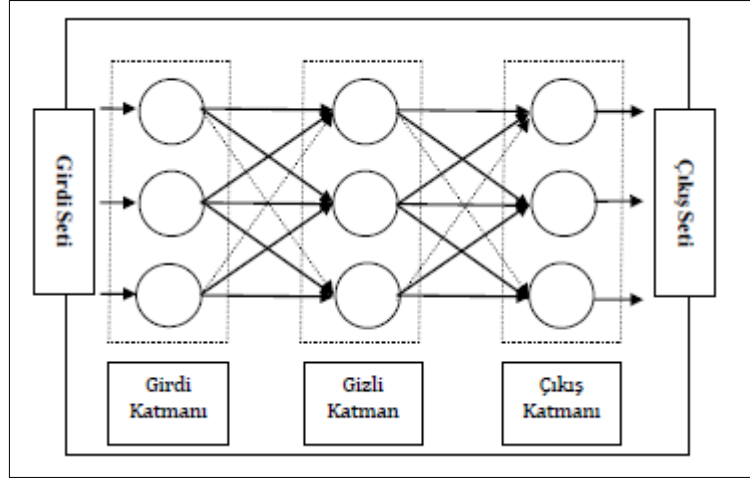
Yarı denetimli öğrenme karma bir ortam olarak etiketsiz verilerin yanında etiketli eğitim verilerinin yalnızca küçük bir kısmına ihtiyaç duymaktadır. Etiketlenmemiş veri örnekleri atılmak yerine öğrenme sürecinde kullanılmaktadır. Yarı denetimli öğrenmenin altında yatan fikir, verinin eğitiminde eksik etiketlerle verileri kullanamayan geleneksel denetimli öğrenmenin bu sınırlılığını ortadan kaldırmak için tercih edilmektedir [39].

2.2.1.4 Pekiştirmeli öğrenme

Diğer makine öğrenme yöntemlerinden farklı olarak pekiştirmeli öğrenmede sistem ön bilgi yerinde gözlem ve seçimde bulunmak üzere kurulmaktadır. Verilen giriş değerlerine karşılık bir çıkış değeri verilmediği için çıkış değerinin doğruluğunu belirleyen bazı ölçütler kullanılmaktadır. Model içinde bulunduğu ortam hakkında yetersiz bilgi olduğu için neden sonuç ilişkisi kurmak için gözlem yapılmaktadır. Gözlem bittikten sonra model bir karar vermeye zorlanır. Karar veya sonuç yönlendirilmek istenen tarafa bir ödül ile şartlandırılmaktadır. Bu sistem ile model her zaman en yüksek ödül neredeyse ona koşullanmış olmaktadır. Ödül alamaması bir ceza olarak görülür. Aynı zamanda aldığı her kararda ve attığı her adımda nasıl ödüle ulaşacağını öğrenmiş olup, deneyim biriktirir. Pekiştirmeli öğrenme modeli gerçek zamanlı karar alabilen bir model geliştirdiği için oyunlarda, robot sistemlerinde, beceri edinme sistemleri gibi öğrenme olanağı olan sistemlerde kullanılmaktadır [40].

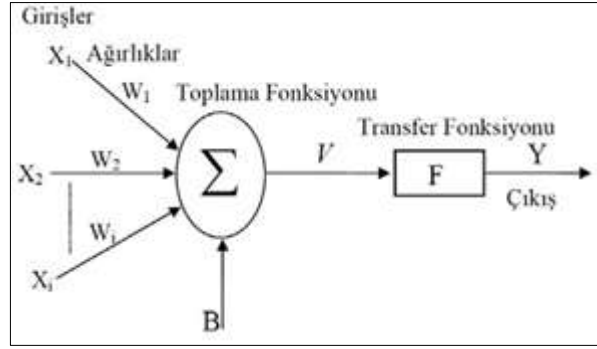
2.3 YAPAY SİNİR AĞLARI

Yapay sinir ağı (YSA), insan zeka yapısının temelinde bulunan sinir ağlarından ilham alınarak düşünülmüş bir tür makine öğrenimi yaklaşımıdır. YSA insanların deneyim ve tecrübelerinden öğrenmesi gibi geçmiş verileri analiz ederek veriler arasındaki kalıp ve ilişkileri öğrenip gelecekle ilgili tahminlerde ve verilerin sınıflandırılmasında kullanılmaktadır [41]. YSA gücünü paralel olarak çalışabilmesinden, genelleme yapabilmesinden ve öğrenebilmesinden almaktadır. YSA özellikle doğrusal olmayan ilişkileri modellemek için daha uygundur. Bu özelliklerinden dolayı günümüzde yapay zeka uygulamaları olan makine çevirisi, görüntü işleme, tıp, savunma sanayisi, haberleşme, finans ve ekonomi, robotik ve mühendislik gibi alanlarında yaygın olarak kullanılmaktadır [42].



Şekil 2.3: Yapay Sinir Ağı [43].

Şekil 2.3'te YSA modeli verilmiştir. YSA modelinde ilk katman olan girdi katmanı dış dünya verilerini alarak (ses, metin, video, görüntü) YSA'nın ilk adımını oluşturur. Modelin ortası denilen ara katmanda bir ya da çok sayıda gizli katman bulunmaktadır. Bu gizli katmanlar girdi verileri üzerinde hesaplamalar yaparak veri setinin tüm öznelik çıkarımını elde edip veri içerisindeki ilişkisel kalıpları yakalamaya çalışmaktadır. Son katman olan çıktı katmanı ise gizli katmanlarda yapılan hesaplamaların sonucunu istatistiksel olarak yorumlayıp bir çıktı değeri vermektedir. YSA'nın doğrusal olmayan sonuçlar üretebilmesi için model içerisinde aktivasyon fonksiyonları kullanılmaktadır. Her gizli katmandan ve çıktı katmanlarından sonra kullanılan aktivasyon fonksiyonları değişiklik göstermektedir. Aktivasyon fonksiyonları nöronlardan gelen ağırlıklı toplamaları hesaplayarak hangi nöronun aktif olup olmayacağına karar vermektedir. Burada doğrusal olmayan dönüşüm yapılarak modelin çok daha karmaşık ilişkileri öğrenmesini sağlanmaktadır. YSA'nın performansını belirleyen birçok parametrik özellik bulunmaktadır. Bunlar; öğrenme oranı, gizli katman sayısı, hata hesaplama fonksiyonlarının seçimi, seçilen aktivasyon fonksiyonları gibi özelliklerdir. Bu özellikler kullanılan veri setinin yapısına ve kullanılan algoritmaya bağlı olarak sürekli değişiklik göstermektedir [43].

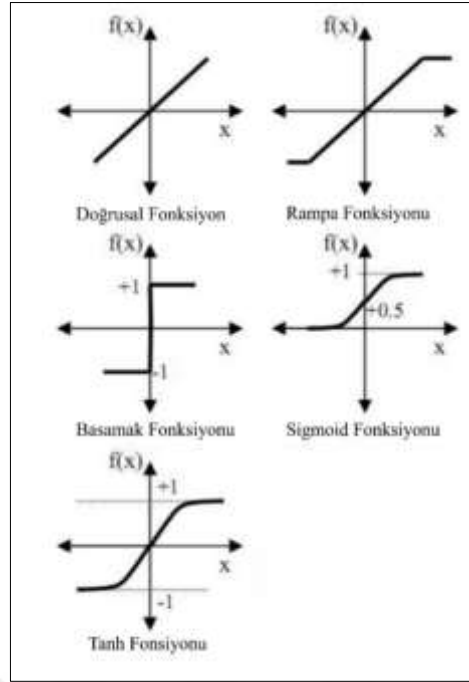


Şekil 2.4: YSA Modelinin Matematiksel İşleyişi [45].

Şekil 2.4'te YSA modelinin matematiksel işleyişi görülmektedir. Burada i giriş değerlerinin sayısını, X değerleri modele sağlanan girdi verilerini, W değeri her bir girişe ait ağırlık değerlerini, B bias değerini, F modelde kullanılan aktivasyon fonksiyonunu, V ilk nörona gelen tüm değerlerin hesaplanması ile oluşan toplam fonksiyonun çıkış değerini, Y ise kullanılan aktivasyon fonksiyonundan geçirilerek elde edilen çıkış değerini ifade etmektedir [46].

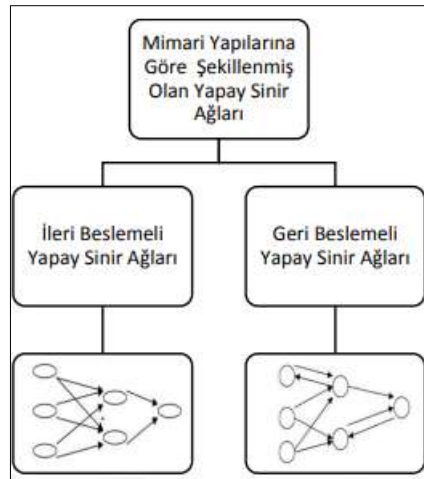
$$V_i = \sum_{j=1}^n x_j w_j + B \quad (2.1)$$

Denklem 2.1'de verilen YSA modelinin hesaplamalar sonucunda elde edilen V_i toplam çıkış değeri modele uygun olarak kullanılan bir aktivasyon fonksiyonundan geçilerek $[0,1]$ ve $[-1,1]$ aralığında değerler alarak nihai çıkış değerlerinin elde edilmesini sağlamaktadır [47].



Şekil 2.5: Aktivasyon Fonksiyonları [48].

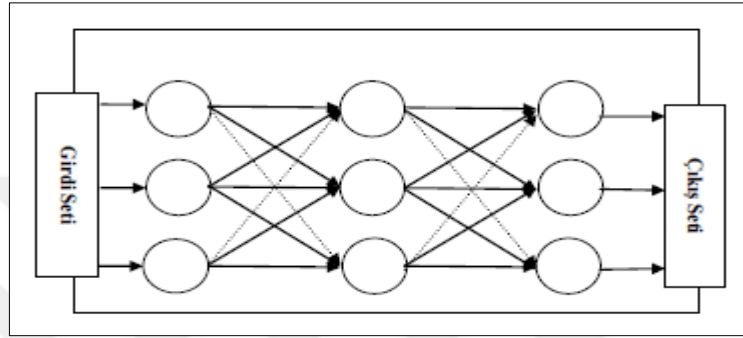
Şekil 2.5'te YSA'da en çok kullanılan aktivasyon fonksiyonları gösterilmektedir. Aktivasyon fonksiyonlarının tercih edilmesindeki asıl neden türevlenebilir ve süreklilik özelliklerinin olmasından dolayı çok daha karmaşık ilişkileri öğrenebilmelerinden kaynaklanmaktadır [48]. YSA ileri beslemeli ve geri beslemeli olmak üzere mimari yapılarına göre ikiye ayrılmaktadır.



Şekil 2.6: Mimari Yapılarına Göre YSA [43].

2.3.1 İleri Beslemeli YSA

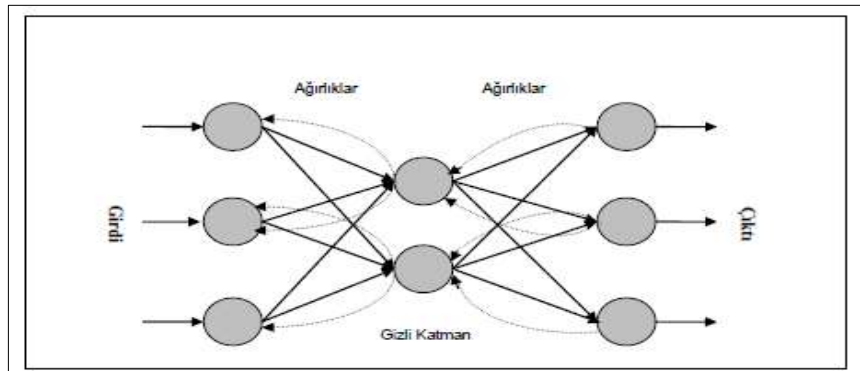
Bu tür YSA'da veri akışı tek yönlü olarak ileri doğru yapılmaktadır. Girdi katmanından verilen veriler gizli katmanlarda işlenerek çıktı katmanına gelerek sonuçlandırılır. Bu tür YSA'da geriye doğru döngüsel bir işleyiş yapılmamaktadır. Bu ağın avantajı, girdi modellerinin nasıl değerlendirileceğinin ve tanınacağını öğrenilmesidir [45]. Şekil 2.7'de ileri beslemeli YSA mimarisi gösterilmiştir.



Şekil 2.7: İleri Beslemeli YSA [43].

2.3.2 Geri Beslemeli YSA

Geri beslemeli YSA modellerinde çıkış katmanına gelen tahmin değerlerinin gerçek değerlerden farkı ile elde edilen hata değerinin azaltılması için, hata değerinin geriye doğru gizli katmanlara ve giriş katmanlarına dağıtıldığı ve bu katmanlarda bulunan ağırlık ve bias değerlerinin güncelleştirilerek optimize edilmeye çalışıldığı bir modeldir. Geri beslemeli YSA'da ağın birbirine bağlı tüm faktörlerinin en iyi değerlerine ulaşılması amaçlanmaktadır [49].



Şekil 2.8: Geri Beslemeli YSA [45].

Şekil 2.8’de geri beslemeli bir YSA görülmektedir. Geri beslemeli YSA modellerinde giriş verileri girdi katmanından alınarak modele verilmektedir. Modelin eğitimi için kullanılan ağırlık değerlerine rastgele bir değer atanmaktadır. Ağırlık değerleri olarak seçilen değerler herhangi bir girdinin modele ne kadar etkisi olması gerektiği hakkında bilgi taşımaktadır. Büyüklüğü ölçüsünde modelin eğitimi aşamasında etkisi çok fazla olmaktadır. Tüm girdiler ve ağırlık değerleri çarpılıp toplanarak ve o katmanda bulunan aktivasyon fonksiyonundan geçirilerek o katmanın çıkış değeri elde edilmektedir. Bu şekilde ileriye doğru diğer katmanlardaki hesaplamalar yapılarak tahmin değeri ile gerçek değer arasındaki fark olan hata değeri hesaplanır. Geri yayılım burada başlamaktadır. Bu hata değerini en aza indirmek için hatayı geri doğru bu katmanlarda bulunan ağırlık ve bias değerlerine dağıtarak modelin en iyi sonucu verecek ağırlık ve bias güncelleştirmeleri yapılmaktadır [50]. Sonuçta bu modeli ifade eden en iyi matematiksel lojistik fonksiyonu elde edilmeye çalışılmaktadır

2.3.3 YSA Öğrenme Kuralları

YSA eğitilirken kullanılan çok farklı öğrenme kuralları bulunmaktadır. Bu kurallar matematiksel hesaplamalar yapılarak YSA’da öğrenmeyi genelleştirmeyi sağlayan ağırlık değerlerinin en iyi duruma getirilmesini sağlamaktadır. YSA ağırlarının verilen girdi değerlerine göre istenen çıktı değerlerine ulaşabilmesi için modelin ağırlıklarının en doğru ve tutarlı değerlere sahip olması gerekmektedir [51]. YSA modellerinde ağırlık değerleri başlangıçta rastgale olarak atanmaktadır. Daha sonraki aşamalarda ağa verilen örneklerle ağın kullandığı öğrenme kuralına göre ağırlık değerleri değiştirilmektedir. Bu şekilde istenilen çıktı değerlerine ulaşana kadar ağırlık değerleri sürekli güncelleştirilmektedir. Bu bağlamda ağırlıkların gerçek değerleri hiçbir zaman tahmin edilememektedir Bu durum için YSA’da kara kutu benzetmesi yapılmaktadır [52]. Şekil 2.9’da YSA kara kutu yakıştırması gösterilmektedir.



Şekil 2.9: YSA Kara Kutu Yakıştırması [52].

2.3.3.1 Hepp öğrenme kuralı

Hepp öğrenme kuralı YSA'da kullanılan ilk öğrenme yöntemlerinden biridir. Bu kural genellikle ileri yayımlı basit YSA modellerinde kullanılmıştır. Bu sinir ağlarında giriş verilerinin alındığı girdi katmanı ve sonuç değerlerinin alındığı çıktı katmanı bulunmaktadır. Bu öğrenme kuralında her girdi örneği için ağırlık değerlerinin aynı işaretli olması durumunda etkisi artırılır zıt işaretli olması durumunda ise etkisi azaltılmaktadır [53].

2.3.3.2 Delta kuralı

Delta kuralı geri yayımlı YSA'da tercih edilen ağırlık değerlerinin güncelleştirilmesini sağlayarak modeli en iyi ifade eden fonksiyonu belirlemek için kullanılan matematiksel bir öğrenme yöntemidir. Bu öğrenme yönteminde amaç gerçek çıktı değerleri ile beklenen çıktı değeri arasındaki karşılaştırmanın yapılarak bu iki değerinin birbirine eşit ya da çok yakın çıkmasını sağlamaktadır. Bu öğrenme yönteminde YSA'nın eğitilmesi aşamasında ağırlık değerlerine rastgele küçük sayılar verilerek model girdi verileri ile eğitime çalışılır. Bu süreç tekrarlı olarak gerçek değer ile beklenen değer arasındaki fark (hata) en az oluncaya kadar ağırlıkların güncelleştirilmesi şeklinde devam etmektedir. Model tarafından kabul edilebilir bir hata seviyesinde ağırlıklar son değerini almış olmaktadır [54].

2.3.3.3 Hopfield kuralı

Hebb öğrenme kuralına benzer yapıda çalışan bir öğrenme kuralıdır. Bu öğrenme kuralında girdi değeri ile istenen çıktı değerlerinin aynı anda aktif veya pasif olması durumuna göre ağırlıklar artırılmakta ya da azaltılmaktadır [43].

2.3.3.4 Kohonen kuralı

1982 yılında Teuvo Kohonen tarafından kullanılmış ve günümüzde çok tercih edilmeyen bir öğrenme kuralıdır. Bu kuralda modelin ağırlıklarının değiştirilmesi için sinir hücreleri en iyi durumlarını oluşturmak için çalışmaktadır. En yüksek değere sahip olan sinir hücresi başarılı olmakta ve komşu sinir hücresinin ağırlıklarının yükseltilmesi için yön göstermektedir [55].

2.4 DERİN ÖĞRENME

Derin öğrenme YSA kullanılarak oluşturulmuş makine öğrenmesinin farklı bir yöntemidir. Bu sinir ağlarına istenilen sayıda gizli katmanların eklenmesiyle daha derin YSA modelleri

oluşturulmaktadır. Çok katmanlı YSA çok önceden biliniyor olsa da derin öğrenme algoritmalarının kullanılması, bilgisayarların işlem gücünün yetersiz olması, algoritmaların eksikliği ve yeteri kadar büyük verinin bulunmaması nedeniyle son zamanlara kadar kullanmak mümkün olmamıştır. Derin öğrenme algoritmaları YSA'nın her gizli katmanında girdi verisi ile ilgili daha karmaşık ve farklı özellikleri tespit edip öğrenebilmekte ve makine öğrenmesi sorunlarına daha iyi çözümler üretebilmektedir. İnsan öğrenmesine benzer şekilde çalıştığı için NLP uygulamalarında daha iyi sonuçlar vermektedir [56]. Derin öğrenmeyi makine öğrenmesinden ayıran en belirgin özellik girdi veri seti ile ilgili özellik çıkarımının algoritma tarafından öğrenilerek insan müdahalesi olmadan yapılması ve çok büyük miktarda veriye ihtiyaç duymasısıdır. Derin öğrenme özellik çıkarımının mümkün olmadığı ve bu işlemin algoritmanın kendisine bırakıldığı resim, ses, video, metin vb. gibi alanlarda etkili bir şekilde kullanılmaktadır. Özellikle günümüzde büyük verinin artması ve grafik işlemcilerindeki gelişmelerin ilerlemesiyle derin öğrenme yöntemleri makine öğrenmesine yöntemlerine göre daha iyi çözümler üretmede önemli bir rol oynamaktadır [49].

2.5 DOĞAL DİL İŞLEME

NLP doğal dillerin yapısının analiz edilerek anlaşılması veya yeniden üretilmesi için yapay zeka algoritmalarının kullanıldığı bir uygulama alanıdır. NLP 1990' yılların başlarında büyük ölçüde kurallara dayalıydı ve bilgisayarların dili nasıl işleyebileceğini belirlemek için dilbilimciler tarafından geliştirilen el yapımı kuralları kullanıyordu. Daha sonraları dil yaklaşımının istatistiksel bir yaklaşımla çözülmesi yönünde gelişme gösterdi. Bu yılların sonlarına doğru NLP dilbilimci temelli bir yaklaşımdan mühendis temelli bir yaklaşıma geçerek, dilbilime dalmak yerine daha çeşitli bilimsel disiplinlerden yararlandı. 2020 yıllarının başlarında NLP klasik dilbilim istatistiksel yöntemler ve yapay sinir ağlarının birlikte iç içe olduğu bir yapı şeklinde kullanılmaktadır. Bu gelişmeler ve bilgi işlem gücündeki gelişmelerle birlikte, doğal dil işleme de çok sayıda gerçek dünya uygulaması geliştirilmiştir. Günümüzde NLP uygulamalarında yaygın olarak YSA tabanlı sistemler kullanılmaktadır [57].

2.5.1 Doğal Dil Anlama

Doğal dil anlama, bilgisayarların doğal dilleri anlamaya çalıştığı uygulamalar oluşturmak için kullanılmaktadır. Doğal dili anlama sisteminin amacı girilen bir metin parçasını yorumlamak ve sınıflandırmaktır. Duygu analizi, spam filtreleme gibi benzer uygulamaları kapsamaktadır [58].

2.5.2 Doğal Dil Üretimi

Doğal dil üretimi, yapılandırılmış ve yapılandırılmamış verilerden doğal yazılı veya sözlü dil üretmek için NLP'yi kullanan bir yazılım sürecidir. Girdi olarak verilen verilere dayanarak çıktı olarak dil uygulaması üreten bir yapay zeka alanıdır. Doğal dil üretimi, bilgisayarların, insanların doğal olarak iletişim kurma şeklini taklit ederek otomatik olarak doğal dilde metin oluşturmasını sağlamaktadır [59]. Makina çevirisi, soru cevaplama, haber toplama, metin üretimi, metin özetleme, Amazon, Apple, Google ve Microsoft gibi birçok büyük firmanın sesli asistanları da bu alanda oluşturulmuş uygulamalardır.

2.5.3 Kural Tabanlı Sistemler

Kural tabanlı yöntemler NLP'nin ilk evrelerinde kullanılan daha çok metin içinde geçen kelimelerin ya da cümlelerin arandığı ve bu kavramlar görüldüğünde belirli kararların verildiği bilgisayar programlarının daha çok kullanıldığı kural yöntemli yaklaşımları içermektedir. Bu yöntemlerde kuralların her zaman manuel olarak oluşturulması ve geliştirilmesi gerekmektedir [60].

2.5.4 Makine Öğrenimi Tabanlı Sistemler

Makine Öğrenimi NLP uygulamalarında yaygın olarak kullanılmaktadır. Bu yaklaşım dili analiz ederek, anlayarak öğrenen algoritmalara dayanmaktadır. Verilen veri setini kullanarak veriler arasındaki ilişkilerden çıkarımlar yaparak istatistiksel yöntemler kullanarak öğrenirler. Makine öğreniminin en önemli avantajı kendi kendine öğrenebilmesinde yatmaktadır. Bu nedenle yüksek beceriler gerektiren manuel kural/dilbilgisi kodlamasına gerek yoktur [61].

2.5.5 Veri Ön İşleme

Dış dünya verilerinin NLP algoritmaları tarafından kullanabilmesi için istenen şekilde temizlenmesi ve istenen formata dönüştürülmesi gerekmektedir. Çünkü NLP algoritmalarının doğru karar vermeleri için verilerin iyi hazırlanmış olmaları gerekmektedir. İnsanlar tarafından oluşturulan metin, resim, video formatındaki veriler dağınık, tutarsız ve eksik bir yapıya sahip olabilmektedir. Bundan dolayı bu veriler makine öğrenmesi algoritmalarından geçirilmeden önce temizlenmeli ve biçimlendirilmelidir [62].

Bir modelin başarısı modele verilen giriş verilerinin kalitesi ve ön işleme adımlarının iyi bir şekilde gerçekleştirilmesine bağlıdır. NLP uygulamalarında metin olarak verilen giriş verilerinin ön işleme adımları yapılarak sayısal vektörlere dönüştürülüp NLP algoritmalarına verilmesi gerekliliği, veri ön işleme adımlarının titizlikle yapılması gerekliliğini zorunlu hale getirmektedir [63].



Şekil 2.10: Doğal Dil İşleme Aşamaları.

Şekil 2.10'da veri ön işleme adımları görülmektedir. Veri ön işleme adımlarında dil bilgisi kurallarına uygun olarak veri seti parçalarına ayrılır. Çalışılan veri üzerinde yazım yanlışlarının düzeltilmesi, verinin küçük harflere dönüştürülmesi, özel karakterlerin, noktalama işaretlerinin, sayıların metinden silinmesi gerçekleştirilmektedir. Kelimelerin köklerinin ve gövdelerinin bulunması, stop words dediğimiz etkisiz kelimelerin çıkartılması, n- gram kullanılarak kelimelerin hece ya da kelime gruplarının oluşturulması ve kelime

vektörleştirme adımlarının gerçekleştirilmesi ile veri tam olarak bir sonraki adım olan modelleme adımlarına hazır hale getirilmektedir [64].

2.5.5.1 Gürültülü veri temizleme

Veri seti içerisindeki gürültülü veriler, NLP algoritmaları tarafından model eğitilirken modelin eğitiminde anlamlı olmayan gereksiz ve modelin performansını düşürecek kelimelerin veri setinden çıkarılmasını ifade etmektedir. Noktalama işaretleri, özel karakterler, sayılar, html etiketleri, alana özel anahtar kelimeler, kaynak kodu kaldırma, web sitesi linkleri, emojiler, konu başlıkları, çeşitli semboller, gürültülü veriler olarak bilinmektedir [65].

2.5.5.2 Duraklama kelimelerinin kaldırılması

NLP uygulamalarında kullanılan veri setlerinde cümlelerin anlamına etkisi olmayan fakat dilin yapısı gereği metin içerisinde sıkça kullanılan kelimelere durak kelimeler denilmektedir. Türkçede edat, bağlaç gibi kelimeler durak kelimeler olarak bilinmektedir. Duraklama sözcükleri herhangi bir insan dilinde bolca kullanılmaktadır. Bu kelimelerin veri setinden kaldırılması veri seti üzerinde herhangi bir olumsuz etki yapmadığından bu düşük anlam taşıyan kelimeleri genellikle veri setinden kaldırılmaktadır. Ancak bu kelimeleri bazı uygulamalarda çıkarmak modelimizin yetersiz öğrenmesine sebep olabilmektedir. Örneğin dikkat mekanizmasının kullanıldığı transformatör tabanlı metin üretme gibi uygulamalarda bu kelimeler arasındaki ilişkiler çok önemli olabilmektedir ve bu kelimelerin çıkarılması modelin iyi eğitilememesine neden olmaktadır [66].

2.5.5.3 Kelime köklerinin bulunması

Kelime köklerinin bulunması kelimeleri eklerinden temizlenerek kelimelerin temel kök durumuna dönüştürme işlemidir. Kelimelerin köklerinin elde edilmesi kullanılan dillere göre değişiklik göstermektedir. Kelimeleri kök haline getirmek büyük veri kümeleri arasındaki ilişkileri ve ortak noktaları belirlemek için kullanılmaktadır [62].

2.5.5.4 Simgeleme–Parçalama

Simgeleme veri setinde bulunan metinlerin doğal dil işleme uygulamalarında kullanabilmek için cümlelere, kelime öbeklerine, kelimelere, harf gruplarına ya da harflere parçalama işlemine denilmektedir. NLP algoritmalarına giriş verilerinin sayısal vektörler olarak

verilebilmesi için metin verilerin anlamlı parçalara ayrılıp sayısal değerlere dönüştürülmesi gerekmektedir. Bunun ilk adımı olarak metin verilerinin ne şekilde anlamlı parçalara ayrılması gerektiğine karar vermek gerekmektedir [65].

2.5.5.5 N-gram özelliklerini ayıklama

N-gram bir metindeki harflerin ya da kelimelerin n adet sayıda bir araya getirilmesi ile oluşturulan grupların elde edilmesini sağlayan bir yöntemdir. Metin içerisinde bulunan kelimeler tek başlarına anlamlı olabildiği gibi farklı kelimelerle birlikte kullanıldığında da farklı anlamlar kazanabilmektedir. Metin içerisinde bulunan kelimelerin ya da harflerin n-gram analizleri yapılarak doğal dil işleme uygulamalarında kullanılan algoritmaların daha iyi öğrenmesi sağlanmaktadır. N-gram teriminde kullanılan n harfi tekrar sayısını ifade etmektedir. Uygulamalarda genellikle 1-gram (unigram), 2-gram (bigram), 3-gram (trigram) gibi yöntemler kullanılmaktadır [67].

2.5.6 Kelime Vektörleri

NLP alanında çalışırken metin halinde bulunan kelimelerin bilgisayarın anlayabileceği bir sayısal vektöre dönüştürülmesi gerekmektedir. Kelimelerin bir özellik çıkartma vektörü haline getirilmesi makine öğrenmesi algoritmalarının kelimelerle ilgili önemli özelliklerin elde edilmesini sağlamaktadır. Kullanılacak makine öğrenmesi algoritmalarına bağlı olarak kelime vektörlerinin gösterimi ve parametreleri değişiklik gösterebilmektedir [68]. NLP alanında kullanılan kelime vektörleri temel olarak iki sınıfa ayrılmaktadır. Bunlar geleneksel (Frekans) tabanlı kelime gösterimi ve yapay sinir ağları (tahmin) tabanlı kelime gösterimi yöntemleridir

2.5.6.1 Geleneksel (frekans) tabanlı kelime gösterimi

Bu kelime gösterimi yönteminde istatistik-frekans tabanlı yöntemler kullanılarak kelimelerin sayısal vektörleri hesaplanmaya çalışılmaktadır. Bu gösterimde metin içerisinde bulunan kelimelerin bulunma sayıları hesaplanır. Bu tür istatistik-frekans tabanlı kelime vektörü oluşturma modellerinde kelimelerin cümle içindeki anlamsal özellikleri ve kelimelerin metin içindeki yerleşiminin bir önemi bulunmamaktadır. Bu yöntem kelimelerin sadece metin içerisinde bulunma sıklığına dikkat edilerek çalışmaktadır [69].

Kelime Torbası (Bag Of Words)

Geleneksel kelime vektörü hesaplama yöntemlerinden biri olan kelime torbası yöntemi sadece metin içerisindeki kelimelerin sayısını hesaplayan kelimelerin anlamı ve sırası ile ilgili hiçbir bilgi taşımayan bir kelime temsil yöntemidir. Sadece kelimelerin metin içerisindeki sayısı ile ilgilendiği için kelime torbası adını almıştır [70].

Tablo 2.1: Kelime Torbası Vektör Dönüşüm Örneği.

	Ateş	Su	Toprak	Hava	Dört	Temel	Elementten	Biridir
Ateş dört temel elementten biridir	1	0	0	0	1	1	1	1
Su dört temel elementten biridir	0	1	0	0	1	1	1	1
Hava dört temel elementten biridir	0	0	0	1	1	1	1	1

Tablo 2.1’de kelime torbası yönteminin nasıl çalıştığı gösterilmiştir. Bu dökümanda kelimelerin benzersiz listesi oluşturulmuştur. Daha sonra kelimelerin metin içindeki geçme sıklıkları belirlenmiştir. Aşağıdaki matrise kelime gösterim matrisi denilmektedir. Bu şekilde kelime vektörleri haline getirilen metinler artık makine öğrenmesi algoritmalarına girdi değeri olarak verilebilmektedir

“Ateş dört temel elementten biridir” = [1,0,0,0,1,1,1,1]

“Su dört temel elementten biridir” = [0,1,0,0,1,1,1,1]

“Hava dört temel elementten biridir” = [0,0,0,1,1,1,1,1]

“Toprak dört temel elementten biridir ” = [0,0,1,0,1,1,1,1]

Terim frekansı – Ters belge frekansı

Terim frekansı – Ters belge frekansı (TF-IDF) NLP uygulamalarında yoğun olarak kullanılan, bir metindeki herhangi bir kelimenin tüm metin içerisinde ne kadar önemli olduğunu hesaplamak için kullanılan istatistiksel bir yöntemdir. Bu yöntemde bir kelime bir dökümanda ne kadar çok geçiyorsa o kelimenin o doküman için çok az bilgi içerdiği düşünülmektedir. Bu nedenle doküman içerisinde çokca geçen durak kelimeler dediğimiz

edat, bağlaç, noktalama işaretleri gibi kelimelerin metin analizinde çok etkisi olmadığı düşünüldüğü için bu kelimelerin ağırlıklarının düşürülerek doküman içerisindeki etkileri azaltılmaya çalışılmaktadır [71].

TF-IDF kelimelerin terim frekansı (TF) ile ters belge frekansı (IDF) değerlerinin çarpılarak kelimelerin vektör değerleri hesaplanmaktadır. TF metin içerisindeki kelimelerin hangi sıklıkta geçtiğini hesaplamaktadır. IDF ise tüm doküman içerisindeki kelimelerin geçme sıklığını hesaplamaktadır. İstenen sonucun hesaplanabilmesi için veri setinde bulunan toplam belge sayısı kelimeyi içinde bulunduran belge sayısının toplamına bölünür ve çıkan sonucun algoritması alınarak hesaplaması yapılmaktadır. Buradan çıkan değer büyük olması o kelimenin bu doküman içindeki anlamının yüksek olması anlamına gelmektedir. Hesaplama sonucunda çıkan sonuç ne kadar sıfır değerine yaklaşırsa o kelimenin doküman içerisinde daha az önemli olduğu anlaşılmış olmaktadır. Bu yöntem kelimelerin doküman için öneminin ağırlıklandırılmasını sağlayan kolay ve basit anlaşılır sezgisel bir yöntemdir. Yalnız bu yöntemde de kelime torbası yöntemi gibi kelimelerin doküman içerisindeki anlamsal bağını ve kelime sırasını hesaba katmamaktadır. Ayrıca boyutsallık lanetinden ötürü gereksiz bellek kullanımının çok fazla olması yöntemin eksik tarafları olarak belirtilmektedir [72].

$$tf(t, D) = \frac{\text{terim sayısı}(t)}{\text{toplam terim sayısı}(D)} \quad (2.2)$$

D dokümanda geçen t terimi için terim frekansı değerini temsil eden $tf(t, D)$ değeri denklem 2.2'deki gibi hesaplanmaktadır.

$$idf(t, D) = \log \frac{N}{n_t} \quad (2.3)$$

N adet dokümandan oluşan bir veri kümesinde D dokümanında geçen t terimi için $idf(t, D)$ değeri denklem 2.3'deki gibi hesaplanmaktadır.

$$tf_{idf} = tf * idf \quad (2.4)$$

Tf_idf değeri ise denklem 2.4'deki gibi hesaplanmaktadır.

N-Gram yöntemi

N-gram yöntemi bir metindeki harflerin ya da kelimelerin n adet sayıda bir araya getirilmesi ile oluşturulan grupların birlikte incelenmesini öngören frekans tabanlı kelime vektörü oluşturma yöntemidir. Bir metni n adet ögeye ayırma işlemidir. Bu yöntem diğer istatistiksel-frekans tabanlı yöntemlere göre az da olsa yapısı gereği kelime bağlamı ve kelime sırasını göz önünde bulundurmaktadır. N-gram modelleri, uzun mesafeli bağlamı zayıf bir şekilde yakalamaktadır. N sayısının sürekli artması yöntemin daha iyi olacağı anlamına gelmemektedir. Değerlendirmeler sonucunda 6 gramdan sonra performans kazanımlarının sınırlı olduğu belirtilmektedir [73]. Aşağıda en çok kullanılan n-gram yöntemleri örnek verilerek açıklanmaya çalışılmıştır.

Tablo 2.2: N-Gram Harf Özellik Çıkarımı.

1- Gram (unigram)	m	a	s	a	d	a	n
2- Gram (bigram)	ma	as	sa	ad	da	an	
3- Gram (trigram)	mas	asa	sad	ada	dan		
4- Gram	masa	asad	sada	adan			

Tablo 2.2 de “masadan” kelimesinin harf düzeyinde n-gram gösterimi verilmiştir.

Tablo 2.3: N-Gram Kelime Özellik Çıkarımı.

1- Gram (unigram)	Yapay	sinir	ağlarının	temel	özellikleri
2- Gram (bigram)	Yapay sinir	sinir ağlarının	ağlarının Temel	temel özellikleri	
3- Gram (trigram)	Yapay sinir ağlarının	sinir ağlarının temel	ağlarının temel özellikleri		
4- Gram	Yapay sinir ağlarının temel	sinir ağlarının temel özellikleri			
5- Gram	Yapay sinir ağlarının temel özellikleri				

Tablo 2.3’te yapay sinir ağlarının temel özellikleri” cümlesinin kelime düzeyinde n gram gösterimi verilmiştir.

Genel olarak, birçok NLP uygulamasında N-Gram yöntemleri kullanılmaktadır. Bunlara örnek olarak; konuşma bölümü etiketleme, doğal dil oluşturma, sözcük benzerliği, duygu çıkarma ve tahmine dayalı metin girişi dahil olmak üzere benzer uygulamalar N-gram modellerinden yararlanmaktadır.

2.5.6.2 YSA tabanlı kelime temsil yöntemleri

YSA tabanlı yöntemler doküman içindeki kelimelerin özniteliklerini tespit etmede geleneksel-frekans tabanlı yaklaşımlara göre yeni olan ve daha başarılı sonuçlar elde edilen yöntemlerdir. Kelimelerin metin içindeki bağlamsal özelliklerini yakalayan ve metin içindeki yerleşimini keşfeden yapay sinir ağları yöntemi kullanılarak kelimelerin sayısal vektörlerinin oluşturulduğu bir yöntemdir. Bir metin içinden kelimelerin en iyi özniteliklerini çıkarmak o metin içerisindeki kelimelerin söz dizimsel yerleşimine, anlamsal bağlamına ve kelimelerin birbiriyle olan ilişkilerine bağlıdır. Bu yöntemlerde metin içerisinde birbirine yakın olan kelimeler vektör uzayında birbirine daha yakın olarak bulunurken birbirine anlamsal olarak farklı olan kelimeler uzak mesafede bulunmaktadır [74]. YSA kullanılarak önceden tanımlanmış kelime vektörleri oluşturmak için Word2vec, Glove, Fasttext, Doc2vec, Elmo, Bert, Gpt-1-2-3 gibi önceden eğitilmiş hazır mimariler kullanılmaktadır.

Word2vec

Word2vec yöntemi 2013 yılında Mikolow ve arkadaşları tarafından ortaya atılmış metin içindeki kelimelerin sayısal vektörlerini oluşturmayı sağlayan denetimsiz olarak çalışan ve yapay sinir ağı kullanan bir yöntemdir. Word2vec yönteminde tek gizli katmanı olan bir sinir ağı kullanılmaktadır. Sinir ağına girdi verisi olarak verilen metinler yapay sinir ağından geçirilerek kelimelerin sayısal vektörleri oluşturulmaktadır. Word2vec yöntemi geleneksel kelime gösterimi olan kelime torbası ve TF-IDF yöntemlerinin yapamadığı kelimelerin anlamsal bağlamını ve kelime yerleşimini hesaplayabilmektedir [74].

Wor2vec yöntemi metin içindeki kelimelerin anlamsal ilişkilerini yakalayabilmek için cbow ve skip-gram denilen iki farklı yöntem kullanılmaktadır. Cbow yönteminde komşu kelimeler

girdi olarak kullanılarak pencere denilen hareket etme parametresinin merkezinde bulunan kelimeyi tahmin etmeye çalışmaktadır. Skip-gram yöntemi ise cbow yönteminin tersi gibi çalışarak verilen kelimenin komşu kelimelerini tahmin etmeye çalışmaktadır. Skip-gram ve Cbow yöntemleri kullanım alanlarına göre farklılık gösterebilmektedir. Skip Gram daha küçük veri setlerinde ve metin içerisinde az görülen kelimelerin analizinde daha iyi sonuçlar vermektedir fakat eğitim süreci yavaş olmaktadır. Cbow yöntemi ise daha büyük veri setlerinde daha iyi sonuçlar vermektedir. Cbow yöntemi skip gram yöntemine göre daha hızlı eğitim süreci gerçekleştirmekte ve metin içerisinde sık görülen kelimelerin analizinde daha iyi sonuçlar vermektedir [75].

Fasttext

Fasttext modeli facebook araştırmacıları tarafından 2016 yılında word2vec modelinin bir başka versiyonu olarak ortaya çıkan kelime gömme yöntemidir. Fasttext modelinin word2vec modelinden farkı metin içerisindeki kelimeleri n-gramlarına ayırarak çözümleme yapmasıdır. Böylece word2vec yöntemiyle yeteri derecede bulunamayan anlamsal bağlılık bu yöntemde daha iyi bulunabilmektedir. Word2vec modelinde metin içindeki kelimeler vektör haline getirilirken fasttext yönteminde metindeki kelimeler n-gramlarına ayrılarak hesaplanır ve daha sonra bu her bir n-gram vektörleri toplanarak kelimelerin kelime vektörleri hesaplanmaktadır. Fasttext modeli Türkçe gibi morfolojik açıdan zengin olan dillerin kelimelerini vektörlerine ayırırken bu dillerin kelime iç yapısını daha iyi analiz edilebilmektedir. Fasttext modeli doküman içindeki az görülen kelimelerin bağlamsal ilişkilerini daha iyi yakalayabilmektedir. Ayrıca doküman içerisinde eğitim esnasında hiç geçmeyen kelimelerin anlamsal bağlılıklarını hesaplayabilmek için kelimeler n-gramlarına ayrılabilen ve çözümlenmesi yapılabilmektedir [76].

Glove

Glove kelime temsil yöntemi word2vec modelinden sonra en çok kullanılan önceden eğitilmiş kelime gömme yöntemlerinden biridir. Stanford Nlp Group araştırmacıları tarafından 2014 yılında geliştirilmiş ve NLP uygulamalarında kullanılabilmek için açık kaynak kodlu olarak dağıtımına açık hale getirilmiştir. Glove kelimelerin vektör temsillerini oluşturmak için global düzeyde kelimelerin ne sıklıkla birlikte geçtiği hesaplamalarını yaparak birlikte geçme olasılıklarını bir oran olarak kabul etmekte ve onu bir kelime temsili

olarak kullanılmaktadır. Glove modelleri kelimelerin oransal istatistiğini kullanarak kelimelerin vektörlerini hesapladığından birbiri ile ilgili kelimeleri ve birbiri ile alakasız kelimelerin ayırt edilmesinde daha iyi sonuçlar vermektedir. Glove modeller word2vec modelleri gibi yerel kelime bağlamını hesaplamak için bir pencere kullanmak yerine metnin tamamını istatistiksel hesaplamalar kullanarak kelimelerin birlikte bulunma matrisini oluşturup daha iyi kelime geliştirmeleri sağlamak ve daha iyi bir öğrenme modeli haline gelmektedir [77].

Doc2Vec

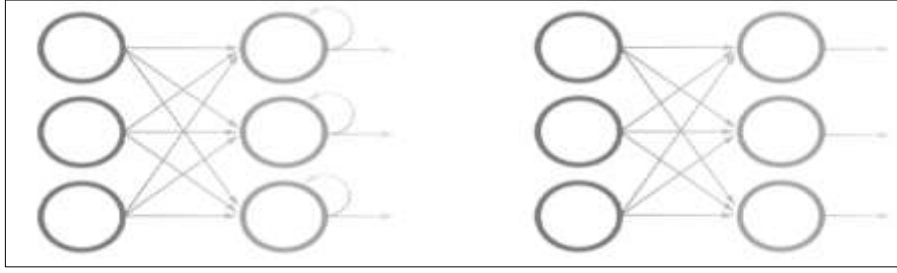
Doc2vec modeli word2vec modelinin benzer mantıkta çalışan farklı bir versiyonudur. Temel amacı cümle ve paragraf yapılarını sayısal değerler içeren vektörlere dönüştürmek olan denetimsiz bir makine öğrenmesi modelidir. Word2vec modeli doküman içindeki kelimeler arasındaki anlamsal ve yerleşim ilişkileri hakkındaki bilgileri tutabilmektedir. Doc2vec yöntemi ise sadece kelimeler arasında değil daha büyük birimler olan cümle ve paragraflar arasındaki ilişkileri öğrenebilmek için kullanılmaktadır. Doc2vec yöntemi kelime temsillerini değil belge temsillerini oluşturmaktadır [78].

2.6 NLP'DE KULLANILAN DERİN ÖĞRENME ALGORİTMALARI

2.6.1 Tekrarlayan Sinir Ağları

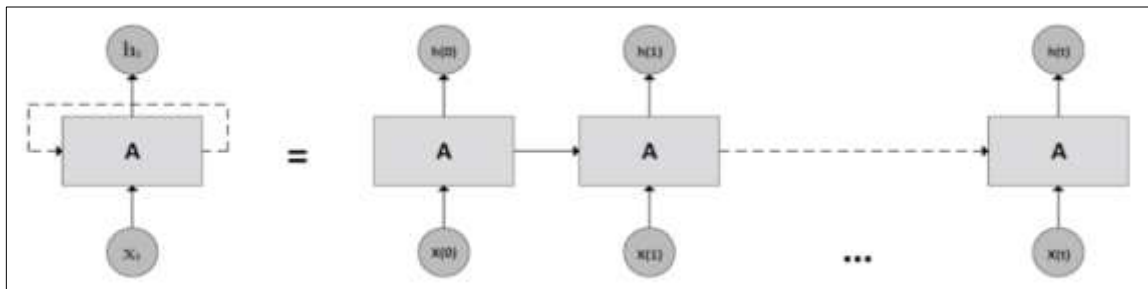
Tekrarlayan sinir ağları (Recurrent Neural Network - RNN) 1980 yıllarında kullanılmaya başlanılan genellikle NLP uygulamalarında kullanılan bir derin öğrenme algoritmasıdır. Sıralı ve zaman serisi şeklindeki zamanla değişen verileri işleyerek öğrenebilmek için tasarlanmışlardır. Geleneksel YSA'da girdilerin sinir ağına giriş sırasının bir etkisi bulunmamaktadır. Girdiler birbirinden bağımsız olarak işlenmekte ve sonuç üretilmektedir. Ayrıca geleneksel YSA bu girdi bilgilerini unutmaktadır. RNN modelleri ise derin öğrenme ağında eğitim gerçekleştirirken bir önceki veriye ait bilgileri hafızasında tutmaktadır. Gerçek bir öğrenmenin olabilmesi için daha önceki bilgilerin unutulmaması ve daha sonraki bilginin tahmin edilmesinde geçmiş bilgilerin hatırlanarak kullanılması gerekmektedir [79].

RNN modellerinin hafızalarının olması çok farklı alanlarda etkili kullanılmasını sağlamaktadır. RNN'ler özellikle NLP alanında zaman serisi ve sıralı veri işlemleri özelliğinden dolayı yoğun bir şekilde kullanılmaktadır [80].



Şekil 2.11: RNN ve İleri Beslemeli YSA Gösterimi [80].

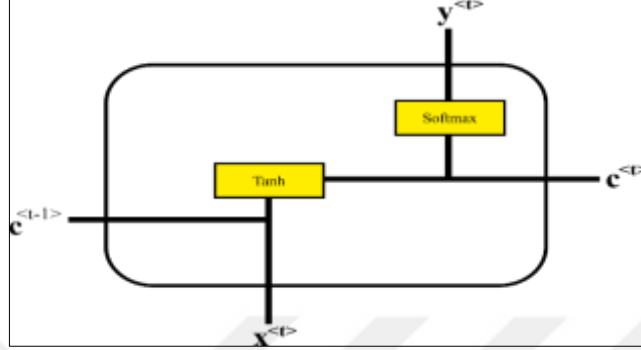
Şekil 2.11’de RNN ağı ile genel yapay sinir ağının çalışma şekli gösterilmektedir. Genel yapay sinir ağında bütün girişler nöronlardan geçerken ağırlık, bias ve aktivasyon fonksiyonlarından geçirilerek bir matematiksel işlem sonucunda bir çıktı üretilmektedir. YSA’dan elde edilen çıkış değerleri olması gereken doğru çıkış değerleri ile karşılaştırılarak aralarındaki hata tespiti yapılmaktadır. Bu hatalar geriye doğru ağırlık ve bias değerlerine dağıtılarak hata oranı sürekli olarak azaltılmaya çalışılıp en uygun YSA modeli oluşturulmaya çalışılmaktadır [80]. Görüldüğü üzere bu YSA’da geçmiş bilgileri hatırlama, zaman ve sıraya bağlı hiçbir kavram bulunmamaktadır. Ayrıca bu ağ yapısında giriş değerlerinin tamamen birbirlerinden bağımsız olarak işlendiği ve geçmiş bilgilerle aralarında hiçbir bağlantının bulunmadığı görülmektedir. RNN modellerinde bir sonraki değer tahmin edilmesinde önceki değerlerin bilgilerinin de hatırlanarak kullanılması gerekmektedir. RNN modelleri sonraki değeri tahmin edebilmek için önceki girdiler arasındaki bağlantıları kullanarak tüm ilişkileri hatırlamakta ve yeni değeri tahmin etmektedirler [81].



Şekil 2.12: Rnn Modeli [81].

Şekil 2.12’de RNN modelinin bir hücresi ve onun çalışma prensibinin açık hali görülmektedir. RNN üniteleri birim zamanda verilen verileri işleyerek elde ettiği çıktıları

tekrar kendisine girdi olarak alıp yeni girdi değerlerini de kullanarak geçmiş bilgileri hatırlamakta ve yeni çıktı değerlerini tahmin etmeye çalışmaktadır. Görselde olduğu gibi üretilen değer bir sonraki adımını beslemekte ve yeni bir değer üretmek için döngüsel bir yapı için bu durum sürekli tekrarlanmaktadır.



Şekil 2.13: Rnn Model İç Yapısı [82].

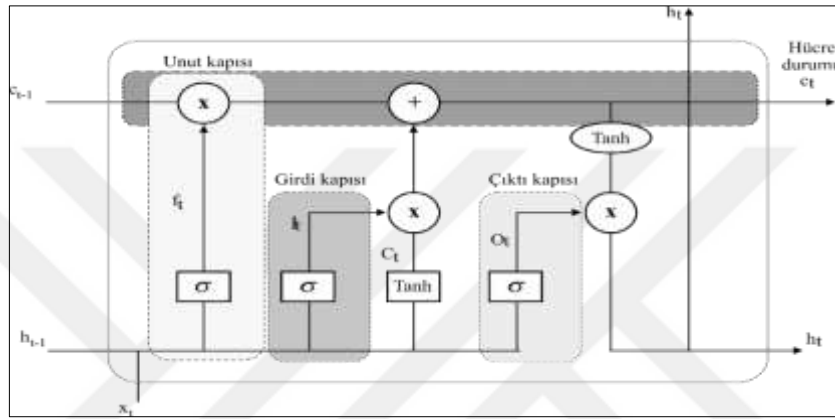
Şekil 2.13'te RNN ağıının birim t zamanda çalışması esnasında yapılacak adımlar belirtmeye çalışılmıştır. $X^{<t>}$ girdi değerini $Y^{<t>}$ çıktı değerini ve $C^{<t>}$ değeri ise RNN modelinin özelliği olan daha önceki girdilerden elde edilen bilgileri tutan parametrelerdir. Bu yapıdan $C^{<t>}$ parametresi çıkarılıp sadece $X^{<t>}$ girdi değerini ve $Y^{<t>}$ çıktı değerini kullansaydık bu yapının genel YSA modelinden hiçbir farkı kalmayacaktır. RNN modelini genel YSA'dan ayıran özellik her $X^{<t>}$ girdi değeri hesaplandığında önceki girdilerden elde edilen sonuçların $C^{<t-1>}$ parametresinde tutulup yeni değer elde edilmesinde bu bilgilerin hesaplama dahil edilmesidir [82].

RNN'ler güçlü ve basit bir model olmasının yanında kendi içinde zorlukları ve eksiklikleri olan bir model olarak bilinmektedir. Modelin en büyük sorunu kaybolan ve patlayan gradyan etkisi sorunudur. Kaybolan gradyan sorunu çok derin yapıları RNN'lerde bir sorun olarak karşımıza çıkmaktadır. Temel sebebi aktivasyon fonksiyonlarının aldığı değerlerin geri yayılım esnasında hataların azaltılması için ağırlıkların güncelleştirilmesi aşamasında başlamaktadır. Buradaki sorun alınan sürekli türev alma işleminde eğimin kaybolacak kadar küçük olması ve ağırlıkların değerlerinin artık güncelleştirilemeyecek duruma gelmesinden kaynaklanmaktadır. Bunun sonucu olarak RNN modellerinin eğitilme aşaması durmaktadır. Bu sorunun çözümü olarak aktivasyon fonksiyonu olarak tanh aktivasyon fonksiyonunun kullanılması önerilmektedir. Çünkü tanh aktivasyon fonksiyonunun aldığı değerler YSA'nın eğitimi ilerledikçe ağırlıkların çarpılması ile oluşan değer çok küçük değere ulaşması diğer

aktivasyon fonksiyonlarına göre daha zordur. Diğer bir yöntem olarak RNN modellerinin gelişmiş hali olan LSTM/GRU gibi modellerin kullanılmasıdır [81].

2.6.2 Uzun Kısa Süreli Bellek

Uzun kısa süreli bellek (Long short-term-LSTM) RNN'lerin bir türüdür. Yapılarında çeşitli kapılar kullanarak metin, ses ve video gibi zaman serilerinin modellenmesinde RNN'lere göre birçok avantajı bulunmaktadır.



Şekil 2.14: LSTM Modeli [79].

Şekil 2.14’de LSTM modeli görülmektedir. LSTM’in temel yapısı hücre durumu (c_t) denilen iletim hattından ve unut kapısı, girdi kapısı, çıktı kapısı olmak üzere üç kapıdan oluşmaktadır. Hücre durumu ağırlık tahmin yapmak için kullanılan ve ağırlık hafızası olarak bilinen ve önceki girdi değerlerinin bilgilerini hat boyunca taşıyan bir iletim hattıdır. Bu sayede LSTM modelleri RNN modellerinin en büyük sorunu olan kısa süreli bellek engelinin önüne geçerek giriş değerlerinin bilgilerini daha uzun süre hafızalarında tutabilmektedir. Hücre durumu hattının taşınması gereken bilgiler LSTM modelinin yapısında bulunan geçit kapıları yardımıyla belirlenir. Bu kapılar bilgilerin hangisinin unutulup bırakılacağını ya da gerekli bilgi olarak taşınacağını belirlemektedir. Kapılar gerekli ve gereksiz bilgilerin tespitini kullandığı sigmoid fonksiyonu ile tespit etmektedirler. Sigmoid fonksiyonundan geçen girdi verileri 0 değeri üretildiğinde unutulur, 1 değeri üretildiğinde ise girdi değerleri hücre durumu hattı üzerinden çıkış değerine doğru ilerlemektedirler [79].

Unutma kapısı; önceki hücrelerden gelen ve mevcut girdi değerlerinin sigmoid fonksiyonundan geçirilerek sonucunun 0 ya da 1 olması durumuna göre hücre durumu ile hat boyunca LSTM hücresi içerisinde taşınmasını sağlamaktadır. Giriş kapısında ise; hücre

durumu tarafından tutulan bilgilerin güncellenmesi yapılır. Önceki ve şimdiki bilgilerin sigmoid aktivasyon fonksiyonundan geçirilerek güncelleme işlemi yapılıp yapılmayacağına karar verilir. Yine bu kapıda tanh aktivasyon fonksiyonu kullanılarak buradan çıkan sonuçla sigmoid fonksiyonundan çıkan sonuç çarpılıp hangi bilgilerin güncelleneceğine karar verilir. Çıkış kapısı ise; LSTM hücresi içindeki bir sonraki zaman adımında kullanılacak gizli durumun taşıyacağı bilginin ne olacağına karar vermektedir [83].

Gizli durum (h_t) bize önceki giriş verisi hakkında bilgiler vermektedir. Gizli durum adımı yeni tahminlerde bulanabilmek için kullanılmaktadır. Çıkış kapısında yapılan işlemler ise gizli durum bilgisinin güncelleştirilmesi için yapılmaktadır. Burada yapılan işlemler gizli durum değeri ile giriş değerini sigmoid ve tanh aktivasyon fonksiyonlarından geçirilerek bu fonksiyonlardan çıkan değerlerin çarpılması sonucunda gizli durum hücresindeki bilgilerin güncelleştirilmesi için yapılmaktadır. Bu işlemler sonucunda artık bir sonraki zaman diliminde kullanılacak gizli durum ve çıkış değerleri diğer adımlara taşınmaktadır. Bir LSTM hücresinin matematiksel denklemleri aşağıda verildiği gibidir [84].

Giriş / güncelleme kapısının aktivasyon vektörünü hesaplamak için kullanılan formül denklen 2.5’de gösterildiği şekildedir

$$i_t = \sigma(x^t U^i + h_{t-1} w^i) \quad (2.5)$$

Unutma kapısı vektörünün değerini hesaplamak için kullanılan formül denklen 2.6’da gösterildiği şekildedir.

$$f_t = \sigma(x^t U^f + h_{t-1} w^f) \quad (2.6)$$

Çıkış kapısının aktivasyon vektörünün değerinin hesaplanması için kullanılan formül denklen 2.7’de gösterildiği şekildedir.

$$o_t = \sigma(x_t u^o + h_{t-1} w^o) \quad (2.7)$$

Hücre girişi aktivasyon vektörünün değerinin hesaplanması için kullanılan formül denklen 2.8’de gösterildiği şekildedir.

$$\hat{c}_t = \tanh(x_t U^g + h_{t-1} w^g) \quad (2.8)$$

Hücre durum vektörünün değerinin hesaplanması için kullanılan formül denklen 2.9'da gösterildiği şekildedir.

$$c_t = \sigma(f_t * c_{t-1} + i_t * \hat{c}_t) \quad (2.9)$$

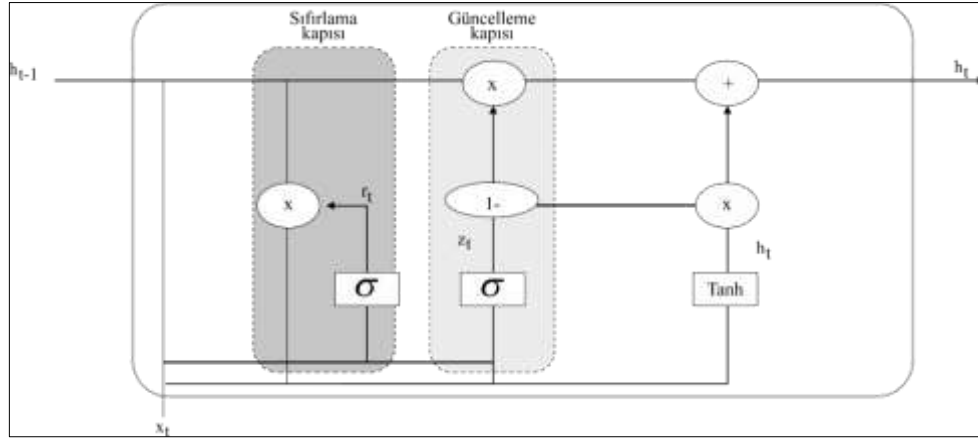
Çıkış vektörü olarak da bilinen gizli durum vektörünün değerinin hesaplanması için kullanılan formül denklen 2.10'da gösterildiği şekildedir.

$$h_t = \tanh(c_t) * o_t \quad (2.10)$$

2.6.3 Geçitli Tekrarlayan Birim

Geçitli tekrarlayan birim (Gated Recurrent Units - GRU) GRU'lar standart olan RNN modellerinin daha ileri seviyesidir. GRU modeli RNN modellerinin en büyük problemi olarak görülen kaybolan gradyan sorununu çözmek için güncelleme ve sıfırlama kapısı olmak üzere iki kapı kullanmaktadır. Bu kapılar çıktı değerinin hesaplanmasında daha önce kullanılan girdi bilgilerinin hatırlanıp bir sonraki adıma iletilmesinin kararını veren temel yapılardır. Güncelleme kapısı LSTM modellerinde bulunan unutma ve giriş kapılarının yaptığı görevi tek kapı olarak yapmaktadır. Bu kapının yaptığı işlev hat boyunca iletilecek geçmiş bilgilerin yani önceki zaman adımlarından gelen bilgilerin ne kadarının kullanılmak üzere geleceğe aktarılacağına belirlendiği kapılardır. Sıfırlama kapısı ise modelden alınan geçmiş bilgilerin ne kadarının unutulacağına kararının verildiği kapı olarak kullanılmaktadır [85].

GRU modeli LSTM modeline göre daha az işlem adımı kullanmakta buna bağlı olarak daha az bellek tüketmekte ve daha hızlı çalışan bir modeldir. LSTM modelleri veri setinin daha büyük ve zaman sorunu olmayan uygulamalarda tercih edilirken, GRU modelleri daha az veri setine sahip ve hızlı çalışması gereken uygulamalarda daha çok tercih edilmektedir. Hangisinin daha performanslı olduğu ile ilgili herhangi bir karşılaştırma yapılamamaktadır. Kullanıldıkları uygulamalara göre performansları değerlendirilmelidir [87]. Şekil 2.15'de GRU modeli görülmektedir.



Şekil 2.15: GRU Modeli [79].

GRU modelinin parametrelerinin matematiksel denklemleri aşağıda gösterilmektedir [87].

Sıfırlama kapısı geçmişten gelen bilgilerin ne kadarının unutulacağını belirlediği kapıdır. Hesaplamak için kullanılan formül denklem 2.11’te gösterildiği gibidir.

$$r_t = \sigma (x_t W^r + h_{t-1} W^r) \quad (2.11)$$

Güncelleme kapısı ise gelen bilgilerden hangilerinin atılıp hangilerinin kullanılacağı hesaplamasının yapıldığı kapıdır. Hesaplamak için kullanılan formül denklem 2.12’da gösterildiği gibidir.

$$z_t = \sigma (x_t W^z + h_{t-1} W^z) \quad (2.12)$$

Geçerli bellek içeriğinin hesaplanması için kullanılan formül denklem 2.13’da gösterildiği gibidir.

$$h_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \quad (2.13)$$

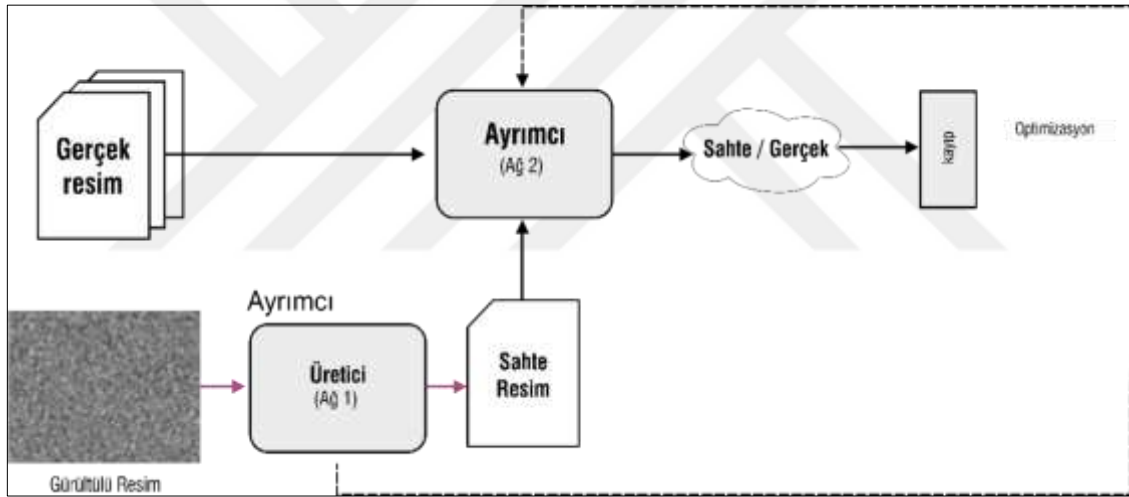
Geçerli zaman adımıdaki son bellek hücresi bu son adım olarak işlem görür mevcut hücre için bilginin tutulduğu ve ağ ortamına iletilen $h(t)$ vektörünün hesaplandığı kısımdır. Kullanılan formül denklem 2.14’de gösterildiği gibidir.

$$h_t = [(1 - Z_t) * h_{t-1} + z_t * h_t] \quad (2.14)$$

2.6.4 Üretken Düşman Ağları

Üretken Düşman Ağları (Generative Adversarial Networks-GAN) derin öğrenme yöntemlerini kullanan üretken modellemeye yönelik bir yaklaşımdır. GAN’lar giriş gerçek verilerine benzer özelliklere sahip veriler üretebilen bir tür denetimsiz derin öğrenme ağıdır.

GAN'lar üretici ağı bir rakiple rekabet etmesi gereken bir oyun teorik senaryosuna dayanmaktadır. GAN mimarisi görüntü üretme, müzik üretme, konuşma üretme veya düzyazı üretme gibi birçok uygulama alanında yaygın olarak kullanılmaktadır [88]. GAN'lar iki sinir ağından oluşmaktadır. Bir Üretici (Generator) ve bir Ayırıcı (Discriminator) olmak üzere, ikisi de düşmanca bir oyun oynuyormuş gibi çalışmaktadır. Üreticinin amacı, eğitim setindekilere benzer veriler üreterek ayırıcıyı kandırmaktır. Ayırıcı, gerçek verilerden sahte verileri belirleyerek kandırılmamaya çalışmaktadır. Üretici ve ayırıcı, her ikisi de sinir ağlarıdır. Eğitim aşamasında her ikisi de birbirleriyle rekabet halinde çalışmaktadır. Kısacası, ayırt edicinin işi, üretilen verinin gerçek mi yoksa sahte mi olduğunu orijinal veriyle karşılaştırarak belirlemektir. Üreticinin işi, orijinal örneğe yakın örnekler üreterek ayırıcıyı kandırmaktır. Adımlar çok kez tekrarlanır ve bunda üretici ve ayırıcı her tekrardan sonra kendi işlerinde daha iyi hale gelmektedir [89].



Şekil 2.16: GAN Mimarisi [89].

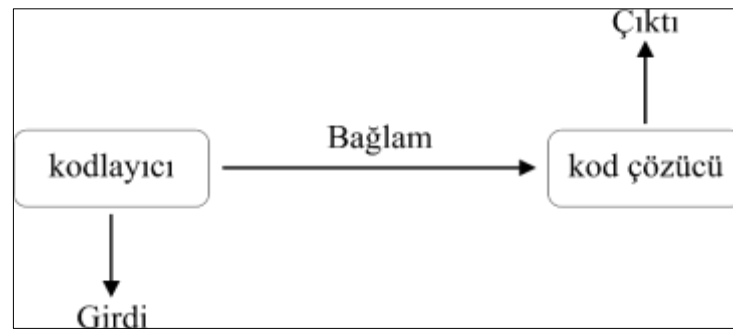
Şekil 2.16’da GAN mimarisi görülmektedir. Üretici verilen eğitim setindekilere benzer veriler üretir. Bu oluşturulan yeni veriler daha sonra gerçek veri kümesinden alınan bir veri akışıyla birlikte ayırıcıya beslenir. Ayırıcı hem gerçek hem de sahte verileri karşılaştırır ve 0 ile 1 arasında bir sayı olan olasılıkları döndürür; burada 1, gerçek olduğuna dair bir öngörüü temsil ederken 0, sahte olduğunu temsil etmektedir. Üreticinin gerçek değerlere benzeyen çıktılar elde edebilmesi için ayırıcının üretilen çıktılarının gerçek mi yoksa sahte mi olduğunu nasıl ayırt edebilmektedir. Model bunu üretici ve ayırıcının kayıp fonksiyonlarının değerine bakarak öğrenmektedir. Kayıp fonksiyonları üretilen veri ile gerçek veri arasındaki benzerlik mesafesini ölçmektedir. Üretici kayıp fonksiyonunun değerini düşürmeye

çalışırken ayırıcı ise yükseltmeye çalışmaktadır. Üretici kayıp fonksiyonun değerini minimize etmek için ayırıcıya bağlıdır. Ayırıcı bu işlemlerden sonra sonuç olarak gerçek ya da sahte çıktıları üretir. Çıktı 0 (sahte) ise, üretici kaybı, ayırıcının sahte olarak sınıflandırdığı bir örnek ürettiği için üreticiyi cezalandırır. Kayıp hesaplandıktan sonra üretici ağırlıkları ayırıcı ağı üzerinden üreticiye geri yayılım yoluyla güncellenerek dağılır. Bu önemlidir, çünkü bir üreticinin parametrelerinin etkisi büyük ölçüde ayırıcınıninkine bağlıdır, çünkü üretici geri bildirim alır ve aynısını daha 'gerçek' görüntüler üretmek için kullanır [90].

2.6.5 Kodlayıcı ve Kod Çözücü Modeller

Günümüzde NLP alanında derin öğrenme yöntemlerinden RNN/LSTM/GRU modelleri etkili bir şekilde kullanılmıştır. Bu modellerin kendi içlerinde eksik yönleri bulunmaktadır bu durumu ortadan kaldırabilmek için yapılarında RNN/LSTM/GRU yapılarının kullanıldığı Kodlayıcı (Decoder) ve Kod çözücü (Encoder) modeller kullanılmaya başlanmıştır. Kodlayıcı ve kod çözücü modeller uzun dizi işlemenin karmaşık olduğu görevleri yerine getirmede başarılı olmaktadır [91].

2014 yılından itibaren NLP uygulamalarında kodlayıcı-kod çözücü modeller standart bir yaklaşım haline gelmiştir. Bu model sayesinde artık uçtan uca değişken uzunluktaki giriş ve çıkış dizelerini işlemek daha kolay ve daha başarılı hale gelmektedir. Bu modelin gücü, farklı uzunluklardaki dizileri birbirine eşleyebilmesinde yatmaktadır [92]. NLP’de verilerdeki kelimelerin sırasının ve anlamının büyük önemi bulunmaktadır. Dolayısıyla bu modeller, özellikle girdi ve çıktıların farklı boyut ve kategorilere sahip olduğu durumlarda, sıra tabanlı herhangi bir probleme çözüm olarak kullanılabilir [91]. Bu modellerin NLP uygulamalarından metin özetleme, soru cevaplama, makine çevirisi, metin üretme, varlık ismi tanıma, sohbet robotları ve zaman serisi tahmini gibi diziden diziye tahmin problemlerinde başarılı olduğu kanıtlanmıştır.



Yukarıda kodlayıcı kod çözücü modelinin bileşenleri görülmektedir. Kodlayıcı kod Çözücü modeli, kodlayıcı ve kod çözücü olmak üzere iki alt ağıdan oluşur. Sol taraftaki kodlayıcı, kaynak dilden girdi olarak dizileri alır ve sonuç olarak, tüm bilgilerini özetlemeye veya yoğunlaştırmaya çalışarak girdi dizisinin kompakt bir temsiliyi üretir. Oluşan bu çıkış değeri kod çözücüye giriş değeri olarak verilir. Her zaman diliminde kod çözücü bu girdiyi ve mevcut durumu kullanarak çıktı dizisinin bir elamanını oluşturur ve sonraki adımlar için kendi bilgisini güncelleştirir. Girdi ve çıktı dizileri sabit boyutta olabileceği gibi girdi dizisinin uzunluğu çıktı dizisinden uzunluğundan farklı olabilmektedir. Bu modelin kritik noktası, kod çözücünün tek bir çıktı ögesindeki giriş dizisinin en eksiksiz ve anlamlı gösterimini kod çözücüye nasıl sağlayacağıdır, çünkü bu vektör veya durum, kod çözücünün üretmek için girdiden alacağı tek bilgidir. Bu girdiden elde edilen çıktı giriş ne kadar uzun olursa, tek bir vektöre bağlamsal olarak sıkıştırmak o kadar zor duruma gelmektedir [93].

2.6.6 Dikkat Mekanizması

NLP uygulamalarında son zamanlarda en çok kullanılan yöntemlerden biri de dikkat (Attention) mekanizmasıdır. Bu yöntem daha önce NLP alanında kullanılan derin öğrenme algoritmalarının çalışma mantığını değiştirmektedir. Dikkat modeli ilk olarak Bahdanau ve arkadaşları tarafından makine çevirisi görevleri için NLP’de kullanılmaya başlanmıştır [94]. Kodlayıcı kod çözücü modeller tüm giriş verilerini bir kerede işleyerek verinin tüm bilgilerin özetini tutan bir çıkış özet vektörü oluşturmaktadır. Bu yaklaşımın çok fazla eksikliği bulunmaktadır. Kodlayıcı modeller giriş verisiyle ilgili yeterli bilgiye sahip olamadıklarında kod çözücüye giriş verisiyle ilgili yetersiz bilgi aktaracağı için kod çözücü metin ile ilgili doğru tahminler yapamayacaktır. Bundan dolayı kodlayıcı-kod çözücü modellerinin giriş ve çıkış verilerin uzunluğu arttıkça bu modellerin performansı yetersiz kalmaktadır [95].

Bu sorunların üstesinden gelebilmek için dikkat mimarisi önerilmiştir. Bu yöntemde kodlayıcı kod çözücü modellerdeki gibi giriş verisinin tüm bilgisini artık tek bir özet vektör olarak oluşturmaya çalışılmamaktadır. Kod çözücü artık sadece kodlayıcı vektöre bağlı kalarak çıktı üretmek yerine her adımda çıktıyı üretirken giriş verilerinin farklı bölümlerine odaklanarak çıktı verisini üretmeye çalışmaktadır. Daha da önemlisi, modelin giriş cümlesine ve şu ana kadar ne ürettiğine bağlı olarak neye dikkat etmesi gerektiğini öğrenmesine izin vermektedir. Kelime vektörlerinin ağırlıklarının hesaplanmasında Bahdanau' tarafından geliştirilen dikkat hesaplama yöntemi kullanılmaktadır. Bu yöntemde

dikkat katmanında, her kelimenin kendisiyle ilişkilendirilmiş bir ağırlığı vardır. Ağırlık daha fazla ise, girdi dizisindeki belirli bir sözcüğün çıktı dizisindeki bir sonraki sözcüğü tahmin etmede daha güçlü bağı olduğu anlamına gelmektedir [94].

2.6.7 Transformer Network

2017 yılında ortaya atılan ve özellikle NLP alanında etkili bir şekil kullanım alanı bulan trasformer modelleri yapısında sadece dikkat mekanizması kullanılmaya başlanmış ve uzun kelime dizeleri arasındaki bağlantı kurma sınırlamasının üstesinden gelmeyi başarmışlardır. Bu durum daha uzun kelime dizelerinin eğitimi kolaylaştırmakta ve hızlandırmaktadır. Ayrıca transformer modeller paralel veri işleme özelliklerine sahip oldukları için daha büyük veri setlerini daha kısa sürede işleyebilmektedirler [96].

2.6.8 Dil Modellerinden Kelime Gömmeleri (ELMO)

Dil modelinden kelime gömmeleri (Embeddings from Language Model - ELMO) modeli AlenNLP araştırmacıları tarafından geliştirilen bir doğal dil işleme modelidir. ELMO kelime vektörlerini iki katmanlı çift yönlü LSTM modellerini kullanarak gerçekleştirilmektedir. ELMO diğer kelime vektörleri olan Word2vec ve Glove modellerinin yapamadığı farklı cümlede kullanılan kelimelerin bağlamını yakalayabilir ve farklı cümlelerde farklı bağlamda kullanılan aynı kelime için farklı yerleştirmeler oluşturabilmektedir. Bu kullanım kelime yerleştirmelerinin daha fazla etkili bilgiye sahip olduğuna ve kelime bağlamlarının daha iyi öğrenileceği anlamına gelmektedir. ELMO modeli gücünü çok büyük veri seti ile eğitilmiş olmasından almaktadır. Bu kadar büyük veri seti ile eğitilen model kullanılan dil ile ilgili çok fazla dil bilgisini ve kelime ilişkilerini iyi bir şekilde öğrenebildiği için NLP uygulamalarında yüksek performans göstermektedir. Bağlama duyarlı kelime yerleştirmelerini ve büyük verinin gücünü iyi kullanan ELMO modeli duygu analizi, metin özetleme, makine çevirisi, soru cevaplama gibi doğal dil işleme alanlarında etkili bir şekilde kullanılmaktadır [97].

2.6.9 Transformers'dan Çift Yönlü Kodlayıcı Temsilleri (BERT)

Transformers'dan Çift Yönlü Kodlayıcı Temsilleri (Bidirectional Encoder Representations from Transformers –BERT) Google AI Language araştırmacıları tarafından 2018 yılında ortaya atılan NLP alanında çok etkili kullanım alanı bulan önceden eğitilmiş bir dil modelidir [98]. BERT modeli temel olarak dikkat mekanizması kullanan bir trasformer modelinin ürünü

olarak ortaya çıkmıştır, BERT modeli eğitimini gerçekleştirirken bir cümledeki kelimeleri maskeleyerek tahmin etmeye çalışmaktadır [99]. Bert modelinin bu şekilde eğitiminin gerçekleştirilmesi tek yönlü bir dil modeline göre kelimelerin anlamsal bağlamlarının elde edilmesinde daha iyi sonuç vermektedir.

BERT modeli GPT3 gibi çok büyük veri setleri ile eğitilmiş transfer öğrenme modellerinden farklı olarak kaynak kodlarına sınırsızca erişilebilmektedir. Bu durum NLP uygulamalarıyla çalışan kişilere çok büyük veri setleri ile çalışmadan önceden eğitilmiş kelimelerden alınan ağırlık değerleri kullanılarak mevcut uygulamalarını çok büyük veri setiyle çalışıyormuş gibi eğitebilme fırsatı vermektedir [100]. NLP alanında BERT modelinin dışında önceden büyük veri setleri üzerinde farklı derin öğrenme yöntemleri kullanılarak eğitilmiş farklı modeller bulunmaktadır. Bunlar; Glove, Elmo, Electra, Xlnet, T5, Turing Nlg, Gpt, Gpt-2, Gpt-3 gibi birçok ön eğitilmiş modellerdir.

2.6.10 Üretken Ön İşlemeli Dönüştürücü Model (GPT-3)

Üretken Ön İşlemeli Dönüştürücü (Generative Pre-trained Transformer – GPT3) GPT3 Google tarafından 2017'de tanıtılmıştır. Bir web içeriği üzerinde eğitilmiş 96 katman ile bugüne kadar yapılmış en büyük dil modeli olmuştur. Daha önce çıkarılan Gpt2 ile aynı mimariye sahip olmasına rağmen Gp3 mimarisinde kullanılan parametre sayısının genişletilmesi ve eğitim için kullanılan veri setinin devasa boyuta ulaşmasından dolayı Gpt3 modelinin performansının üst seviyelere çıktığı belirtilmektedir [101].

GPT-3 modeli NLP alanında insanların kullandıkları doğal diller üzerinde ürettikleri metinlere benzer metinler üretmek için oluşturulmuş önceden çok büyük veriler kullanılarak denetimsiz olarak eğitilmiş modellerdir. GPT-3 kelime özetleme, metinden kod oluşturma, şiir yazma, soruları yanıtlama, tercüme etme, fikir üretme, bilgisayar kodu üretme, romanlar ve haber makaleleri gibi dil görevlerini yerine getirmek gibi benzer uygulamalar geliştirmek için kullanılmaktadır. GPT-3 gibi modellerin olumsuz yönleri de mevcuttur genellikle internet tabanlı içerikler üzerinden eğitilmiş olmalarından dolayı Gpt3 uygulamalarının önyargılı davranarak yanlış sonuçlar ürettiği bilinmektedir. Bunlar arasında kişileri yanlış yönlendirmek, cinsiyet ayrımcılığı, terörizm, aşırıcılık, terörist faaliyetlere teşvik gibi zararlı sonuçlar üretmesi modelin korkulan tarafı olmaktadır [102].

3. GEREÇ VE YÖNTEM

3.1 NLP DİL MODELLERİ

NLP alanında metin üretme uygulamaları oluşturmak için farklı derin öğrenme algoritmaları kullanılmaktadır. Metin üretme uygulamalarında bir dil modeli geliştirmek için en çok kullanılan derin öğrenme yaklaşımları şunlardır: [103].

- a. Tekrarlayan Sinir Ağları (RNN, LSTM ve GRU)
- b. Kodlayıcı - Kod Çözücü ve Dikkat Mekanizması
- c. Transformatörler
- d. Üretken Düşman Ağları (GAN)

Yukarıdaki NLP uygulamalarında kullanılan derin öğrenme modellerinden hangi dil modelini kullanırsak daha iyi sonuçlar alınabilir sorusunun net bir cevabı bulunmamaktadır. Çünkü bu modellerin kendi içinde olumlu ve olumsuz tarafları bulunmaktadır. Öyle ki en yeni ve güçlü bir model olan transformer modellerini kullanarak yapılacak bir metin üretme uygulamasında bile çok büyük veri setlerine ve güçlü donanımlara ihtiyaç duyulmaktadır. Bu modellerde yetersiz veri ve güçsüz donanımlarda istenilen başarı alınamamaktadır [104]. RNN modelleri daha çok küçük veri setlerinde kullanılmakta, büyük veri setlerinde ve uzun metin dizelerinde çalışırken hafızaları yetmemekte ve başta ne dediklerini unuttukları için sona doğru çok farklı şeyler söylemektedirler. LSTM ve GRU modelleri ile daha uzun diziler hatırlansa da yeteri başarı sağlanamamaktadır. Kodlayıcı ve kod çözücü modellerinin yapısında LSTM ve GRU modelleri kullanılmaktadır. Bu modellerin bağlantı şekillerinin iyileştirilmesi ve parametre özelliklerinin değiştirilmesi ile uzun metinlerin hatırlanmasındaki başarıyı biraz artmakta fakat yeterli olmamaktadır. Kodlayıcı ve Kod Çözücü modellerinde daha uzun metinlerin hatırlanmasındaki başarıyı artırmak için dikkat mekanizması kullanılmaktadır. GAN'lar NLP'de yakın zamanda kullanılmaya başlanılan eğitim süreçlerinin çok zorlu olduğu bazen de hiç eğitilemeyen modellerdir [105]. Burada dikkat edilmesi gereken veri setinin boyutuna, yapılacak uygulamaya ve kullanılacak bilgisayar donanımının gücüne göre uygun bir dil modeli seçmektir. NLP alanında yukarıda bahsedilen derin öğrenme algoritmaları kullanılarak oluşturulan uygulamalara metnin

özetini çıkarma, metin seslendirme, konuşmayı yazıya dökme, soru yanıtlama, makine çevirisi, konuşma tanıma, resim altyazısı, video altyazısı, sesli asistanlar ve sohbet robotları, otomatik tahmin ve düzeltme, hedef kitleye reklam ve doğal dil üretimi gibi uygulamalar örnek verilebilmektedir.

3.2 KULLANILACAK YÖNTEMLERİN İNCELENMESİ

NLP uygulamalarında dil modeli geliştirirken genel yapay sinir ağları başarılı olamamaktadır. Çünkü bu sinir ağları tüm girdilere eşit derecede dikkat etmekte ve girdi ile çıktı arasındaki ilişkisinin zaman, sıra ve geçmiş bilgilerle olan bağlantılarını öğrenemediği için girdi ve çıktıyı ilişkilendirmekte eksik kalmaktadır. Bundan dolayı NLP uygulamalarında derin öğrenme algoritmalarından olan özyinelemeli YSA tercih edilmektedir. Bu sinir ağları adım adım çalışmakta ve bir adımı bitirmeden diğer adıma geçmemekte ve geçmiş bilgileri hatırlayarak öğrenmektedir. Bu yöntemlerden biride bu tür problemlerin çözümünde çok sık kullanılan kodlayıcı-kod çözücü yaklaşımıdır. Bu yaklaşım sayesinde artık uçtan uca değişken uzunluktaki giriş ve çıkış dizelerini işlemek daha kolay ve daha başarılı hale gelmektedir [103].

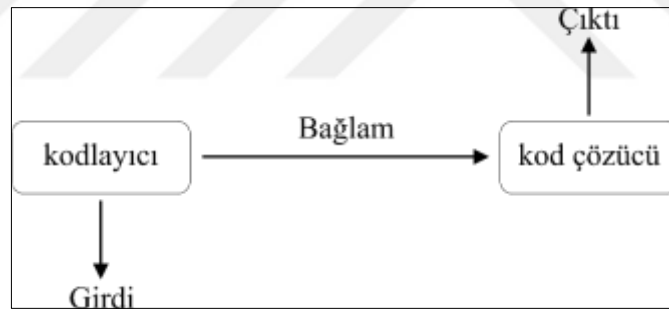
Metin üretimi uygulamalarında kullanılan kodlayıcı kod çözücü modellerinin yapılarında genellikle LSTM ağları tercih edilmektedir. Bundan dolayı LSTM ağlarının çalışma mantığını biraz daha yakından incelemek gerekmektedir. LSTM modellerinin yapısında modelin performansını belirleyen, LSTM'in çıktısının vektör boyutunun belirlendiği unit parametresi, LSTM katmanının çıktılarını belirleyen return parametreleri bulunmaktadır. LSTM verilen bu parametrelere göre 4 farklı sonuç/durum kümesi döndürmektedir [104]. Bunlar:

- a. Default: Son gizli durum (Son zaman adımının gizli durumu)
- b. Return_sequences = True : Tüm gizli durumlar (Tüm zaman adımlarının gizli durumu)
- c. Return_state = True : Son gizli durum + Son gizli durum (tekrar) + Son hücre durumu (Son zaman adımının hücre durumu)
- d. Return_sequences = True + Return_state = True : Tüm gizli durumlar (Tüm zaman adımlarının gizli durumu) + Son gizli durum + Son hücre durumu (Son zaman adımının hücre durumu)

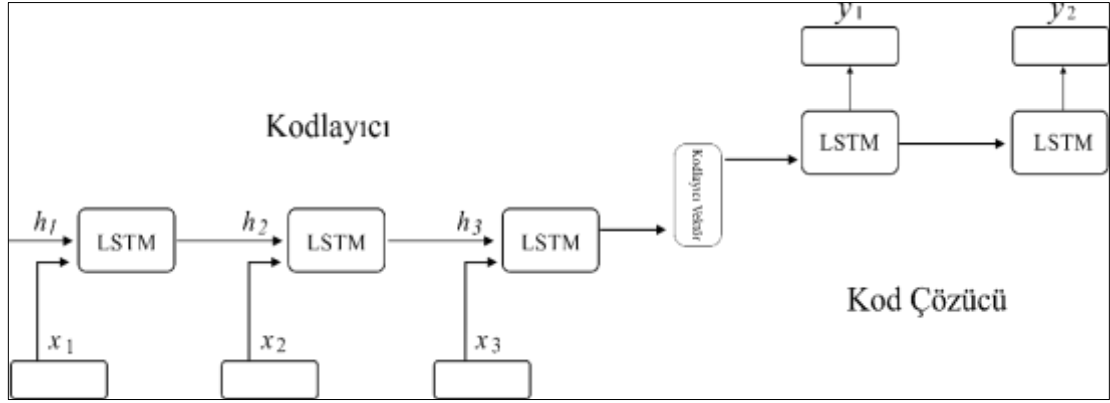
LSTM modellerinin bu parametrelerini deęiřtirerek LSTM modellerinin performansını arttırmak mnkndr. Fakat zyilemeli YSA modelleri doęası gereęi dizilerin boyu arttıķa gemiř bilgileri hatırlamakta zorlanmaktadır. Ayrıca bu yaklařımda uzun dizelerdeki iliřkileri keřfetmek iin daha fazla katmana ve eęitim srecine ihtiya duymaktadır. Bu ve buna benzer sebeplerden dolayı yeni bir model mimarisine ihtiya duyulmaktadır. Bu mimarinin mmkn olduęu kadar esnek ve bařarımının daha yksel olmasını hedeflenmektedir. ok daha yksek doęrulukla ok daha uzun dizelerdeki iliřkileri hatırlamalarını ve bařarılı sonular retmeleri beklenmektedir. Bundan dolayı metin retimi uygulamalarında daha iyi sonular verdięi iin kodlayıcı kod zc ve dikkat mimarisi yaygın olarak kullanılmaktadır.

3.2.1 Kodlayıcı Kod zc Modeller

2014 yılından itibaren NLP uygulamalarında kodlayıcı-kod zc modeller standart bir yaklařım haline gelmiřtir. Bu model sayesinde artık utan uca deęiřken uzunluktaki giriř ve ıkıř dizelerini iřlemek daha kolay ve daha bařarılı hale gelmektedir [106].



Yukarıda Kodlayıcı Kod zc modelinin bileřenleri gsterilmektedir. İřlevsel olarak, kodlayıcıve kod zc modellerde birlikte alışan kodlayıcı, kod zc ve baęlam (ierik) vektr olmak zere  ana bileřen bulunmaktadır.



Şekil 3.1: Kodlayıcı Kod Çözücü Model.

Sekil 3.1’de kodlayıcı kod çözücü modelinin genel yapısı görülmektedir. Kodlayıcı kendisine girdi olarak verilen veriyi kendi içinde adım adım işlemekte, bütün adımlar bittikten sonra bir içerik vektörü oluşturmaktadır. Kod çözücü ise bu içerik vektörünü kullanarak adım adım çıktı değerini üretmektedir. Dolayısıyla kodlayıcı bütün girdi değerlerini bitirdikten sonra bir tane içerik vektörü üretip durmakta, kod çözücü ise içerik vektörünü kullanarak tek tek bir döngü içerisinde çıktısını üretmeye çalışmaktadır. Burada kodlayıcı ile kod çözücü arasındaki bilgi akışı sadece içerik vektörü sayesinde olmaktadır. Bu vektöre içerik vektörü, giriş dizisinin bağlam vektörü, giriş dizisinin yeni temsili, giriş dizisinin özeti şeklinde ifade edilmektedir [107].



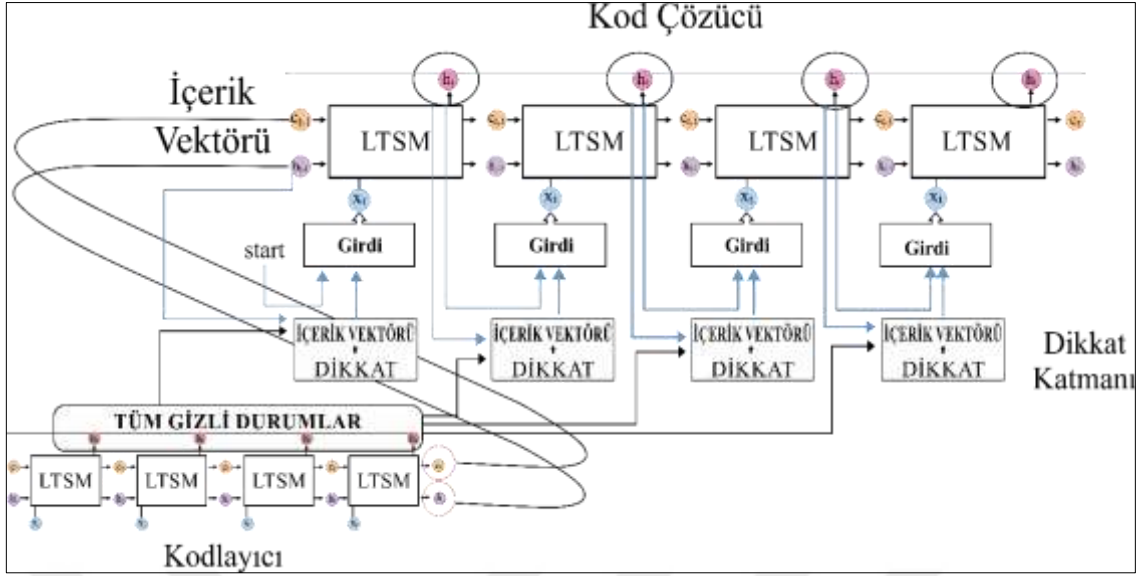
Şekil 3.2: Kodlayıcı Kod Çözücü Örneği.

Şekil 3.2’de modelin çalışma şekli bir örnek ile gösterilmektedir. Modele bir girdi olarak bir cümle verildiğinde LSTM kodlayıcı burada her bir karakteri alarak teker teker işlemekte ve durmaktadır. Ve bize kodlayıcının son gizli ve hücre durumlarını içerik vektörü olarak döndürmektedir. Kod çözücü modeli ise bu içerik vektörünü kullanarak çalışmaya başlamaktadır. Kod çözücü modelini başlatmak için bir start girdisi verilmektedir. Daha sonra ilk içerik vektörünü ve ilk girişi kullanarak kod çözücü ilk çıkışı üretecek/tahmin etmektedir. Bir sonraki zaman adımı için, kod çözücü içerik vektörü olarak kendi son gizli

ve hücre durumlarını ve girdi olarak önceki zaman adımında üretilen/tahmin edilen çıktıyı kullanacaktır. Bir kere başladı mı kendi ürettiği değerlerle sonraki adımları döngüsel olarak üretmeye başlamaktadır. Stop gibi özel bir kelimeyi gördüğünde ise işlemi sonlandırmaktadır. Burada önemli olan içerik vektörünün boyutuna doğru karar verebilmektir. Çünkü girdi verisinin tüm bilgilerini özet halinde üzerinde bulunduran içerik vektörüdür. Bu özet ne kadar detaylı ve büyük olursa o kadar anlamlı bir sonuç için bize yardımcı olmaktadır. İçerik vektörünün boyutunu arttırmak için kodlayıcı içerisindeki LSTM hücrelerinin birim boyutlarını arttırmak ve LSTM katmanlarının sayısını arttırmak gerekmektedir [11].

3.2.2 Dikkat Mekanizması

Kodlayıcı-Kod çözücü yaklaşımında sabit uzunluklu bir içerik vektörü oluşturulup üst üste çıktılar üretilmeye çalışılmaktadır. Kodlayıcı ve kod çözücü modelinde çıktı değerimizin doğrudan girdi değeriyle bir ilişkisi bulunmamaktadır. Bundan dolayı kodlayıcı kod çözücü modeller daha uzun girdilerde başarısız olmaktadır. Kodlayıcı kod çözücü modellerde çıktı değeri en son LSTM hücresi ve bir önceki LSTM hücrelerinin gizli durum ve hücre durumu değerleri üzerinden tahmin edilmektedir. Bu şekilde geriye doğru giderek girdi değeri hakkındaki bilgileri sadece içerik vektörü yardımıyla alınabilmektedir. Bu durumda kodlayıcı ve kod çözücü modellerinde dizi uzadığında girdi değeri ile en son çıktı değeri arasındaki ilişkiler kaybolmaktadır. Bunun üzerine kodlayıcı kod çözücü modellerde daha uzun dizilerde çalışabilmek için dikkat mekanizması kullanılmaya başlanmıştır [108]. Dikkat mekanizması bunu her adımda kod çözücü bir kelimeyi tahmin ettiğinde kodlayıcıdaki bütün gizli durumları dikkate alarak yapmaktadır. Yani model içerisindeki adım sayısı artsa bile her adımda modelin ürettiği çıktı değerlerinde dikkat mekanizması sayesinde girdi değerlerine sürekli göz atıldığı için başarı artmaktadır. Bu şekilde dikkat mekanizması sayesinde uzun dizilerde girdi ve çıktı değerleri arasındaki ilişkilerin daha iyi hatırlanması sağlanmaktadır. Kısacası dikkat mekanizması sayesinde girdi ile çıktı arasındaki ilişki her adımda hatırlanmaktadır [11].



Şekil 3.3: Kodlayıcı Kod Çözücü ve Dikkat Mimarisi.

Şekil 3.3'te kodlayıcı ve kod çözücü modellerde dikkat mekanizmasının nasıl kullanıldığı gösterilmektedir. Dikkat mimarisi sayesinde kodlayıcı artık sadece son gizli ve hücre durumlarını değil, aynı zamanda tüm zaman adımlarında üretilen kodlayıcının tüm gizli durumlarını da kod çözücüye aktarmaktadır. Kod çözücü ise çıktı oluşturmak için ilk giriş ve kodlayıcının tüm gizli durumlarını kullanarak dikkat vektörünü hesaplamaktadır. Daha sonra dikkati kodlayıcının tüm gizli durumlarına uygulamakta ve içerik vektörünü hesaplamaktadır. Kod çözücü ise girdisini oluşturmak için içerik vektörü ve ilk girişi birleştirmektedir. Sonra model bir döngüde halinde çalışmaktadır [109].

3.2.2.1 İçerik vektörünün hesaplanması

Kodlayıcının tüm gizli durumlarını ve kod çözücünün çıktısını ilişkilendirmek için bir puan (skor) değerinin hesaplanması gerekmektedir. Dikkat mekanizmasının ana fikri kodlayıcının tüm gizli durumlarını dikkate almaktır. Yani, her bir kod çözücünün çıktısını üretmek için kodlayıcının tüm çıktıları işleme katmak gerekmektedir.

Dikkat mimarisinde skor hesaplamasını yapabilmek için Bandanau tarafından geliştirilmiş hesaplama yöntemi kullanılmaktadır [11].

Denklem 3.1'de Bandanau tarafından geliştirilmiş olan skor hesaplama yönteminin formülü gösterilmektedir.

$$score = (v_a \tan h(W_1 h_t + W_2 h_s)) \quad [Bandanau's \text{ Yöntemi}] \quad (3.1)$$

hs: Kodlayıcının tüm gizli durumları

ht: Kod çözücünün önceki gizli durumları (önceki zaman adımı çıkışı)

ct: İçerik vektörü

w: Ağırlık matrisi

tanh: Aktivasyon fonksiyonu

v: Bir tek gizli katmanlı ağ modeli

Denklem 3.1'nin bir sonucu olarak, modelin eğitim sırasında uygun bir puanın nasıl hesaplanacağını öğrenmesini beklenmektedir. Bu puanlar, kodlayıcının tüm gizli durumları ile kod çözücünün çıktısı arasındaki ilişki düzeyini ortaya koyan bir puan olarak düşünülmektedir. Yani kod çözücünde üretilen değer ile kodlayıcıdan alınan değer çok ilişkili ise bu skor değeri yüksek olmakta ilişkisiz ise bu skor düşük olmaktadır. Algoritma bunu kendi kendine öğrenerek girdi ile çıktı arasında ilişkili olup olmadığına karar vermektedir. Burada her bir girdi vektörlerinin ağırlık değerlerinin çarpılıp toplanması, aktivasyon fonksiyonundan geçirilmesi ve hataların geri yayılım yapılarak hesaplanması sonucunda modelin en iyi ilişkileri yakalaması sağlanarak ilişki skorları hesaplanmaktadır [109].

$$\alpha_{ts} = \frac{\exp(score(h_t, h_s))}{\sum_{s'=1}^S \exp(score(h_t, h_{s'}))} \quad [Dikkat \text{ ağırlıkları}] \quad (3.2)$$

Denklem 3.2'te yapılan bir softmax işlemidir. Hesaplanan puanları normalleştirerek dikkat ağırlıkları hesaplanmaktadır. Bu şekilde girdi değerleri ile çıktı değerlerinin birbiriyle ne kadar ilişkili oldukları hesaplanmaya çalışılmaktadır [104].

$$C_t = \sum_s \alpha_{ts} h_s \quad [İçerik \text{ vektörü}] \quad (3.3)$$

Denklem 3.3'te dikkat ağırlıklarını kod çözücünün gizli durumlarına uygulayarak bağlam vektörünü hesaplanmaktadır. Burada her bir gizli durum değerinin yukarıdaki softmax işlemiyle üretilen değer ile olan olasılıksal değerini bu değer ile çarpıp topladığımızda içerik vektörünü elde edilmiş olunmaktadır. İçerik vektörü kodlayıcının herhangi bir gizli durumu ile bir önceki adımda ürettiğimiz gizli durum ilişkisi yüksek ise bu çarpımdan çıkan sonuç yüksek olmakta ve içerik vektörüne etkisi büyük olmaktadır [104].

3.3 MODELİN OLUŞTURULMASI

3.3.1 Metin Üretimi

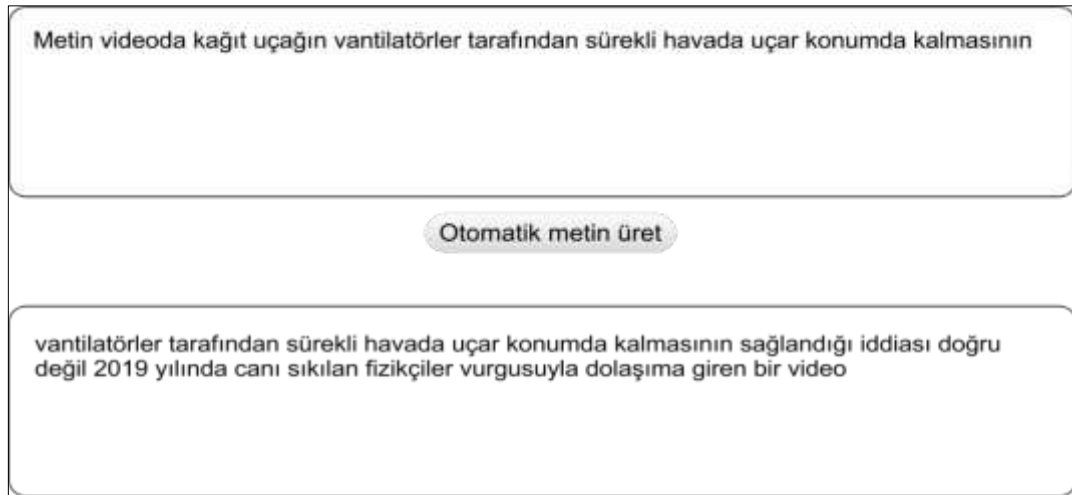
Temelde metin üretimi otomatik metin üretimi ve kontrol edilebilir metin üretimi olmak üzere iki tür yaklaşım kullanılarak üretilmektedir.

3.3.1.1 Kontrol edilebilir metin üretimi

Bir metin üretirken metnin özel bir kural ya da beklentiye göre üretilmesidir. Örneğin oluşturulacak metnin zamanı, içindeki duygusu, dilin yapısı, dil bilgisi kuralları ile ilgili ya da içerisinde bazı anahtar kelimelerin geçip geçmediği ile ilgili yönlendirmelerin bulunduğu kelime üretme yaklaşımıdır [110].

3.3.1.2 Otomatik metin üretimi

Dil modeline bir metin parçası verdiğimizde herhangi bir özel kural veya beklenti tarafından sınırlandırılmadan veya yönlendirilmeden her hangi bir metin oluşturmakta özgürdür. Amacımız bu metni tutarlı bir şekilde tamamlayacak dil bilgisi kurallarına uygun, anlaşılabilir bir metin üretmeyi gerçekleştirmektir.



Metin videoda kağıt uçağın vantilatörler tarafından sürekli havada uçar konumda kalmasının

Otomatik metin üret

vantilatörler tarafından sürekli havada uçar konumda kalmasının sağlandığı iddiası doğru değil 2019 yılında canı sıkılan fizikçiler vurgusuyla dolaşıma giren bir video

Şekil 3.4: Otomatik Metin Üretimi Örneği.

Şekil 3.4’de otomatik metin üretimi yapan bir örnek arayüz görülmektedir. Burada modele başlangıç olarak bir metin parçası verildiğinde eğitilmiş olan dil modeli verilen metin parçasına göre uygun bir metin üretmektedir. Buradaki beklentinin üretilen metnin kendi içerisinde tutarlı, anlamlı ve dil bilgisi kurallarına uygun olmasıdır.

Metin tokenizasyonu

Metin içerisinde kullanılan karakterlere, kelimelere, sayılara, özel işaretlere ve sembollere token denilmektedir. Metin tokenizasyonu dil modelinin eğitimi için kullanılan metin verisinin istenen tokenlere ayrılmasını ifade etmektedir. Bir dil modeli geliştirilirken metin verisi token (harf, kelime, sembol, sayı, karakter grubu ya da kelime grubu) denilen parçalara ayrılarak dil modeline verilmekte ve modelin eğitilmesi gerçekleştirilmektedir. NLP uygulamalarında tokenlerin nasıl kullanılacağı değişkenlik gösterebilmektedir. NLP uygulamalarında kullanılan tokenizasyon için çoğunlukla 3 seviye kullanılmaktadır. Bunlar:

- a. Kelime
- b. Harf
- c. N-gram

Doğru bir tokenizasyon işlemi yapabilmek için ilk önce metin verisinin istenmeyen tokenlerden temizlenmesi gerekmektedir. Dil modelinin eğitimi için kullanılacak token boyutuna (kelime, karakter veya n-gram) karar verilmelidir. Daha sonra metin içerisinde geçen çok farklı kelime ve karakterlerden oluşan bir sözlük oluşturmak gerekmekte ve bu sözlüğün boyutunun ne kadar olacağına karar verilmektedir. Bu durum donanımın gücüne, modelin gücüne ve veri setinin büyüklüğüne bağlı bir durumdur. Sözlükte maksimum sayıda bulunacak token sayısının ve sözlükteki her bir tokenin nasıl temsil edileceği belirlenmeli ve tamsayılardan oluşan bir dizin listesi oluşturmak gerekmektedir. Tokenizasyon işlemi sonucunda dil modelinde kullanacak (girdi ve çıktı olarak eğitilecek) token dizeleri elde edilmiş olunmaktadır. Yukarıdaki adımların tümüne metin tokenizasyonu adı verilmektedir [104]. Bu adımları yapılandırılmış bir şekilde işlemek için python programlama dili ve derin öğrenme kütüphaneleri kullanılmaktadır

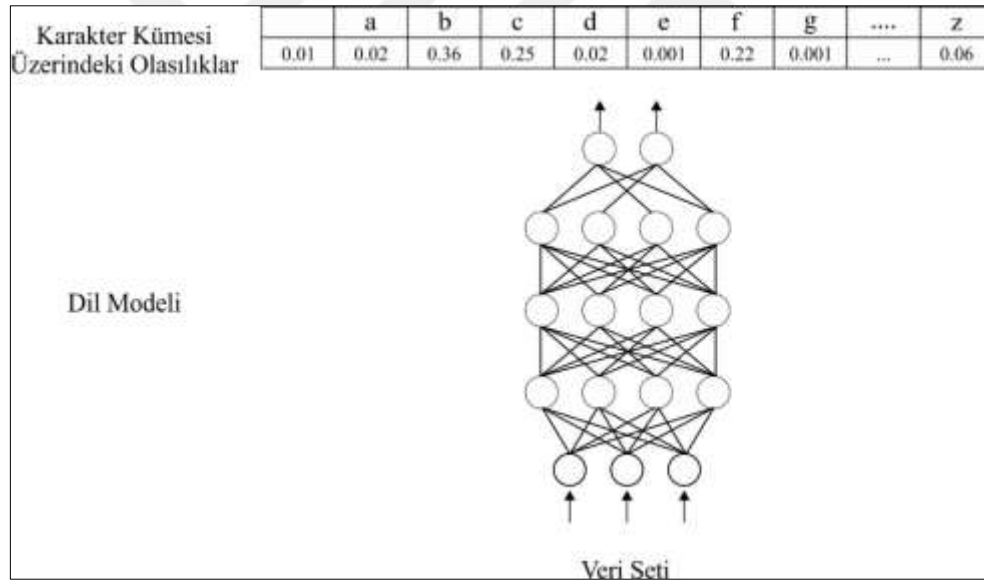
3.3.1.3 Karakter tabanlı ve kelime tabanlı metin oluşturma

Metin üretimi yapılırken token yapısını nasıl belirleyeceğimiz çok önemlidir. Yani tokenlerin büyüklüğü kelime, karakter ya da n-gram mı olacak buna karar vermek gerekmektedir. Yalnız bunun net bir cevabı bulunmamaktadır. Çünkü her birinin kendine göre avantaj ve dezavantajı bulunmaktadır. Elinizdeki probleme, hedeflediğiniz sonuca,

elinizdeki verinin büyüklüğüne ve tasarladığınız modelin yeteneklerine göre seçilecek yöntem değişebilmektedir [103].

3.3.1.4 Dil modeli

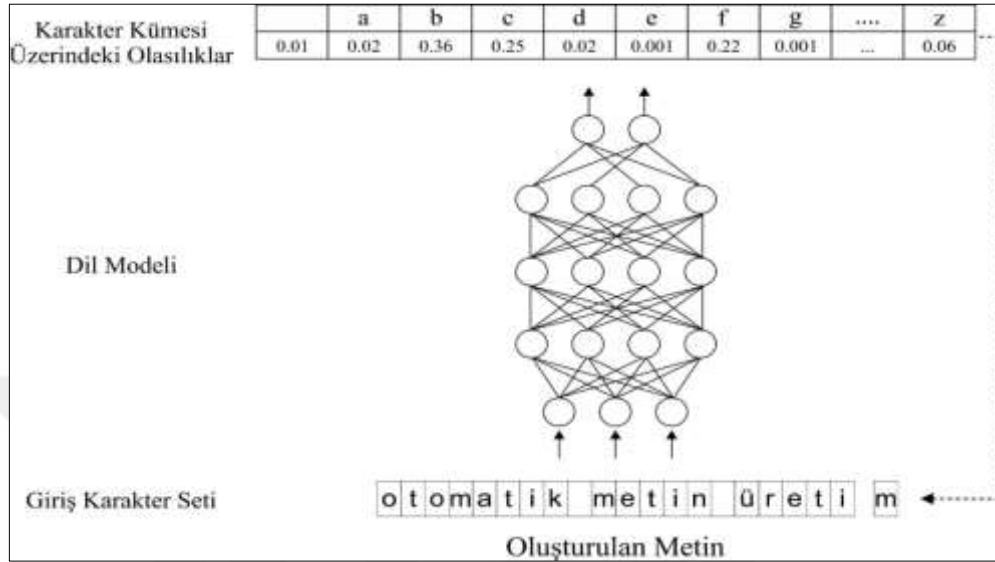
NLP uygulamalarının merkezinde dil modeli bulunmaktadır. NLP uygulamalarından olan metin üretiminin başarılı bir şekilde eğitilmesinin altında dil modelleri yatmaktadır. Metin üretimi uygulaması yapabilmek için ilk olarak bir dil modeli tasarlayıp eğitmemiz gerekmektedir. Dil modeli derin öğrenme algoritmaları kullanılarak oluşturulan, verilen veri setinin token haline getirilmiş birimlerinin istatistiksel hesaplamalarını yaparak metin üretimi gerçekleştiren bir eğitilmiş modeldir. Bu dil modellerini eğitebilmek için ilk olarak geniş veri setlerine ihtiyaç duyulmaktadır [104]. Metin tabanlı bir dil modeli eğitilirken veri setinde bulunan cümleler, kelimeler, harfler ya da n-gram denilen kelime ya da harf grupları kullanılmaktadır [104].



Şekil 3.5: Dil Modeli Örneği.

Şekil 3.5’de bir dil modelinin çalışma mimarisi karakter düzeyinde gösterilmektedir. Bir dil modelini eğitirken veri setinde bulunan bütün karakterler, dil modelinin mimarisini oluşturan derin öğrenme algoritmalarına verilerek tüm karakterlerin anlamsal, sırasal ve birliktelik ilişkileri olasılıksal olarak hesaplanmakta ve hangi kelimelerin birbiriyle ilişkili olduğu ya da olmadığı ile ilgili hesaplama puanları dil modeli tarafında eğitim bittikten sonra öğrenilmektedir. Dil modeli eğitim bittikten sonra ona verilen cümleden sonra hangi karakterin geleceğini dil modelinin sözlüğünde bulunan tüm karakterlerin şartlı olasılık

değerlerine bakarak tahmin etmektedir. Bu şekilde tekrarlı olarak dil modelinin sözlüğünde bulunan her bir karakterin bize verilen cümle dizisine göre bir sonraki karakteri hesaplanarak tahmin edilmiş olunmaktadır.



Şekil 3.6: Karakter Tabanlı Dil Modeli Çalışma Örneği.

Şekil 3.6’de dil modelinin eğitimi bittikten sonra kendisini verilen bir girdiden kendi sözlüğünde bulunan karakterlerden hesaplanan şartlı olasılık değerlerine göre örnek seçme yöntemlerinden biri kullanılarak en uygun karakteri tahmin etmektedir. Daha sonra bu seçtiği karakteri girdi dizisine eklemekte ve bu yeni girdi dizisine göre tekrar örnek alma yöntemlerini kullanarak yeni bir karakteri seçmekte ve girdi dizisine eklemektedir.

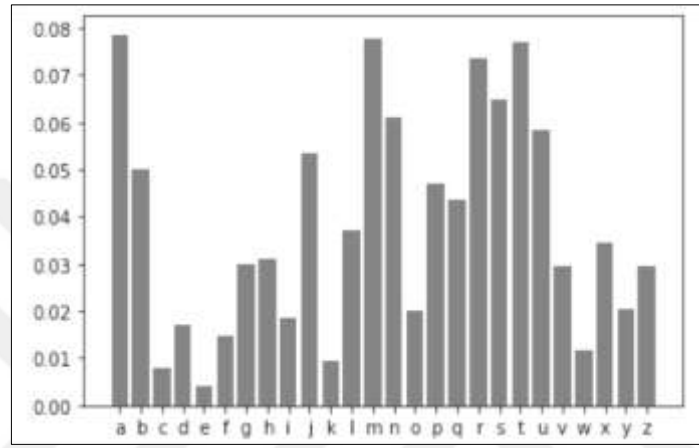
3.3.1.5 Örnek alma (sampling) yöntemleri

Metin üretme aşamalarından biride örnek alma işlemidir. Dil modeli kendisine verilen girdiye göre sözlükteki her bir karakter için oluşturduğu şartlı olasılık değerlerine bakarak ve örnek alma yöntemlerinden birini kullanarak sözlükten bir karakteri seçmektedir. Seçilen bu karakter girdi olarak verilen metne eklenmektedir. Metin üretme uygulamalarında kullanılan farklı örnek alma yöntemleri bulunmaktadır. Metin oluşturmada örnek almak için kullanılan temel yöntemler şunlardır [111].

- i. Açgözlü Arama (Maksimizasyon)
- ii. Sıcaklık Örnekleme
- iii. Top-K Örnekleme

Aç Gözlü Arama

Bu yöntem dil modelinin oluşturduğu sözlükteki karakterlerin şartlı olasılık değerlerini, dil modeline verilen girdi metnindeki karakterleri dikkate alarak en yüksek olasılık dağılımına sahip karakteri seçerek girdi metninin sonuna eklemektedir. Bu yöntemin sakıncalı yönü sürekli olarak sözlükteki en yüksek olasılık değerine sahip karakteri seçtiği için girdi metninin sonuna aynı karakteri getirmektedir [112].

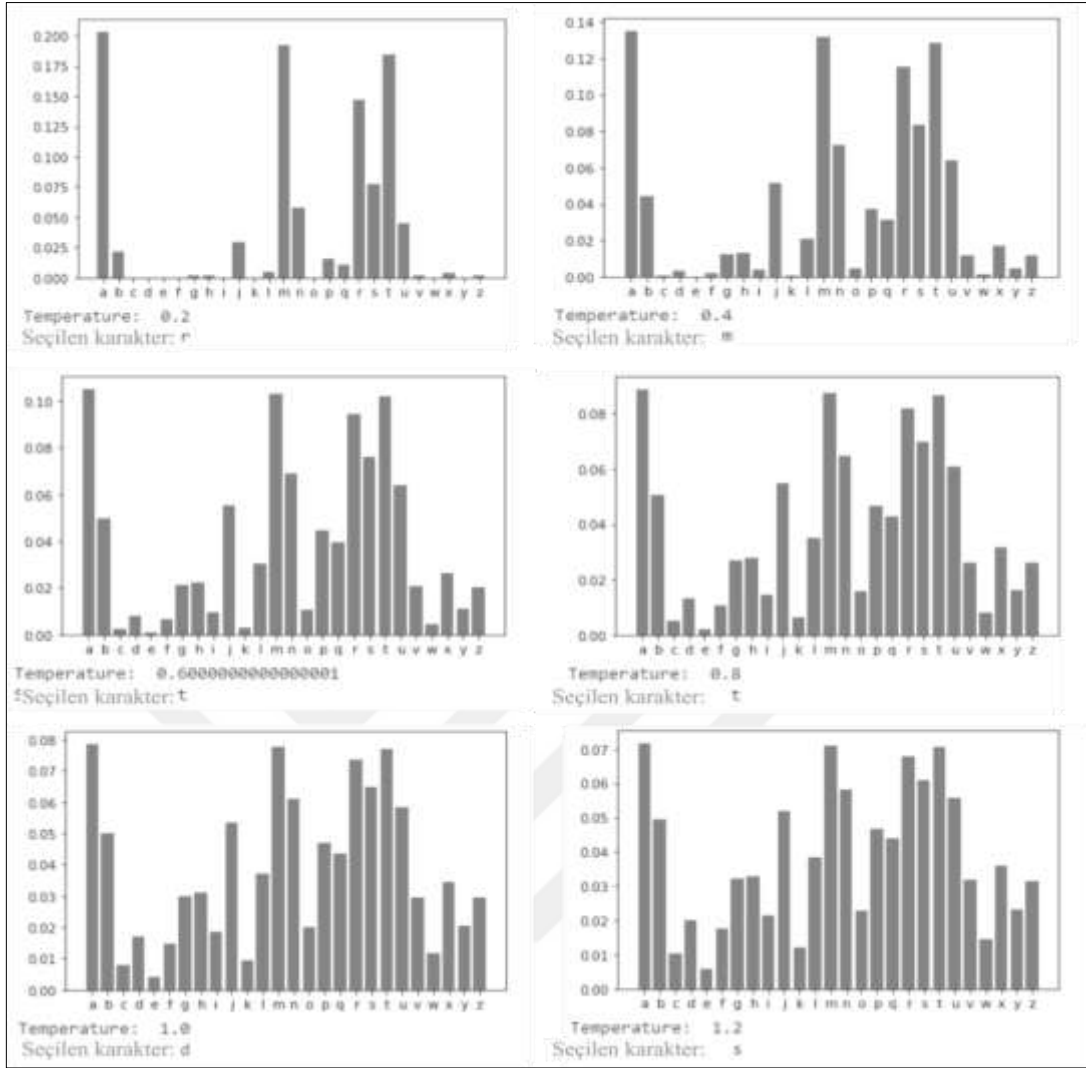


Şekil 3.7: Aç Gözlü Arama Örneği.

Şekil 3.7’te sözlük içerisindeki en yüksek şartlı olasılık değerine sahip karakter seçilerek verilen girdi kelimesinin sonuna eklemektedir.

Sıcaklık Örnekleme

Sıcaklık örnekleme istatistiksel termodinamikten esinlenerek kullanılan bir yöntemdir. Bu yöntemde göre yüksek sıcaklıklar düşük enerji seviyelerini göstermektedir. Dil modelinin ürettiği sözlükteki kelimelerin değerlerini verilen sıcaklık değerlerine göre bir normalizasyon işleminden geçirildiğinde ve düşük sevide olan kelimeler sözlükten çıkarıldığında geriye yüksek seviyeye sahip karakterler kalmaktadır. Bu yüksek seviyeli karakterlerin değerlerini tekrar bir softmax fonksiyonundan geçirerek sıfır ile bir olasılık aralığına dağıtıldığında ve yeni değerlere göre örnek alma işlemi gerçekleştirildiğinde daha iyi sonuç alınabildiği görülmektedir. Sıcaklık örnekleme yönteminin dayandığı mantık budur [113].



Şekil 3.8: Sıcaklık Örneklemesi Örneği.

Şekil 3.8’de dil modelinin sözlüğünde bulunan karakterler için verilen sıcaklık değerine göre bir tahminde bulunulmakta ve tahminler arasındaki farklılıklar görülmektedir.

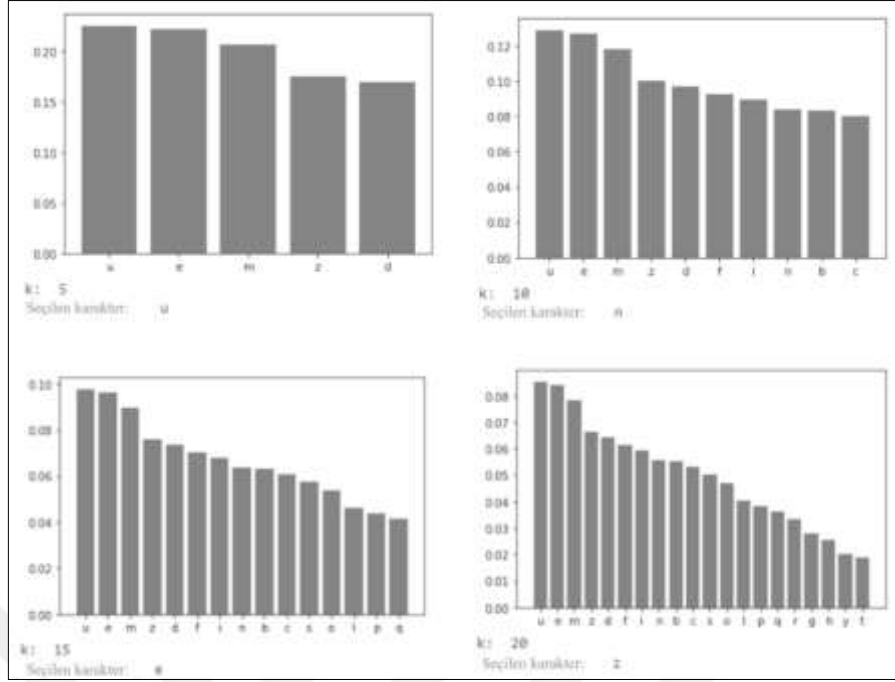
- Sıcaklık 0,2 değeri için sözlükten seçilen karakterlerin dağılımı görülmektedir. Burada birçok değer sıfıra yaklaştırılmış ve yüksek değerler arasından ‘r’ karakteri seçilmiştir.
- Sıcaklık 0,4 seçildiğinde sözlükten daha çok karakter seçilmekte ve daha az değer sıfıra yaklaştırılmakta ve yüksek adaylar arasında ‘m’ karakteri seçilmektedir.
- Sıcaklık 0,6 seçildiğinde ise çok daha fazla aday seçilmekte ve daha az sayıda karakter sıfıra yaklaştırılmaktadır. Yüksek adaylar arasında ‘t’ karakteri seçilmektedir.
- Sıcaklık 0,8 seçildiğinde ise çok daha fazla aday seçilmekte ve daha az sayıda karakter sıfıra yaklaştırılmaktadır. Yüksek adaylar arasında ‘t’ karakteri seçilmektedir.

- e. Sıcaklık 1.0 seçildiğinde grafiğimiz orijinal grafikle aynı duruma gelmektedir. Çünkü karakterlerin şartlı olasılık değerlerini sıcaklık değerlerine böldüğümüz için orijinal değerler değişmemektedir. Yüksek adaylar arasında 'd' karakteri seçilmektedir.
- f. Sıcaklık değerini 1.2 yaptığımızda ise yine sözlükteki tüm karakterler seçilebilir durumda olmakta ve yüksek değerler arasındaki fark azalmaya başlamaktadır. Ve rastgelelik iyice artmaktadır. Yüksek adaylar arasında 's' karakteri seçilmektedir.

Sıcaklık örnekleme yönteminde sıcaklık değerleri düşük tutulduğunda az aday içinden, sıcaklık değerleri yüksek tutulduğunda ise çok daha fazla aday içinden seçim yapmak durumunda kalınmaktadır. Yani sıcaklık 0 seçildiğinde aç gözlü arama yöntemine benzer şekilde seçim yapılmakta, sıcaklık değeri çok yüksek seçildiğinde ise rastgelelik arttırılmış olmaktadır. İyi sonuçlar alabilmek için sıcaklık değerini arada bir değer olarak belirlemek gerekmektedir. Doğru değeri bulabilmek için birçok sıcaklık değeri ile uygulama üzerinde deneme yapmak gerekmektedir [111].

Top K Örnek Alma

Diğer örnek alma yöntemi top k yöntemidir. Bu yöntem en yüksek k adet dağılım içerisinde rastgele bir karakterin seçilmesini sağlamaktadır. Top k örneklemesini uygulamak için sözlükteki en yüksek şartlı olasılık değerine sahip k tane karakter seçilerek diğer karakterler için olasılıklar sıfırlanmaktadır. Seçilen karakteri softmax fonksiyonundan geçirerek tekrar bu karakterler için sıfır ile bir arasında olasılık değerleri hesaplanmaktadır. Daha sonra karakterler içerisinde rastgele bir karakter seçilerek verilen girdi metninin sonraki karakteri olarak eklenmekte ve metin oluşturmaya bu şekilde devam edilmektedir. Bu yöntemin hangi k değeri için daha iyi sonuç verdiği deneme yanılma yöntemi yapılarak belirlenmektedir [113].



Şekil 3.9: Top K Örneği.

Şekil 3.9’da top k örnek alma yönteminin farklı k değerleri için dil modelinin ürettiği sözlük içinden karakter seçme yöntemi gösterilmektedir.

- İlk örnekte 5 değeri için sözlükte şartlı olasılık değeri en yüksek olan 5 karakter getirilmekte ve karakterlerin değerlerini tekrar kendi aralarında 0 ile 1 arasında olacak şekilde normalize ederek rastgele bir karakter seçilmektedir. Seçilen karakter ‘u’ harfi olmaktadır.
- İkinci örnekte ise sözlükte şartlı olasılık değeri en yüksek olan 10 karakter seçilmekte ve dağılım tekrar kendi içinde yapılmakta ve içinden rastgele bir karakter seçilmektedir. Seçilen karakter ‘n’ harfi olmaktadır.
- Üçüncü örnekte ise sözlükte şartlı olasılık değeri en yüksek olan 15 karakter seçilmekte ve dağılım tekrar kendi içinde yapılmakta ve içinden rastgele bir karakter seçilmektedir. Seçilen karakter ‘e’ harfi olmaktadır.
- Dördüncü örnekte ise sözlükte şartlı olasılık değeri en yüksek olan 20 karakter seçilmekte ve dağılım tekrar kendi içinde yapılmakta ve içinden rastgele bir karakter seçilmektedir. Seçilen karakter ‘z’ harfi olmaktadır.

Top K yönteminde dikkat edilmesi gereken konu k değerinin sözlükteki toplam karakter sayısına göre dikkatli seçmek gerekmektedir. K değerini çok yüksek seçtiğimizde sözlükteki

çok sayıda karakter arasından bir tercih yapılacağı için rastgeleliği arttırmış olmaktadır. Bunun tersi olarak k değerini çok küçük seçtiğimizde de hep aynı karakterleri seçme durumu ortaya çıkmaktadır. Bundan dolayı k değerini ortalama bir değer olarak seçmek daha doğru metinler üretmeyi sağlamaktadır [111].

3.3.2 Kodlama Yapısı ve Mimarisi

Veri Seti

Bu çalışmada internette toplanan Türkçe haber veri seti üzerinde çalışma yapılmıştır. Toplanan haberler birleştirilerek tek bir “txt” uzantılı metin dosyası haline getirilmiştir. DDİ alanında yapılan çalışmalarda metin verilerinin istenen sevide işlenebilir düzeyde olmamasından dolayı veri seti veri ön işleme adımlarından geçirilerek çalışmaya uygun bir düzeye getirilmektedir. Bu şekilde düzenlenmiş ve temizlenmiş bir veri seti üzerinde yapılan çalışmalar daha başarılı sonuçlar vermektedir. Veri temizleme işlemleri Python programlama dili ve NLTK kütüphanesi kullanılarak gerçekleştirilmiştir. Kullanılan veri setinden tüm noktalama işaretleri, html etiketleri url ve @ işareti kaldırılmış ve tüm harfler küçük harfe çevrilmiştir.

Karakter tabanlı kelime üretimi uygulaması python programlama dili, python üzerinde kullanılan derin öğrenme kütüphaneleri (TensorFlow, Keras) ve IDE olarak GoogleColap arayüzü kullanılarak oluşturulmaktadır. Kütüphanelerden TensorFlow kütüphanesinin veri hattını (Data pipeline) ve Keras’ın veri ön işleme katmanı olan TextVectorization katmanı kullanılarak uçtan uca çalışan bir model oluşturulmaktadır. TextVectorization katmanı metin ön işleme adımlarını yapabilmek için kullanılırken, Tensorflow kütüphanesinin veri hattı ise otomatikleştirilmiş olarak veriyi işlememizi sağlayan bir yaklaşım olarak kullanılmaktadır. Bir veri hattında verinin çıkarılması, dönüştürülmesi ve doğruluğunun kontrol edilmesi gibi uçtan uca bir bağlantı sağlanarak işlemlerin otomatik olarak gerçekleştirmesi sağlanmaktadır. Bu yapı aynı zamanda verilerin paralel olarak çalışmasını sağlayarak oluşabilecek donanımsal darboğazı ortadan kaldırarak hız kazandırmakta ve oluşabilecek hataları ortadan kaldırarak modelin performansını artırmaktadır. Bu sayede TensorFlow tf.data API’sinin bize sunduğu tüm komutlar sayesinde büyük miktarda veriyi kısa sürede işleyebilmekte farklı veri biçimlerini okuyabilmekte ve karmaşık dönüşümleri hatasız yapabilmektedir.

Bir veri hattında şu işlemler yapılmaktadır.

- İlk olarak veri hattına veri seti eklenmektedir.
- Veri seti karakterlere dönüştürülmektedir.
- Metin içindeki noktalama işaretleri, html etiketleri, beyaz boşluklar vb. gibi istenmeyen karakterler kaldırılarak veri seti temizlenmektedir.
- Karakter dizileri giriş (x) ve çıkış (y) çiftleri haline dönüştürülmektedir.
- Girdi (x) ve çıktı (y) birleştirilerek tek bir veri seti haline getirilmektedir.
- Karakterler bilgisayarın anlayacağı sayısal değerlere (embedding) dönüştürülmektedir.
- Modelin eğitimi için veri, dil modeline verilmektedir.

Veri Seti

Metin Videoda Kağıt Uçağın Vantilatörler Tarafından Sürekli Havada Uçar Konumda Kalmasının Sağlandığı İddiası Doğru Değil 2019 yılında canı sıkılan fizikçiler vurgusuyla dolaşıma giren bir video, 2022 yılında tekrar gerçek sanılarak sosyal medyada dolaşıma sokulmuş. Bilgisayar ortamında görsel efekt kullanılarak oluşturulan bahse konu video Videonun kaynağı, Instagram'da vernbestintheworld uzantılı profilinde, ürettiği CGI içeren videoları paylaşan kullanıcı Profili incelendiğinde bahsi geçen kullanıcının diğer paylaşımlarının da CGI içerdiği görülüyor. Kaynak: Malumatfuruş (@malumatfurusorg) September 9, 2019 Söz konusu videonun kağıt uçağın dönelişinin vantilatörlerin oluşturduğu hava akımıyla sağlandığı anı vantsitörün canan coşval

Girdi x & Çıktı y (Karakter)

Girdi x (Girdi Karakterleri)	--->	Çıktı y (Çıktı karakteri)
metin videoda kağıt	---	u
in videoda Kağıt Uça	---	ğ
videoda Kağıt Uçağın	---	
eoda kağıt uçağın va	---	n
a kağıt Uçağın vant	---	l

Sayısal Vektörleştirme (Karakter)

[18 10 3 16 6 14 3 9 15 18 18 7 9 5 8 19 2 4 11 6]	--->	4
[16 6 14 3 9 15 18 18 7 9 5 8 19 2 4 11 6 4 2 4]	--->	10
[3 9 15 18 18 7 9 5 8 19 2 4 11 6 4 2 4 10 15 4]	--->	11
[18 18 7 9 5 8 19 2 4 11 6 4 2 4 10 15 4 11 2 5]	--->	9
[9 5 8 19 2 4 11 6 4 2 4 10 15 4 11 2 5 9 2 6]	--->	2

Şekil 3.10: Veri Hattı.

Şekil 3.10'da veri hattı yapısının işleyişi görülmektedir. İlk olarak veri seti okunmakta daha sonra metin hatasız karakterlere ayrılması için tek satır haline getirilmektedir. Metin önışleme adımları yapılarak istenmeyen karakterler kaldırılmaktadır. Veri seti girdi ve çıktı verisi olarak iki parçaya ayrılmaktadır. İlk yirmi karakteri girdi (x) olarak almakta, yirmi

birinci karakteri ise çıktı (y) değeri olarak almaktadır. Daha sonra her bir karakter bir tamsayı ile ifade edilerek modelin işleyeceği bir formata dönüştürülmektedir. Artık kullanılacak modellerin işleyebileceği bir formata getirilen veri seti LSTM, GRU ve kodlayıcı kod çözücü ve dikkat mimarisi kullanılarak dil modelleri eğitilmektedir. Eğitilen bu dil modellerinin kod yapısı ve parametreleri aşağıda verilmektedir.

```
inputs = tf.keras.Input(shape=(sequence_length), dtype="int64")
x = layers.Embedding(max_features, embedding_dim)(inputs)
x = layers.Dropout(0.5)(x)
x = layers.LSTM(512, return_sequences=True)(x)
x = layers.Flatten()(x)
predictions= layers.Dense(max_features, activation='softmax')(x)
model_LSTM = tf.keras.Model(inputs, predictions,name="model_LSTM")
model_LSTM.compile(loss='sparse_categorical_crossentropy',
                    optimizer='adam', metrics=['accuracy'])
print(model_LSTM.summary())
Model: "model_LSTM"
```

Layer (type)	Output Shape	Param #
input_7 (InputLayer)	[(None, 20)]	0
embedding_6 (Embedding)	(None, 20, 200)	24000
dropout_6 (Dropout)	(None, 20, 200)	0
lstm (LSTM)	(None, 20, 512)	1460224
flatten_6 (Flatten)	(None, 10240)	0
dense_6 (Dense)	(None, 120)	1228920

```

Total params: 2,713,144
Trainable params: 2,713,144
Non-trainable params: 0

```

Şekil 3.11: LSTM Modeli Parametreleri.

Şekil 3.11’de LSTM modelinin kod yapısı görülmektedir. LSTM Modelinin ilk satırında modelin kullandığı girdi değerlerinin boyut bilgisi 20 olarak verilmektedir. Daha sonra bir embedding katmanı kullanılarak her bir karaktere karşılık gelen sayısal değerler 200 boyutlu bir vektörle gösterilmeye çalışılmaktadır. Modelin eğitim veri setini ezberlememesi için

Dropout katmanı kullanılmaktadır. LSTM katmanının hücre sayısı 512 belirlenerek LSTM çıktısının 512 olması sağlanmaktadır. Return_sequence değerini true yapılarak 20 adımda yaptığı her bir gizli durum (ht) değerlerinin işleme sokulması gerektiği belirtilmektedir. Daha sonra tüm değerler düz bir tek vektör haline getirilerek dense katmanına verilip karakter sayısı kadar çıktı alınmakta ve aktivasyon fonksiyonundan geçirilip en olası karakter seçilmektedir. Uygulamada 120 farklı karakter girdi olarak kullanılmaktadır. Model toplamda 2.7 milyon parametreyi işleyerek eğitilmektedir.

```
inputs = tf.keras.Input(shape=(sequence_length), dtype="int64")
x = layers.Embedding(max_features, embedding_dim)(inputs)
x = layers.Dropout(0.5)(x)
x = layers.GRU(512, return_sequences=True)(x)
x = layers.Flatten()(x)
predictions = layers.Dense(max_features, activation='softmax')(x)
model_GRU = tf.keras.Model(inputs, predictions, name="model_GRU")
model_GRU.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['sparse_categorical_accuracy'])
print(model_GRU.summary())
Model: "model_GRU"
```

Layer (type)	Output Shape	Param #
input_6 (InputLayer)	[(None, 20)]	0
embedding_5 (Embedding)	(None, 20, 200)	24000
dropout_5 (Dropout)	(None, 20, 200)	0
gru_5 (GRU)	(None, 20, 512)	1096704
flatten_5 (Flatten)	(None, 10240)	0
dense_5 (Dense)	(None, 120)	1228920

```
=====
Total params: 2,349,624
Trainable params: 2,349,624
Non-trainable params: 0
```

Şekil 3.12: GRU Modeli Parametreleri.

Şekil 3.12’de GRU modelinin kod yapısı görülmektedir. GRU modeline LSTM modeline benzer şekilde ilk satırda girdi değerlerinin boyut bilgisi 20 olarak verilmektedir. Embedding katmanında her bir karaktere karşılık gelen sayısal değerler 200 boyutlu bir vektörle gösterilmektedir. GRU katmanının hücre sayısı 512 belirlenerek GRU çıktısının 512 olması sağlanmaktadır. Return_sequence değerini true yapılarak 20 adımda yaptığı her bir gizli

durum (ht) değerlerinin işleme sokulması gerektiği belirtilmektedir. Daha sonra tüm değerler düz bir tek vektör haline getirilerek dense katmanına verilip karakter sayısı kadar çıktı alınmakta ve aktivasyon fonksiyonundan geçirilip en olası karakter seçilmektedir. GRU modeli LSTM modelinden daha hızlı çalışan ve küçük veri setlerinde daha başarılı olan bir modeldir. Model toplamda yaklaşık 2.3 milyon parametreyi işleyerek eğitime çalışılmaktadır.

```
encoder_inputs = tf.keras.Input(shape=(sequence_length,), dtype="int64", name='encoder_inputs')
embedding = layers.Embedding(max_features, embedding_dim)
embedded= embedding(encoder_inputs)
encoder_lstm = layers.LSTM(LSTMoutputDimension,return_sequences=True, return_state=True, )
encoder_outputs, encoder_state_h, encoder_state_c = encoder_lstm(embedded)
encoder_states = [encoder_state_h, encoder_state_c]
attention= BahdanauAttention(LSTMoutputDimension, verbose=verbose)
decoder_inputs = layers.Input(shape=(1, (embedding_dim+LSTMoutputDimension)))
decoder_lstm = layers.LSTM(LSTMoutputDimension, return_state=True, name='decoder_lstm')
decoder_dense = layers.Dense(max_features, activation='softmax', name='decoder_dense')
all_outputs = []
inputs = np.zeros((batch_size, 1, max_features))
inputs[:, 0, 0] = 1
decoder_outputs = encoder_state_h
states = encoder_states
context_vector, attention_weights=attention(decoder_outputs, encoder_outputs)
context_vector = tf.expand_dims(context_vector, 1)
inputs = tf.concat([context_vector, tf.dtypes.cast(inputs, tf.float32)], axis=-1)
decoder_outputs, state_h, state_c = decoder_lstm(inputs,initial_state=states)
outputs = decoder_dense(decoder_outputs)
outputs = tf.expand_dims(outputs, 1)
all_outputs.append(outputs)
inputs = outputs
states = [state_h, state_c]
decoder_outputs = layers.Lambda(lambda x: K.concatenate(x, axis=1))(all_outputs)
model_encoder_decoder_Bahdanau_Attention = Model(encoder_inputs,
decoder_outputs, name='model_encoder_decoder_Attention')
```

Şekil 3.13: Kodlayıcı Kod Çözücü ve Dikkat Modeli Parametreleri.

Şekil 3.13’de modelin kod yapısı verilmektedir. Kodlayıcı kod çözücü ve dikkat modelinde ilk satırında kodlayıcı modelinin girdi değerlerini, girdi dizisinin uzunluğunu ve veri tipinin integer sayı tipinde olduğu belirtilmektedir. Embeding katmanı kullanılarak integer değerlerinin 200 boyutta kelime vektörleri ile temsil edilmesi sağlanmaktadır. LSTM katmanını kullanarak her bir girdi arka arkaya işlenmekte ve çıktılar elde edilmektedir. Bu çıktılar en son katmanda elde ettiğimiz gizli durum değerleri, her bir adımda elde ettiğimiz gizli durum değerleri ve her bir adımda elde ettiğimiz hücre durumu değerleridir. Daha sonra her bir adımda elde ettiğimiz gizli durum değerlerini ve her bir adımda elde ettiğimiz hücre durumu değerlerini bir araya getirerek bir sonraki katman olan LSTM katmanının girdisi olarak kullanılmaktadır. Her bir adımda elde ettiğimiz gizli durum değerleri ise bize verilen girdi değerlerinin özetlenmiş durumunu temsil etmektedir. Kodlayıcının ürettiği bu çıktı değerleri kullanılarak ve daha önceden hazırladığımız dikkat katmanını burada çağırarak kod çözücü katmanı çalıştırılmaktadır. Her adımda bir çıktı elde edeceğimiz için bu değerler bir değişkende toplanmaktadır. Dikkat katmanına kodlayıcının çıktıları ile kod çözücünün ürettiği özet içerik verilmektedir. Dikkat katmanı bize içerik vektörü ve dikkat ağırlıklarını döndürmektedir. İçerik vektör artık neye dikkat etmesi gerektiği bilgisini taşıyan bir vektör durumunda bulunmaktadır. Daha sonra kod çözünün girişini oluşturmak için içerik vektör ile girdi değerini birleştirerek yeni girdi değeri oluşturulmaktadır. Kodlayıcının çıktı değerini tahminde kullanmak için daha önce hazırladığımız dense katmanını vererek sözlükte bulunan karakterlerden kendisine verdiğimiz değerlerin softmax değerlerini hesaplayarak bir dağılım üretmekte ve bize tahminde bulunmaktadır. Çıktı değerlerini adım adım oluşturulduğu için mevcut durumu kaydedip tahminler birleştirilmektedir. Burada bir sonraki döngü için çıktıyı girdi olarak tekrar LSTM hücresine verilmekte ve katmandan gelen yeni gizli durum (state_h) ile hücre durumu (state_c) değerlerini kod çözücünün bir sonraki adımında kullanmak için içerik vektörü güncelleştirilmektedir. Daha sonra kodlayıcının girdileri ve kod çözücünün çıktıları kullanılarak model eğitilmektedir.

Model: "model_encoder_decoder_Attention"			
Layer (type)	Output Shape	Param #	Connected to
encoder_inputs (InputLayer)	[(None, 20)]	0	[]
embedding_1 (Embedding)	(None, 20, 200)	24000	['encoder_inputs[0][0]']
encoder_lstm (LSTM)	[(None, 20, 512), (None, 512), (None, 512)]	1460224	['embedding_1[0][0]']
bahdanau_attention_1 (Bahdanau Attention)	((None, 512), (None, 20, 1))	525825	['encoder_lstm[0][1]', 'encoder_lstm[0][0]']
tf.expand_dims_2 (TFOpLambda)	(None, 1, 512)	0	['bahdanau_attention_1[0][0]']
tf.concat_1 (TFOpLambda)	(512, 1, 632)	0	['tf.expand_dims_2[0][0]']
decoder_lstm (LSTM)	[(512, 512), (None, 512), (None, 512)]	2344960	['tf.concat_1[0][0]', 'encoder_lstm[0][1]', 'encoder_lstm[0][2]']
decoder_dense (Dense)	(512, 120)	61560	['decoder_lstm[0][0]']
tf.expand_dims_3 (TFOpLambda)	(512, 1, 120)	0	['decoder_dense[0][0]']
lambda_1 (Lambda)	(512, 1, 120)	0	['tf.expand_dims_3[0][0]']
=====			
Total params: 4,416,569			
Trainable params: 4,416,569			
Non-trainable params: 0			

Şekil 3.14: Kodlayıcı Kod Çözücüve Dikkat Modeli Parametreleri.

Şekil 3.14’de kodlayıcı kod çözücü ve dikkat modelinin bir özeti görülmektedir. Modelin 20’lik bir girdi değeri aldığı, embedding katmanında herbir girdinin 200’lük karakter vektörü haline dönüştürüldüğü görülmektedir. LSTM katmanının hücre sayısı 512 olarak belirlenmekte ve 20 karakterlik girdi değerlerinin işleme sokulduğu görülmektedir. Çıktıya bakıldığında 1 karakter tahmin edildiği ve tahminin 120 farklı karakter içesinden yapıldığı görülmektedir. Modelin yaklaşık 4.4 milyon parametresi bulunmaktadır. Model bu toplam parametreleri kullanarak öğrenmeyi gerçekleştirmektedir.

4. BULGULAR

Literatür incelendiğinde, herhangi bir makine öğrenmesi modelinin performansının verilerin kalitesiyle doğru orantılı olduğu bilinmektedir. Dolayısıyla derin öğrenme modelinin kalitesi ile kullanılan verilerin kalitesi arasındaki güçlü bir ilişki bulunmaktadır. Ayrıca derin öğrenme modelinin oluşturulması ile ilgili sürecin büyük bölümü veri hazırlama oluşturmaktadır. Veri hazırlama sürecinin veri kalitesini arttırdığı ve modelleme sonuçlarının başarısını iyileştirdiği bilinmektedir. Bu bağlamda elde edilen sonuçların daha kaliteli ve yeterli büyüklükteki bir veri seti ile daha iyi sonuçlar alınabileceği düşünülmektedir.

Veri seti ve modelleme ile ilgi olarak aşağıdaki hususların göz önünde bulundurulması, öneri ve sonuçların daha doğru değerlendirilmesini sağlayacaktır;

- a. Türkçe'ye ait geniş ve güvenilir veri setlerinin bulunmaması, literatürde daha çok İngilizce diline ait geniş kapsamlı veri setlerinin yer alması çalışmanın veri seti noktasındaki en temel sınırlılığını oluşturmaktadır.
- b. Türkçe'nin sondan eklemeli, diğer dillere göre daha karmaşık gramer kurallarına sahip olması hem karakter tabanlı hem de kelime tabanlı tahminlerin güvenilirlik düzeyi özelinde belirli bir sayısal sınırdan kalmasına sebep olmaktadır.
- c. Literatür incelendiğinde metin üretiminde genellikle kelime düzeyinde çalışmaların olduğu görülmektedir. Türkçe literatürde karakter tabanlı daha az araştırmanın yer alması bu tezin özgünlük ve öncüllük değerlerini yükseltmektedir.

Günümüzde metin üretimi uygulamalarında dil modelleri eğitilirken derin öğrenme algoritmalarından yinelemeli yapay sinir ağları kullanılmaktadır. Bu modeller özellikle girdi ve çıktıların farklı boyut ve kategorilere sahip olduğu durumlarda, girdi ve çıktı değerleri arasındaki anlamsal ve sırasal ilişkileri geçmiş bilgileri hatırlayarak öğrenebildiği için metin üretiminde yaygın olarak kullanılmaktadır. Bu çalışmada dil modellerinin oluşturulmasında yinelemeli yapay sinir ağlarından LSTM, GRU ve dikkat merkezli kodlayıcı kod çözücü sinir ağları kullanılmaktadır. Oluşturulan dil modelleri karakter tabanlı olarak eğitilmektedir. Dil modeli eğitilirken modele veri setinden ardışık olarak k-1 tane giriş karakteri ve k. doğru çıktı karakteri verilerek dil modelinin eğitilmesi gerçekleştirilmektedir. Bu işlemler tüm veri seti içinde adım adım ilerleyerek yapılmakta ve bu şekilde tüm veri setinin eğitimi

gerçekleştirilmektedir. Çalışma kapsamında oluşturulan tüm modellemelere ait güvenilirlik düzeyleri (G.D), girdi olarak belirlenmiş bir metin parçasından elde edilen tahmin sonuçları baz alınarak oluşturulmaktadır. Söz konusu düzeylerin belirlenebilmesi adına veri seti içerisinde alınan aşağıdaki metin parçasının ilk 20 karakteri girdi olarak modele verilmektedir;

“Öncelikle her üniversite her bölüm için öğrenci kontenjanı belirliyor.”

Literatür çalışmalarına paralel olarak bu çalışmada elde edilen sonuçların başarı seviyelerini ölçebilmek için “Accuracy” başarı ölçütü hesaplama yöntemi kullanılmıştır. Accuracy değeri modelde doğru tahmin edilen kelime sayısının üretilen toplam kelime sayısına oranı olarak hesaplanmaktadır; Tahmin edilen anlamlı kelime sayısı (TEAKS) ve tahmin edilen manasız kelime sayısı (TEMKS) parametreleri güvenilirlik metriğinin hesaplanmasında kullanılmıştır. Söz konusu hesaplama için elde edilen formül denklem 4.1’de gösterildiği gibidir.

$$G.D = \frac{TEAKS}{TEAKS + TEMKS} \quad (4.1)$$

Aşağıda verilen tablolarda LSTM, GRU, kodlayıcı kod çözücü ve dikkat modellerinin farklı epoch değerleri ve sıcaklık örnek alma yönteminin farklı eşik değerleri için üretilen kelimeler ve modellerin başarıları yüzdeleri gösterilmektedir.

Tablo 4.1: LSTM Modelinin 100 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
LSTM	100	0.2	'Öncelikle', 'her', 'üniversite', 'milletvekili', 'Şiirleri', 'şapka', 'vururken', 'görüntüler', 'ortaya', 'çıkarak', 'da', 'bu', 'fotoğraf', 'yayınlandı', 'olması', 'İçin', 'yakın', 'kopmasının', 'en', 'temel', 'olduğu', 'için', 'bir', 'açıklama', 'yaptı', 'farkla', 'bulunan', 'kişi', 'ile', 'birlikte', 'arama', 'yaptığımızda', 'ise', 'çocuk', 'bayramı', 'olarak', 'sağlamıştı', 'bu', 'İddianın', 'senesinde', 'fotoğraf', 'aktardığı', 'koç', 'haberlerde', 'de', 'bu', 'iddianın', 'doğruluğu', 'Spotify', 'olduğu', 'anlarına', 'ait', 'değil', 'güzleri', 'yılında', 'arama', 'yapmak', 'isteyen', 'düşünde', 'kutlanay'	'Haliforphi', 'metunlar', 'karaboğ'cu'nın', 'turkuvapovice', 'var'an', 'teşekkü'
LSTM	100	0.3	'Öncelikle', 'her', 'üniversite', 'ile', 'aynı', 'anda', 'etkilenen', 'an', 'olmadığını', 'gösteriyor', 'Çakı', 'sağlamak', 'için', 'daha', 'az', 'farklı', 'dinlerinin', 'kaybetmiş', 've', 'ilgili', 'bir', 'bir', 'uyarı', 'olan', 'üzerine', 'de', 'konuyla', 'ilgili', 'bilgi', 'akaryakıt', 'görüntülerde', 'bu', 'yönde', 'bir', 'durum', 'yapan', 'herkes', 'teyitçi', 'kaynaklarından', 'birinde', 'yayınlanmış', 'kolay', 'parlamentosu', 'videonun', 'etmek', 'için', 'kullanıldığını', 'görüyoruz', 'ayında', 'ilgili', 'bir', 'çalışmamızda', 'kullanılan', 'kurum', 'halinde', 'olduğu', 'gibi', 'bir', 'devam', 'tanımları'	'varsayımlaray', 'çöpüçürüÜyormaz', 'havayesinde', 'refmi', 'tesadlük', 'soyynlı', 'siyasiği',
LSTM	100	0.5	'Öncelikle', 'her', 'üniversite', 'ile', 'aynı', 'adımın', 'atatürk', 'için', 'diğer', 'ilgili', 'dönemde', 'çiftlik', 'satın', 'alan', 'isimlerini', 'tespit', 'etmiş', 've', 'bu', 'gibi', 'internet', 'sitesindeki', 'bir', 'milletvekili', 'toplumu', 'kuruluşu', 'ilgili', 'iddiaları', 'çıkarıldı', 've', 'bunlara', 'artıyor', 'olabilir', 'gibi', 'rahatsız', 'edici', 'paranın', 'görüntülerin', 'bir', 'yapıya', 'Çalışını', 'gösteriyor', 'kendisine', 'hatalı', 'terin', 'edildiğinde', 've', 'bilgilerin', 'gelir', 'enerji', 'gelişmeler', 've', 'karşılığı', 'olmadığını', 'görüyoruz', 'haline', 'gelen', 'uçuş', 'yer', 'verilmiş', 'kutum', 'aşamasında', 'açı'	'sonuattık', 'cumaauri', 'gorgan', 'eskaşan', 'oz', 'gebmez',
LSTM	100	0.7	'Öncelikle', 'her', 'üniversitede', 'alman', 'işçi', 'bilinen', 'bir', 'sonuca', 'ulaşmak', 'tarihinde', 'çekilen', 'fotoğraflarla', 'geçmişle', 'ya', 'da', 'vaha', 'insanların', 'olduğu', 'iddiası', 'yanıltıcı', 'yönü', 'korkmaz', 'göç', 'önem', 'katındır', 'bu', 'videonun', 'görüntülerde', 'bir', 'açıklama', 'yaptı', 'da', 'etkili', 've', 'semi', 'hallerini', 'de', 'meydana', 'gelen', 'yerinde', 'hal', 've', 'metro', 'ahmet', 'Öncelikle', 'videonun', 'prens', 'yılında', 'bir', 'açıklama', 'yaptım', 'üstünde', 'platformunda', 'risklerin', 'ilk', 'imasıyla', 'aşık', 'teyitçi', 'gazatesinde',	'gougunla', 'beşire', 'hayavamımı', 'hatislari', 'prensiz', 'zortunu', 'hayvin', 'pultini', 'geliren', 'hişler'e',

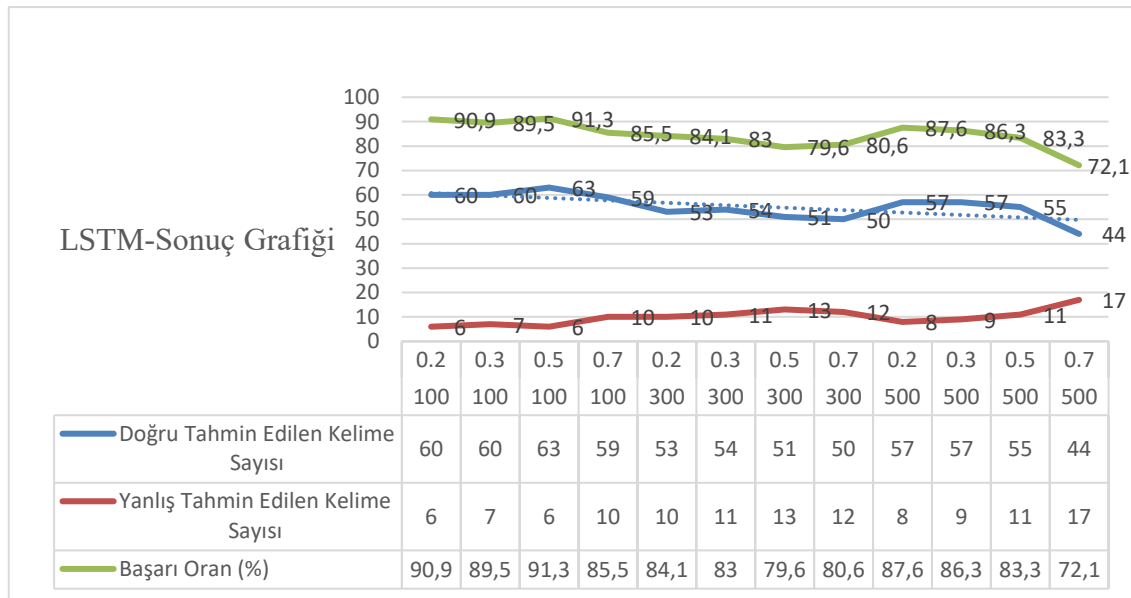
Tablo 4.2: LSTM Modelinin 300 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
LSTM	300	0.2	'Öncelikle', 'her', 'üniversitede', 'de', 'bu', 'kadar', 'yüksek', 'yıkılan', 'olarak', 'gelen', 'olmadığı', 'kayda', 'bunu', 'yansıttığı', 'iddia', 'edildi', 'olmuş', 'uygulamasında', 'erkek', 'konutun', 'fiyatı', 'yılında', 'de', 'bir', 'kararlı', 'yılında', 'aştığı', 'yükte', 'görüntüleriyle', 'hayatını', 'da', 'bir', 'türenin', 'olmasına', 've', 'ediliyor', 'yolu', 'yanlılıkları', 'üzerine', 'de', 'başladığı', 'yaşlarla', 'geliştirmek', 'için', 'kullanılan', 'tamamen', 'uygulamalarından', 'birine', 'göre', 'bir', 'ilgisi', 'fizik', 'tarafından',	'kadrolarış', 'grent', 'yazdımagütü', 'değilterh'de', 'görüsmüyorfok', 'erzaşın', 'rağie', 'esni', 'urtic', 'uygusunuyla',
LSTM	300	0.3	'Öncelikle', 'her', 'üniversite', 'gündemine', 'yapılan', 'yanlış', 'bilgileri', 'durumların', 'kitabında', 'türkiye'de', 'olduğu', 'gibi', 'hadise', 'olayları', 'ile', 'olduğu', 'bir', 'arama', 'motorlarında', 'aramak', 'iletirme', 'tespit', 'etmeye', 'çalışıyor', 'birlikte', 'alınmış', 'emisyenları', 'gerektiğine', 'işaret', 'ederek', 'teyit', 'dolaşıma', 'görüntüleyen', 'olan', 've', 'son', 'kentine', 'bir', 'geçmişe', 'sahip', 'olduğu', 'görüntüler', 'içeriği', 've', 'birçok', 'ülkede', 'neden', 'kaldıkları', 'görülebiliyor', 'ülkede', 'paylaşılmış', 'ver', 'yolu', 'bağırırssa',	'pekklerinin', 'namiy', 'çöğüz', 'çağarımı', 'tarabında', 'tachi', 'sokağan', 'piyosu', 'piice', 'Örünü[UNK]düg', 'kull'
LSTM	300	0.5	'Öncelikle', 'her', 'üniversitede', 'konut', 'piyasası', 'da', 'güncelleme', 'buğdayından', 'ayrılmamıştır', 'yayınının', 'üyeliliğini', 'çıkışını', 'doğuşu', 'olabilir', 'olduğu', 'etmiş', 'kurumu', 'kadar', 'açıklamaya', 'gelen', 'çıkarcak', 'olursak', 'yine', 'bu', 'da', 'kullanılmış', 'geçirmesi', 'aracında', 'da', 'yanıtı', 'vermek', 'için', 'bu', 'gibi', 'bir', 'yanlış', 'bilgileri', 'durduğu', 'iddia', 'gibi', 'görüntülerin', 'gerçek', 'bir', 'nato', 'fiyaka', 'biliyordu', 'pağanın', 'sunuyor', 'temyiz', 'uzak', 'çocuk'	'spatt', 'geci', 'scavy', 'milletive', 'aksiyegi', 'sufy', 'uşrua', 'memiu', 'donhakımluyun', 'araçgada', 'edildiİçdağının', 'apğı', 'çabışmış',
LSTM	300	0.7	'Öncelikle', 'her', 'üniversitede', 'bulunması', 'gerektiğini', 've', 'paylaşımlarda', 'konut', 'açık', 'ama', 'araçların', 'çekilmiş', 'olduğundan', 'uygulanacak', 'raporundaki', 'olaylara', 'ulaşıyoruz', 'başka', 'bir', 'versiyonunu', 'yansıttığı', 'yanlış', 'bilgileri', 'durduğu', 'için', 'biliyordu', 'veya', 'yedenin', 'haberinin', 'bir', 'mesajlarından', 'dolayı', 'ne', 'yazıyor', 'söylediği', 'programı', 'var', 'Öncelendiğini', 'söyledi', 'erik', 'görüntü', 'elektriğin', 'günlük', 'kayıt', 'sayfasında', 'da', 'teslimleri', 've', 'zelenski'nin', 'değiştirilerek',	'baslıngı'ya', 'şışket', 'amd', 'donguru', 'İmitorlaşarakı', 'jemi', 'arbasan', 'gebi', 'izelenmiş', 'fazül', 'bunlarl', 'ulaşılabiliorbundan',

Tablo 4.3: LSTM Modelinin 500 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
LSTM	500	0.2	'Öncelikle', 'her', 'üniversite', 'ilk', 'kez', 'bulunamadık', 'şiir', 'üzerinde', 'de', 'yeterliliklerini', 'gösterdiği', 'yapak', 'anlaşıyor', 'olacağını', 'geliyorlar', 'ayaklarından', 'kalmak', 'adına', 'kaynaklandığı', 'belirtilmiş', 'değil', 'ayrı', 'olduğunda', 've', 'raporlara', 'da', 'arkadaşlarıyla', 'ilgili', 'başlıyor', 'gibi', 'ifadeleri', 'kullanıyor', 'şekilde', 'de', 'yeni', 'tam', 'de', 'uzun', 'ziyaret', 'de', 'adım', 'alacağımızda', 'divanı', 'temizle', 'iletişime', 'geçmek', 'üzere', 'sonunda', 'uygulanacak', 'bir', 'geri', 'dönüş', 'tarihi', 'de', 'büyük', 'dayattı', 'evrim',	'getçer', 'regtorgal', 'lojco'na', 'oço', 'talafandırılması', 'buğuraalan', 'yünya', 'babımıyor...'
LSTM	500	0.3	'Öncelikle', 'her', 'üniversite', 'ile', 'bir', 'alan', 'tüteniyle', 'tekrar', 'nedeniyle', 'kilovat', 'hakkında', 'şiddetli', 'bir', 'yerinde', 'gerçekçi', 'toplum', 'cenevre', 'binası', 'türkiye'deki', 'İzli', 'videoyu', 'bu', 'konuyla', 'ilgili', 'bilgi', 'alanı', 'olmadığı', 'görülüyor', 'veya', 'ilişkili', 've', 'gibi', 'suriyelilere', 'yayılan', 'iddia', 'eki', 'yeni', 'zara', 'iki', 'dola', 've', 'kanunu', 'ifade', 'ediyor', 'cinsel', 'tahsin', 'merkezli', 'görüntüsü', 'arma', 'katı', 'yazdıklarını', 'içeren', 'önlemi', 'ilerisi', 'oranları', 'bulunduğunu', 'söyleşi'	'teşekkilenlerin', 'kastonularının', 'mikvaşatı', 'bocaygüm'ün', 'fitmisayla', 'dağrasi', 'muyar', 'hatırlatmış', 'deği'
LSTM	500	0.5	'Öncelikle', 'her', 'üniversite', 'ilk', 'sorun', 'var', 'ağır', 'bir', 'önemler', 'yananlarını', 'bir', 'yandan', 'doğru', 'çıkan', 'halk', 'sonucunda', 'alan', 'verilerin', 'haz', 'sosuyla', 'öldürüldüğünde', 'belirtilen', 'isimden', 'olmadığı', 'ya da', 'halinde', 'getirdiği', 'iddia', 'edilen', 'fotoğraf', 'yorumlara', 've', 'ifadesidir', 'zaten', 'konuk', 'fiyatları', 'kanununda', 'konuşman', 'ajansı', 'fotoğrafları', 'para', 'uygulayan', 'paylaşım', 'için', 'değil', 'sorunlarındaki', 'tüm', 'mülteci', 'çağlı', 'bir', 'için', 'de', 'düşürülen', 'görüntüleriyle',	'işlenileri', 'googled', 'etrebsiz', 'glib', 'kanutların', 'anbimes'in', 'centerdin', 'youtubei', 'bülde', 'itabzen', 'hafızlamda',
LSTM	500	0.7	'Öncelikle', 'her', 'üniversite', 'ilk', 'kabil', 'sanır', 'e', 'ile', 'ilgili', 'politikaları', 'silahçı', 'tahsin', 'içeriklerin', 'ilgili', 'için', 'değildir', 'kuburlarında', 'uyuyorum', 'kullanımının', 'yanı', 'sıra', 'tümür', 'ediyor', 'bin', 'olacak', 'ilanı', 'edildi', 'yapması', 'sergilendiğini', 'belirtiyor', 'de', 'doğalgaz', 'uçacağını', 'belirtmiş', 'alışveriş', 'olduğu', 'anlaşılabiliyor', 'ya', 'de', 'fotoğraf', 'üzerinden', 'değil', 'suçtun', 'beşikçik',	'ağlandırma', 'izizi', 'düzengiler', 'binazın', 'yasılı', 'gülü', 'butgoki', 'siyatinde', 'bilinmeleriyleymte n', 'natk'in', 'uzmayı', 'müyaza', 'muydur', 'kahı'nın', 'dereğeri', 'asbera', 'andes'a'taki'

Karakter tabanlı olarak eğitilen LSTM dil modeli 100,300,500 epoch değerlerinde ve sıcaklık örnek alma yönteminin farklı eşik değerlerinde çalıştırıldığında alınan sonuçlar tablo 4.1, 4.2, 4.3,'te ayrıntılı olarak gösterilmektedir. Tablolar incelendiğinde LSTM modelinin veri seti içerisindeki karakterlerin birbiri ile olan anlamsal ve sırasal ilişkileri yakalayabildiği ve modelin anlamlı kelime üretiminde başarılı olduğu görülmektedir. Modelin uzun ve kısa kelimeleri (bu,ile,için,yeterliliklerini, kaynaklandığı, anlaşılabilir, öldürüldüğünde) üretmede başarılı olduğu ve yanlış ürettiği çoğu kelimeyi ise bir kaç karakter ile yanlış olarak ürettiği (tarifinden, söyleşiği, uzmayı, 'youtubei, hafızlamda, işlenileri, getçer, kanutların, yünaya, büşünde vb.) görülmektedir. Modelin Türkçe dil bilgisi kurallarına uygun olarak ayrı yazılan (ya da) ve özel isimlere getirilen ekleri (zelenski'nin, türkiye'deki) öğrenebildiği ve doğru bir şekilde ayırabildiği görülmektedir. LSTM modelinin anlamlı kelime grupları üretmede başarılı olduğu da görülmektedir. (fotoğraf yayınlandı, çocuk bayramı, anadolu ajansı, bu iddianın doğruluğu, internet sitesindeki, rahatsız edici görüntülerin, karşılığı olmadığını görüyoruz, savaş uçakları suriye'de, twitter hesabı paylaşımından, arama motorlarında aramak, tespit etmeye çalışıyor, bir geçmişe sahip olduğu, konut piyasasın da güncelleme, zelenski'nin raporundaki olaylara ulaşıyoruz, gerçek hayatta olduğuna da)



Şekil 4.1: LSTM Karakter Tabanlı Modele Ait Sonuç Grafiği.

Şekil 4.1'de LSTM dil modelinin farklı epoch değerlerinde ve sıcaklık örnek alma algoritmasının farklı eşik değerlerinde doğru üretilen anlamlı kelime sayısı, yanlış üretilen

kelime sayısı ve modelin başarı yüzdesi gösterilmektedir. LSTM modeli 100 epoch çalıştırıldığında ve 0.2 sıcaklık eşik değerinde 90,9%, 0.3 sıcaklık eşik değerinde 89,5%, 0.5 sıcaklık eşik değerinde 91,3% ve 0.7 sıcaklık eşik değerinde 85,5% değerinde sonuçlar vermektedir. LSTM modeli 100 epoch sonucunda 0.5 sıcaklık eşik değerinde 91,3% değeri ile en başarılı sonucu vermektedir. LSTM modeli 300 epoch çalıştırıldığında ve 0.2 sıcaklık eşik değerinde 84,1%, 0.3 sıcaklık eşik değerinde 83%, 0.5 sıcaklık eşik değerinde 79,6% ve 0.7 sıcaklık eşik değerinde 80,6% değerinde sonuçlar vermektedir. Modelin 300 epoch sonucunda 0.2 sıcaklık eşik değerinde 84,1% değeri ile en başarılı sonucu verdiği görülmektedir. LSTM modeli 500 epoch çalıştırıldığında ise 0.2 sıcaklık eşik değerinde 87,6%, 0.3 sıcaklık eşik değerinde 86,3%, 0.5 sıcaklık eşik değerinde 83,3% ve 0.7 sıcaklık eşik değerinde 72,1% değerinde sonuçlar vermektedir. Modelin 500 epoch sonucunda 0.2 sıcaklık eşik değerinde 87,6% değeri ile en başarılı sonucu verdiği görülmektedir. LSTM dil modelinin farklı epoch değerlerinde ve sıcaklık örnek alma algoritmasının farklı eşik değerlerinde alınan sonuçları değerlendirildiğinde modelin en başarılı sonucu 100 epoch değerinde ve örnek alma yönteminin 0.5 eşik değerinde vermektedir. Diğer epoch ve sıcaklık değerlerinde başarının yaklaşık 72,1% - 90,9% aralığında değiştiği görülmektedir. Modelin sıcaklık örnek alma yönteminin 0.2, 0.3, 0.5 eşik değerleri için daha başarılı sonuçlar verdiği, eşik değeri arttığında ise rastgeleliğin artarak ('bilinmeleriyleymten', 'İmitorlaşarakı', 'hayavamımı') anlamlı kelime üretiminde çok da başarılı sonuçlar vermediği görülmektedir. Ayrıca modelin 300 ve 500 epoch değerlerinde başarısının artmasının beklendiği halde başarısının düşmesinin nedeni olarak veri setinin kalitesinin ve veri seti boyutunun yetersiz olmasından kaynaklandığı düşünülmektedir.

Tablo 4.4: GRU Modelinin 100 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
GRU	100	0.2	'Öncelikle', 'her', 'ilişkin', 'inliyor', 'ya da', 'araçlarını', 'tartıştığı', 'iddia', 'edildi', 'de', 'kaydedilmesini', 'yapanların', 'yapmak', 'için', 'bir', 'açıklama', 'yapmadıkları', 'için', 'tamamlanmasından', 'sonra', 'ona', 'ait', 'olduğu', 'iddiasıyla', 'görsellerin', 'kullanıcıları', 'görebiliyoruz', 'yani', 'bir', 'videonun', 'farklı', 'sonuçları', 'gerçek', 'kaynakları', 'açıklamıştı', 'yanlış', 'bilginin', 'yer', 'aldığı', 'görülüyor', 'yardım', 'konut', 'fiyat', 'artışı', 'ile', 'ilgili', 'altında', 'bir', 'açıklama', 'yapmadıkları', 'için', 'tamamlanmasından', 'sonra', 'ona', 'ait'	'ünivere', 'geogle', 'ketkilerini', 'interel', 'paylaşılmışvokal', 'pa'
GRU	100	0.3	'Öncelikle', 'her', 'ilişkin', 'izliyor', 'tarifelerde', 'kaldı', 'sunan', 'bir', 'kadının', 'olmayan', 'tabloya', 'yer', 'verdiği', 'yazıyor', 'yanı', 'türk', 'devletleri', 'için', 'başka', 'bir', 'varlık', 'yapıyor', 'diş', 'kullanmak', 'için', 'bir', 'açıklama', 'yapmadığı', 'için', 'bu', 'yüzde', 'olan', 'kuruluyor', 'isimli', 'birbirinin', 'farklı', 'konut', 'primine', 'harekete', 'gececeği', 'müslümanların', 'konuşmalarının', 'doğru', 'olmadığı', 'yönünde', 'bilgi', 'genç', 'bir', 'açıklama', 'yaptı', 'da', 'yöntemlerini', 'taşıyan', 'kadınlara', 'sebep', 'olan', 'olay',	'ünivere', 'geogle', 'koloylu', 'ılanı', 'mocunu', 'konyolu', 'guuphe', 'renfi', 'kışlı', 'yapmadıkl'
GRU	100	0.5	'Öncelikle', 'her', 'ilişkin', 'uygulamak', 'olarak', 'bir', 'konuştu', 'ilgili', 'geliştirdikleriyle', 'bilgilendirme', 'yapmış', 'olduğunu', 'söyledi', 'başka', 'bir', 'youtube'a', 'yüklenen', 'ise', 'suç', 'hayatı', 'kadınlar', 'ise', 'bu', 'daha', 'önce', 'ki', 'videoyu', 'anlattığı', 'açıklamada', 'son', 'dönemlerde', 'tablonun', 'şair', 've', 'sayfalarına', 'yönlendirerek', 'türkiye'deki', 'görev', 'kullanıcıların', 'fotoğrafları', 'gerekli', 'etkileşim', 'kurumu', 'tarihli', 'açıklaması', 'göre', 'iddianın', 'kaynaklarına', 'konu', 'olabilir', 'yerde', 'enerji', 'paylaşıldığını', 'görüyoruz', 'da'	'ünivere', 'hastaneke', 'tanzmı', 'maltep', 'aradağımızda', 'komüs', 'yokıddiaya'
GRU	100	0.7	'Öncelikle', 'her', 'iliyor', 'din', 'bir', 'tek', 'diş', 'mücadeledeki', 'için', 'esir', 'olmadığını', 'ifade', 'etti', 'şöyle', 'aktarılmış', 'bir', 'aslında', 'gösteren', 'fotoğraf', 'yarını', 'ortaya', 'atılmıştı', 'eğer', 'olmasa', 'da', 'ukrayna', 'vermiş', 'fiyatla', 'birlikte', 'arasında', 'geçen', 'açıklaması', 'şu', 'şekildeydi', 'zaman', 'zammı', 'da', 'şeklindeki', 'te', 'yayınlanan', 'gergin', 'kaydetmeye', 'çalıştığı', 'biliniyor', 'dahil', 'olmak', 'üzere', 'kaynaklarının', 'de', 'papasını', 'ise', 'önce', 'sitelerine', 'daha', 'önce', 'de', 'belirleniyor', 'başlıklı', 'da'	'ünivere', 'mohkmm', 'replike', 'cevab', 'statisi', 'dezankonu', 'gemilçra', 'nazar', 'markup', 'giplektis', 'alayk'

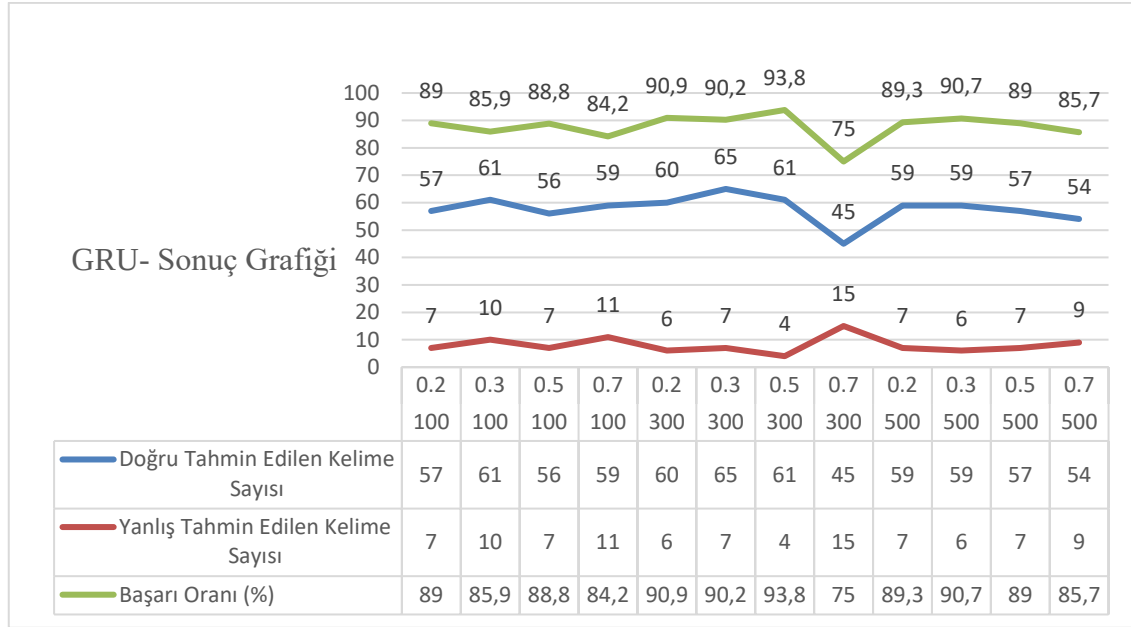
Tablo 4.5: GRU Modelinin 300 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
GRU	300	0.2	'Öncelikle', 'her', 'üniversite', 'yoğunların', 'aracı', 'desteklemek', 'için', 'kaleme', 'aldığı', 'zeminden', 'yıkılabilirler', 'ancak', 'ele', 'geçirmek', 'veya', 'kolay', 'tersine', 'görsel', 'arama', 'yöntemiyle', 'aktarılmalar', 'yapıldı', 'bir', 'yazıda', 'olduğu', 'da', 'ileri', 'site', 'kullanıcı', 'tarafından', 'yayınlanan', 'görmesiyle', 'teşkilat', 'teklif', 'edilmedi', 'ancak', 'milletvekili', 'bir', 'the', 'alınacak', 'bir', 'başka', 'bir', 'videoya', 'ulaşan', 'edebiyatının', 'silahı', 'teknoloji', 'yeni', 'aylık', 'yılına', 'kadar', 'kurtarıken', 'görüldüğü', 'iddia', 'edilen', 'cezaevi', 'bakanlar', 've', 'fotoğraf'	'wergtn', 'adeyle', 'kanut', 'Çifine', 'yapıldığı', 'kullandıcılar'
GRU	300	0.3	'Öncelikle', 'her', 'üniversite', 'yoğun', 'bir', 'diğer', 'idi', 'duyurumun', 'ilk', 'olarak', 'de', 'bir', 'vatandaşlık', 'İptal', 'değil', 'görselin', 'ama', 'bir', 'restoranda', 'sosyal', 'medyada', 'dolaşan', 'bir', 've', 'ismin', 'değil', 'ama', 'en', 'güçlerden', 'biri', 'de', 'diğer', 'üç', 'köylerin', 'bilimsel', 've', 'aldıkları', 'sözleniyor', 'aynı', 'dünya', 'başbakanının', 'milyon', 'dolar', 'özel', 'olarak', 'bir', 'resmi', 'hastalık', 'hareket', 'eden', 'bir', 'açıklama', 'yaptı', 'da', 'paylaşımlarda', 'da', 'kullanılabilir', 'daha', 'sağlanmıştı', 've', 'bu', 'da', 'enflasyon', 'oranında', 'düzenlene'	'paylaşılmışvan', 'zenii', 'öğüregini', 'oncan', 'yapılabilir', 'afam', 'çoruklar'
GRU	300	0.5	'Öncelikle', 'her', 'üniversite', 'yoğun', 'bir', 'diğer', 'karanlık', 'olarak', 'gözlemlenebilir', 'görüntülerin', 'gerçek', 'olmadığını', 've', 'aralık', 'de', 'türkiye'de', 'bir', 'açıklama', 'yapmadı', 'bilgi', 'vermek', 'için', 'kaldı', 'değiştirildiğini', 'anlama', 'ergininin', 'ilk', 'kaynakta', 'kullanıyor', 'geçerle', 'dikilir', 'internette', 'yapılan', 'aramada', 'uluslararası', 'fotoğrafta', 'görülebilir', 've', 'medya', 'platformları', 've', 'bazı', 'ölçüsü', 'ya da', 'takip', 'etmek', 'için', 'ele', 'geçirildiğini', 'belirttiği', 'baskı', 'açık', 've', 'yanıtlı', 'en', 'başka', 'bir', 'teknikte', 'hasta',	'tÜİK'in', 'susyal', 'kuzayı', 'kollanıcının',
GRU	300	0.7	'Öncelikle', 'her', 'bir', 'kadının', 'böyleydi', 'arabalığındaki', 'bir', 'sağlığı', 'çekildiği', 'gibi', 'olabildiğini', 've', 'önerdiği', 'bir', 'halka', 'verdiği', 'de', 'enflasyon', 'sonrası', 'dijital', 'düzenleme', 'içeriyordu', 'oluşuyordu', 'bu', 'yana', 'türkiye'deki', 'mümkün', 'değil', 've', 'yüzüme', 'anlatılara', 'İnsan', 'olduğu', 'açıklama', 'uçacağının', 'rahatmış', 'geçiş', 'ödeme', 'yapmak', 'konusunda', 'uygulanacak', 'çok', 'sağlaya',	'üniverek', 'borakakı', 'ecivnin'de', 'Çarıştırdığımızla', 'getirselerinden', 'aklıymaz', 'güzetini', 'çrumag', 'şabok', 'öldü'den', 'fenkub', 'ekronca', 'biyokü', 'tanımlanıyoraynı', 'gerekiyoraogetleri'

Tablo 4.6: GRU Modelinin 500 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
GRU	500	0.2	'Öncelikle', 'her', 'üniversite', 'görselin', 'temellendiğinde', 'ile', 'görüntüleri', 'gerektiğini', 'görüyoruz', 'sonuçları', 'görülebiliyor', 'geçmişte', 'anlaşılmadığını', 'söyledi', 'bir', 'alay'ın', 'bir', 'kadın', 'görüntülerin', 'karşılığı', 'da', 'bulunuyor', 'olduğunu', 'söyledi', 'bir', 'ajansı', 've', 'maliyetini', 'işlemleri', 'yanlış', 'bilgileri', 'de', 'yapılan', 'açıklamaları', 'gerektiğini', 'açıkladı', 'artısı', 'da', 'büyükşehir', 'açıklama', 'belediyesi'nde', 'ortaya', 'çıkan', 'aşağıdaki', 'fiyatlarının', 'yüzde', 'ben', 'güçlü', 'bir', 'arama', 'motorlarında', 'arama', 'konusunda', 'bir', 'karşılığı', 'motor'	'juhe', 'ila', 'kalabalıyor', 'ini', 'bilgise', 'aramal', 'bulunduğuk'
GRU	500	0.3	'Öncelikle', 'her', 'üniversite', 'gerçek', 'olduğu', 'iddiası', 'yanıltıcı', 'yönü', 'Öyküm', 'hüma', 'keskin', 'teyitçi', 'Çatışma'da', 'daha', 'önce', 'olmadığını', 'belirtmişti', 'aşısı', 'için', 'hesapları', 'için', 'yaptığı', 'için', 'kaydedilmiş', 'değil', 'ortamlarında', 'araç', 'halinde', 'son', 'yıldan', 'satıcı', 'olmadığı', 'gibi', 'aşı', 'da', 'aşılama', 'olabilir', 'olduğunu', 'söylediklerini', 'gösterdiği', 'iddia', 'edildi', 'özel', 'iki', 'karşılaştırma', 'yapmak', 'zorunda', 'kalın', 'değerleri', 'gerektiren', 'bir', 'kaynak', 'olarak', 'bilinen', 'bir', 'fotoğrafın', 'görüntülerin', 'bir',	'etiyor', 'olabiliri', 'karşılıkla', 'dek', 'olabilin', 'gerekiorr'
GRU	500	0.5	'Öncelikle', 'her', 'üniversite', 'yapılan', 'bir', 'teknoloji', 'haline', 'gelişmiş', 'de', 'uzmanlar', 'haberin', 'bir', 'kaynak', 'olmadığını', 'belirtmiş', 'artıyoruz'da', 'yaşananların', 'insanların', 'aldıkları', 'gerektiğini', 'ifade', 'etmiş', 'Şubat'ta', 'yayınlanan', 'ifadelerin', 'konusunda', 'yakın', 'iz', 'de', 'bu', 'ilk', 'polisi', 'yılında', 'insan', 'erişimlere', 'ait', 'olduğunu', 'söyledi', 'bir', 'arama', 'yaptığımızda', 'bilinen', 'bir', 'kaynak', 'oluşturuyor', 'Çatışma', 'insanları', 'doğu', 've', 'kayıtların', 'gerçek', 'olan', 'tarama', 'buluşuyor', 'olmasını', 'soru', 'bilgi', 'karşılışması'	'abirin', 'kamşulama', 'dedemiye', 'gerekt', 'ayırmaza', 'işleti', 'aki'
GRU	500	0.7	'Öncelikle', 'her', 'üniversite', 'görsel', 'arama', 'ile', 'şirketi', 'tutuklanan', 'komple', 'teyitçi', 've', 'kayıtta', 'milli', 'eğitim', 'Çatışma', 'popüler', 'içeriklerinden', 'birinin', 'ya da', 'aşının', 'ilk', 'madde', 'gelişmiş', 'olmadı', 'uzadı', 'dolaşımda', 'ilgili', 'kullanılmış', 'amacıyla', 'ulemanın', 'bin', 'yılı', 'yanıltıcı', 'yönü', 'ali', 'osman', 'arabacı', 'mezarlık', 'için', 'görüntüleri', 'türkiye'deki', 'aradığımız', 'bazı', 'haberler', 'gerçek', 'dağılmış', 'olmayan', 'halinde', 'de', 've', 'alkol', 'de', 'görüntüleri', 'gibi', 'gelişmiş'	'sayfalgı', 'bağlantıdır', 'johnef', 'intisal', 'iştirilerek', 'seles', 'alen', 'değişikli'n', 'mernes'i'

GRU modelinin 100,300,500 epoch değerlerinde ve sıcaklık örnek alma yönteminin farklı eşik değerlerinde alınan sonuçlar tablo 4.4, 4.5, 4.6'da ayrıntılı olarak verilmektedir. GRU modeli LSTM modeline benzer yapıda çalışan, LSTM modeline göre daha az işlem adımı kullanan buna bağlı olarak daha az bellek tüketen ve daha hızlı çalışan bir modeldir. Bu nedenle sonuçların LSTM modeline yakın çıkması beklenmektedir. Sonuçlar incelendiğinde GRU modelinin LSTM modeline benzer şekilde anlamlı kelime üretiminde başarı olduğu görülmektedir. Modelin uzun ve kısa kelimeleri üretmede başarılı olduğu (ait, bu, çok, bir, ama, tamamlanmasından, müslümanların, kaynaklarının, konuşmalarının, değiştirildiğini) ve yanlış ürettiği çoğu kelimeleri ise LSTM dil modeli gibi bir kaç karakter ile yanlış olarak ürettiği (hakkında, kanut, yapıldığı, kullandıcılar, kollanıcının, Çarıştırdığımızla, güzetini vb.) görülmektedir. Modelin Türkçe dil bilgisi kurallarına uygun olarak ayrı yazılan (ya da) ve özel isimlere getirilen ekleri (belediyesi'nde, türkiye'deki) ayırmayı öğrenebildiği ve doğru bir şekilde ayırabildiği görülmektedir. Modelin aynı zamanda bazı özel isimleri (üniverek, geogle, hastaneke sosyal, kuzayı, wergtn, statisi, maltep) doğru tahmin edemediği görülmektedir. Modelin kelimeler arasında anlamsal sıra ve birliktelik yakaladığı kelime grupları ise şu şekildedir; (türk devletleri, internette dolaşıma sokulmuş, bu konuyla ilgili bilgi almak için bir kaynak, harekete geçeceği, müslümanların konuşmalarının doğru olmadığı yönünde bilgi, bilgilendirme yapmış olduğunu söyledi, daha önce ki videoyu anlattığı medyada dolaşan bir, enflasyon oranında düzenleme, görüntülerin gerçek olmadığını, internette yapılan aramada uluslararası fotoğrafta görülebiliyor, söz konusu fotoğraf yer alıyor, sosyal medyada yeniden dolaşan, açıklamada son dönemlerde başarılı olduğu görülmektedir)



Şekil 4.2: GRU Karakter Tabanlı Sonuç Grafiği.

Şekil 4.2’de GRU modelinin farklı epoch ve sıcaklık örnek alma algoritmasının farklı eşik değerlerinde üretilen anlamlı kelime, anlamsız kelime sayısı ve modelin başarı yüzdesi gösterilmektedir. GRU modeli 100 epoch çalıştırıldığında 0.2 sıcaklık eşik değerinde 89%, 0.3 sıcaklık eşik değerinde 85,9%, 0.5 sıcaklık eşik değerinde 88,8% ve 0.7 sıcaklık eşik değerinde 84,2% değerinde sonuçlar vermektedir. GRU modeli 100 epoch sonucunda 0.2 sıcaklık eşik değerinde 89% değeri ile en başarılı sonucu vermektedir. GRU modeli 300 epoch çalıştırıldığında 0.2 sıcaklık eşik değerinde 90,9%, 0.3 sıcaklık eşik değerinde 90,2%, 0.5 sıcaklık eşik değerinde 93,8% ve 0.7 sıcaklık eşik değerinde 75% değerinde sonuçlar vermektedir. Modelin 300 epoch sonucunda 0.5 sıcaklık eşik değerinde 93,8% değeri ile en başarılı sonucu verdiği görülmektedir. GRU modeli 500 epoch çalıştırıldığında ve 0.2 sıcaklık eşik değerinde 89,3%, 0.3 sıcaklık eşik değerinde 90,7%, 0.5 sıcaklık eşik değerinde 89% ve 0.7 sıcaklık eşik değerinde 85,7% değerinde sonuçlar vermektedir. Modelin 500 epoch sonucunda 0.3 sıcaklık eşik değerinde 90,7% değeri ile en başarılı sonucu verdiği görülmektedir.

Tablo 4.7: Kodlayıcı Kod Çözücü ve Dikkat Alg. Modelinin 100 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
Dikkat Alg.	100	0.2	'Öncelikle', 'her', 'üniversite', 'kullanıcıları', 'hatalı', 'ağızları', 'harekete', 'geliyor', 've', 'internet', 'bilinmiyor', 've', 'bunu', 'kullandığı', 'haberleri', 'içerisinde', 'bulunan', 'ilahiyaçıya', 've', 'rusya', 've', 'gergininin', 'kuruluşunda', 'ileri', 'ile', 'daha', 'önce', 'de', 'plakalı', 'bir', 'pandemi', 'görüntüleriyle', 'aktarmış', 've', 'haberler', 'geldi', 'alman', 'tarafında', 'sağlıyor', 've', 'ulusal', 'yapmaya', 'devam', 'ediyor', 've', 'kar', 'mı', 'bu', 'yangının', 'detaylı', 'adını', 'eylemiş', 've', 'hangi', 'içerik', 'bu', 'değerlendirme', 'bir', 'kullanıcı', 'covid	'çetğimizi', 'nin', 'afp', 'egemaldır', 'yağıldı', 'genyizle', 'ünivince', 'yapmayısı'
Dikkat Alg.	100	0.3	'Öncelikle', 'her', 'üniversite', 'kullanıcıları', 'gelişmenin', 'sağladığı', 'biliniyor', 'ima', 'ediyor', 'türkiye'nin', 'toplandığı', 'iddia', 'edilen', 'sosyal', 'medya', 'platformlarında', 'paylaşılan', 'bazı', 'bir', 'paylaşım', 'görüntüsü', 'youtube', 'yüklenmiş', 'break', 've', 'araçlarını', 'kullandığında', 'bu', 'konuda', 'insanları', 'oldukları', 'bir', 'yayılmaya', 'dair', 'gücü', 'gibi', 'bilgilerin', 'geliştirilmesi', 'destekte', 'yedi', 'kararlı', 'bir', 'şeklinde', 'uygulama', 'yapmak', 'üzere', 'kullanılan', 'kuruşlar', 'da', 'daha', '	'lik', 'olduğunu', 'edebiyor', 'emil', 'dötti', 'enunmetteçman', 'ka'
Dikkat Alg.	100	0.5	'Öncelikle', 'her', 'üniversite', 'kullanıcılarına', 'çoklukla', 'aynı', 'bir', 'aradaki', 'ile', 'karşılaşmıyoruz', 've', 'hayatta', 'alınmadığını', 'bekliyoruz', 'iş', 'arama', 'motorlarına', 'yaptığımızda', 'ise', 'daha', 'önce', 'hangi', 'sunulmuş', 'olabilir', 'nasıl', 'yaşandığını', 'belirtti', 'yedi', 've', 'siteleri', 'bir', 'sosyal', 'medya', 'platformları', 'siyer', 've', 'ya', 'da', 'özgürlüğünü', 'ayrılan', 'kanıt', 'araştırma', 'devlet', 'başkanı', 'vladimir', 'putin'in', 'sanatçı', 'olarak', 'kullanılıyor', 'olabilir', 've', 'sahte', 'oldukları', 'yapmayacak'	'çıplama', 'endef', 'statikin', 'sertifikalisy', 'radyorken', 'rengü'
Dikkat Alg.	100	0.7	'Öncelikle', 'her', 'üniversite', 'kullanıcılarına', 'destek', 'sembollerinin', 'ya da', 'önünde', 've', 'kanallarda', 'bir', 'virüs', 'yoktur', 'potansiyel', 'yapılarak', 'inşa', 'virüs'ü', 'kutlamıştır', 'artırmaların', 'bir', 'kayıt', 'gösterdiğimizde', 'ulaşılabilirim', 'araştırma', 'yapmadığı', 've', 'küresel', 'ön', 'de', 'açıklamaları', 'sınırsız', 've', 'devrimi', 'olması', 'şaka', 'sahibinin', 'ikinci', 'bir', 'kaniyor', 'varsayım', 'mı', 'bahsi', 'gidiyoruz', 'benzer', 'kullanıcılar', 'tarafından', 'kullanılmış', 'terim', 'bir',	'ynınlı', 'kayasii', 'edilmişaryoloji', 'yoğan', 'yandayı', 'kiliten', 'protokule', 'durimçel', 'üçtürksen', 'unman', 'hiv'

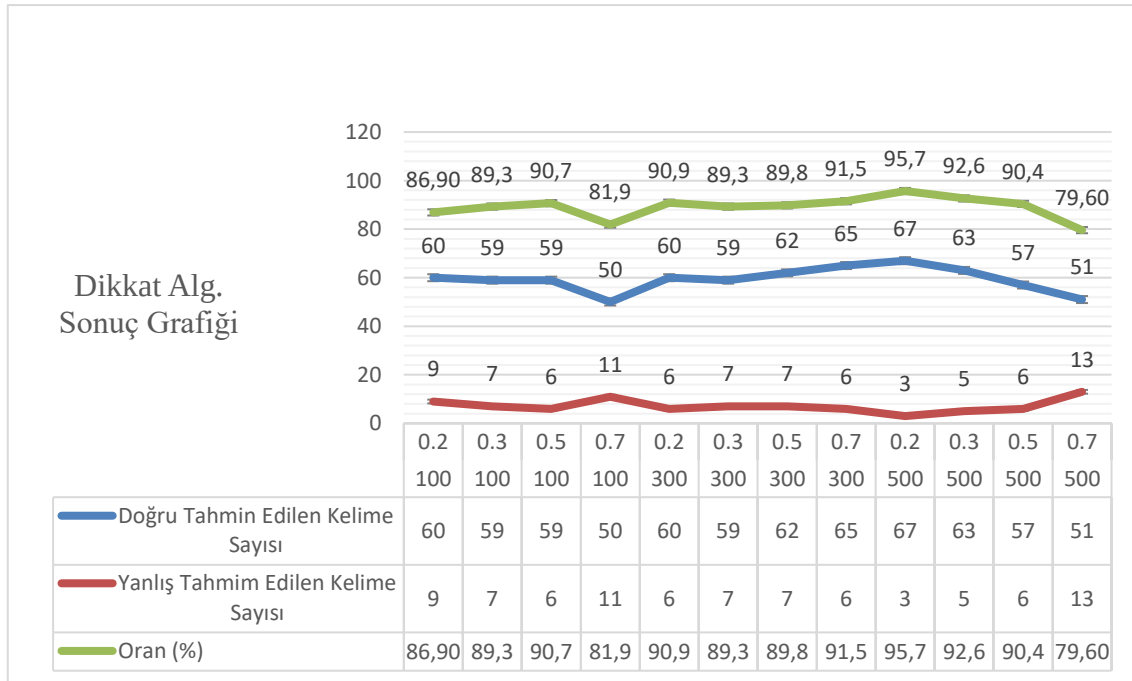
Tablo 4.8: Kodlayıcı Kod Çözücü ve Dikkat Alg. Modelinin 300 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
Dikkat Alg.	300	0.2	'Öncelikle', 'her', 'üniversite', 'hareket', 'edeceğiniz', 'arama', 'motorlarında', 'da', 'fazla', 'virüsün', 'temsil', 'edebiliyor', 'miyiz', 'devrim', 'ya', 'amansa', 'de', 'bilgi', 'göre', 'yüksek', 'bir', 'kanıt', 'idam', 'edilmesini', 'en', 'iyi', 'yılında', 'kuzey', 'yaygın', 'olarak', 'yaygın', 'olurlar', 'tanımlanabilecek', 'en', 'köşe', 'isimlerinden', 'öncesine', 'açılmaya', 'başladı', 've', 'altyapı', 'bu', 'iddiaların', 'fotoğrafları', 'için', 'davet', 'olarak', 'da', 'bildirilen', 've', 'kullanıcılara', 'yazının', 'için', 'değerlendirilmişti', 'analitti', 'videodaki', 'sıklıkla', 'bağlantılı', 'olarak',	'salmanya'ya', 'şişen', 'kurumu'na', 'holis', 'tra'
Dikkat Alg.	300	0.3	'Öncelikle', 'her', 'üniversite', 'gibi', 'kimyasal', 'gelişinden', 'yola', 'çıkmasın', 'mümkün', 'değil', 'kar', 'ya', 'da', 'arttığımızda', 'araştırma', 'hakkı', 'etme', 'bulunsa', 'da', 'için', 'bir', 'yolcu', 'olan', 'var', 've', 'uygulanacağını', 'açıklamıştı', 'kadın', 'iddia', 'edildi', 'dün', 'ileri', 'geçişimizde', 'yalan', 'söylüyor', 'sanatçısını', 'paylaşmıştı', 'tanıdığını', 'söyledi', 'biliyorlar', 'yok', 'geliştirmek', 'için', 'gerekli', 'araçlarda', 'olan', 'komşulardan', 'birinin', 'kaynağı', 'deneyle', 'intiharlara', 'göre', 'az', 'artmış', 'değil', 'gitmeyi', 'koşmalarının', 'sunulan', 'cesetlendiği'	'kumacak', 'mariupol', 'metabei', 'devr', 'mebleğine', 'farukları'
Dikkat Alg.	300	0.4	'Öncelikle', 'her', 'üniversite', 'kaldı', 'etkileşimin', 'aktif', 'olduğu', 'kalsa', 'birimin', 'üzerinde', 'olduğu', 'rivayeti', 'olarak', 'inanılan', 've', 'hatalı', 'insanlar', 'yeniden', 'paylaşılsın', 'durumun', 'ardından', 'hale', 'gelen', 'oluşmuş', 'romanı', 'dair', 'diyordu', 'ancak', 'bildirildiğinde', 'o', 'fotoğraf', 'bir', 'film', 'sistemi', 'hakkında', 'konut', 'sağlamak', 'için', 'özel', 'almak', 'sayılan', 'bir', 'halde', 'görüntülenen', 'yarış', 've', 'hakkında', 'saçma', 'ilhanları', 'kapsayan', 'anlamına', 'gelmiyor', 'ancak', 'bir', 'uçak', 'ağır', 'süredir', 'kayı', 'içinde', 'bıraktırmış', 'haber', 'ya da'	'kutlusal', 'harbiyatında', 'kolo', 'eriisi', 'makupo'da', 'aşkam', 'cumhu'
Dikkat Alg.	300	0.7	'Öncelikle', 'her', 'üniversite', 'hatalı', 'ile', 'aktarıyoruz', 'sosyal', 'medyada', 'gündeme', 'geldi', 'diye', 'kamu', 'yoksun', 'kurumu'na', 'dair', 'olduğu', 've', 'bun', 'iç', 'yeşil', 'iddiaları', 'olan', 'bir', 'hat', 'üye', 'ülke', 've', 'araç', 'satışlarını', 'bile', 'yazdırılmış', 've', 'youtube', 'kanalına', 'uygulanacağı', 'röportajların', 'olduğu', 'iddia', 'çeşitli', 'herhangi', 'bir', 'yoldan', 'sonra', 'yapılmışsa', 'daha', 'fazla', 'kanunda', 'olarak', 'idare', 've', 'ukrayna'ya', 'devam', 'edildi', 've', 'hayattaki', 'haberini', 'tarafalarına', 'göre', 'yanlış', 'bilgi', 'paylaşıldı', 'aşağıdaki', 'görsel', 'yapan', 'ipuçla'	'icraril', 'edildi illicler', 'dörüşünün', 'görevsenin', 'yayını', 'disenere',

Tablo 4.9: Kodlayıcı Kod Çözücü ve Dikkat Alg. Modelinin 500 Epoch Sonuçları.

Model	Epoch	Sıcak	Anlamlı Kelimeler	Anlamsız Kelimeler
Dikkat Alg.	500	0.2	'Öncelikle', 'her', 'üniversiteler', 'için', 'da', 'nice', 'yüzde', 'olabilecek', 'bir', 'yeni', 'dağıtım', 'da', 'geliyor', 'fiyat', 'artışı', 'ile', 'yaşanabilir', 've', 'temelleri', 'olmak', 'üzere', 'yılında', 'kurulan', 've', 'oy', 'veriyor', 'konutu', 'ile', 'bu', 'kadar', 'akraba', 'miktarları', 'ile', 'de', 'yapılan', 'bir', 'çalışmada', 'kaldıklarını', 'gösterdiği', 'iddia', 'edildi', 'danışmanlık', 'artışı', 'hem', 'de', 'daha', 'önceki', 'yılmaz', 'okuya', 've', 'sayılı', 'kanuna', 'korunması', 've', 'yaşamak', 'önerilen', 'ekim', 'tarihinde', 'etçildi', 'görülüyor', 'yanı', 'sıra', 'düşünmeden'	'yazılırılır', 'temalet', 'tayıl'
Dikkat Alg.	500	0.3	'Öncelikle', 'her', 'üniversiteler', 'için', 'da', 'nice', 'yüzde', 'enin', 'alınarak', 'teşvik', 'böyle', 'bir', 'ifade', 'kullanıcı', 'tarafından', 'yürütülen', 'herkesin', 'gazetesinde', 'yer', 'alan', 'isimlerle', 'ilgili', 'internette', 'arama', 'yaptığımızda', 'ise', 'oluşturulabileceğini', 'gösterdiği', 'iddia', 'edildi', 've', 'maske', 'etmek', 'için', 'dikkat', 'ettiği', 'gelir', 'vergisinde', 'kullanıcı', 'da', 'tersi', 'görmektedir', 'her', 'alan', 'bu', 'iyi', 'değil', 'öteye', 'veriyor', 've', 'onlardan', 'mehmet', 'sahnesi', 'artan', 'yeni', 'bir', 'virüs', 'numarası', 'ile', 'anlatılıyor', 'olabilir', 'resmi', 'ilan'	'yazılırılır', 'metabında', 'anelinde', 'semonal', 'kırırladığını'
Dikkat Alg.	500	0.5	'Öncelikle', 'her', 'üniversiteler', 'için', 'denetlemeler', 'için', 'yer', 'alan', 'pankartlardan', 'hemen', 'piyasadan', 'yararlanabildikleri', 'anlamına', 'gelmiyor', 'ayrıca', 'anlamını', 'satmaya', 'çalışıyor', 've', 'telefonun', 'arasında', 'gerçek', 'hisler', 'verilmediği', 've', 'ilaç', 'sanılarak', 'bu', 'yönde', 'bir', 'ayrılık', 'oranı', 'yüzde', 'ile', 'kilovat', 'yapmış', 'kitli', 'belli', 'şirketine', 'iddiası', 'yanıltıcı', 'yönü', 'Öyküm', 'keskin', 'Çatışma', 'milyon', 'dolar', 've', 'biden', 'hava',	'yazılırılır', 'sokakip', 'civaba', 'takladığı', 'hüma', 'dezektirilios'
Dikkat Alg.	500	0.7	'Öncelikle', 'her', 'üniversitelerinden', 'biri', 'de', 'anlamında', 'çalışmalardan', 'başladığını', 'geçen', 'bir', 'üye', 'yapılmadı', 'deniyor', 'tekliflerinin', 'dergisine', 'bir', 'konumun', 'ne', 'dinlendiğini', 'söylüyor', 'bulunuyor', 'musun', 'restorana', 'gelmediğini', 've', 'çekilen', 'ifade', 'ülke', 've', 'ihraca', 'ithal', 'etmeye', 'başlamaları', 'için', 'aşağıdaki', 'gibi', 'bir', 'tablonun', 'dayanımına', 'dair', 'bir', 'yanlış', 'bilgi', 'yılı', 'aralık', 'de', 'pek', 'bir'	'kaydedilimi', 'söylediv', 'kinskide', 'dilifari'nden', 'papon', 'sahide', 'oluntalara', 'peklende', 'saviye', 'artin', 'teyitçilerin', 'çocuklar',

Kodlayıcı-kod çözücü ve dikkat mimarisi kullanılarak oluşturulan dil modeli 100,300,500 epoch değerlerinde ve sıcaklık örnek alma yönteminin farklı eşik değerlerinde çalıştırıldığında alınan sonuçlar tablo 4.7, 4.8, 4.9,'ta ayrıntılı olarak verilmektedir. Modelin GRU ve LSTM modelleri gibi uzun ve kısa kelimeleri üretebildiği ve diğer modellere göre daha uzun anlamlı kelimeler üretebildiği görülmektedir. (ileri, ile, daha, önce, bir, ya da, iyi, yeni, görüntüleriyle, ilahiyatçıya, görüntüleriyle, kullanıcılarına, sorumluluklarının, yararlanabildikleri karşılaşmıyoruz, oluşturulabileceğini, değerlendirilmişti) Yanlış ürettiği çoğu kelimeleri ise bir kaç karakter ile yanlış olarak üretmektedir. (afp, çocuklar, seviye, sahade, söylediv, civaba, takladığ, aşkam, mebleğine, salmanya'ya, vb.) Modelin oluşturduğu anlamlı kelime gruplarına bakıldığında GRU ve LSTM modellerine göre daha uzun kelime grupları üretmede daha başarılı olduğu görülmektedir. (türkiye'nin toplandığı iddia edilen, sosyal medya platformlarında paylaşılan bazı vakalı bir paylaşım görüntüsü youtube'a yüklenmiş , kararlı bir şekilde uygulama yapmak üzere kullanılan, sosyal medya platformları, devlet başkanı vladimir putin'in, arama sonuçlarında fatih sultan mehmet'in, müslüman toplumunun konuyla ilgili paylaşımlarda yer aldığı anlar oluşmaktadır, youtube kanalına uygulanacağı röportajların olduğu iddia edildi, hayattaki haberin taraflarına göre yanlış bilgi paylaşıldı, gazetesinde yer alan isimlerle ilgili internette arama yaptığımızda)



Şekil 4.3: Karakter Tabanlı Dikkat Sonuç Grafiği.

Şekil 4.3’de Kodlayıcı-kod çözücü ve dikkat mimarisi kullanılarak oluşturulan dil modelinin 100,300,500 epoch değerlerinde ve sıcaklık örnek alma yönteminin farklı eşik değerlerinde çalıştırıldığında üretilen anlamlı kelime asayısı, yanlış üretilen kelime sayısı ve modelin başarı yüzdeleri gösterilmektedir. Model 100 epoch çalıştırıldığında ve 0.2 sıcaklık eşik değerinde 86.9%, 0.3 sıcaklık eşik değerinde 89.3%, 0.5 sıcaklık eşik değerinde 90.7% ve 0.7 sıcaklık eşik değerinde 81.9% değerinde sonuçlar vermektedir. Model 100 epoch sonucunda 0.5 sıcaklık eşik değerinde 90.7% değeri ile en başarılı sonucu vermektedir. Model 300 epoch çalıştırıldığında ve 0.2 sıcaklık eşik değerinde 90.9%, 0.3 sıcaklık eşik değerinde 89.3%, 0.5 sıcaklık eşik değerinde 89.8% ve 0.7 sıcaklık eşik değerinde 91.5% değerinde sonuçlar vermektedir. Model 300 epoch sonucunda 0.7 sıcaklık eşik değerinde 91.5% değeri ile en başarılı sonucu verdiği görülmektedir. LSTM modeli 500 epoch çalıştırıldığında ve 0.2 sıcaklık eşik değerinde 95.7%, 0.3 sıcaklık eşik değerinde 92.6%, 0.5 sıcaklık eşik değerinde 90.4% ve 0.7 sıcaklık eşik değerinde 79.6% değerinde sonuçlar vermektedir. Modelin 500 epoch sonucunda 0.2 sıcaklık eşik değerinde 95.7% değeri ile en başarılı sonucu verdiği görülmektedir. Modelin farklı epoch değerlerinde ve sıcaklık örnek alma algoritmasının farklı eşik değerlerinde alınan sonuçları değerlendirildiğinde modelin en başarılı sonucu 500 epoch değerinde ve örnek alma yönteminin 0.2 eşik değerinde 95.7% ile verdiği görülmektedir. Diğer epoch ve sıcaklık değerlerinde başarının yaklaşık 79.6% - 92.6% aralığında değiştiği görülmektedir. Kodlayıcı-kod çözücü ve dikkat mimarisi kullanılarak oluşturulan dil modelinin anlamlı kelime üretiminde ve daha uzun kelime gruplarının oluşturulmasında diğer modellere (LSTM,GRU) göre daha yüksek başarı elde edildiği görülmektedir.

5. TARTIŞMA VE SONUÇ

Bu çalışmada karakter tabanlı kelime üretimi uygulaması için derin yapay sinir ağlarından LSTM, GRU ve Kodlayıcı-kod çözücü ile birlikte dikkat mimarisi kullanılarak bir dil modeli geliştirilmekte ve verilen metin parçasına bağlı olarak kelimeler üretilmektedir. Çalışmada bu modellerin tercih edilmesindeki neden ise metin üretiminde öğrenmenin olabilmesi için daha önceki bilgilerin hatırlanarak sonraki bilgilerin doğru tahmin edilmesinde kullanılması zorunluluğudur.

LSTM ve GRU modelleri kullanılarak oluşturulan karakter tabanlı dil modellerinin sonuçları incelendiğinde modeller 100,300,500 epoch değerlerinde ve sıcaklık örnek alma yönteminin farklı eşik değerlerinde çalıştırılmaktadır. Alınan sonuçlar incelendiğinde anlamlı kelime üretiminde LSTM ile oluşturulan dil modelinin en başarılı sonucu 100 epoch değerinde ve örnek alma yönteminin 0.5 eşik değerinde 90% ile verdiği görülmektedir. Diğer epoch ve sıcaklık değerlerinde başarının yaklaşık 68.1% - 86.9% aralığında değişmektedir. GRU ile oluşturulan dil modelinin sonuçları değerlendirildiğinde ise modelin en başarılı sonucu 300 epoch değerinde ve örnek alma yönteminin 0.5 eşik değerinde 93.9% ile en yüksek sonucu vermektedir. Diğer epoch ve sıcaklık değerlerinde başarı yaklaşık 73.3% - 90.9% aralığında değişmektedir. LSTM ve GRU modellerinin bu sonuçları incelendiğinde sıcaklık örnek alma yönteminin 0.2, 0.3, 0.5 eşik değerleri için daha başarılı sonuçlar verdiği, eşik değeri arttığında ise rastgeleliğin artarak anlamlı kelime üretiminde başarılı sonuçlar vermediği görülmektedir. LSTM ile oluşturulan modelin 300 ve 500 epoch değerlerinde başarısının artması beklenirken, başarısının düşme nedeni olarak veri seti kalitesinin ve veri ön işleme adımlarının yetersiz ve veri seti boyutunun yeterli büyüklükte olmamasından kaynaklandığı düşünülmektedir. GRU modeli ile oluşturulan dil modelinde 300 ve 500 epoch değerlerinde elde edilen başarının LSTM modeli ile elde edilen sonuçlar kadar düşmesmesinin nedeni olarak ise Gru modellerinin küçük veri setlerinde daha başarılı sonuçlar vermesinden kaynaklandığı düşünülmektedir.

Kodlayıcı-kod çözücü ve dikkat mimarisi kullanılarak oluşturulan karakter tabanlı dil modellerinin sonuçları incelendiğinde modelin 100,300,500 epoch değerlerinde ve sıcaklık örnek alma yönteminin farklı eşik değerlerinde anlamlı kelime üretiminde başarının LSTM ve GRU modellerine göre arttığı görülmektedir. Model en başarılı sonucu 500 epoch değerinde ve örnek alma yönteminin 0.2 eşik değerinde 95.7% ile en yüksek başarıyı elde

ettiği görülmektedir. Diğer epoch ve sıcaklık değerlerinde başarının yaklaşık 79.6% - 92.6% aralığında değiştiği görülmektedir. Dikkat mimarisinin çalışma prensibi düşünüldüğünde uzun dizilerde girdi ve çıktı değerleri arasındaki ilişkiler daha iyi hatırlayabildiği için bu modelde üretilen anlamlı kelime, kelime grubu ve modelin başarı yüzdelerinin oldukça tatmin edici olduğu düşünülmektedir.

Yapay zeka uygulamalarının başarısının veri setinin kalitesi ve büyüklüğü ile doğru orantılı olduğu düşünüldüğünde ve Türkçe dilinin kök, gövde ve ek yapısının diğer dillere oranla daha karmaşık olması sonuçların belirli başarı seviyesinde kalmasına neden olmaktadır. Kullanılan dil modelinin sonuçlarının iyileştirmek için ise daha geniş ve kaliteli bir eğitim veri seti kullanmak, LSTM birim sayısını arttırmak, eğitim iterasyon (epoch) sayısını arttırmak ve yüksek boyutlu kelime vektörleri ile temsil etmek gibi iyileştirmelerin modelin başarısını attırabileceği düşünülmektedir.

REFERANSLAR

- [1] R. Kibble, "Introduction to natural language processing" University of London, 2013.
- [2] N. Indurkha, & F. J. Damerau, "Handbook of natural language processing", *Chapman and Hall/CRC.*, 2010.
- [3] D. Khurana, A. Koli, K. Khatter , & S. Singh, "Natural language processing: state of the art, current trends and challenges" *Multimedia Tools and Applications*, 1 - 32., 2022.
- [4] M. Agarwal, "An Overview of Natural Language Processing" *International Journal for Research in Applied Science and Engineering Technology*. (IJRASET), 7, 2811-2813, 2019
- [5] D. Nadeau, S. Sekine, "A Survey of Named Entity Recognition and Classification" *Linguisticae Investigationes*. 30(1):3-26., 2007.
- [6] D. A. Dahl, "Natural language processing: past, present and future" In *Mobile speech and advanced natural language solutions*, Springer, New York, NY., pp. 49-73, 2013.
- [7] J. Li, T. Tang, W. X. Zhao, & J.-r. Wen, "Pretrained Language Models for Text Generation: A Survey" *IJCAI.*, 2021.
- [8] A. N. Shewalkar, "Comparison of rnn, lstm and gru on speech recognition data", 2018.
- [9] S. Kostadinov. "Understanding Encoder-Decoder Sequence to Sequence Model", 2019. [towardsdatascience.com:https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346](https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346)
- [10] Z. Wang, X. Su and Z. Ding "Long-Term Traffic Prediction Based on LSTM Encoder-Decoder Architecture," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6561-6571, Oct., 2021.
- [11] S. Chaudhari V. Mithal, G. Polatkan, , & R. Ramanath, "An attentive survey of attention models" *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(5), 1-32., 2021.

- [12] Poulos, J., & Valle, R., Character-based handwritten text transcription with attention networks, *Neural Computing and Applications*, vol. 33, no.16, pp.10563-10573, 2021.
- [13] A. Eriguchi, K. Hashimoto, & Y. Tsuruoka, Character-based decoding in tree-to-sequence attention-based neural machine translation, In *Proceedings of the 3rd Workshop on Asian Translation (WAT 2016)*, pp. 175-183, 2016.
- [14] Y. Feng, C. Hu, H. Kamigaito, H. Takamura, & M. Okumura, Improving Character-Aware Neural Language Model by Warming Up Character Encoder under Skip-gram Architecture, In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pp. 421-427, 2021.
- [15] A. Renduchintala, P. Shapiro, K. Duh, and P. Koehn, Character-aware decoder for translation into morphologically rich languages, In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pp. 244–255, 2019.
- [16] Z. Yang, W. Chen, F. Wang, & B. Xu, A character-aware encoder for neural machine translation, In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, Osaka-Japan, pp. 3063-3070, 11-17 December, 2016.
- [17] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, & Y. Bengio, End-to-end attention-based large vocabulary speech recognition, In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, Shanghai-China, pp. 4945-4949, 20-25 March, 2016.
- [18] Z. Meng, Y. Gaur, J. Li, & Y. Gong, Character-Aware Attention-Based End-to-End Speech Recognition. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Singapore, pp. 949-955, 14-18 December, 2019.
- [19] A.R. Johansen, J.M. Hansen, E.K. Obeid, C.K. Sønderby, & O. Winther, Neural Machine Translation with Characters and Hierarchical Encoding, Technical University of Denmark (DTU), 2800 Lyngby, Denmark Bioinformatics Centre, Department of Biology, University of Copenhagen, Copenhagen, Denmark ArXiv, abs/1610.06550, 2016.
- [20] B.J. Copeland, The Turing Test*. *Minds and Machines* 10, 519–539 , 2000.

- [21] Bildirici, F. *Turing testi nedir? Yapay Zeka Eğitim Serisi — Bölüm 3*, (2018, 4 1).
<https://medium.com/@fatihbildirici.dev/yapay-zeka-e%C4%9Fitim-serisi-b%C3%B6l%C3%BCm-3-69884059e2c1> adresinden alındı
- [22] B.A. Buchanan, (Very) Brief History of Artificial Intelligence. *Ai Magazine*, 26(4), 53-53, 2005.
- [23] R. Yampolskiy, Turing Test as a Defining Feature of AI-Completeness. *Artificial Intelligence, Evolutionary Computing and Metaheuristics.*, 2013
- [24] M. A. Ali, A Brief Introduction on Artificial Intelligence. *International Journal of Advance Research, Ideas and Innovations in Technology*, 6, 254-257., 2020.
- [25] B. Goertzel, Artificial general intelligence: concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 5(1), 1., 2014
- [26] L. Lu, X. Zhang, K. Cho, & S. Renals, *A Study of the Recurrent Neural Network Encoder-Decoder*. *Centre for Speech Technology Research.*, 2015.
- [27] M. Jordan, & M. Thomas, Machine learning: Trends, perspectives, and prospects. *Science* 349, 255 - 260., 2015.
- [28] A. Blum, & P. Langley, Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2), 245-271., 1997.
- [29] T. O. Ayodele, Machine Learning Overview. *New Advances in Machine Learning*, 2, 9-18., 2010.
- [30] T. Meng, X. Jing, & Z. A. Yan, survey on machine learning for data fusion. *Information Fusion*, 57, 115-129., 2020.
- [31] P. Cunningham, M. Cord, & S. Delany, Supervised Learning. In: Cord, M., Cunningham, P. (eds) *Machine Learning Techniques for Multimedia. Cognitive Technologies.*, 2008
- [32] Z.H. Zhou, & M. Li, Semi-supervised learning by disagreement. *Knowledge and Information Systems*(24), 415-439. doi:<https://doi.org/10.1007/s10115-009-0209-z>., 2010.

- [33] J. Novaković, A. Veljović, S. Ilić, Ž. Papić, & T. Milica, Evaluation of Classification Models in Machine Learning. *Theory and Applications of Mathematics & Computer Science*, 7(1), 39-46. 2017.
- [34] M. Mohri, A. Rostamizadeh, & A. Talwalkar, *Foundations of machine learning*. MIT press., 2018.
- [35] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, & K. J. Kim, A survey of deep learning-based network anomaly detection. *Cluster Computing*, 22(1), 949- 961., 2019
- [36] M. A. Rao, & J. Srinivas *Neural networks: algorithms and applications*. Alpha Science., 2003.
- [37] D. A. Dahl, Natural language processing: past, present and future. In *Mobile speech and advanced natural language solutions* (pp. 49-73). Springer, New York, NY., 2013.
- [38] P. Harrington, *Machine learning in action*. Simon and Schuster., 2012.
- [39] C. Olivier, B. Schölkopf, & Z. Alexander, *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*, MIT Press, Cambridge., 2006.
- [40] I. H. Sarker, *Machine learning: Algorithms, real-world applications and research directions*. SN Computer Science, 2(3), 1-21., 2021.
- [41] V. S. Dave, & K. Dutta, Neural network based models for software effort estimation: a review. *Artificial Intelligence Review*, 42(2), 295-307., 2014.
- [42] M. Jahnavi, *Introduction to Neural Networks, Advantages and Applications, towards Data Science* . 2017.
- [43] E. Öztemel, *Yapay Sinir Ağları*, İstanbul: Papatya Yayıncılık, 2006.
- [44] A. Kilani, A.B. Hamida, & H. Hamam, *Artificial Intelligence Review, Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation, and Human-Computer Interaction*, (2019).
- [45] S. Haykin, & N. A. Network, *comprehensive foundation*. Neural networks, 2,41., 2004.

- [46] B. Çelik, YSA Metodolojisi ile Zaman Serisi Analizi: Teori ve Uygulama. Marmara Üniversitesi Yüksek Lisans Tezi. İstanbul, 2008.
- [47] E. Davalo, & P. Naım, Multi-layer Neural Networks. In Neural Networks (pp. 36-62). Macmillan Education UK. 1991.
- [48] J. Lederer, Activation Functions in Artificial Neural Networks: A Systematic Overview. ArXiv, abs/2101.09957., 2021.
- [49] Y. LeCun, Y. Bengio, & G. Hinton, Deep learning. Nature, 521(7553), 436 - 444., 2015.
- [50] P. S. Rajpal, K. S. Shishodia, & G. S. Sekhon, An artificial neural network for modeling reliability, availability and maintainability of a repairable system. Reliability Engineering & System Safety, 91(7), 809-819. 2006.
- [51] M. A. Rao, & J. Srinivas, Neural networks: algorithms and applications. Alpha Science., 2003.
- [52] S. V. Kartalopoulos, & S. V. Kartakapoulos, Understanding neural networks and fuzzy logic: basic concepts and applications. Wiley-IEEE Press. 1997.
- [53] B. Çelik, YSA Metodolojisi ile Zaman Serisi Analizi: Teori ve Uygulama. Marmara Üniversitesi Yüksek Lisans Tezi. İstanbul, 2008.
- [54] L. Vanneschi, & M. Castelli, Delta Rule and Backpropagation. Encyclopedia of Bioinformatics and Computational Biology. 2019.
- [55] Z. Lo, Y. Yu, & B. Bavarian, Two theorems for the Kohonen mapping neural network. [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, 4, 755-760 vol.4. 1992.
- [56] Y. Chen, F. Li, J. Wang, B. Tang, & X. Zhou, Quantum recurrent encoder–decoder neural network for performance trend prediction of rotating machinery. Knowledge-Based Systems, 197, 105863., 2021.
- [57] N. B. Albayrak, Opinion and sentiment analysis using natural language processing techniques (Doctoral dissertation, Master’s thesis, Fatih University). 2011

- [58] A. Porshnev, & I. Redkin, Analysis of Images, Social Networks and Texts. Communications in Computer and Information Science, 436(April), 190–197. 2014.
- [59] D. Khurana, A. Koli, K. Khatter, & S. Singh, Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications, 1 - 32., 2022.
- [60] D. Nadeau, S. Sekine, A Survey of Named Entity Recognition and Classification. Linguisticae Investigationes. 30(1):3-26., 2007.
- [61] M. Jordan, & M. Thomas, Machine learning: Trends, perspectives, and prospects. Science 349, 255 - 260., 2015.
- [62] A. Hotho, A. Nürnberger, & G. Paaß, A brief survey of text mining. In Ldv forum (Vol. 20, No. 1, pp. 19-62), 2005.
- [63] S. Kannan, V. Gurusamy, S. Vijayarani, J. Ilamathi, M. Nithya, Kannan, S., & V. Gurusamy, Preprocessing techniques for text mining. International Journal of Computer Science & Communication Networks, 5(1), 7-16., 2014.
- [64] J. Nouri, & R. Yangarber, A Novel Evaluation Method for Morphological Segmentation. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), 3102–3109., 2011.
- [65] K. E. Lochbaum, & L. A. Streeter, Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval., 1989
- [66] V. V. Nabiyeu, Yapay Zeka, Seçkin Yayıncılık, Ankara., 2012.
- [67] Ö. Çoban, Metin Sınıflandırma Teknikleri İle Türkçe Twitter Duygu Analizi. Yüksek Lisans Tezi, Erzurum., 2016
- [68] M. F. Amasyalı, S. BALCI, M. E. T. E. Emrah, & E. VARLI, “Türkçe metinlerin sınıflandırılmasında metin temsil yöntemlerinin performans karşılaştırılması”. EMO Bilimsel Dergi, 2(4), 2012.
- [69] S. Yıldırım, & T. Yıldız, A comparative analysis of text classification for Turkish language. Pamukkale Univ Muh Bilim Derg, 24(5), 879-886., 2018.

- [70] E. A. C. J nior, V. Q. Marinho, & L. B. dos Santos, NILC-USP at SemEval-2017 Task 4: A Multi-view Ensemble for Twitter Sentiment Analysis. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) (pp. 611-615), 2017.
- [71] F. Akba, “Duygu Analizinde  znitelik Se me Metriklerinin Değerlendirilmesi: T rk e Film Eleştirileri”. Y ksek Lisans Tezi, Hacettepe  niversitesi / Fen Bilimleri Enstit s  / Bilgisayar M hendisliđi Ana Bilim Dalı. Ankara., 2014.
- [72] H. Sch tze, C. D. Manning, & P. Raghavan, Introduction to information retrieval (Vol. 39). Cambridge University Press Cambridge., 2008.
- [73]  .  oban, Metin Sınıflandırma Teknikleri İle T rk e Twitter Duygu Analizi. Y ksek Lisans Tezi, Erzurum., 2016.
- [74] Y. Ren, R. Wang, & D. Ji, A topic-enhanced word embedding for twitter sentiment classification. Information Sciences, 369, 188-198., 2016.
- [75] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, & V. Stoyanov, SemEval-2016 task 4: Sentiment analysis in Twitter. In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016) (pp. 1-18), 2016.
- [76] S. Reddy, GloVe and fastText — Two Popular Word Vector Models in NLP. Retrieved from blogs.sap.com: <https://blogs.sap.com/2019/07/03/glove-and-fasttext-two-popular-word-vector-models-in-nlp/> adresinden alındı., (2019, 8 3).
- [77] T. Ganegedara, Intuitive Guide to Understanding GloVe Embeddings. towardsdatascience.com: <https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010> adresinden alındı., (2019, 5 5).
- [78] G.A. Shperber, gentle introduction to Doc2Vec. medium.com: <https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e> adresinden alındı., (2017, 8 26).
- [79] E. E. Eryılmaz, T rk e İstenmeyen E-Postaların Derin  ğrenme ile Tespit Edilmesi. Ondokuz Mayıs  niversitesi, Bilgisayar M hendisliđi. Samsun: Ondokuz Mayıs  niversitesi., 2021.

- [80] A. Güneş, Türkçe'de Varlık İsmi Tanıma. İstanbul: İstanbul Teknik Üniversitesi., 2018
- [81] E. Keçeci, Yinelemeli Sinir Ağları İle Finansal Veri Tahmini. Ankara: Hacettepe Üniversitesi., 2020.
- [82] L. Lu, X. Zhang, K. Cho, & S. Renals, (A Study of the Recurrent Neural Network Encoder-Decoder. Centre for Speech Technology Research., 2015.
- [83] Rana, R. Gated recurrent unit (GRU) for emotion classification from noisy speech. arXiv preprint arXiv:1612.07778., 2016.
- [84] L. Lu, X. Zhang, K. Cho, & S. A. Renals, Study of the Recurrent Neural Network Encoder-Decoder. Centre for Speech Technology Research., 2015.
- [85] J. Chung, C. Gulcehre, K. Cho, & Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv: Neural and Evolutionary Computing., 2014.
- [86] P. March, How to generate text: using different decoding methods for language generation with Transformers. huggingface.co: <https://huggingface.co/blog/how-to-generate> adresinden alındı., (2018, 3 18).
- [87] R. Rana, Gated recurrent unit (GRU) for emotion classification from noisy speech. arXiv preprint arXiv:1612.07778., 2016.
- [88] H. Alqahtani, M. Thorne, & G. Kumar, Applications of Generative Adversarial Networks (GANs): An Updated. Archives of Computational Methods in Engineering. 28(2), 525-552 2021.
- [89] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, & A. Bharath, Generative Adversarial Networks: An Overview. IEEE Signal Processing Magazine, 35(1), 53-65., 2018.
- [90] Z. Tavası, W. Yu, X. Yi, A., Han, F. Yuan & Y. Zheng, Recent Progress on Generative Adversarial Networks (GANs): A Survey. IEEE Access., 2019.

- [91] S. Kostadinov, Understanding Encoder-Decoder Sequence to Sequence Model. <https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346> adresinden alındı., (2019, 2 5).
- [92] N. BM, What is an encoder decoder model? [towardsdatascience.com: https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a](https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a) adresinden alındı., (2020, 10 7).
- [93] Y. Chen, F. Li, J. Wang, B. Tang, & X. Zhou, Quantum recurrent encoder–decoder neural network for performance trend prediction of rotating machinery. Knowledge-Based Systems, 197, 105863., 2021.
- [94] S. Chowdary, Understanding Attention mechanism and Machine Translation Using Attention-Based LSTM (Long Short Term Memory) Model. [www.marktechpost.com: https://www.marktechpost.com/2021/04/14/understanding-attention-mechanism-and-machine-translation-using-attention-based-lstm-long-short-term-memory-model/](https://www.marktechpost.com/2021/04/14/understanding-attention-mechanism-and-machine-translation-using-attention-based-lstm-long-short-term-memory-model/) adresinden alındı., (2021, 4 14).
- [95] D. Bahdanau, K. Cho, & Y. Bengio, Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473., 2014.
- [96] S. Sushant, & M. Ausif, The NLP Cookbook: Modern Recipes for. IEEE Access, 68675-68702., 2021.
- [97] J. Wei, ELMo: Why it's one of the biggest advancements in NLP. [https://towardsdatascience.com/: https://towardsdatascience.com/elmo-why-its-one-of-the-biggest-advancements-in-nlp-7911161d44be](https://towardsdatascience.com/elmo-why-its-one-of-the-biggest-advancements-in-nlp-7911161d44be) adresinden alındı., (2020, 10 16).
- [98] J. Devlin, M.-W. Chang, K. Lee, & K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for. Google AI Language., 2018.
- [99] R. Horev, BERT Explained: State of the art language model for NLP. [https://towardsdatascience.com/: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270](https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270) adresinden alındı., (2018, 11 10).

- [100] U. K. Uçar, BERT Modeli ile Türkçe Metinlerde Sınıflandırma Yapmak. <https://medium.com/@toprakucar/bert-modeli-ile-t%C3%BCrk%C3%A7e-metinlerde-s%C4%B1n%C4%B1fland%C4%B1rma-yapmak-260f15a65611> sitesinden alındı., (2020, 28 4).
- [101] R. Dale, GPT-3: What's it good for?. Natural Language Engineering, 27(1), 113-118., 2021.
- [102] L. Floridi, & M Chiriatti, GPT-3: Its nature, scope, limits, and consequences. Minds and Machines, 30(4), 681-694., 2020.
- [103] N. Fatima, A. S. Imran, Z. Kastrati, S. M. Daudpota, A. Soomro, & S. ShaikhA Systematic Literature Review on Text Generation Using Deep Neural Network Models. IEEE Access., 2022.
- [104] M. Karakaya, Sampling in Text Generation. Retrieved from colab.research.google.com: (2021, 3 19).
<https://colab.research.google.com/drive/1yK5LO4jAPb8LHAoQWw0svH6p8QSiDcwC?usp=sharing> adresinden alındı., (2021, 3 19).
- [105] K. Chintapalli, Generative Adversarial Networks for Text Generation . Retrieved from becominghuman.ai: <https://becominghuman.ai/generative-adversarial-networks-for-text-generation-part-1-2b886c8cab10> adresinden alındı., (2019, 3 6)
- [106] Y. Jiang, Augmenting Neural Encoder-Decoder Model for Natural Language Generation Tasks., 2018.
- [107] W. Ou, C. Chen, & J. Ren, An Encoder-Decoder Model for Topic-Based Natural Language Generation. NLDB. (pp. 143-151)., 2018.
- [108] S. Gupta, M. Kaur, & M. Khattar, Application-based Attention Mechanisms in Deep Learning. 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), (pp. 363-366). IEEE., 2020.
- [109] Z. Niu, G. Zhong, & H. YuA review on the attention mechanism of deep learning. Neurocomputing, 452, 48-62., 2021.

- [110] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, & E. P. Xing, Toward controlled generation of text. In International conference on machine learning (pp. 1587-1596). PMLR., 2017.
- [111] J. A. Stokes, guide to language model sampling in AllenNLP. [blog.allenai.org: https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3](https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3) adresinden alındı., (2020, 11 18).
- [112] J. Brownlee , How to Implement a Beam Search Decoder for Natural Language Processing. Retrieved from [machinelearningmastery.com: https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/](https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/) adresinden alındı., (2018, 1 5).
- [113] B. Mann, How to sample from language models. [towardsdatascience.com: https://towardsdatascience.com/how-to-sample-from-language-models-682bceb97277](https://towardsdatascience.com/how-to-sample-from-language-models-682bceb97277) adresinden alındı., (2019, 5 25).