MAKİNE ÇEVİRİSİ İLE TÜRKÇE SÖZEL İFADELERİN PYTHON SÖZDİZİMİNİN OLUŞTURULMASI

Mehmet BOZDEMİR

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÇEVİRİSİ İLE TÜRKÇE SÖZEL İFADELERİN PYTHON SÖZDİZİMİNİN OLUŞTURULMASI

Mehmet BOZDEMİR
0000-0003-1467-6047

Doç. Dr. Metin BİLGİN
(Danışman)

YÜKSEK LİSANS
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2022
Her Hakkı Saklıdır

ÖZET

Yüksek Lisans Tezi

MAKİNE ÇEVİRİSİ İLE TÜRKÇE SÖZEL İFADELERİN PYTHON SÖZDİZİMİNİN OLUŞTURULMASI

Mehmet BOZDEMİR

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Metin BİLGİN

Bu çalışmada, Türkçe sözel ifadelerin, Python programlama dilindeki söz dizimsel ifadesinin otomatik oluşturulması hedeflenmiştir. Yapılan çalışma ile Python kodunun öğrenilmesi kolaylaştırılarak hem eğitim-öğretim ortamlarında kodlama dersi veren öğretmenler için büyük bir kolaylık sağlanması hem de bu dersi alan öğrencilerin hızlı ve kolay bir şekilde kodlama öğrenme imkânı sağlanması hedeflenmiştir. Bu amacın gerçekleştirilebilmesi için yeni bir yaklaşım benimsenmiştir. İlk olarak, Türkçe sözel olarak söylenen cümleler yazıya çevrilmektedir. Yazıya çevirme işlemi, Python dilinde bulunan "speech recognition" kütüphanesi kullanılarak gerçekleştirilmektedir. İkinci olarak, yazıya çevrilen Türkçe cümleler, LSTM, BiLSTM ve GRU gibi derin öğrenme algoritmaları kullanılarak İngilizceye çevrilmektedir. Makine çevirisi yapılırken, Seq2seq mimarisini içerisinde barındıran Encoder-Decoder modeli kullanılmaktadır. Encoder-Decoder modeli, girişte Türkçe olarak aldığı cümleleri, çıkışta İngilizceye çevirmektedir. İngilizceye çevrilen cümleler, daha sonra GPT-3 modeline girdi olarak verilmektedir. GPT3 modeli, kendine gelen İngilizce cümleyi alarak Python kodunu üretmektedir. Çalışmanın son kullanıcılar tarafından da kullanılabilmesi amacıyla "Django" ve "PyQt5" kütüphaneleri kullanılarak hem web tabanlı hem de Windows tabanlı uygulaması geliştirilmiştir. Çalışma sonuçlarının değerlendirilmesinde BLEU, METEOR ve TER metrikleri kullanılmıştır. Çalışma sonucunda, BiLSTM modeli, tüm metriklerde diğer modellere göre daha yüksek skorlar üretmeyi başarmıştır.

Anahtar Kelimeler: LSTM, BiLSTM, GRU, Encoder-Decoder yapısı, GPT-3 modeli, makine çevirisi
2022, viii + 63 sayfa.

ABSTRACT

MSc Thesis

GENERATION OF PHYTON SYNTAX OF TURKISH VERBAL EXPRESSIONS WITH MACHINE TRANSLATION

Mehmet BOZDEMİR

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Assoc.Prof. Metin BİLGİN

In this study, the goal is automatically to create the syntax term in the Python programming language of Turkish verbal expressions is targeted. The study aims to simplify the learning of Python code, easiness for software teachers in educational environments, as well as providing a remarkably swift to learn code for students who take the software course. A new approach has been taken to bring this about. First of all, sentences pronounced in Turkish are translated into text. The transcription is carried out with the help of the "speech recognition" library in Python. Secondly, the Turkish sentences translated into the text are translated into English using deep learning algorithms such as LSTM, BiLSTM and GRU. During machine translation, the Encoder-Decoder template, which includes the Seq2seq architecture, is utilized. The Encoder-Decoder model translates sentences in Turkish at the entrance into English at the exit. The English translated sentences are then transferred as input to the GPT-3 template. The GPT-3 template generates the Python code based on the English text sent to itself. Both the web-based and Windows-based applications have been generated using the libraries “Django” and “PyQt5” to allow the study to be used by end-users. The study results have appraised by BLEU, METEOR and TER metrics. As the study results, the BiLSTM template have succeeded in generating higher ratings on all metrics than others.

Key words: LSTM, BiLSTM, GRU, Encoder-Decoder structure, GPT-3 model, machine translation
2022, viii + 63 pages.

TEŞEKKÜR

Tez yazım süreci boyunca her daim yanımda olan, aldığım kararları her zaman destekleyen, zorluklarla karşılaştığımda beni cesaretlendiren, moral ve desteğini esirgemeyen sevgili eşim Aliye BOZDEMİR'e teşekkür ederim.

Bir yıllık çalışma sürecinde her anlamda yol gösterici olan, olumlu tavrı ile beni cesaretlendiren, bilgi ve birikimi ile her türlü katkıyı sağlayan, kendisi ile çalışmaktan büyük bir onur ve gurur duyduğum değerli danışman hocam Doç. Dr. Metin BİLGİN'e sonsuz teşekkür ederim.

Çalışmalarım boyunca maddi manevi destekleriyle beni hiçbir zaman yalnız bırakmayan aileme de sonsuz teşekkürler ederim.

Bu çalışma FYL-2022-803 kodu ile Bursa Uludağ Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından desteklenmiştir.

Mehmet BOZDEMİR
…/…/…….

İÇİNDEKİLER

Sayfa

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
SİMGELER ve KISALTMALAR DİZİNİ v
ŞEKİLLER DİZİNİ vi
ÇİZELGELER DİZİNİ viii
1. GİRİŞ 1
2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI 3
2.1. Dil Nedir 3
2.2. Dil Türleri Nelerdir 3
2.3. Makine Çevirisi ve Tarihi 5
2.4. Makine Çeviri Yöntemleri 6
2.4.1. Kural tabanlı makine çevirisi 6
2.4.2. İstatistiksel makine çevirisi 7
2.4.3. Sinirsel makine çevirisi 8
2.4.4. Sinirsel makine çevirisinde seq2seq mimarisi ve tarihi 9
2.5. Makine Çevirisinin Uygulama Alanları 11
2.6. Makine Çevirisinin Kullanılabilirliği 11
3. MATERYAL VE YÖNTEM 13
3.1. Özyinelemeli Sinir Ağı 13
3.2. Uzun Kısa Süreli Bellek 16
3.3. Kapı Özyinelemeli Geçitler 18
3.4. Çift Taraflı Uzun Kısa Süreli Bellek 19
3.5. Üretken Ön İşlemeli Dönüştürücü-3 Mimarisi 21
3.6. Konuşmadan Yazıya Çevir 27
3.7. Kullanılan Kütüphaneler 32
3.7.1. Tensorflow 32
3.7.2. Keras 33
3.7.3. Numpy 35
3.7.4. Matplotlib 36
3.7.5. Speech recognition 37
3.7.6. OpenAI 37
3.8. Kullanılan Teknolojiler 39
3.8.1. Anaconda Navigator 39
3.8.2. Django framework 41
3.8.3. PyQt5 framework 44
3.9. Deneysel Çalışmalar 45
3.9.1. Veri kümesi 45
3.9.2. Eğitim süreci 47
3.9.3. Lstm için eğitim süreci 47
3.9.4. Gru için eğitim süreci 53
3.9.5. Bidirectional lstm için eğitim süreci 55
4. BULGULAR 58
5. TARTIŞMA ve SONUÇ 62
KAYNAKLAR 63
ÖZGEÇMİŞ 66

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	Açıklama
m	Referans cümle içinde geçen her bir kelimenin hipotez cümle içerisindeki sayısı
w_t	Hipotez cümle içindeki kelime sayısı
w_r	Referans cümle içindeki kelime sayısı
c	Kaynak ve hedef arasındaki eşleşen sözcük öbeklerinin sayısıdır.

Kısaltmalar	Açıklama
ABD	Amerika Birleşik Devletleri
ALPAC	Auotomatic Language Processing Advisory Committee
API	Application Programming Interface
AR-GE	Araştırma Geliştirme
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short Term Memory
BiRNN	Bidirectional Recurrent Neural Network
BLUE	Bilingual Evaluation Understudy
CPU	Central Processing Unit
CTC Loss	Connectionist Temporal Classification Loss
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
GTP-2	Generative Pre-trained Transformer-2
GTP-3	Generative Pre-trained Transformer-3
GUI	Graphical User Interface
HMM	Hidden Markov Model
HTTP	Hyper Text Transfer Protocol
IBM	International Business Machines
LSTM	Long Short Term Memory
MATLAB	Matrix Labaratory
METEOR	Metric For Evaluation of Translation with Explicit Ordering
MT	Machine Tranlation
RNN	Recurrent Neural Network
Seq2seq	Sequence to Sequence
STFT	Short-Time Fourier Transform
TER	Translation Error Rate
URL	Uniform Resource Locator

ŞEKİLLER DİZİNİ

		Sayfa
Şekil 2.1.	MT'nin Gelişim Grafiği (Abdul et al., 2021)……	9
Şekil 2.2.	Seq2seq Model Kullanım Alanları (Keneshloo et al., 2020'den değiştirilerek alınmıştır)…….	10
Şekil 3.1.	RNN'nin İç Yapısı (Onyedi et al., 2020)……	13
Şekil 3.2.	RNN Modeli (Zivkovic Strahinja, 2020)……...	14
Şekil 3.3.	RNN'nin Kullanım Alanları (ÇELİK & ODABAS, 2020)…….....	15
Şekil 3.4.	RNN Geri Yayılım (Chen, 2016)……	16
Şekil 3.5.	LSTM Hücre Örneği (Terzi, 2021'den değiştirilerek alınmıştır)……	17
Şekil 3.6.	Encoder-Decoder Yapısı ……...……	18
Şekil 3.7.	GRU Hücre Örneği (Zivkovic Strahinja, 2020b)……	19
Şekil 3.8.	BiLSTM Network Yapısı-1 (Moharm et al., 2020)……	20
Şekil 3.9.	BiLSTM Network Yapısı-2 (Wu et al., 2021)……	20
Şekil 3.10.	Transformer Network Yapısı (Vaswani et al., 2017)……..	22
Şekil 3.11.	Kelime Benzerliklerinin Vektörel Gösterimi …….	23
Şekil 3.12.	Embeding ve Position Encoding Vektörlerinin Birleştirilmesi……..	24
Şekil 3.13.	Multi-Head Attention Linear Dense Matris Çıktısı ……...	25
Şekil 3.14.	Q, K ve V Vektörlerinin Elde Edilmesi……..	26
Şekil 3.15.	Denklem 3.14'e göre QKT İşleminin Gerçekleştirilmesi……...	26
Şekil 3.16.	Ses Frekansı……...	28
Şekil 3.17.	Short-time Fourier Transform (STFT) Dönüşümü (Jeon et al., 2020)	29
Şekil 3.18.	Spectrogram……...	30
Şekil 3.19.	Ses Sinyallerinde RNN Modelinin Yapısı ve Gösterimi (Hannun et al., 2014)……	30
Şekil 3.20.	Tensorflow Kütüphanesini İmport Etme……	32
Şekil 3.21.	Tensorflow Kütüphanesinin Model Metodunu Kullanma……..	32
Şekil 3.22.	Eğitim İçin Model Oluşturma…….	33
Şekil 3.23.	Fonksiyonel API Kullanımı……...	34
Şekil 3.24.	Sequential API Kullanımı……..	34
Şekil 3.25.	Numpy Olmadan Matris Çarpımı……...	35
Şekil 3.26.	Numpy Kullanılarak Matris Çarpımı…….	35
Şekil 3.27.	Matplotlib ile Oluşturulmuş Model Doğruluk Grafiği……	36
Şekil 3.28.	Konuşma Tanıma Kütüphanesinin Kullanımı……	37
Şekil 3.29.	OpenAI Kütüphanesi Kullanımı……	39
Şekil 3.30.	Anaconda Navigator……..	40
Şekil 3.31.	Anaconda Navigator Yüklü Paketler……..	41
Şekil 3.32.	Django Framework İç Yapısı (Django Introduction - Learn Web Development | MDN, n.d.)…….	42
Şekil 3.33.	Django Framwork'ü Kullanarak Oluşturulan Arayüz……	43
Şekil 3.34.	PyQt5 Kullanılarak Oluşturulan Ara yüz……...	44
Şekil 3.35.	Veri Kümesi Kelime Sayı Dağılımları……...	46
Şekil 3.36.	Encoder-Decoder Model İç Yapısı (Szucs & Huszti, 2019)……...	49
Şekil 3.37.	Encoder-Decoder Modelinin Uçtan Uca Eğitilmesi……...	50
Şekil 3.38.	Encoder-Decoder Modeli İçin Hiper Parametre Ayarı……...	50
Şekil 3.39.	LSTM Model Parametre Sayısı……..	51
Şekil 3.40.	LSTM Eğitim Doğruluğu Grafiği……...	52

Şekil 3.41. LSTM Kayıp Fonksiyonu Grafiği…….. 52
Şekil 3.42. GRU Hücresi İç Yapısı (Qian et al., 2021)……. 53
Şekil 3.43. GRU Model Parametre Sayısı…… 54
Şekil 3.44. GRU Eğitim Doğruluğu Grafiği……. 55
Şekil 3.45. GRU Kayıp Fonksiyonu Grafiği…… 55
Şekil 3.46. BiLSTM Model Parametre Sayısı…….. 56
Şekil 3.47. BiLSTM Eğitim Doğruluğu Grafiği……... 57
Şekil 3.48. BiLSTM Kayıp Fonksiyonu Grafiği…….. 57

ÇİZELGELER DİZİNİ

		Sayfa
Çizelge 3.1.	Kelime Benzerlikleri……	24
Çizelge 3.2.	Phonome Veri Kümesi (The CMU Pronouncing Dictionary, n.d.)..	31
Çizelge 3.3.	Veri Kümesi…….	46
Çizelge 3.4.	LSTM İçin N-Gram BLUE Skorları……	47
Çizelge 4.1.	BLUE Skor……..	61
Çizelge 4.2.	METEOR Skor……	61
Çizelge 4.3.	TER Skor…….	61

1. GİRİŞ

Python programlama dili, dünyada en çok kullanılan programlama dillerinden biridir. ABD'de son dört yılın işverenlerince en çok tercih edilen programlama dilleri arasında zirvede yer almaktadır (*En Popüler Programlama Dilleri • Turhost Blog*, n.d.). Ayrıca yapay zekâ alanındaki gelişmelere paralel olarak, bu dilin popülaritesi her geçen gün artmaktadır. Dilin iyi bir şekilde bilinmesi, eğitim-öğretim ortamlarında Python programlama dersi veren öğretmenler için de önem arz etmektedir. Python programlama dili, diğer programlama dillerine kıyasla daha basit bir yapıya ve daha kolay anlaşılır bir söz dizimine (syntax) sahiptir. Python kodunu yazarken söz dizimini, dile tanımlanmış şekilde eksiksiz olarak yazmak gerekmektedir. Diğer programlama dillerine oranla daha kolay bir söz dizimi yapısı olmasına rağmen dilin mevcut fonksiyonları, döngü yapıları, karar blokları gibi birçok yapı bazen hatırlanamayabilir. Örneğin, bir fonksiyon oluşturmak istenildiğinde "def topla()" şeklinde bir söz dizimi yazılması gerekmektedir. Burada yazılımcı, "topla()" şeklinde bir ifade yazarak Python sentaksını (syntax) başka programlama dillerinin söz dizimi ile karıştırabilmektedir. Programlama dilleri seviyelerine göre yüksek ve düşük olarak ikiye ayrılmaktadır. Kullanıcının ana diline en yakın diller yüksek seviyeli dillerdir. Bu yönü itibari ile Python yüksek seviyeli bir programlama dili olarak ifade edilir (Bogdanchikov et al., 2013a). Bunlardan bazıları C#, C++, Delphi, Java, JavaScript, Python, Go, R'dır. Günümüzde "Full Stack" çalışan geliştiricilerin birden fazla programlama diline vâkıf olmalarının bir zorunluluk haline gelmiştir. Benzer şekilde, eğitim kurumlarında programlama dersi veren öğreticilerin de aynı anda birden fazla programlama dilini öğretebilmeleri gerekmektedir. Aslında bütün programlama dilleri, algoritma mantığı açısından birbirlerine benzemektedirler. Aralarındaki temel farklılık, söz dizimi yapılarındaki değişikliklerdir. Programcı, inşa edeceği algoritmayı zihninde tasarlamış ve hatta ilgili bileşenlerini kurgulamış olsa dahi farklı bir programlama diline geçtiğinde o dile alışması zaman almaktadır. Yapay zekanın bir alt dalı olan doğal dil işleme alanında, son yıllarda büyük bir gelişme yaşanmıştır. OPENAI firması tarafından geliştirilen GPT-3 modeli sayesinde, kod yazmak başta olmak üzere birçok farklı alanda ilerlemeler kaydedilmiştir. Kod yazma işi, klavye aracılığıyla yazılan kelimelere ihtiyaç duyulmadan, ağızdan çıkan konuşma cümleleri ile mümkün hale getirilmiştir. Bu ilerlemenin ardındaki en büyük etken, daha yüksek

performans sergileyen bilgisayarların geliştirilmesidir. Bunun yanında, derin öğrenme alanında kullanılan algoritmalarda yaşanan değişiklikler de bu gelişime katkı sunmuştur. Derin öğrenme, ilk olarak 1943'te, Warren McCulloch ve Walter Pitts tarafından sinir ağlarını çoğaltan bir bilgi işletim sistemi kurmak için matematik algoritmalarının kullanılmasıyla ortaya çıkmıştır (Mcculloch & Pitts, 1990).Daha sonraki yıllarda bu alanda birçok gelişme yaşanmış ama teoriden öteye gidememiştir. Çünkü derin öğrenme üzerinde hesaplama yapabilmek için çok sayıda veri ve bu verileri işleyebilecek donanıma sahip hızlı bilgisayar sistemlerine ihtiyaç vardır. Bu gibi teknolojik eksikliklerden ötürü, derin öğrenme alanında uzunca bir süre dikkate değer bir gelişme yaşanmamıştır. Bu alandaki ilk kayda değer gelişme, 2006 yılında Terim, Geoffrey Hinton ve Ruslan Salakhutdinov tarafından, çok katmanlı bir sinir ağının nasıl eğitilebileceğinin açıklandığı makalenin yayımlanmasıdır(Hinton & Salakhutdinov, 2006). Özellikle 1990'lardan sonra bilgisayar sistemlerinin performans artırımını sağlayan teknolojik gelişmeler, derin öğrenme alanında bir dönüm noktası olmuştur. 2005 yılından sonra Google, eski tip makine çeviri yöntemlerini terk ederek derin öğrenme tabanlı çeviri sistemine geçiş yapmıştır. Yaşanan değişim ve gelişimlerle birlikte, nispeten insan gibi görebilen, konuşabilen ve hareket edebilen sistemler geliştirilmiştir.

Sunulan bu çalışmada, derin öğrenme alanındaki gelişim paralelinde bir seyir izlenmektedir. Python dilinin öğretiminde yaşanan problemlerin, bu çalışma ile ortadan kaldırılması amaçlanmaktadır. Çalışmanın sonunda, yapılması istenilen bir komut, son kullanıcılar tarafından sözlü olarak modele söylenmekte ve model, sözlü cümlenin Pythondaki komut karşılığını döndürmektedir. Bu çalışma, yukarıda sayılan kodun hatırlanmasının zorluğu, çabuk unutulması gibi problemlere yönelik bir çözüm sunmakta ve geliştirmektedir. Bu çalışma, eğitim-öğretim hizmetlerinde bulunan eğitici ve öğrenciler için de son derece kullanışlı bir araç haline gelecektir.

2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

2.1. Dil Nedir

Dil, insanların birbiriyle iletişim kurabilmesi için kullandığı bir araçtır. Bu araç sayesinde duygu ve düşünceler karşı tarafa kolaylıkla aktarılabilir. İnsanların hayatlarını idame ettirebilmesi için dile ihtiyaçları vardır. Toplum tarafından oluşturulan yazılı ve yazısız kuralların anlaşılabilmesi için de dile ihtiyaç vardır. Devletleri ve milletleri ayakta tutan yine dildir. Dil, sürekli kendini yenileyen ve değiştiren bir araçtır. Örneğin herhangi bir dil incelendiğinde, yüzyıl önce kullanılan kelime, cümlecik, cümle ve söz öbeklerinin, bugünkü dil yapılarından farklılıklar arz ettiği ortaya çıkacaktır. Zaman geçtikçe yeni buluşların, yeni sistemlerin ortaya çıkmasıyla bunları karşılayacak yeni kelimelerin bulunması gerekecektir. Bu sayede dil, değişikliğe uğramış olacaktır. Bundan yüzyıl önce bilgisayar kavramının olmadığı, ama günümüzde bu kavramın sıklıkla kullanıldığı bilinmektedir. Bu yönü itibari ile de dile canlı bir varlık denilmektedir. Türk dili uzmanı Türkolog Prof. Dr. Muharrem Ergin'e (1977) göre dil, "*Dil, insanlar arasında anlaşmayı sağlayan tabiî bir vasıta, kendisine mahsus kanunları olan ve ancak bu kanunlar çerçevesinde gelişen canlı bir varlık, temeli bilinmeyen zamanlarda atılmış bir gizli antlaşmalar sistemi, seslerden örülmüş içtimaî bir müessesedir.*" şeklinde tanımlanmaktadır. Sonuç itibari ile dil, insanların toplumsal olarak yaşamını sağlıklı bir şekilde yürütebilmesi için gerekli olan bir araçtır.

2.2. Dil Türleri Nelerdir

Diller, kendi içerisinde çok farklı alt başlıklara ayrılmaktadır. Genel olarak bir dilin türleri, ana dil, yazı dili, konuşma dili ve yapay diller olarak; alt birimleri ise lehçe, şive ve ağız olarak gruplandırılabilir.

Ana Dil: İnsanların doğuştan itibaren anne ve babasıyla etkileşime girmek suretiyle öğrendiği dildir. Bu dili öğrenirken çocuk herhangi bir zorluk çekmez. Bireyin yaşadığı toplum ve yakın çevresiyle iletişim kurabilmesi için gerekli olan dildir (König, 1991'den değiştirilerek alınmıştır).

Yazı Dili: Bir dilin ağızdan çıkan ses ve söz dizimlerinin yazıya geçirilmesi ile oluşturulan dildir. Kalıcı görsel göstergelerle sabitleştirilmiştir. Konuşma dilinin aksine bu dilde seçme olanağı vardır. Eline makale alan bir kişi isteğine göre istediği makaleyi okuyabilir. Bu avantaj nedeniyle yazı dili önemini korumaktadr. Yazı dili, toplumların düzen ve işleyişlerini kurmalarında önemli bir yere sahiptir. Örneğin devletlerin yönetilebilmesi için geçerli olan kaide, yazı dilidir. Bir kuruma şikâyet ya da istek başvurusunda bulunurken asıl olan dil, yazı dilidir. Konuştuğumuz ses ve sözcüklerin yazı diline çevrildiği ağız ise İstanbul ağzı olarak adlandırılmaktadır (Sönmez, 1990'dan değiştirilerek alınmıştır).

Konuşma Dili: İki ya da daha fazla dinleyici arasında karşılıklı gerçekleştirilen bir iletişim aracıdır (Sönmez, 1990). Gündelik hayatta kullanılan ağızdan ağıza farklı biçimlerde söyleyişlere sahip olan bir dildir. Örneğin yazı dilinde "geliyorum" şeklindeki bir ifadeyi konuşma dilinde "gelirem", "gelim", "geliyem", "geliyom" gibi birçok farklı şekilde ifade etmek mümkündür.

Yapay Diller: İnsanlar tarafından oluşturulan dillerdir. Martin Scheyer tarafından oluşturulan Volapük dili ve Polonyalı Doktor Zemanhof'un oluşturduğu Esperanto dili, yapay dillere örnek verilebilir (Ünalan Saniye Uysal, 2013).Bilgisayar programlama dilleri de bu kategoride ele alınabilir. C, C++, Java, Python vb. diller, insanlar tarafından yapay olarak oluşturulmuştur.

Lehçe: Aynı dilin değişik alanlarda ve yerine göre, değişik çağlarda, yine aynı toplum tarafından konuşulan değişik biçimi olarak tanımlamaktadır (König, 1991). Türkçenin, Çavuşça ve Yakutça adında iki tane lehçesi bulunmaktadır. Dilin yapısında büyük farklar oluştuğu için aynı milletin fertleri olan iki insan birbiriyle anlaşamazlar.

Şive: Şive bir dilin, bilinen tarihi seyri içinde ayrılmış olup bazı ses ve şekil ayrılıkları gösteren kolları ve bir kavmin ayrı kabilerlerinin birbiriden farklı olan konuşmaları olarak tanımlanmaktadır (Demir, 2006). Aynı ana dile sahip olan milletlerin şiveleri farklılık göstermektedir. Burada daha çok ses ve şekil farklılıkları söz konusudur. Şiveleri farklı olan devletler, çok iyi olmasa da genel anlamda birbirleri ile anlaşabilirler. Örneğin

Türkiye ve Azerbaycan Türkçesi, iki ayrı şivedir. Bu iki devletin insanları birbirlerini anlayabilmektedir.

Ağız: Yazılı geleneği olmayan farklı dil biçimlerine ağız adı verilmektedir. Bu farklılıklar bölgeden bölgeye, yöreden yöreye, şehirden şehire değişiklik göstermektedir. Hatta aynı şehirde yaşayan insanlar arasında bile bu farklılıklardan bahsedilebilmektedir. Örneğin Çocuk kavramının değişik yörelerde uşak, baba, bebe, balak gibi anlamlara gelebilmektedir. Başka bir örnek'te ise İstanbul ağzı, İzmir ağzı ve Doğu ağzında kelimeler ve sözcükler farklı şekillerde ifade edilebilmektedir. Ağız farklılığı olan insanlar birbirleri ile iletişim kurabilmektedir. İzmir ağzıyla "gitcem" şeklinde ifade edilen bir sözcük, İstanbul ağzıyla "gideceğim" şeklinde ifade edilmektedir (König, 1991'den değiştirilerek alınmıştır).

2.3. Makine Çevirisi ve Tarihi

Makine çevirisi (MT- Machine Translation), kaynak içeriği hedef dillere çevirebilen, tam otomatik yazılım anlamına gelir. İnsanlar, MT'yi başka bir dilde metin yazmalarına veya konuşma yapmalarına yardımcı olmak için kullanabilirler. MT araçları, geleneksel yolla tercüme edilmekte zorlanılan, ağırlıklı olarak terim sözcükler içeren büyük metinleri çevirmek için kullanılır. MT'nin kökeni, bu alanda kullanılan kriptanaliz, frekans analizi ve istatistiksel tekniklerin yanı sıra 9. yüzyılda yaşayan Arap kriptografı Al-Kindi'nin çalışmalarına kadar götürülebilir (Láng, 2018).1629 yılında, Rene Descartes, bir sembolün farklı dillerde eşdeğer fikirler öne sürebildiğini söylemiştir (Salmanova, 2019). 1933 yılında, Sovyet bilim insanı Peter Troyanski, bir dilden diğerine çeviri yapabilen makineyi sunmuştur. Buluş, 4 farklı dilde olmak üzere kartlar, daktilo ve eski bir film kamerasından oluşmaktadır. Operatör, çevrilecek olan metinden bir kelimeyi alıp bu kelimeye karşılık gelen kartın fotoğrafını çeker ve bunun morfolojik özelliklerini daktiloya yazardı. Bu şekilde bir cümle ve bu cümlenin morfolojik özellikleri, bir dizi halinde makineye verilerek bir dilden diğerine çeviri gerçekleştirilirdi. Ancak bütün bu çalışmalara rağmen Sovyet Rusya, bu buluşun kullanışsız olduğunu ileri sürmüştür. 7 Ocak 1954'te, New York'ta Georgetown-IBM deneyi başlamıştır. IBM (International Business Machines) 701 bilgisayarı, tarihte ilk defa altmış Rusça cümleyi, otomatik olarak İngilizceye çevirmiştir. 1966 yılında, ABD (Amerika Birleşik Devletleri) ALPAC

(Auotomatic Language Processing Advisory Committee) komitesi, bir rapor yayımlayarak MT'yi çok pahalı ve doğru çevirmeyen bir sistem olarak tanımlamıştır. Komite, bu alanda çalışma yapan araştırmacılara sunulan desteğin geri çekilerek sözlük çalışmalarına ağırlık verilmesi gerektiğini söylemiştir. 1970 yılına gelindiğinde, kural tabanlı MT fikri ortaya atılmıştır. Burada bir dilin gramer kuralından yola çıkarak sistemler geliştirilmiştir. Birbirine gramer olarak benzeyen dillerin başarılı sonuçlar verdiği, benzemeyenlerin ise başarısız sonuçlar verdiği gözlemlenmiştir. 1990'ların başında IBM araştırma merkezi, sisteme kuralların öğretilmediği, istatistiksel tabanlı bir MT tanıtmıştır. Burada amaç, çeşitli istatistiksel metotlardan yararlanılarak tahminleme yapmaktır. Bu yöntem, bir kurala bağlı olmadığı için sisteme kural tanımlama zorunluluğu yoktur. Bu yönüyle kullanışlı bir yöntem olarak gözükmektedir. 2014 yılına gelindiğinde ise artık MT'de sinirsel ağların kullanılmasıyla beraber bu alanda bir devrim gerçekleşmiştir. Bu yöntem, şu ana kadar olan bütün yöntemlerden daha iyidir. İnsan beynindeki sinirsel yapıdan esinlenerek oluşturulmuş bu öğrenme yöntemi sayesinde MT, çok kesin ve doğru sonuçlar verebilmiştir. Örneğin Google Translate, 2014 yılına kadar istatistiksel MT'yi kullanmıştır. İstatistiksel MT kullanılırken gerçekleştirilen çeviriler çok başarılı değilken; sinirsel MT'ye geçişle birlikte çevirilerdeki başarı oranı hızlı bir yükselişe geçmiştir (Bouguesmia, 2020).

2.4. Makine Çeviri Yöntemleri

Makine çevirisi için genel anlamda kullanılan 3 türlü yöntemden bahsedilebilir. Bunlar; kural tabanlı makine çevirisi, istatistiksel makine çevirisi ve sinirsel makine çevirisidir.

2.4.1. Kural tabanlı makine çevirisi

Kural tabanlı MT yaklaşımı ile yapılan çeviriler, kaynak ve hedef dillerin birbirlerine gerek sözcük yapıları gerek dil bilgisi kuralları gerekse de sözcük dizimi ilkelerinin buluştuğu ortak paydaların çokluğu ile doğru orantılı olarak iyi sonuç vermektedirler. Aynı ya da benzer dil ailesine mensup diller arası yapılan çeviriler, daha olumlu sonuçlar verirken; farklı dil ailelerine bağlı ve yapısal özellikleri ortak olmayan diller arasında yapılan çeviriler olumsuz sonuç vermektedir. "Systran", "Apertium" ve "GramTrans", kural-odaklı MT programlarına örnek olarak gösterilebilir (Sadıkov Taşpolat, 2021).

Türkçe ve Kırgızca için basit bir cümle örneği verilirse:

Hedef Cümle: Misafirler eve geldi.
Ek ve köklerinin hedef dille ilişkilendirilmesi için önce ayrıştırılması gerekir.
Misafir-ler ev-e gel-di (kök, gövde, ek, tür, zaman vb.)
Misafir: konok (isim)
-ler: +lAr; +dAr, +tAr; +lOr, +dOr, -tOr. Çokluk eki.
ev: üy (isim)
-e: +gA; +gO; +kA, +kO (yönelme eki)
gel-: kel- (fiil, yüklem)
-di: +dI, +dU; +tI, +tU (belirli geçmiş zaman eki)
Çevrilmiş Cümle: Konok-tor üy-gö kel-di (Sadıkov Taşpolat, 2021).

2.4.2. İstatistiksel makine çevirisi

İstatistiksel MT, kaynak dilde verilen cümlenin hedef dildeki en olası karşılığının tespit edilmesidir. Genelde bu MT'de olasılıklar üzerine sistemler oluşturulmaktadır. Bayesci istatistik, n-gramlar, hidden markov model, viterbi algoritması, entropi teorisi, bellek tabanlı öğrenme vb. yaklaşımlar genelde istatistiksel MT'de kullanılmaktadır. Bu yaklaşımla, çok karmaşık modellerin daha doğru ve etkili sonuçlar verdiği gözlemlenmiştir. Kural tabanlı modellerde, ilgili dillere ait bütün gramer kurallarının en ince ayrıntısına kadar bilinmesi gerekmektedir. Hâlbuki istatistiksel modelde, olasılıklar üzerinden gidilerek etkili ve doğru bir sonuca varmak mümkün hale getirilmiştir. Çeviri yapmak için yapılması gereken, bir cümlenin olasılığını bulmak ya da bir kelime sekansından sonra gelebilecek kelimenin olasılığını bulmaktır. Şayet bir Bayesci istatistikte, bir cümlenin bir metinde geçme olasılığı bulunmak isteniyorsa;

$$P(W) = P(w1, w2, \dots, wn) \tag{3.1}$$

formülü kullanılır. Eğer bir kelime sekansından sonra başka bir kelimenin gelme olasılığı bulunmak isteniyorsa;

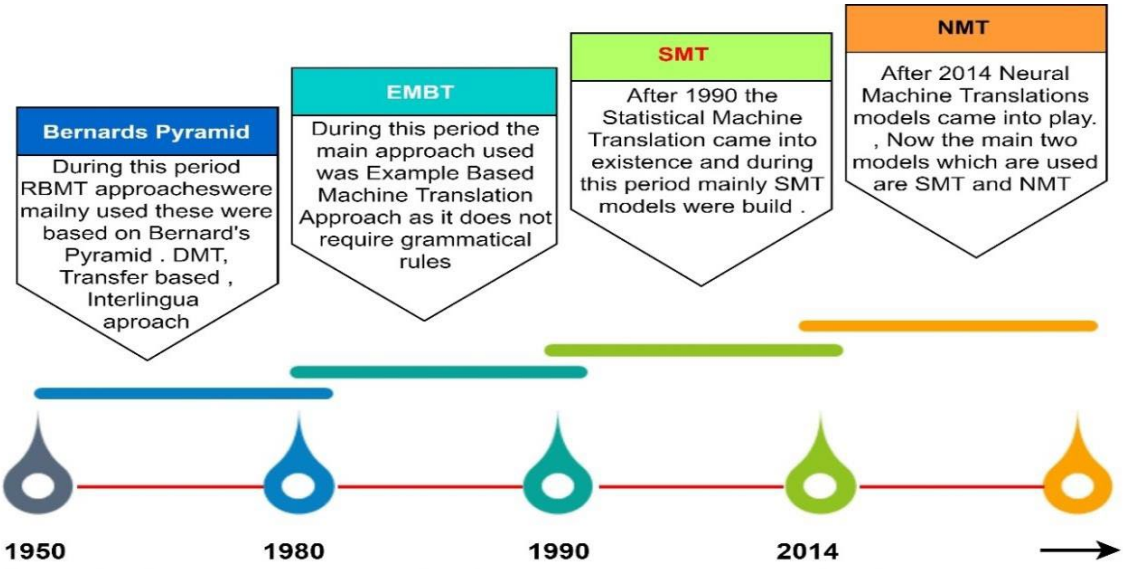
$$P(w5|w1, w2, w3, w4) \tag{3.2}$$

formülü kullanılır. Örneğin “I am tired because I am hungry.” cümlesine bakarak “I am” yapısından sonra “tired” gelme olasılığı;

$\mathrm{P}(\text{tired}|\text{I am}) = \frac{\mathrm{C}(\text{I am tired})}{C(I\ am)}$ = 0,5 olarak bulunacaktır.

2.4.3. Sinirsel makine çevirisi

Sinirsel MT ile cümleler çevrilirken kural tabanlı ya da olasılıksal yöntemler kullanılmamaktadır. Bu modeller sadece kelimenin birkaç öncesi ve birkaç sonrasına bakarak tahminleme yapmaktadır. Kural tabanlı ya da olasılıksal yöntemlerde veri kümesi büyüdükçe, geçmişteki bilgilerin hatırlanması zorlaşmaktadır. Bu da modelin tam ve doğru sonuç vermemesine sebep olmaktadır. Sinirsel makine çevirisinde ise bir kelimenin bir cümle içindeki bütün durumlarına bakılır. Kelimenin sadece bir öncesine ya da bir sonrasına bakılarak karar verilmez. Örneğin bir kelimenin, ilişkili olduğu diğer bütün kelimelere göre bir vektörü çıkartılır. Model, sinirsel katmanlı bir yapıdan oluşur. Bu katmanlar, ileri yayılım yapılarak çeşitli ağırlık değerleri ile çarpılır. Her bir katmanda çıkan sonuç kendisinden sonraki katmana iletilir. Elde edilen değer en son katmana geldiğinde gerçek değeri ile karşılaştırılarak bir kayıp değeri hesaplanır. Daha sonra geri yayılım yapılarak bu kayıp değeri ağırlık değerlerine dağıtılır. Böylece her bir katman için doğru olan ağırlık değerleri hesaplanmış olur. Sinirsel MT’nin ortaya çıkmasıyla bu alanda daha doğru ve daha etkili sonuçlar alınmıştır. Örneğin eskiden Google Translate, çok doğru bir şekilde çeviri yapamazken sinirsel MT ile daha doğru sonuçlar verdiği gözlemlenmiştir.



Şekil 2.1. MT'nin Gelişim Grafiği (Abdul et al., 2021).

Şekil 2.1'e bakıldığında, 1990 ve 2014 yılları arası istatistiksel makine çevirisi kullanılmaktaydı. 2014 yılından sonra ise sinirsel makine çevirisi, kural ve istatistiksel tabanlı makine çevirilerini geride bırakarak bu alanda lider olmayı başarmıştır. Google Translate, çeviri başarısını, sinirsel MT'den almaktadır.

2.4.4. Sinirsel makine çevirisinde seq2seq mimarisi ve tarihi

Sutskever NIPS 2014 tarafından önerilen yaklaşım, klasik yöntemlerle mukayese edilebilecek performansa sahip olan ilk sinirsel makine tercüme sistemidir (Sutskever et al., 2014). Bu algoritma, Google tarafından makine çevirisinde (MT) kullanılmak üzere geliştirilmiştir. Şirket, sistemin karmaşık problemleri çözmek hususunda Mathematica, MATLAB (Matrix Labaratory) gibi ortamlardan daha iyi sonuçlar verdiğini belirtmiştir. Geliştirilen bir LSTM (Long Short Term Memory) sinir ağı vasıtasıyla daha verimli ve etkili sonuçlar elde etmeyi başaran Google, 2020 yılında "Meena" adında, yaklaşık 2,6 milyar parametreden oluşan ve 341 GB'lık veri setine sahip bir sohbet robotu (chatbot) yayınlamıştır. Google, bu sohbet robotunun, 2020 yılında OpenAI tarafından çıkartılan GTP-2 (Generative Pre-trained Transformer-2) modelinden 1,7 kat daha fazla model kapasitesine sahip olduğunu açıklamıştır. Son olarak OpenAI tarafından 175 milyar parametre ve 45 TB veriye sahip GPT-3 (Generative Pre-trained Transformer-3) adında bir model üretilmiştir (Panchbhai & Pankanti, 2021). Seq2seq model, sıralı bir dizi

halinde verilen ifadelerin yine sıralı bir dizi şeklinde çıktısının alındığı bir mekanizmadır. Burada modele verilen sequance (sıralı diziler), zamanla değişen verileri ifade etmektedir. Bu model kullanılarak MT, soru cevaplama, sohbet robotları ve metin özetleme yapılabilmektedir.

SUMMARY OF DIFFERENT APPLICATIONS OF SEQ2SEQ MODELS.

Application	Problem Description	Input	Output
Machine Translation	Translating a sentence from a source language to a target language	A sentence (sequence of words) in language X (e.g., English)	Another sentence (sequence of words) in language Y (e.g., French)
Text Summarization Headline Generation	Summarizing a document into a more concise and shorter text	A long document like a news article (sequence of words)	A short summary/headline (sequence of words)
Question Generation	Generating interesting questions from a text document or an image	A piece of text (sequence of words) or image (sequence of layers)	A set of questions (sequence of words) related to the text or image
Question Answering	Given a text document or an image and a question, find the answer to the question	A textual question (sequence of words) or an image (sequence of layers)	A single word answer from a document or the start and end index of the answer in the document
Dialogue Generation	Generate a dialogue between two agents e.g., between a robot and human	A dialogue from the first agent (sequence of words) or audibles (sequence of speech units)	A dialogue from the second agent (sequence of words) or audibles (sequence of speech units)
Semantic Parsing	Generating automatic SQL queries from a given human-written description	A human-written description of the query (sequence of words)	The SQL command equivalent to that description
Image Captioning	Given an image, generate a caption that explains the content of the image	An image (sequence of layers)	The caption (sequence of words) describing that image
Video Captioning	Given a video clip, generate a caption that explains the content of the video	A video (sequence of images)	The caption (sequence of words) describing the video

Şekil 2.2. Seq2seq Model Kullanım Alanları (Keneshloo et al., 2020'den değiştirilerek alınmıştır).

Seq2seq (Sequence to Sequence) mimarisinde girdi, değişken sayıda veriyi ifade etmektedir. Değişkenlikten kasıt, girdinin sıralı bir şekilde sürekli değişmesidir. Örneğin konuşma dilleri, doğal diller, ardı ardına gelen dizilerden oluşmaktadır. Bir video kaydı, karelerin (frame) sürekli olarak arka arkaya gelerek oluşturduğu bir dizidir. İşte bu gibi örnekler incelendiğinde, verilerin birbirini takip eden durumlardan oluştuğu gözükmektedir. Bu noktada, konunun daha iyi anlaşılması açısından, bilgisayarlı görü alanına değinilmesinde yarar vardır. Bilgisayarlı görü de yapay zekânın bir alt dalını oluşturmaktadır. Amaç, sisteme verilen görsellerin sınıflandırılması ya da tanımlanması gibi işlemlerin gerçekleştirilmesidir. Bilgisayarlı görü alanında seq2seq mimarisi çok yaygın bir şekilde kullanılmamaktadır. Çünkü kullanıcı tarafından oluşturulan modelde, verilerin diziden diziye çevrilmesi söz konusu değildir. Giriş verisi sabittir. Sürekli olarak değişmemektedir. Netice itibari ile sabit bir sonuç dönmektedir. Seq2seq mimarisinde ise giriş ve çıkış sabit değildir. Sürekli olarak değişen değerler vermektedir. Şekil 2.2'de

seq2seq modelinin kullanıldığı birkaç alan gösterilmiştir. Bu mimarinin kullanılması ile doğal dil işleme alanındaki ana problemlerden olan MT, metin özetleme ve soru cevaplama gibi konularda başarılı sonuçlar elde edilmiştir. Son yıllarda, derin öğrenme alanındaki gelişmelerin seq2seq mimarisine eklemlenmesi ile birlikte doğal dil işleme alanında bir sıçrama gerçekleşmiştir.

2.5. Makine Çevirisinin Uygulama Alanları

MT'nin sektörde kullanım alanlarına bakıldığında; günlük ortalama 2000 kelime çeviren bir tercümanın yerine de kullanılabileceği öngörülmekle birlikte, günlük milyonlarca kelime çevrilebileceği dikkate alındığında herhangi bir güncel MT yönteminin, bir tercümandan daha hızlı ve daha fazla sayıda kelimeyi tercüme edeceği gerçeği de ön plana çıkmaktadır. Günümüzde Google Translate ve Yandex çeviri gibi sistemler incelendiğinde, genel anlamda çok ağır bir akademik dil olmamak şartıyla doğru çeviri yaptıkları görülmektedir. Bu anlamda, akademik bir çeviri yapılmak istenildiğinde MT sistemleri tam anlamıyla başarılı olamadıkları için tercümana olan ihtiyaç devam etmektedir. Diğer bir kullanım alanı ise e-postaların okunması, gelen bu e-postalara cevap verilmesi ya da şirketlerin müşteri profilini değerlendirmek için kullandığı kısa ölçekli sorulara verilen yanıtları okumak için kullanılabilmesidir. Burada dikkat edilmesi gereken husus, içeriğin sınırlı ve az sayıda olmasıdır. İçerikteki bilgi miktarı arttığında ya da daha karmaşık hale geldiğinde MT sistemleri kullanılamayacaktır. İçinde bulunduğumuz zaman dilimi itibari ile MT'nin, tam anlamıyla insan yerine geçebilecek bir potansiyeli olduğunu söylemek mümkün değildir.

2.6. Makine Çevirisinin Kullanılabilirliği

İçeriğin kalitesi, dil becerilerinin yansıtılması, kültüre olan duyarlılık, dilin değişimi ve insan elinin değmesi gibi kriterler bir MT sisteminin henüz kullanılabilir olmadığını göstermektedir. İçeriğin kaliteli olması, bir insanın metnin özünü anlayıp dilde var olan farklılıkları tespit ederek ortaya bir ürün çıkarması ile mümkün hale gelmektedir. Fakat bir makine, metnin özünü ve dilde var olan farklılıkları tespit etmede sorunlar yaşayabilmektedir. Dil becerisi, okuyucunun olayı adeta kendi yaşıyormuş gibi hissetmesinin sağlanmasıdır. Bu ise ancak, dilde kullanılan söz sanatları ile mümkün

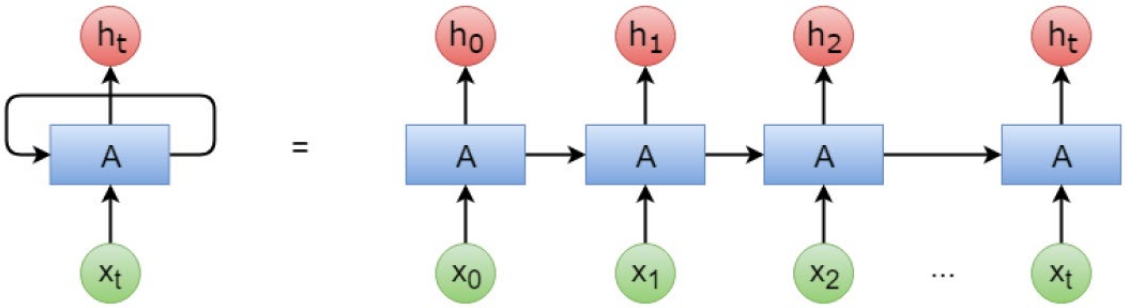
olabilir. Fakat bir makineden bu gibi becerilerin beklenmesi, günümüzde, mümkün gözükmemektedir. Her bir milletin kültürü, yaşam tarzı, gelenek ve görenekleri farklılık göstermektedir. Her dilin kendine özgü atasözleri ve deyimleri bulunmaktadır. İşte bu gibi konular, bir insan tarafından bir dilden başka bir dile mükemmel bir şekilde yansıtılabilirken, makinelerin bunları yapması pek olası değildir. Dilin değişimi ile bir dildeki var olan kelimelerin zaman içerisinde değişikliğe uğraması, yeni anlamlar kazanması mümkün olabilmektedir. Dolayısıyla her bir kelimenin, derin bir tarihsel birikimi olabilir. İnsan çeviriciler bu gibi durumları rahatlıkla kavrayıp analiz edebilirken, makineler için aynı şeyleri söylemek ve MT'den aynı kaliteyi beklemek mümkün değildir. Son aşamada, MT aracılığıyla yapılan tüm çevirilerin bir insan elinden geçmesi gerekmektedir. Çünkü insan, çevrilen metne bütünsel bir bakış açısıyla bakarak konunun bütünlüğünü ve tutarlılığını analiz edebilir. Fakat makinler bunu yapamazlar. İşte sayılan ve açıklanan bütün bu sebeplerden dolayı bir makinenin bir tercümanın yerini tam anlamıyla alması günümüz itibari ile mümkün gözükmemektedir. Ancak geçmiş yıllara bakıldığında ciddi anlamda ilerlemeler kaydedildiğini düşünürsek, gelecekte belki de insanın tam anlamıyla yerini alabilecek sistemler geliştirmek mümkün olabilecektir.

Sayılan tüm bu olumsuzluklara rağmen MT hızlı, tutarlı ve az bir bütçeyle çok iş yapılmasına olanak sağlaması sebebiyle bir tercümandan çok daha fazla tercih edilmektedir. Örneğin 1 milyon kelimelik bir verinin, sınırlı bir süre içinde (1 gün gibi) çevrilmesi talebi ile karşı karşıya gelinmesi halinde, bir tercümanın bu iş için ayıracağı zaman ve harcayacağı emek ile makine kullanılarak işin gerçekleştirilmesi durumunda geçecek zaman ve sarf edilecek iş gücü karşılaştırıldığında, bu işi dakikalar hatta kullanılan makine ve algoritmanın performansına göre saniyeler içerisinde yapabilecek makinelerin, sistemlerin, daha fazla tercih edilmelerinin sebebi anlaşılmış olacaktır. Kelimelerin tam bir gramer uyumu içinde olması gibi durumlarda, makineler, çeviride daha tutarlı olabilmektedir. İnsan bu gibi uyumları bazen gözden kaçırabilmektedir. Son olarak, daha az bir maliyetle, makineler milyonlarca satırlık bir veriyi işleyebilmektedir. Bu işin insan eliyle yapılabilmesi için çok sayıda tercümanı işe almak gerekecektir. Haliyle bu da ekstra maliyet anlamına gelmektedir.

3. MATERYAL VE YÖNTEM

3.1. Özyinelemeli Sinir Ağı (Recurrent Neural Network, Rnn)

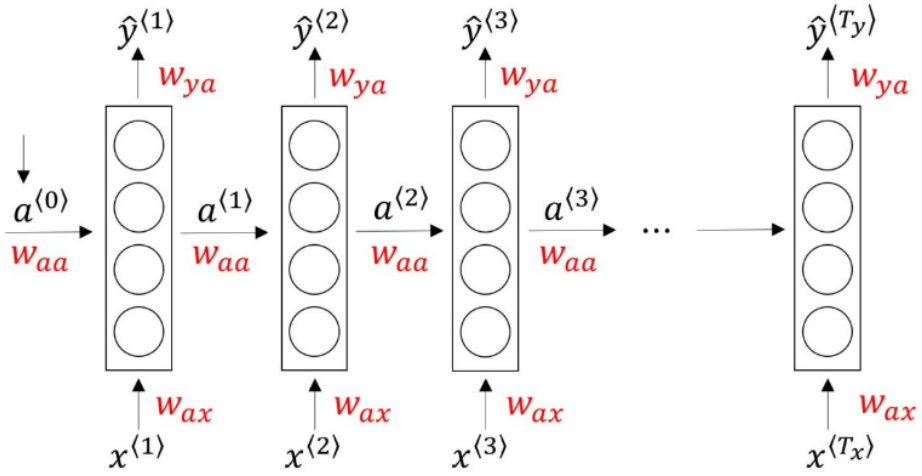
RNN (Recurrent Neural Network)'ler sinirsel MT'nin temelini oluşturan yapılardır (Xiao Jianqiong, 2020). Bu yapılar, geçmişte kullanılan geleneksel makine çevirilerinin yerini alarak, bir devrimin başlangıcına öncülük etmiştir. RNN, girdi ve çıktının değişkenlik gösterdiği durumlarda kullanılmaktadır. Örneğin, bir dilden bir dile çeviri sistemlerinde, girdi değişken uzunlukta ve birbirini takip eden sekanslardan oluşmaktadır. Çıktı da bunun gibi birbirini takip eden sekanslardan oluşabilmektedir. Burada bir verinin sürekli olarak değişmesi söz konusudur. Yine başka bir örnek olarak borsa tahmini gibi sistemler verilebilir. Borsa tahmin sistemlerinde, geçmişte sürekli bir dizi halinde gelen verilere bakılarak geleceğe yönelik tahminlerde bulunulabilmektedir. RNN tıpkı sesten yazıya veya sesten sese çeviri benzeri süreklilik arz eden durumlarda kullanılabildiği gibi videoların işlenmesi durumlarında da kullanılabilmektedir. Çünkü videolar birbiri ardına gelen framelerden oluşmaktadır. Özellikle doğal dil işlemede, bir önceki adımda girilen kelime ya da sözcüklerin hafızada tutulup bu bilgiye dayanarak bir sonraki adımın tahmin edilmesi istenmektedir.



Şekil 3.1. RNN'nin İç Yapısı (Onyedi et al., 2020).

Şekil 3.1'e bakıldığında bir RNN'nin iç yapısı görülmektedir. Bu yapıda "input" olarak bir x girdisi alınarak bir "hidden state" (h0, h1, h2, ..., ht) değeri üretilir. Ardından, bir sonraki hücre için bir önceki durumdan gelen "hidden state" ve mevcut durumun x girdisi

alınarak, bir sonraki hücreye gerekli bilgiler taşınmış olur. Bu şekilde kendini yenileyen bir sinir ağı kurulmuş olur. Bir RNN modeli şekil 3.2'deki gibi formülleştirilebilir.



Şekil 3.2. RNN Modeli (Zivkovic Strahinja, 2020).

Şekil 3.2'ye göre sıfır (0) anında herhangi bir veri olmadığı için $a^{<0>}$değeri 0 olarak verilir.

$$a^{<0>} = 0$$

$$a^{<1>} = g_1(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a)$$

$a^{<1>}$ Değerini hesaplamak için bir önceki adımdan gelen $W_{aa}a^{<0>}$ çarpımı ve mevcut durumdaki $W_{ax}x^{<1>}$ çarpımı bir bias(b_a) değeri toplandıktan sonra g_1 adındaki bir aktivasyon fonksiyonundan geçirilerek bir sonraki adıma girdi olarak verilecek olan $a^{<1>}$ değeri elde edilir. Formül genelleştirildiğinde;

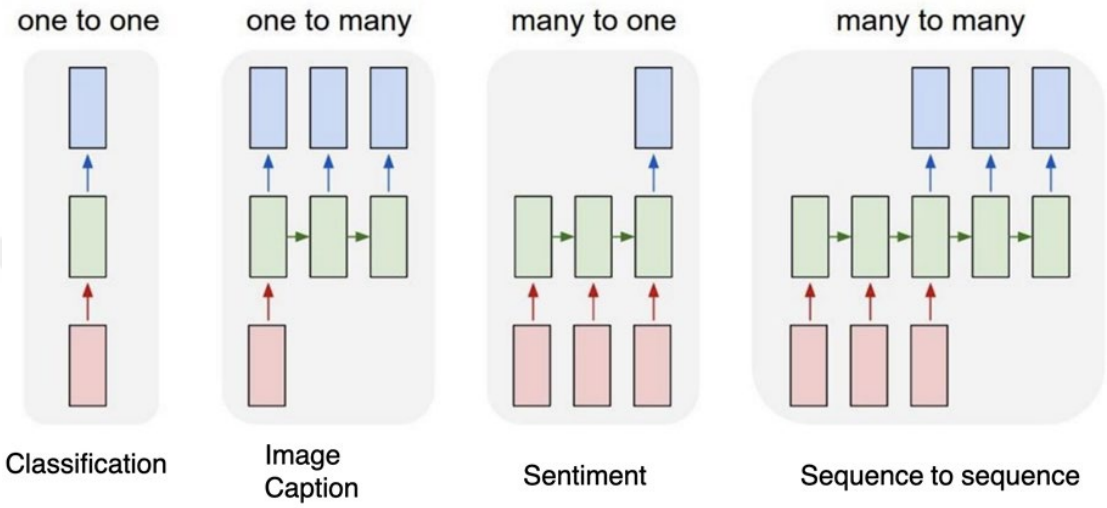
$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad (3.3)$$

$$\hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + +b_y) \quad (3.4)$$

sonucuna ulaşılır.

RNN gibi yapılar, şekil 3.3'te gösterildiği gibi sınıflandırma, resim yakalama, duygu analizi ve MT alanlarında kullanılmaktadır. Burada kullanılan yapıya göre RNN'ler farklı

girdi ve çıktılar alabilmektedir. Örneğin duygu analizi yapılmak istenildiğinde, cümleler sekanslar halinde girdi olarak alınacaktır. Fakat çıktı olarak sadece cümlenin olumlu ya da olumsuz olduğu sonucu bulunacaktır. Yine seq2seq mimarisi kullanıldığında hem girdi hem de çıktı sekanslar halinde olacaktır. Buradan da anlaşıldığı üzere RNN'ler farklı tip mimarilerde değişik şekillerde kullanılabilmektedir.



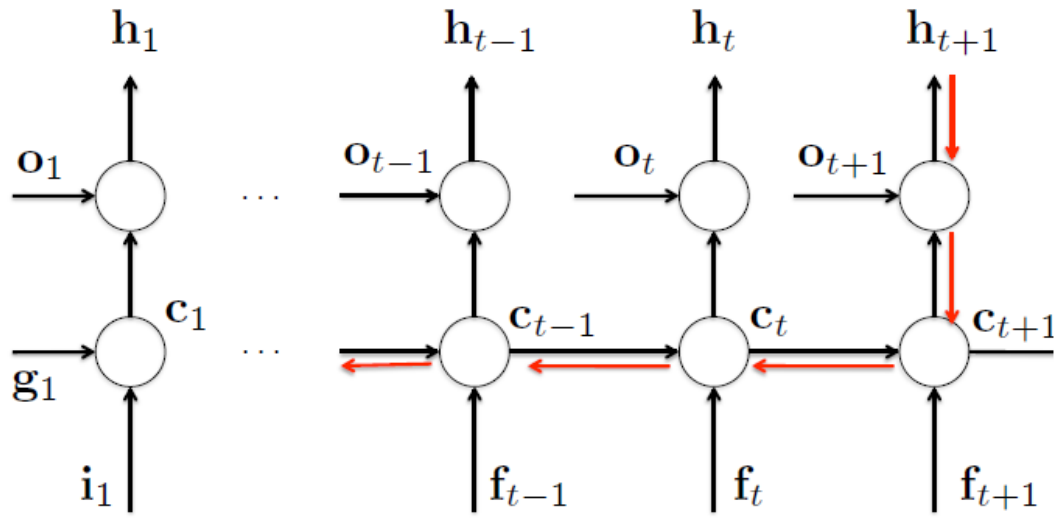
Şekil 3.3. RNN'nin Kullanım Alanları (ÇELİK & ODABAS, 2020).

RNN dil modelleri, çok geçmişten bilgi alma konusunda yeterli olamamıştır. Büyük bir derlem kullanılarak eğitilen dil modelleri başarılı sonuçlar verememiştir. Bunun sebebi ise vanishing gradient (gradyanların yok olması) ve exploding gradient (gradyanların patlaması) olarak ifade edilebilir.

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>}) \tag{3.5}$$

Denklem (3.5)'te kayıp fonksiyonu verilmiştir. Bu fonksiyona göre hesaplanan kayıp değeri, şekil 3.4'te gösterildiği gibi, geri yayılım yapılarak ağırlık değerleri güncellenmektedir. Fakat geri yayılım yapılırken sekans çok uzun olursa gradyanlar yok olabilmektedir. Örneğin, hesaplanan kayıp fonksiyonunun, sıfıra yakın bir değer çıktığı durumlarda, geri yayılım yapılırken, arka arkaya türev almanın doğal bir sonucu olarak, kayıp değeri küçülerek sıfır değerine ulaşabilmektedir. Bu hâl, ağırlık değerlerinin

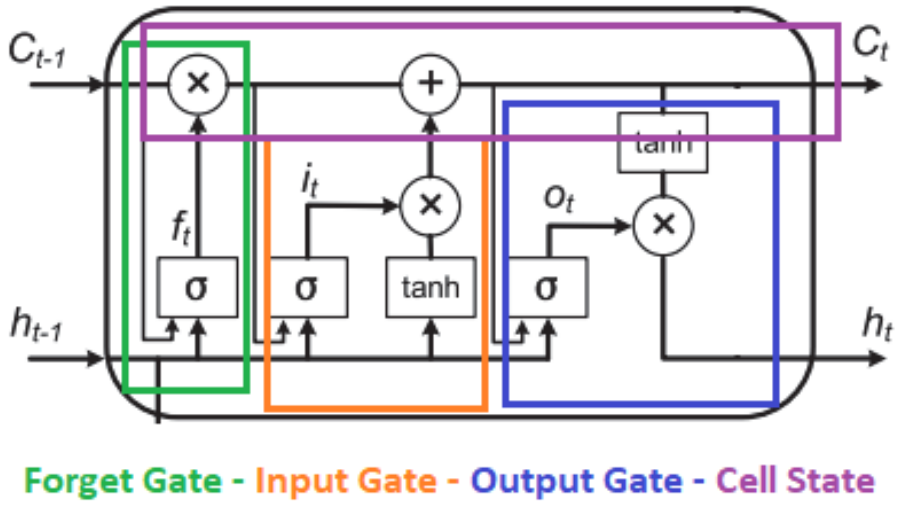
değişmemesi anlamına gelir ki artık burada öğrenme işlemi gerçekleşmemektedir. Diğer türlü kayıp değeri büyük bir sayı çıktıysa arka arkaya çarpmanın bir sonucu devasa değerlere ulaşacaktır. Bu olaya da gradyanların aşırı patlaması denilmektedir. İşte bu gibi vanishing ve exploding gradient sorunlarından ötürü klasik RNN yapıları terk edilerek LSTM ve GRU (Gated Recurrent Unit) gibi yapılara geçilmiştir.



Şekil 3.4. RNN Geri Yayılım (Chen, 2016).

3.2. Uzun Kısa Süreli Bellek (Long Short Term Memory, Lstm)

LSTM ile aslında klasik RNN kullanılarak oluşturulan bir modelin eksikliği giderilmiştir (Yuan et al., 2020). RNN, verilen bir metindeki ifadelerden sadece bir ya da iki öncesine kadar olanları başarılı bir şekilde hatırlamaktadır. Eğer çok fazla kelimenin hatırlanması gerekiyorsa burada RNN "vanishing gradient" dediğimiz bir problem sebebiyle işe yaramayacaktır. Gradyanların yok olması, ağırlık değerlerinin bir noktadan sonra değişmemesi anlamına gelir ki öğrenme gerçekleşmiyor demektir. RNN'in bu eksikliğinin ortadan kaldırılması için LSTM'ler geliştirilmiştir. LSTM ile geçmişteki bilgiler hatırlanabilmektedir. Bu da oluşturulan modelin daha doğru sonuçlar vermesini sağlamaktadır.



Şekil 3.5. LSTM Hücre Örneği (Terzi, 2021'den değiştirilerek alınmıştır).

Şekil 3.5'te bir LSTM hücresinin iç yapısı sunulmuştur. Şekilde, üç adet kapı bulunmaktadır. "Forget gate" ile önceki hücrelerden gelen bilginin unutulup unutulmayacağına, "Input gate" ile hangi bilginin hafızada tutulacağına ve nihayetinde "Output gate" ile de hangi bilginin çıkış olarak bir sonraki hücreye gönderileceğine karar verilmektedir.

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \quad (3.6)$$

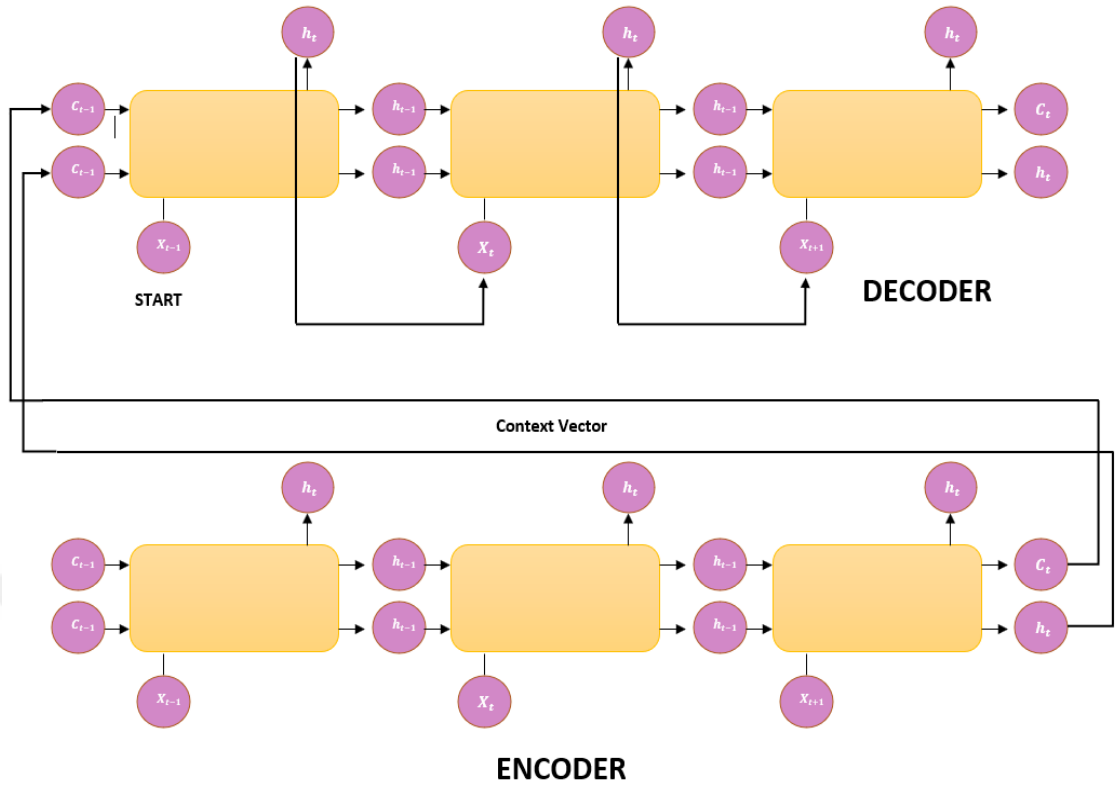
$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i) \quad (3.7)$$

$$C_t = tanh(W_c . [h_{t-1}, x_t] + b_c) \quad (3.8)$$

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o) \quad (3.9)$$

$$h_t = o_t * tanh(C_t) \quad (3.10)$$

3.6'daki denkleme göre forget kapısı için f_t değeri sıfır gelirse önceki hücrelerden gelen değer unutulur. Eğer bir gelirse önceki hücrelerden gelen değer bir sonraki hücreye aktarılır. 3.7'deki denkleme göre i_t değeri eğer sıfır gelirse mevcut durumdaki bilgi giriş olarak alınmaz. Eğer i_t değeri bir gelirse mevcut durumdaki bilgi giriş olarak hesaba katılır. 3.9'daki denkleme göre o_t değerinin bir ve sıfır olması durumlarına göre hücreden çıkacak bilginin ilgili hücrenin devamında gelecek hücreye iletilip iletilmeyeceğine karar verilir.

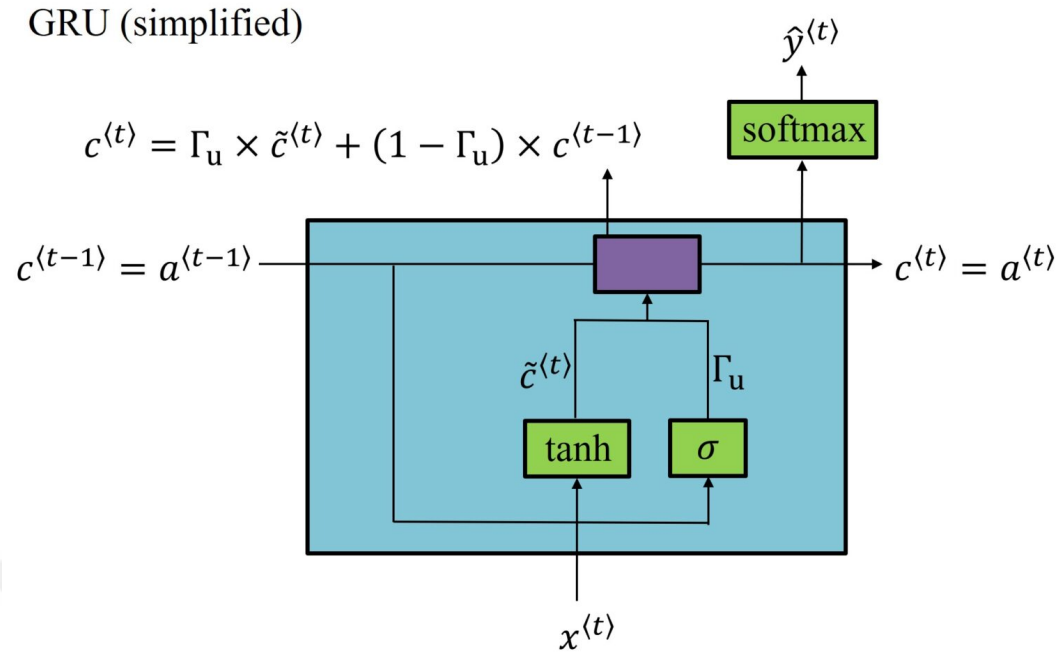


Şekil 3.6. Encoder-Decoder Yapısı.

Şekil 3.6'da gösterilen Seq2Seq mimarisi bir Encoder-Decoder yapısı içerisinde kullanılmaktadır (Szucs & Huszti, 2019). Encoder-Decoder yapısında birden fazla LSTM, GRU ya da BiLSTM hücreleri bulunmaktadır. Encoder yapısına gelen giriş cümleleri, bu hücrelerin en sonunda, bir düşünce vektörü olarak çıktı vermektedir. Sonra bu vektör, decoder modeline gönderilmektedir. Decoder modeli, gelen bu düşünce vektörüne göre bir çıktı üretmektedir.

3.3. Kapı Özyinelemeli Geçitler (Gated Recurrent Units, Gru)

GRU, geçmiş bilgileri hatırlamak için LSTM gibi kullanılan bir yapıdır (Qian et al., 2021). Fakat LSTM'den farklı olarak içinde kullanılan parametreler daha az ve sadedir. Bu da GRU'nun LSTM'ye göre daha hızlı çalışmasını sağlamaktadır.



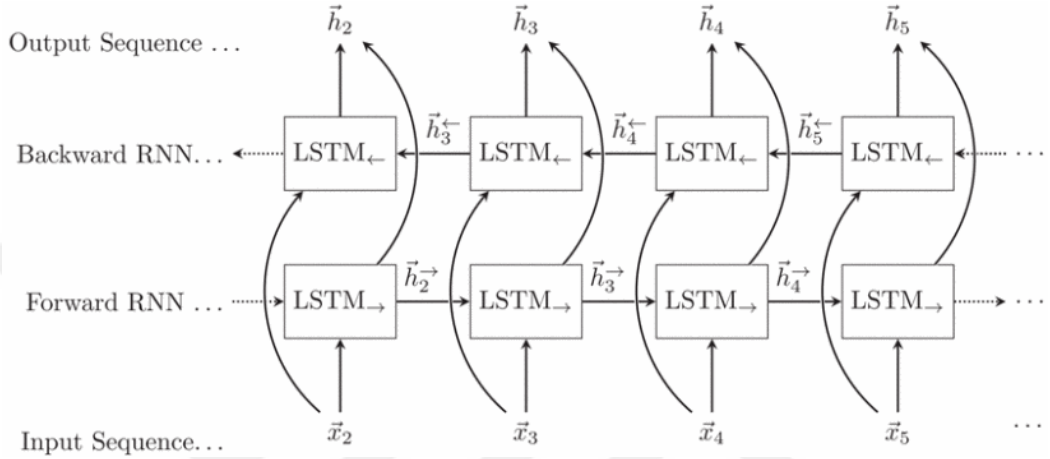
Şekil 3.7. GRU Hücre Örneği (Zivkovic Strahinja, 2020b).

Şekil 3.7'de bir GRU'nun iç yapısı sunulmuştur. Şekilde verilen $c^{(t-1)}$ bir önceki adımdan gelen değer $x^{(t)}$ ise mevcut durumdaki girdiyi ifade etmektedir. $c^{(t-1)}$ ve $x^{(t)}$ değerleri $tanh$ fonksiyonundan geçirilerek $\tilde{c}$'ye aday olan değer oluşturulur. Çünkü hala gerçek c değeri değildir. Gerçek c değeri ise güncelleme geçidinden geçtikten sonra elde edilmektedir. $r_u = 1$ yapılarak önceki adımlarda hafızada tutulması istenen bir değer sonraki adımlarda kullanılabilmektedir. Eğer gradient hesaplamasında 0 gelmişse, sonraki kelimeler için $r_u = 0$ olacak ve buna ait $c^{(t-1)}$ değerleri hesaplamaya dahil edilmeyecektir. Sequence to sequence mimarisinin GRU ile kullanımı LSTM ile aynıdır. Sadece LSTM hücreleri yerine GRU hücreleri kullanılmaktadır (Zivkovic Strahinja, 2020b).

3.4. Çift Taraflı Uzun Kısa Süreli Bellek (Bidirectional Long Short Term Memory, Bilstm)

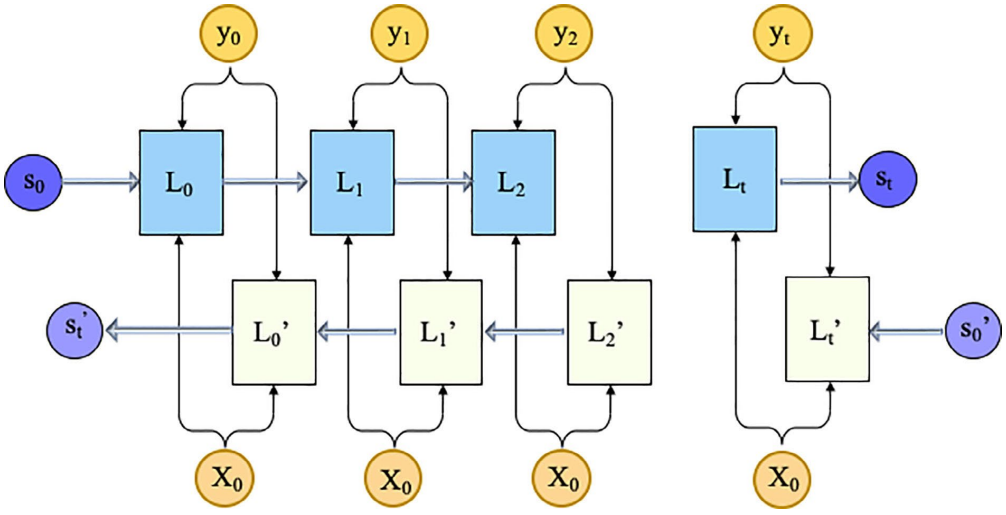
BiLSTM (Bidirectional Long Short Term Memory) yapısı çift yönlü çalışan LSTM olarak ifade edilebilir. LSTM yapısı ile sadece geçmişteki bilgiler toplanılarak geleceğe ait bir tahminlemede bulunulabilirken, BiLSTM yapısı ile hem geçmişteki hem de gelecekteki bilgiler kullanılarak bir tahminleme yapılabilmektedir. Örneğin "He said, Teddy bears on

sale" cümlesinde, Teddy kelimesinin oyuncak ayı olduğu bilgisine sadece “He said” cümlesine bakılarak karar verilemez. Karar verebilmek için “Teddy” kelimesinden sonra gelen cümleye de bakılması gerekmektedir (Al-Sabahi et al., 2018). İşte bu gibi durumlarda BiLSTM yapısı kullanılmaktadır. Şekil 3.8’de BiLSTM hücresinin network yapısı gösterilmiştir.



Şekil 3.8. BiLSTM Network Yapısı-1 (Moharm et al., 2020).

Şekil 3.9’da y_2 değerini hesaplayabilmek için hem geçmiş zamandan gelen L_2 değerlerine hem de gelecekten gelen $L_{2'}$ değerlerine ihtiyaç vardır.



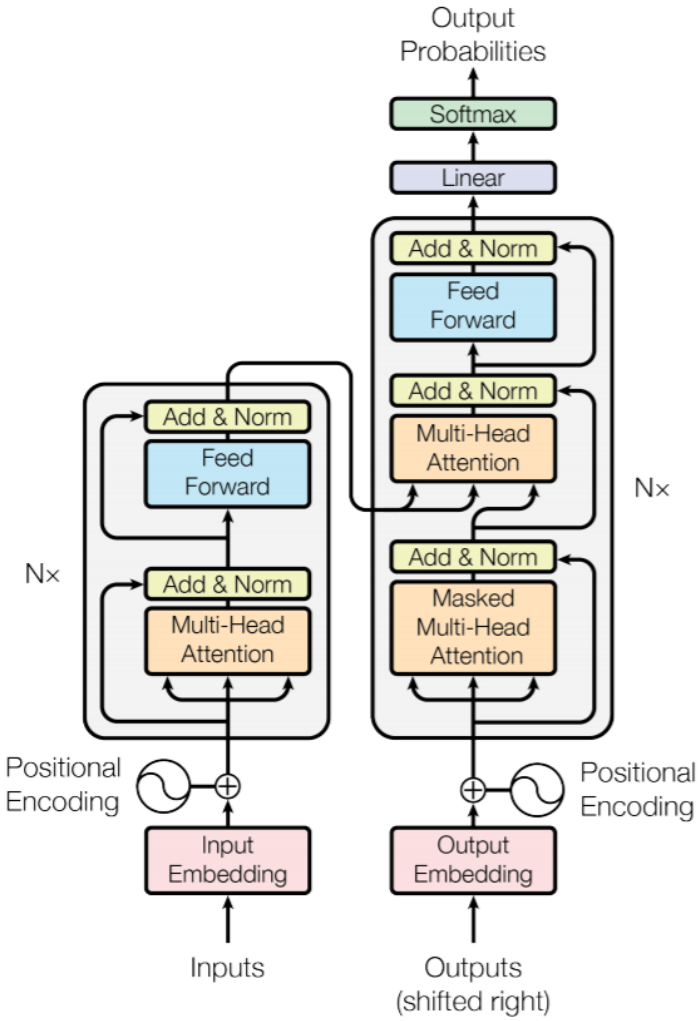
Şekil 3.9. BiLSTM Network Yapısı-2 (Wu et al., 2021).

y_2 değeri, uygun ağırlık değerleri ile çarpılıp bir bias değeri ile toplandıktan sonra aktivasyon fonksiyonundan geçirilerek denklem 3.11'deki gibi tahmin değeri bulunmaktadır.

$$y_t = g(W_y[L, L'] + b_y) \tag{3.11}$$

3.5. Üretken Ön İşlemeli Dönüştürücü-3 (Generative Pre-Trained Transformer-3, Gpt-3)

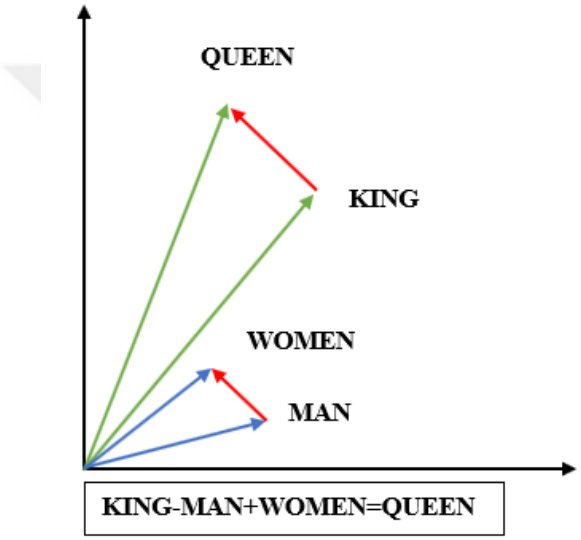
GPT-3 modeli, aslında transformer mimarisini kullanan daha gelişmiş bir modeldir. Dolayısı ile bu modeli anlayabilmek için öncelikle transformer mimarisine bakılması gerekmektedir. Transformer mimarisi kullanılarak GPT-2, GPT3 ve Google'ın kullandığı BERT (Bidirectional Encoder Representations from Transformers) gibi dil modelleri geliştirilmiştir. GPT-3 modeli, OPENAI tarafından geliştirilen 175 milyar parametreye sahip bir dil modelidir. Transformer mimarisi, ilk olarak 2017 yılında Google tarafından geliştirilmiştir. Bu model ile birlikte MT alanında adeta bir devrim yaşanmıştır. Çünkü bu model şimdiye kadar olan dil modellerinden daha iyi bir performansa sahiptir. Transformer mimarisi, geçmişte kullanılan RNN, LSTM ve GRU gibi yapılar terk edilerek self-attention ve Encoder-Decoder mimarisi temelli bir yapıda geliştirilmiştir. RNN gibi dil modelleri kullanılırken, mimarilerinin yapısı gereği, bir işlemi yapmadan önce başka bir işlemin yapılması gerekmekteydi (Bir sonraki adımı hesaplayabilmek için bir önceki adıma ihtiyaç vardır.) Transformerda ise, işlemlerin çoğu paralel olarak yapılabilmektedir (Vaswani et al., 2017).



Şekil 3.10. Transformer Network Yapısı (Vaswani et al., 2017).

Şekil 3.10 incelendiğinde, çift taraflı bir blok ile karşılaşılmaktadır. Sol taraftaki Encoder, sağ taraftaki ise Decoder bloğunu temsil etmektedir. Klasik Encoder- Decoder yapısında, giriş değişken uzunluktaki cümlelerden oluşuyordu. Encoder sabit uzunlukta bir düşünce vektörü üreterek Decoder bloğuna gönderiyordu. Decoder ise Encoderden gelen düşünce vektörü ve bir başlangıç parametresi ile çalışmaya başlıyordu. MT gibi işlemlerde Encoder-Decoder yapısı kullanılmasının nedeni ise sabit bir giriş çıkış elde edebilmekti. Örneğin, İngilizcedeki "What are you doing" cümlesinin, Türkçeye "Ne yapıyorsun" şeklinde çevrilmesi işleminde; dört kelimeden oluşan İngilizce girdi ile iki kelimeden oluşan Türkçe çıktı cümlelerinin değişkenliği görülmektedir. Nöral ağlar sürekli değişken bir yapıda olmadığından, bu değişkenliğin Encoder-Decoder gibi sistemler kullanılarak sabit bir yapıya dönüştürülmesi gerekmektedir. Fakat transformer

mimarisinde, yukarıda anlatılanın aksine, Encoder modele gönderilecek giriş bilgisi sabit hale getirilmiştir. Giriş olarak verilen bir cümle Word Embeding (Kelime Gömme) yöntemi ile vektör haline getirilmekte ve bir kelimenin diğer kelimeler ile olan bağlantısı tespit edilmektedir. Bu alanda, sıklıkla verilen örneklerden biri, kral ve kraliçe örneğidir. Şekil 3.11'e bakıldığında kral, kraliçe, erkek ve kadın vektörleri görülmektedir. Bu şekilde, birbirine yakın kelimeler bir alanda toplanmaktadır. Burada kral (KING) vektöründen erkek (MAN) vektörü çıkartılıp kadın (WOMAN) vektörü eklendiğinde, sonucun şaşırtıcı bir şekilde kraliçe (QUEEN) olduğu görülmüştür.



Şekil 3.11. Kelime Benzerliklerinin Vektörel Gösterimi.

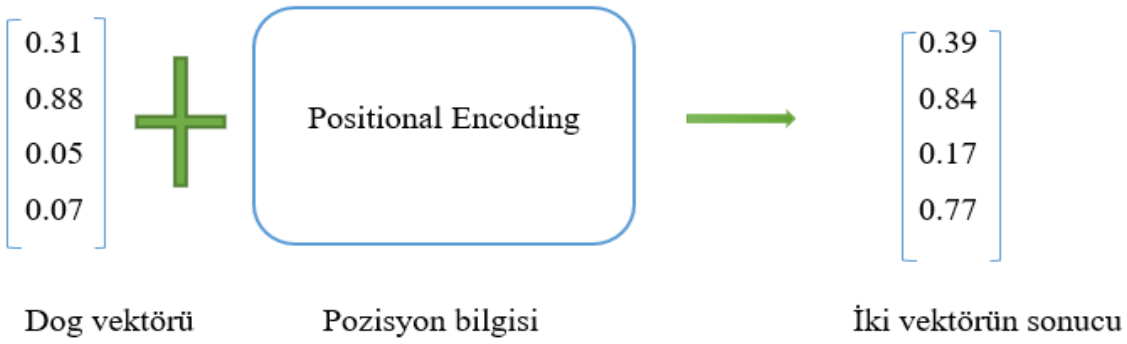
Sürecin başında girdi cümlesi "word embeding" işlemine sokulmaktadır. Ardından "Positional Encoding" (pozisyon kodlama) ile her bir kelimenin bulunduğu pozisyon bilgisine ait vektörü çıkartılmaktadır. RNN, LSTM gibi dil modellerinde, kelimeler bir sekans halinde, sırayla modele verildiği için pozisyon bilgisine gerek duyulmamaktadır. Transformer modelinde ise, cümle toplu olarak modele verildiği için, her bir kelimenin pozisyon bilgisinin hesaplanıp verilmesi gerekmektedir. Pozisyon bilgisi ise denklem 3.12 ve 3.13'te verildiği gibi hesaplanmaktadır.

$$PE_{(pos,2i)} = \sin\left(pos/10000^{\frac{2i}{d_{model}}}\right) \tag{3.12}$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{\frac{2i}{d_{model}}}\right) \tag{3.13}$$

Pozisyon bilgisi hesaplanırken, bir kelimenin birden fazla yerde olması durumunda yaşanacak tekrarın önüne geçmek için, indeks bilgisine bakılmamaktadır. Hâlbuki mükerrer her bir kelime, cümle içinde konumlandığı yere göre, farklı anlama gelebilmektedir. Başka bir açıdan, mükerrer kelimenin cümle içindeki kullanım anlamları aynı olsa dahi, çok sayıdaki tekrar, verinin normalleştirilmesi sürecinde problemler çıkartmaktadır. Bu gibi sebeplerden dolayı kelime bilgisi aynı olsa bile indeks bilgisinin farklılık göstermesi gerekmektedir. Her bir kelimenin farklı bir indeks bilgisine sahip olacak şekilde hesaplanması, denklem 3.12 ve 3.13'te gösterildiği gibi yapılmaktadır. Tek olan pozisyon bilgisi hesaplanırken denklem 3.12, çift pozisyon bilgisi hesaplanırken ise denklem 3.13 kullanılmaktadır.

Örneğin, "Ali'nin köpeği sevimlidir" cümlesindeki "köpek" kelimesinin "word embeding" ve "positional encoding" ile birleştirilmesi, aşağıdaki gibi bir görünüm oluşturacaktır.



Şekil 3.12. Embeding ve Position Encoding Vektörlerinin Birleştirilmesi.

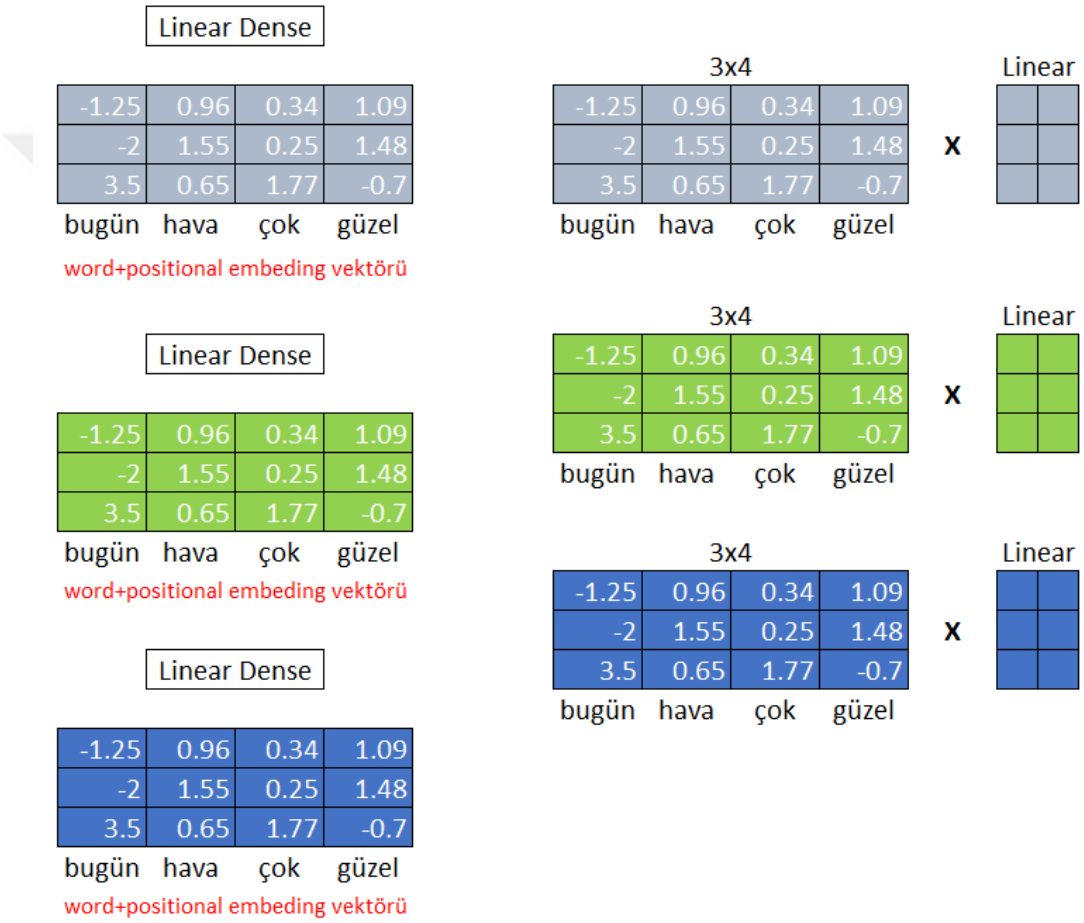
Şekil 3.12'deki gibi bir çıktısı olan iki vektörün birleşimi, "Multi-Head Attention" (Çok Başlı Dikkat)'a gönderilerek her bir kelimenin cümledeki hangi kelimeler ile ilgili olduğu bilgisi hesaplanır. Örneğin, Çizelge 3.1'de "Bugün hava çok sıcak" cümlesi, her bir kelime benzerliklerine göre vektörel olarak ifade edilmiştir.

Çizelge 3.1. Kelime Benzerlikleri

bugün →	bugün	Hava	çok	sıcak	$[0.99,0.58,0.10,0.12]^T$
hava →	bugün	Hava	çok	sıcak	$[0.58,0.95,0.13,0.27]^T$
çok →	bugün	Hava	çok	sıcak	$[0.10,0.13,0.89,0.55]^T$
sıcak →	bugün	Hava	çok	sıcak	$[0.12,0.27,0.55,0.90]^T$

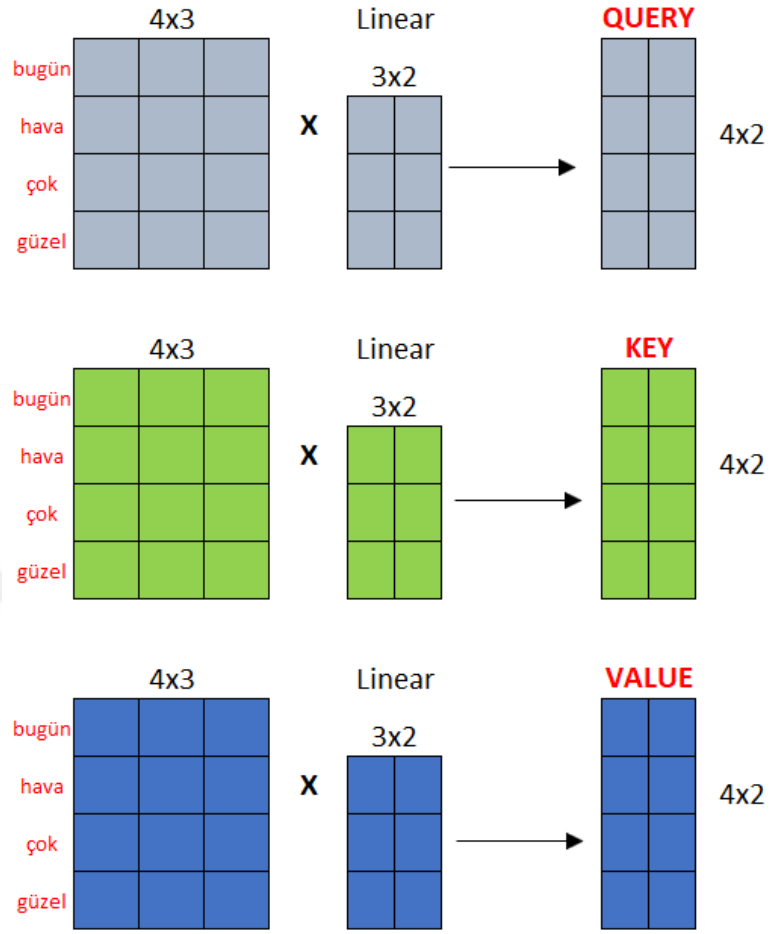
Kelime benzerliklerini hesaplayabilmek için Q (Query), K (Key) ve V (Value) değerlerinin bulunması gerekmektedir. Bu değerlerin nasıl bulunacağı aşağıdaki örnekle açıklanmaktadır:

Şekil 3.13'te Multi-Head Attention'un içeriği gösterilmiştir. Bu bölümde, kendisine gelen kelime ve pozisyon bilgisi bir matris şeklinde ifade edilmektedir. Örnekte "bugün hava çok güzel" matrisi, pozisyon bilgisi ile birlikte lineer dense ile çarpılmaktadır.

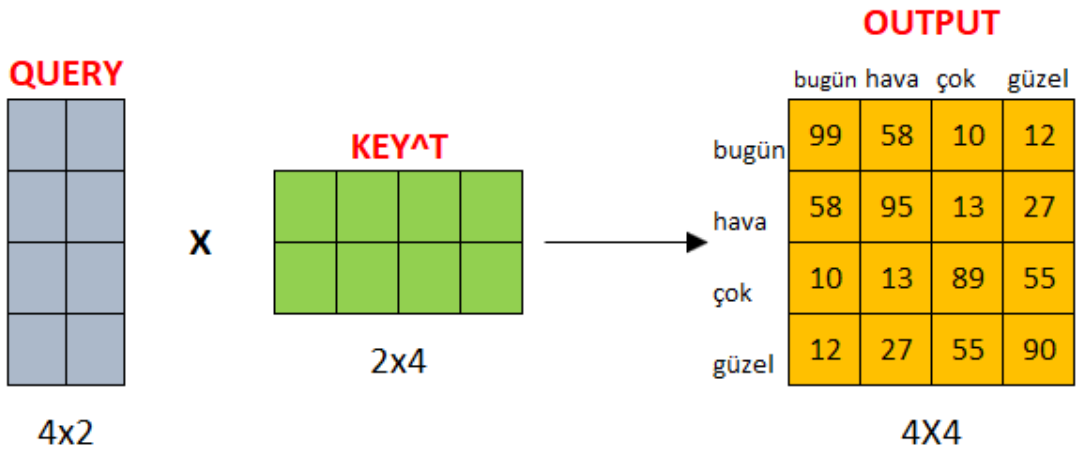


Şekil 3.13. Multi-Head Attention Linear Dense Matris Çıktısı.

Çarpım işlemini gerçekleştirebilmek için "word+positional Embeding" matrisinin transpozu alınmalıdır. Q, K ve V değerleri, Şekil 3.14'te görülebileceği üzere, matrisin transpozu alınıp çarpım işlemi gerçekleştirildikten sonra elde edilmektedir.



Şekil 3.14. Q, K ve V Vektörlerinin Elde Edilmesi.



Şekil 3.15. Denklem 3.14'e göre QK^T İşleminin Gerçekleştirilmesi.

Output olarak elde edilen vektör ile birbirine yakın olan kelimelerin benzerlik oranları hesaplanmaktadır. Bu işlemlerden sonra hesaplanan QK^T değeri, $\sqrt{d_k}$ değerine bölünmektedir. Burada d_k değeri Q, K veya V vektörünün boyutuna eşittir. Çıkan sonuç, denklem 3.14'te verildiği gibi, softmax fonksiyonundan geçirilmekte ve ardından V (value) matrisi ile çarpılarak Attention (dikkat) hesaplanmaktadır.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.14}$$

Bütün bu hesaplamalar yapıldıktan sonra, birden fazla cümle arasındaki benzerliklerin de tamamı hesaplanarak uç uca ekleme işlemi yapılmaktadır. Aslında "Multi-Head Attention", cümledeki kelime benzerliklerinin bulunmasından sonra her birinin uç uca eklendiği süreç olarak tanımlanabilir. Uç uca ekleme yapılmasının sebebi, her bir cümlenin benzerliklerinin kaybolmamasıdır. "Multi-Head Attention"dan çıkan vektörler, "Feed Forward" (İleri besleme) nöral yapısına gönderilmektedir. Buradaki amaç, Encoderden çıkan "input" (giriş) bilgisinin Decoder tarafında kabul edilebilir bir bilgiye dönüştürülmesidir. Yani, giriş verisinin boyutu azaltılarak daha sade bir hale getirilmektedir. LSTM yapısını kullanan bir Encoder-Decoder yapısında, bütün cümleyi temsil eden, birbirine bağımlı tek bir düşünce vektörü elde edilmektedir. RNN gibi dil modellerinde bütün metni temsil eden tek bir vektör oluşturulurken; attention mekanizmasında ise her bir cümleyi ayrı ayrı temsil eden bir düşünce vektörü oluşturulmaktadır. Böylece her bir cümlede nereye dikkat edileceği bilgisi, Decodere gönderilmektedir. Decoder ise kendisine gelen düşünce vektörü ve bir "start" ifadesi ile çalışmaya başlayarak, sırasıyla, girişte verilen cümlenin çevirisini yapmaktadır.

3.6. Konuşmadan Yazıya Çeviri (Speech To Text Translation)

Verilen sesli bir komutun metin haline dönüştürülmesi işine "speech to text" denilmektedir. Yoğun mühendislik gerektiren birçok işlemi kapsamaktadır. Bu amaç için HMM (Hidden Markov Model), akustik model gibi yaklaşımlar kullanılmaktadır. Bu çalışmada akustik model anlatılacaktır. Akustik model, derin öğrenme yaklaşımını

kullanan etkili bir yöntemdir. Yapılan çeviriler daha başarılıdır. Genel anlamda, sesin bir metne dönüştürülmesi işlemi aşağıdaki süreçlerden oluşmaktadır.

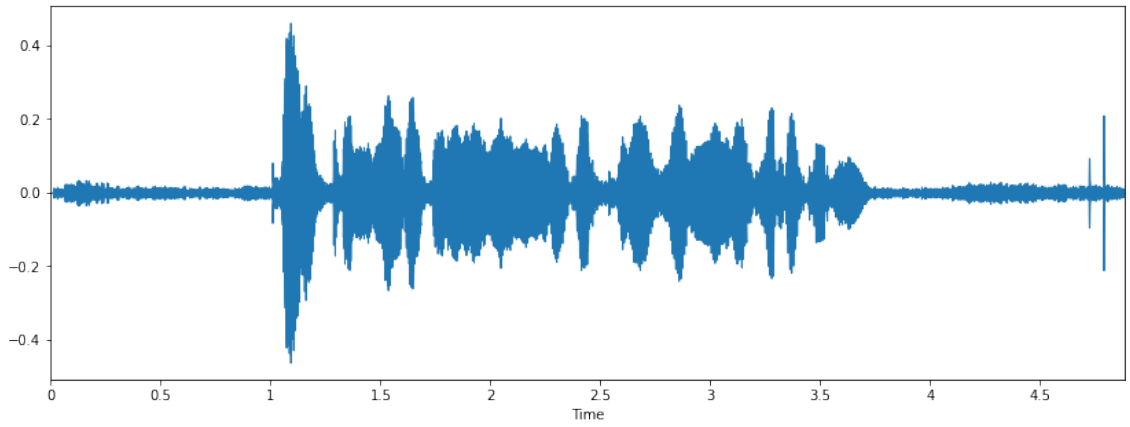
- Ağızdan çıkan titreşimler analog bir sinyaldir. Yönü ve şiddeti zamana göre değişkenlik göstermektedir. Bu ses sinyalleri, işlenebilmesi için dijital sinyale dönüştürülmelidir.

- Analog dijital dönüştürücüler, bu ses sinyallerini alır, ayrıntılı bir şekilde, gerekli filtreleme işlemlerini yapar.

- Alınan bu sesler, saniyenin yüzde veya binde biri küçüklüğünde segmentlere ayrılır. Bu segmentlerdeki ses sinyalleri ilgili fonemler ile ilişkilendirilir. Burada fonem, bir dildeki bir sözcüğü başka bir sözcükten ayırt eden ses birimini ifade etmektedir.

- Daha sonra fonemleri, cümlelerler ve sözcüklerle karşılaştıran bir dil modeli oluşturulur.

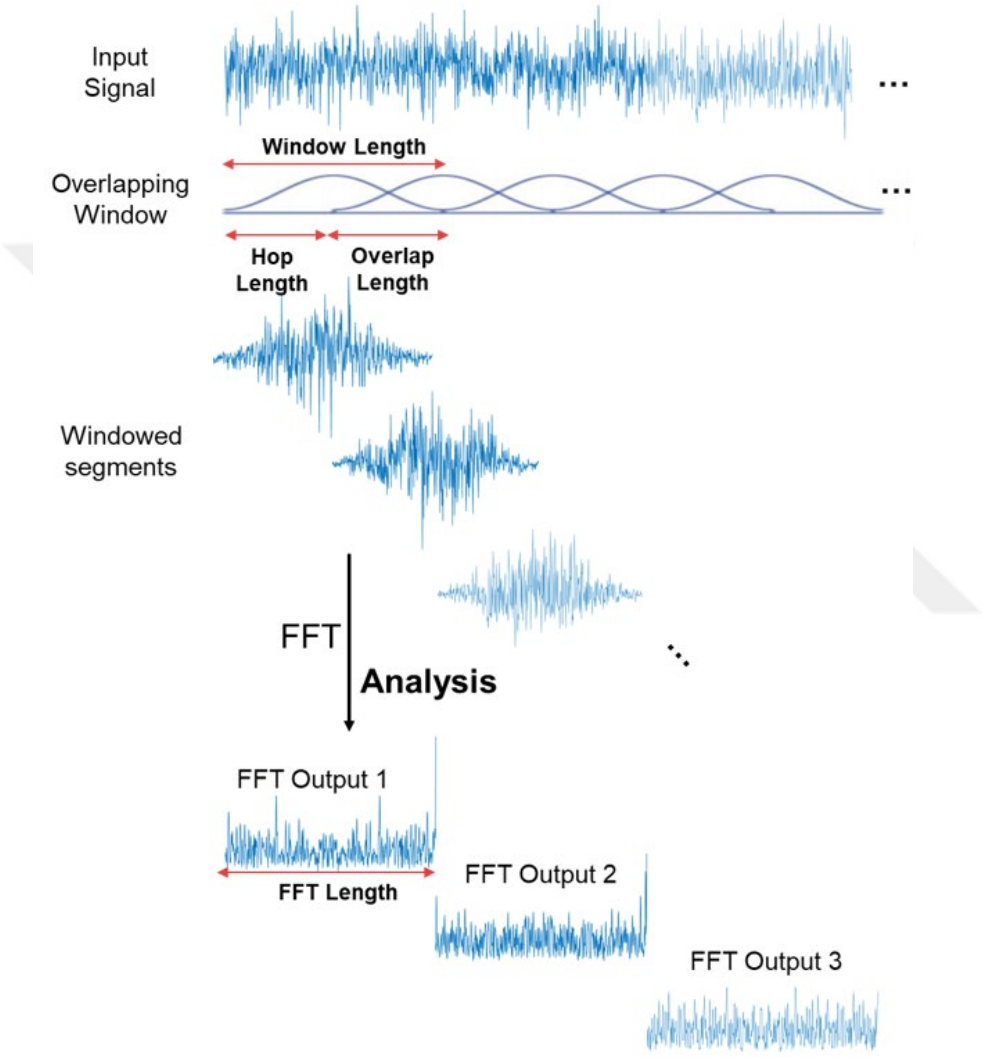
- Son olarak, modele verilen ses sinyalleri ilgili sese ait metni döndürür (*Konuşmayı Metne Dönüştürme Nedir? - Yeni Başlayanlar İçin Transkripsiyon Kılavuzu - AWS*, n.d.).



Şekil 3.16. Ses Frekansı.

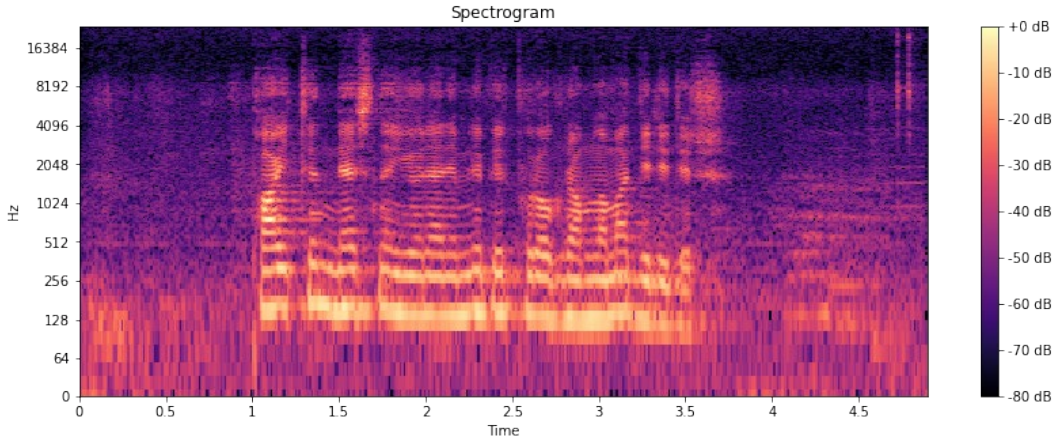
Şekil 3.16'da bir ses sinyali analog olarak gösterilmektedir. Bu şekilde analog olarak verilen sinyaller, bilgisayar tarafından, dijital sinyale çevrilmektedir. Ardından, ses sinyalindeki akustik olarak isimlendirilen öznitelikleri çıkartılmaktadır. Öznitelik

çıkarmadaki ana hedef, konuşmacıyı tanımlayan akustik özelliklerden ödün verilmeksizin konuşulan veriyi en iyi şekilde ifade etmektir. Ses verisini, sinyal düzleminden frekans düzlemine dönüştürmek için "short-time fourier transform" (STFT) yöntemi kullanılmaktadır. Bu yöntem ile ses, pencere dilimlerine ayrılarak işlenebilir hale getirilmektedir (Şekil 3.17).



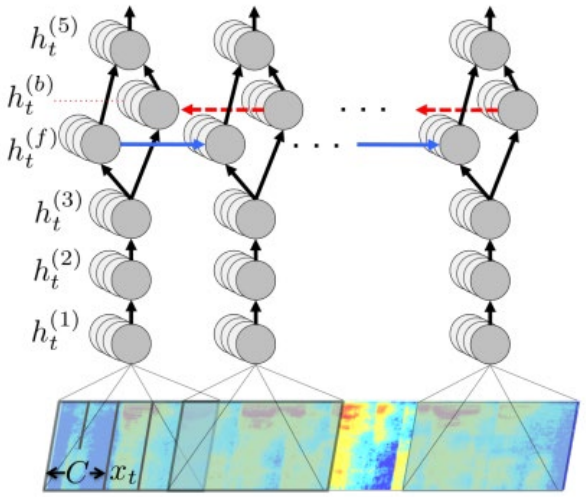
Şekil 3.17. Short-time Fourier Transform (STFT) Dönüşümü (Jeon et al., 2020).

STFT dönüşümü tamamlanan ses sinyallerinin derin öğrenme uygulamalarında kullanılabilmesi için spectrogramları çıkartılmakta ve böylece ses sinyalleri, bir görüntü şekline indirgenmiş olmaktadır. Şekil 3.18'de STFT dönüşümü yapılan bir ses sinyalinin spektrogramı görülmektedir. Aslında bu süreçte gerçekleştirilen işlem, ses sinyalinin STFT yardımıyla bir resme dönüştürülmesidir.



Şekil 3.18. Spectrogram.

Şekil 3.19'da küçük parçalara ayrılan spektrogram verilerinin her bir parçası RNN modeline verilmektedir. Dilimlenen küçük parçalara karşılık gelen fonem tahminleri ise çıktı olacaktır.



Şekil 3.19. Ses Sinyallerinde RNN Modelinin Yapısı ve Gösterimi (Hannun et al., 2014).

Burada bir, iki ve üçüncü katmanlar "fully connected layers" (tam bağlı katmanlar)'dan oluşmaktadır. Üçüncü katmandan sonra, veri RNN yapısına girdi olarak verilmektedir. RNN katmanındaki sağa ve sola doğru giden oklar, BiRNN (Bidirectional Recurrent Neural Network) yapısının kullanıldığını göstermektedir. Hem geçmiş hem de gelecekteki bilgiler hesaba katılarak bir tahmin yapılmaktadır. BiRNN'den sonra üretilen

iki adet çıktı toplanıp, lineer katman olan beşinci katmandan geçirilerek bir tahmin elde edilmektedir. Modeli eğitebilmek için ise çizelge 3.2'de bir kısmı verilen bir veri seti kullanılmaktadır. Veri setine bakıldığında, her bir örnek kelimenin nasıl bir fonem (ses) yapısından oluştuğu görülmektedir.

Çizelge 3.2. Phonome Veri Kümesi (*The CMU Pronouncing Dictionary*, n.d.).

Phonome	Example	Translation
AA	Odd	AA D
AE	At	AE T
AH	Hut	HH AH T
AO	Ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
ER	hurt	HH ER T

Girişte söylenen "hello" kelimesi spektrogramlara çevrilerek oluşturulan modele verilmektedir. Çıktı "hhhhhheeeeellllooooo" şeklinde olabilmektedir. Konuşmacının vurgusundaki fonem uzunluğu, mükerrer harflerle gösterilmektedir. Eğer "h" harfi uzatılarak söyleniyorsa, çıktı olarak verilen "h" harflerinin sayısı da fazla olacaktır. "hhhhhheeeeellllooooo" çıktısındaki birden fazla tekrarlanan harfler, tek harf olarak alınırsa, bu sefer çıktı "helo" olmaktadır. Bu noktada, "hello" beklerken "helo" çıktısı alınmış, yani bir sorunla karşılaşılmıştır. Yan yana gelen harflerde böyle bir problem yaşanmaktadır. Bu problemi çözmek için "Connectionist Temporal Classification Loss" (CTC Loss) denilen bir yöntem kullanılmaktadır. CTC ile her iki karakterin arasına bir epsilon karakteri koyularak bir ayırma işlemi gerçekleştirilmektedir. "Hello" kelimesi "h ε e ε l ε l ε o" şekilde bir yapıya bürünmekte ve ardından, epsilon karakterleri atılarak doğru sözcük elde edilmektedir.

3.7. Kullanılan Kütüphaneler

3.7.1. Tensorflow

Tensorflow, "Deep Learning" (Derin Öğrenme) uygulamalarında sıklıkla kullanılan bir kütüphanedir. Yapay zekâ uygulamalarının daha efektif bir şekilde gerçekleşmesini sağlamaktadır. Google'ın AR-GE (Araştırma Geliştirme) birimi tarafından yapılan çalışmalar neticesinde, 2015 yılının kasım ayında açık kaynak olarak dünyanın erişimine açılmıştır (Abadi et al., 2016). Tensorflowun esnek yapısı sayesinde, derin öğrenme uygulamaları kolay bir şekilde gerçekleştirilebilmektedir. Tensorflow, CPU (Central Processing Unit) ya da GPU (Graphical Processing Unit) kullanarak program işlemlerini yürütmektedir. Python programlama dili özelinde tasarlanmış olsa da C++, JavaScript, C#, R ve Java gibi birçok programlama dili tarafından kullanılabilmektedir. Bu kütüphanenin kullanılabilmesi için paketinin yüklenmesi gerekmektedir. Windows ortamında konsol ekranı açılıp "pip install tensorflow" komutu yazılarak kütüphane yüklenmektedir. Kütüphane yüklendikten sonra kullanılabilmesi için "import tensorflow as tf" kodunun yazılması gerekecektir. Buradaki "tf" takma bir ad olarak kullanılmıştır.

```
[1]  import matplotlib.pyplot as plt
     import tensorflow as tf
     import numpy as np
     import math
     import os
```

Şekil 3.20. Tensorflow Kütüphanesini İmport Etme.

```
# from tf.keras.models import Model  # This does not work!
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, GRU, Embedding,LSTM,Bidirectional
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Şekil 3.21. Tensorflow Kütüphanesinin Model Metodunu Kullanma.

Şekil 3.20'de kütüphanenin nasıl import edildiği kod üzerinde gösterilmektedir. Şekil 3.21'de ise kütüphanenin Model adındaki metodu çağrılmaktadır.

```
[68] model_train = Model(inputs=[encoder_input, decoder_input],outputs=[decoder_output])
```

Şekil 3.22. Eğitim İçin Model Oluşturma.

Şekil 3.22'de model metodu, kendine gelen parametreler doğrultusunda bir eğitim modeli oluşturmaktadır.

3.7.2. Keras

Keras, matematiksel işlemler, sayısal teknikler ve optimizasyon yöntemleri hakkında bir sinir ağı oluşturmada kullanılan, Pythonda yazılmış, bir derin öğrenme kütüphanesidir (Erhandı, 2020). Keras Tensorflow, Theano, R, Microsoft Cognitive Tolkit ile de beraber çalıştırılabilmektedir. Keras ile beraber bir model oluşturmak daha kolay hale gelmiştir. Özellikle derin öğrenme alanında çalışacak insanlar için çok büyük kolaylıklar sağlamaktadır. Çünkü yapılması gereken onlarca düşük seviyeli matematiksel hesaplamalardan geliştiricileri kurtarmaktadır. Bu sayede geliştiriciler, daha hızlı bir şekilde, detaylara dalmadan, yapay zekâ uygulamaları geliştirebilmektedirler. Kerasta derin öğrenme uygulamaları geliştirebilmek için genel olarak;

- Model oluşturma,
- Aktivasyon fonksiyonu ayarlama,
- Loss Fonksiyonu hesaplama,
- Layer oluşturma, gibi işlemlerin gerçekleştirilmesi gerekmektedir.

Keras, bu gibi işlemlerin basitçe yapılmasına fırsat vermektedir. Kerasta, yukarıda sayılan işlemleri gerçekleştirebilmek için, ilki "Sequential API" ikincisi ise "Functional API" olmak üzere iki adet API (Application Programming Interface) kullanılabilmektedir. Şekil 3.23'de, Encoder-Decoder yapısı içerisinde LSTM kullanılarak model oluşturulmuştur. Burada dikkat edilecek olursa katmanlar ayrı ayrı oluşturulmuştur. Katmanlar daha sonra birbiri arkasına bağlanmıştır. Bu şekilde oluşturulan bir yapıya fonksiyonel API denilmektedir. Şekil 3.24'te ise, model daha sade ve daha basit bir görüntüdedir. Model içine katman eklemek için, sadece "model.add()" metodunu

kullanmak yeterli olabilmektedir. Fakat üzerinde değişiklik yapılmak istendiğinde tam yetki vermemektedir. Bu şekilde kullanılan bir yapıya "Sequential API" denilmektedir.

```
[ ]  batch_size = 64  # batch size for training
     epochs = 1000  # number of epochs to train for
     latent_dim = 917  # latent dimensionality of the encoding space
```

```
[ ]  encoder_inputs = keras.Input(shape=(None, num_encoder_tokens))
     encoder = keras.layers.LSTM(latent_dim, return_state=True)
     encoder_outputs, state_h, state_c = encoder(encoder_inputs)
     encoder_states = [state_h, state_c]
```

```
[ ]  decoder_inputs = keras.Input(shape=(None, num_decoder_tokens))
     decoder_lstm = keras.layers.LSTM(latent_dim, return_sequences=True, return_state=True)
     decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
     decoder_dense = keras.layers.Dense(num_decoder_tokens, activation="softmax")
     decoder_outputs = decoder_dense(decoder_outputs)
```

```
[ ]  model = keras.Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

Şekil 3.23. Fonksiyonel API Kullanımı.

```
# Hyperparameters
learning_rate = 0.003

# Build the layers
model = Sequential()
# Embedding
model.add(Embedding(english_vocab_size, 256, input_length=input_shape[1],
                    input_shape=input_shape[1:]))
# Encoder
model.add(Bidirectional(LSTM(410)))
model.add(RepeatVector(output_sequence_length))
# Decoder
model.add(Bidirectional(LSTM(441, return_sequences=True)))
model.add(TimeDistributed(Dense(500, activation='relu')))
model.add(Dropout(0.5))
model.add(TimeDistributed(Dense(english_vocab_size, activation='softmax')))
model.compile(loss=sparse_categorical_crossentropy,
              optimizer='adam',
              metrics=['accuracy'])
```

Şekil 3.24. Sequential API Kullanımı.

3.7.3. Numpy (Numerical Python)

Numpy, diziler ile çalışmak için kullanılan bir Python kütüphanesidir. Ayrıca lineer cebir, matris hesaplamaları ve karmaşık fonksiyon hesaplamalarında sıklıkla kullanılmaktadır. Bu kütüphane 2005 yılında Travis Oliphant tarafından oluşturulmuştur (Virtanen et al., 2020). Python'da kullanılan diziler genel olarak yavaş çalışmaktadır. Dizilerin aksine Numpy kütüphanesi, makine diline yakın bir dil olan C++ dilinde yazıldığı için hızlı çalışmaktadır. Numpy, geleneksel Python listelerinden 50 kat daha hızlı çalışmaktadır. Özellikle veri bilimi, derin öğrenme, makine öğrenmesi ve daha genel manada yapay zekâ uygulamalarında sıklıkla kullanılmaktadır. Örneğin, görüntü işleme uygulamasındaki bir resim, satır ve sütunlarında sayıların olduğu pikseller ile ifade edilmektedir. Buradaki piksellerin hesaplamalarda birbirleri ile çarpılması gerekebilmektedir. İşte bu çarpma işlemleri, Numpy ile kolay ve hızlı bir şekilde yapılabilmektedir. Bir Numpy kütüphanesini kullanabilmek için, öncelikle, konsol ekranını açıp "pip install numpy" komutu ile kütüphanenin yüklenmesi, ardından, programa "import numpy as np" kodu yazılarak kütüphanenin çağrılması gerekmektedir.

```
import numpy as np
a = [[2, 3], [5, 6]]
b = [[1, 5], [0, 6]]
x = np.array([[0, 0], [0, 0]])
for i in range(0,2):
    for j in range(0,2):
        for k in range(0,2):
            x[i][j] = a[i][k]*b[k][j] + x[i][j];
print(x)
```

Şekil 3.25. Numpy Olmadan Matris Çarpımı.

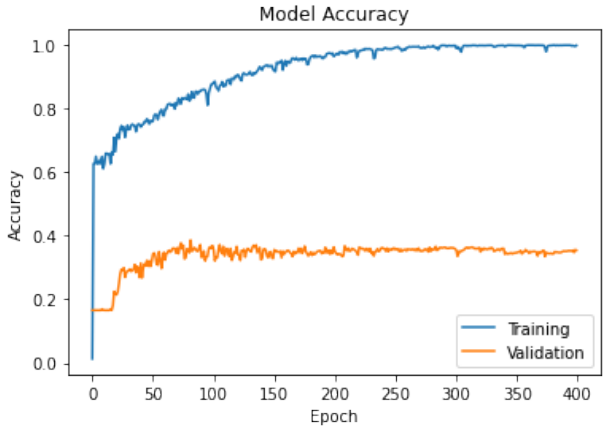
```
import numpy as np
a = np.array([[2, 3], [5, 6]])
b = np.array([[1, 5], [0, 6]])
print(np.dot(a,b))
```

Şekil 3.26. Numpy Kullanılarak Matris Çarpımı.

Şekil 3.25'te, Numpy kütüphanesi kullanılmadan, iki ayrı matrisi çarpmak için içe içe iki for döngüsü kullanmak gerektiği görülmektedir. Şekil 3.26'da ise Numpy kütüphanesi kullanılarak bu işlemin daha kısa ve daha anlaşılır bir şekilde yapılması sağlanmıştır. Sonuç olarak, Numpy hem matematiksel işlemlerde hız kazandırmakta hem de daha sade bir kod görüntüsü sunmaktadır.

3.7.4. Matplotlib

Matplotlib kütüphanesi, Pythonda veri görselleştirmek için kullanılmaktadır. John Hunter tarafından 2002 yılında tanıtılmıştır. Yapılan uygulamaların sonuçları histogram, daire, bar ya da lineer grafikler yardımı ile kolayca gösterilebilmektedir. Dolayısıyla uygulamaların bilimsel sonuçları, göze hitap etmekte ve daha anlaşılır bir hale gelmektedir. Genellikle bu kütüphane kullanılarak iki boyutlu grafikler çizilmektedir. Bu kütüphanenin de kullanılabilir olması için, öncelikle Windows ya da Linux ortamında konsol ekranı açılıp "pip install matplotlib" komutu ile yüklenmesi gerekmektedir. Ardından "import matplotlib as plt" kodu ile programa import edilen modül kullanılabilir olacaktır. Genellikle yapay zekâ uygulamalarında, bu kütüphaneden sıklıkla faydalanılmaktadır. Örneğin, bir model oluşturduktan sonra modelinizin doğruluk ve kayıp değerlerini grafiksel olarak görmek isteyebilirsiniz. Şekil 3.27'de model doğruluğuna ait bir grafik görülmektedir.



Şekil 3.27. Matplotlib ile Oluşturulmuş Model Doğruluk Grafiği.

3.7.5. Speech recognition (Konuşma Tanıma)

Speech Recognition, Pythonda kullanılan ücretsiz bir kütüphanedir. Verilen sesli komutların yazıya çevrilmesini sağlar. Burada sesli komutlar, mikrofondan canlı konuşma veya bir ses dosyası içinde kaydedilmiş şekilde verilebilmektedir. Verilen ses komutları, Google sunucularına gönderilerek, metin haline çevrilmiş şekilde geri döndürülmektedir. Speech Recognition kütüphanesini yüklemek için de diğer kütüphanelerde olduğu gibi, konsol ekranına "pip install speech_recognition" komutunun yazılması ve "import speech_recognition as sr" kodu ile çağrılarak projeye dâhil edilmesi gerekmektedir.

```
59  import speech_recognition as sr
60  r = sr.Recognizer()
61  with sr.Microphone() as source:
62      print("Birşeyler Söyle!")
63      audio = r.listen(source)
64      r.adjust_for_ambient_noise(source)
65  try:
66      soyledigim=r.recognize_google(audio,language="tr-TR")
67      print("You said: " + soyledigim)
68  except sr.UnknownValueError:
69      print( "Google Speech Recognition Sesini Tanımlayamadı")
70  except sr.RequestError as e:
71      print( "Google Speech Recognition Servisinden sonuç istenemedi; {0}".format(e))
72
```

Şekil 3.28. Konuşma Tanıma Kütüphanesinin Kullanımı.

Şekil 3.28'deki kodun 63'üncü satırında "listen" komutu ile dış ortamdaki sesler dinlenilmektedir. Bir alt satırda ise oluşabilecek gürültüler azaltılmaktadır. 66'ncı satırda ise gürültüden arındırılmış sesler Google sunucularına gönderilmektedir. Geriye sesin yazıya çevrilmiş hali döndürülmektedir. Burada "language=tr-TR" olarak seçilen parametre, konuşulan dilin Türkçe olduğunu ifade etmektedir. Çeviri işlemi eş zamanlı gerçekleşmemektedir. Sesler toplu olarak alınıp tek seferde Google sunucularına gönderilmektedir. Bu yüzden bir miktar beklenilmesi gerekmektedir.

3.7.6. OpenAI

OpenAI kütüphanesi kullanılarak birçok uygulama geliştirmek mümkündür. İlk zamanlarında bu kütüphaneden yararlanabilmek için OpenAI firmasına kütüphanenin hangi gerekçeler ile kullanılacağına dair bir yazı gönderilmesi gerekmiştir. Eğer onay

verilirse bu kütüphane kullanabilmiştir. Şu an itibari ile herkesin kullanımına sunulmuştur. Ancak kütüphaneye sınırlı bir erişim imkânı verilmiştir. Çünkü bu model, 175 milyar parametreden oluşmaktadır. Bu haliyle kullanmak, yasal olmayan sonuçlar doğurabilecektir. Ayrıca modelin, aylık bir token limiti bulunmaktadır. Yani çeviri yapılabilecek içerikler sınırlıdır. Sınırsız bir şekilde kullanabilmek için ücret ödenmesi gerekmektedir. Kullanıcılara birçok içerik sunulmaktadır. Bunların bazıları aşağıdaki gibi sıralanabilir:

- Soru-cevap sistemleri
- Metin özetleme
- Metni program koduna çevirme
- Bir dilden başka bir dile çeviri
- Gramer hatalarını düzeltme
- Verilen kod bloğunu açıklama
- Koddaki hataları bulma
- Birçok dilde program yazabilme (Java, Python, C# vb.)
- Arkadaşça chat yapabilme
- Tweet'leri sınıflandırma

Kütüphanenin kullanımı oldukça basittir. OpenAI kütüphanesi, konsol ekranına "pip install openai" komutu yazılarak yüklenmektedir. Uygulamada kullanılabilmesi için ise "import openai" kodu yazılmalıdır. Şekil 3.29'da verilen bir cümle, Python koduna çevrilmek istenmektedir. "Create a list" ifadesi modele verilerek Pythonda bir liste oluşturulmak istenmektedir. Bunu yapabilmek için ise 75'inci satırdaki gibi bir API olması gerekmektedir. OpenAI web sitesine kayıt olunduktan sonra bu API elde edilebilmektedir. 77'nci satırda kullanılacak engine (motor) belirlenmesi gerekiyor.

Burada birden fazla motor belirlenebiliyor. Sunulan bu çalışmada "text-davinci-002" motoru belirlenmiştir. Çünkü bu motor özelliği GPT-3 modelini desteklemektedir. Yaklaşık 4000 token desteği vermektedir (*Models - OpenAI API*, n.d.). 78'inci satırdaki promp kısmında ise "convert to python code" yazısı ile modele verilen cümleyi Python koduna çevirmesi gerektiği söylenmektedir. Bu şekilde "max_tokens", "frequency_penalty" gibi ayarları da yapıldıktan sonra kod çalıştırıldığında Python dilinde bir liste oluşturulmaktadır.

```
73  import openai
74  cevrilmisCumle="create a list"
75  openai.api_key = "sk-HyfewfewPHS9fefe3rlT3BlbkFJMyJdUeeSIGHTrwVt6S"
76  response = openai.Completion.create(
77  engine="text-davinci-002",
78  prompt="""convert to python code\nExample:"""+cevrilmisCumle+"\nOutput: """,
79  temperature=0,
80  max_tokens=100,
81  top_p=1,
82  frequency_penalty=0.2,
83  presence_penalty=0
84  )
85
86  for i in response["choices"]:
87      sonuc=i['text'].strip()
```

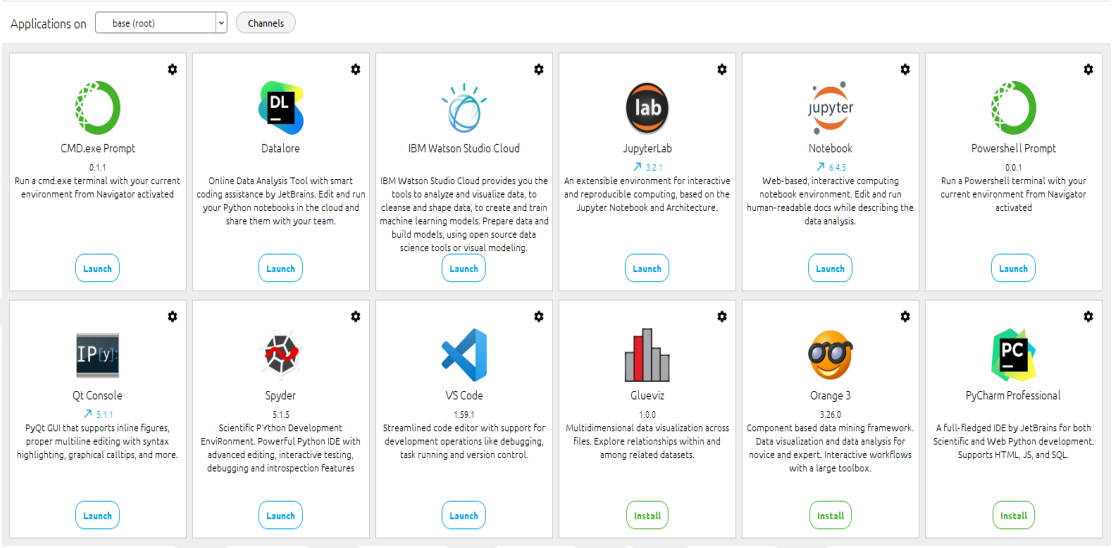
Şekil 3.29. OpenAI Kütüphanesi Kullanımı.

3.8. Kullanılan Teknolojiler

3.8.1. Anaconda Navigator

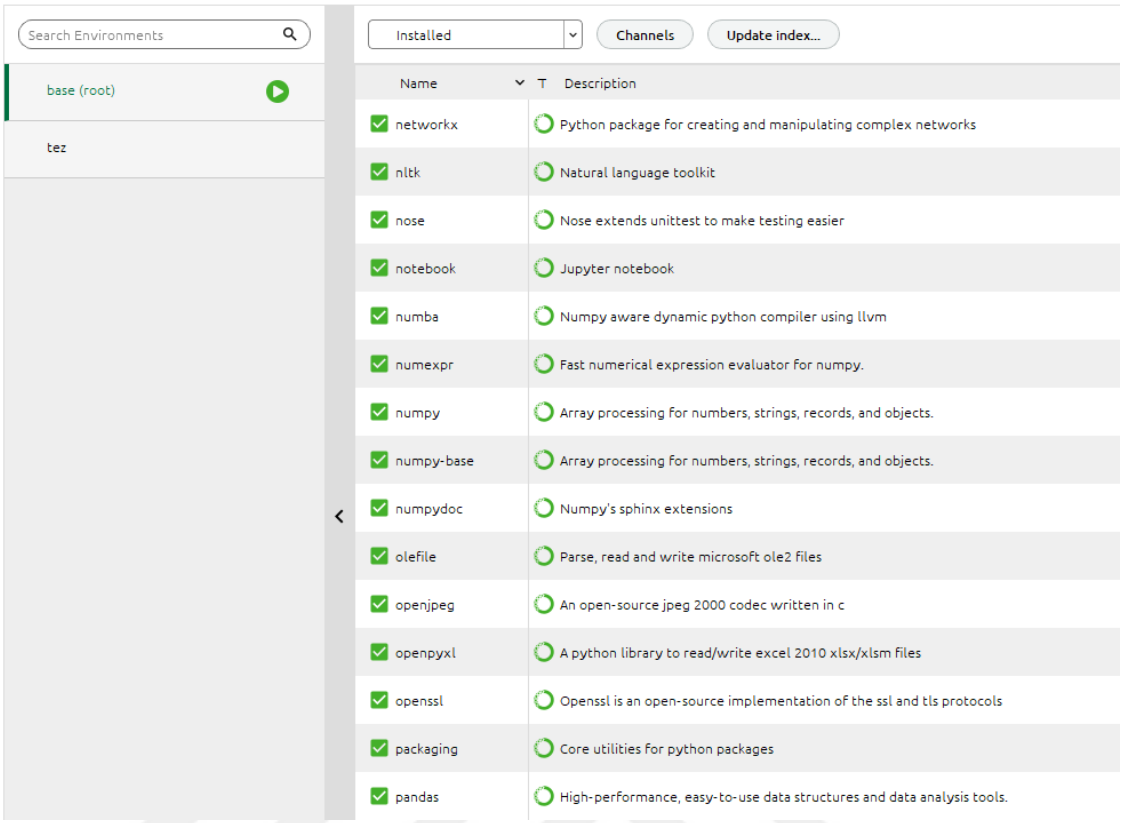
Anaconda, veri bilimi, makine öğrenmesi ve derin öğrenme gibi yapay zekâ uygulamaları için özellikle ortaya çıkartılmış bir geliştirme aracıdır. Açık kaynak kodlu ve ücretsiz bir uygulamadır. Bu uygulama bir GUI (Graphical User Interface) hizmeti sunmaktadır. Paketleri yüklemek için komutlara ihtiyaç duyulmaz. Kullanıcı, uygulamanın sağladığı ara yüz sayesinde bu gibi işlemleri kolaylıkla yapabilmektedir. Bu platformda, Python için gerekli olan temel paketler hazır bir şekilde gelmektedir. Örneğin, Python geliştirmek için ihtiyaç duyulan Juptyer Notebook, Pycharm, Spdyer, Vscode gibi uygulamaların birçoğu Anaconda ile yüklü olarak gelmektedir. Ayrıca bir proje gerçekleştirildiğinde, ilgili ve gerekli bütün kütüphanelerin bir ortam içerisinde tutulması gerekmektedir. Bu yüzden environment (ortam) oluşturulur. Anaconda ile environmet oluşturmak için komut

kullanmaya gerek yoktur. Kod kullanmadan bu işlem gerçekleştirilebilir. İçerisinde hazır olarak Numpy, Matplotlib, Sklearn, Keras, Pandas, Tensorflow, Scipy, Sqlite ve benzeri birçok kütüphane hazır olarak gelir. Bu sayede, makine öğrenmesi uygulamaları çok hızlı ve etkin bir şekilde gerçekleştirilebilir.



Şekil 3.30. Anaconda Navigator.

Şekil 3.30'da Anaconda Navigator içerisinde hazır olarak gelen birçok paket görülmektedir. Şekil 3.31'de ise base (root) ortamında bulunan kütüphanelerin listesi gösterilmiştir. Burada base (root) ortamı, Anacondanın içerisinde varsayılan olarak gelir. İçindeki paketler kurulum esnasında yüklenmektedir.

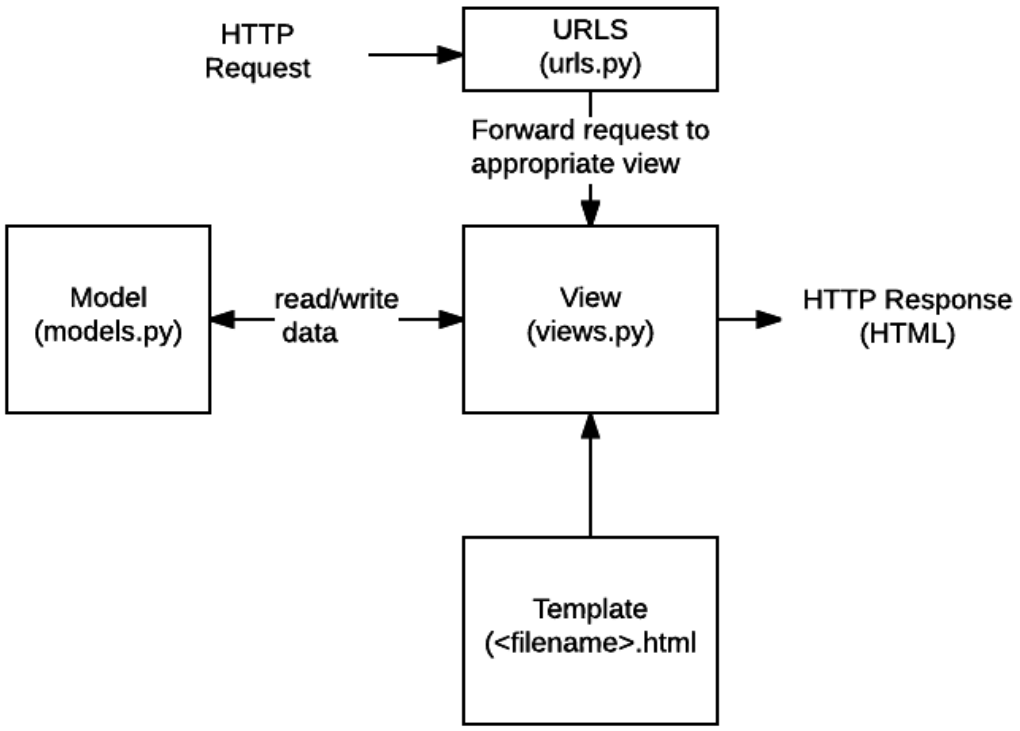


Şekil 3.31. Anaconda Navigator Yüklü Paketler.

3.8.2. Django framework

Django, 2003-2005 yılları arasında, ABD'de bir gazetenin web sayfasından sorumlu bir teknik ekip tarafından Python programlama dili kullanılarak geliştirilmiştir (Plekhanova, 2009). Framework açık kaynak kodludur. Birçok platformda çalıştırılabilir. Örneğin Windows, Linux, MacOs gibi. Bu Frameworkun geliştirmesinin sebebi, sürekli aynı kalıpları kullanmaktan kaynaklanan vakit kaybını önlemektir. Örneğin, web sayfasında bir admin paneli yaptığınızı düşünün. Admin panelleri, genel olarak bütün web sayfalarında ortaktır. Bu paneli, her web sayfasında ilk adımdan itibaren tekrar oluşturmak vakit kaybına neden olacaktır. Veri tabanına bağlantı kurma, URL (Uniform Resource Locator) yolu oluşturma gibi işlemler de tekrarlı bir şekilde yapılmaktadır. Django Framework, sürekli olarak yapılan bu işlemleri tekrar etmemek için oluşturulmuştur. Framework ile yapılan işi kolaylaştıran bütün işlemler bir çatı altına alınmaktadır. Bu Frameworkun popüler olmasının ana sebebi, Python programlama

dilinde yazılmış olmasıdır. Djangonun popüler olmasının diğer bir sebebi ise arkasında bulunan topluluğun sayısıdır. Dünya genelinde çok gelişmiş bir yapılanmaya sahiptir. Bu yüzden kendini değişen şartlara ve durumlara çok hızlı bir şekilde adapte edebilmektedir. Örneğin, Frameworkun bir bölümünde hata oluştuğunda, topluluk kısa sürede hatayı bulup düzeltebilmekte ve değişen şartlara uygun olarak kodlar anında revize edilebilmektedir. Geleneksel web sitelerinde, bir web tarayıcısı HTTP (Hyper Text Transfer Protocol) isteklerini "POST" ya da "GET" metotları ile alır. Alınan değerler, önem derecesine göre veri tabanında bir sorgu işlemine tabi tutulur. Gerçekleşen sorgunun sonucu olarak geriye dönen değer, web sayfasında ilgili yer tutucular vasıtası ile gösterilir.



Şekil 3.32. Django Framework İç Yapısı (*Django Introduction - Learn Web Development | MDN*, n.d.).

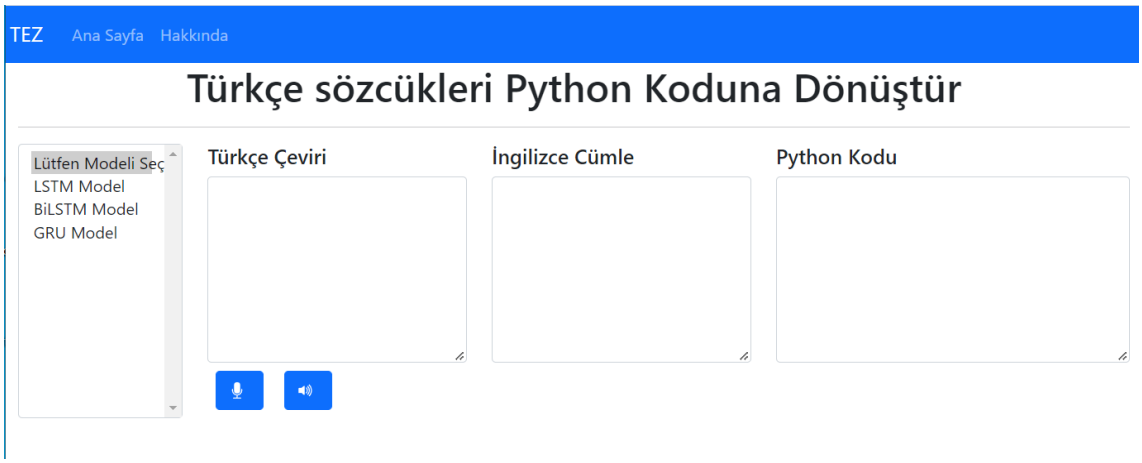
Şekil 3.32'de yukarıda geleneksel bir web sitesinin yapımı için sayılan aşamalar, bir Python dosyasının içerisinde modüler olarak kullanıcılara sunulmuştur.

Url: HTTP istekleri istek URL'sine göre ilgili View'a yönlendirmek için urls.py dosyası kullanılmaktadır. Hatta URL kalıpları yanında bir sayı ya da dizi olacak şekilde verileri alıp yine başka bir View'a iletilebilir. Bu şekilde istenilen URL istenilen bir View'a dinamik bir şekilde kolayca yönlendirilebilecektir.

View: URL'den gelen istekler alınıp ilgili işlemler yapılarak indeks sayfasına geri döndürme işlemi yapılır.

Models: Veri tabanındaki kayıtları yönetmek için hazır mekanizmalar sağlayan dosyadır. Bu sayede veri tabanı ile ilgili olan listeleme, silme, güncelleme, ekleme gibi işlemler daha kolay bir şekilde gerçekleştirilebilecektir.

Templates: Bir HTML dosyasını daha verimli bir şekilde yönetmek için birtakım kolaylıklar sağlar. Örneğin, sitede bulunan "header" ve "footer" bölümleri genel olarak sabittir. Değişen kısım, bu ikisi arasında kalan bölümdür. "Header" ve "footer" yapıları sabit tutulup diğer bölüm dinamik bir şekilde değiştirilebilir. Djangonun kullanılabilmesi için konsol penceresine "pip install django" komutu yazılarak paketin yüklenmesi gerekmektedir. Bu işlemi bir environment (çevre) oluşturarak yapmak gerekecektir. Çünkü Djangonun kendi içerisinde çok sayıda farklı paketler mevcuttur. Dolayısıyla bütün bu paketler bir environment içinde olmalıdır. Projeyi çalıştırmak için ise "python manage.py runserver" komutunu yazmak yeterli olacaktır.



Şekil 3.33. Django Framwork'ü Kullanarak Oluşturulan Arayüz.

Şekil 3.33'te bu çalışmaya konu proje için oluşturulan ara yüz gösterilmektedir. Sol kısımdan çeviri işleminin gerçekleştirileceği model seçildikten sonra mikrofon butonuna basılarak verilen cümlenin Python kodu bulunmaktadır.

3.8.3. PyQt5 framework

PyQt, Pythonda Qt GUI çerçevesinin kullanılmasına izin veren bir kitaplıktır. Qt'nin kendisi C++ dili ile yazılmıştır. Python kullanılarak C++'ın sağladığı hızdan ödün verilmeksizin ivedilikle uygulamalar geliştirilebilmektedir. Qt şirketi, Python 2 ve 3 sürümlerinde kullanılmak üzere, kullanıcılara bir GUI hizmeti sunmaktadır. Bütün işletim sistemlerinde çalışmaktadır. Buna Android telefonlar da dâhildir. PyQt, ticari ve açık kaynak olmak üzere iki adet lisansa sahiptir. Eğer yazılan programdan kâr amacı güdülmeyecekse ücretsiz olan lisans kullanılabilmektedir. Fakat kâr elde etmek amaçlanıyorsa ücretli lisansının kullanılması gerekmektedir. PyQt'nin, PyQt4 ve PyQt5 olmak üzere iki adet sürümü mevcuttur. Konsol ekranına "pip install PyQt5" komutunun yazılması bu kitaplığı kullanmak için yeterli olacaktır. Bu çalışmaya konu modele PyQt5 kütüphanesi kullanarak bir arayüz gerçekleştirilmiştir. Bu sayede proje, kullanıcılar tarafından kolaylıkla kullanılabilir hale gelmiştir.



Şekil 3.34. PyQt5 Kullanılarak Oluşturulan Ara yüz.

Şekil 3.34'te çalışmaya konu projenin tasarımı PyQt5 kütüphanesi kullanılarak gerçekleştirilmiştir. LSTM, GRU ve BiLSTM gibi üç farklı model ile çeviri işlemi yapılmasına imkân veren seçenekler mevcuttur. Hangi model ile çeviri yapılacaksa o model seçilmektedir. Daha sonra seçilen modele göre mikrofon butonuna basılarak çevrilmesi istenen Python kodu söylenmektedir. Bunun yanında, mikrofondan konuşmak istenmediği durumlarda yazı butonuna basılarak sadece yazılan komutların çevrilmesi de sağlanabilmektedir.

3.9. Deneysel Çalışmalar

Deney, LSTM, BiLSTM ve GRU hücrelerinin bir Encoder-Decoder yapısı içerisinde kullanılmasıyla oluşturulmuştur. Üç farklı model için eğitimler gerçekleştirilerek performans değerlendirilmesi yapılmıştır. Model, sözlü olarak verilen bir Türkçe cümleyi ilk önce İngilizceye çevirmekte; ardından İngilizceye çevrilen bu cümle, GPT-3 modeline girdi olarak verilmektedir. GPT-3 modeli, kendine gelen bu İngilizce cümlenin Python dilindeki söz dizim karşılığını bulmaktadır. Burada iki aşamalı bir işlem tasarlanmasının sebebi, GPT-3 modelinin İngilizce olarak eğitilmesidir. GPT-3 modeli, İngilizce olarak eğitildiği için daha başarılı sonuçlar verecektir. GPT-3 modeli, sadece bir API olarak kullanılmıştır. Çünkü bu modelin tam kullanım hakkı henüz verilmemiştir. Burada ana hedef LSTM, BiLSTM ve GRU modellerinin MT'de ne kadar başarılı olduklarını tespit etmektir.

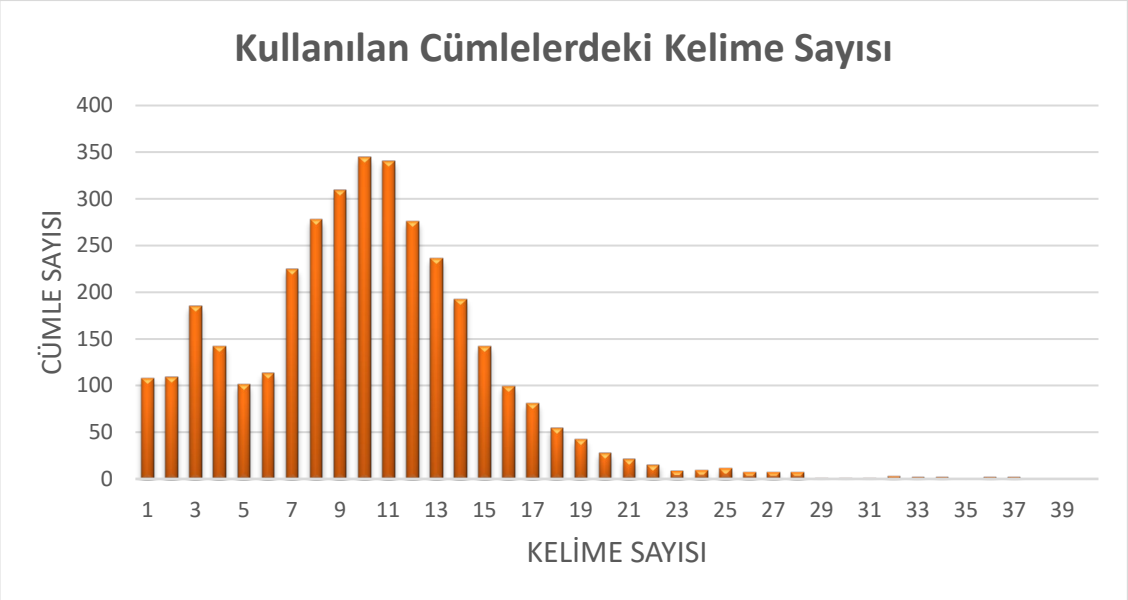
3.9.1. Veri kümesi

Model oluşturmak için github sitesinden indirilen bir veri kümesi kullanılmıştır. Bu sitedeki veri kümesi, İngilizce bir cümle ve bu İngilizce cümleye karşılık gelen Python kodundan oluşmaktadır. Yapılmak istenen şey ise Türkçe bir cümlenin İngilizceye çevrilmesidir. Bunun için İngilizce dilindeki Python koduna karşılık gelen sözlü ifadeler alınıp etiket olarak Python kodu yerine Türkçe çevirisi yazılmıştır. Bu veri kümesinde 3014 adet veri bulunmaktadır. 491 adet veri ise sıfırdan oluşturularak bu veri kümesine eklenmiştir. Sonuç olarak, veri kümesinde 3505 adet veri oluşturulmuştur.

Çizelge 3.3. Veri Kümesi

Türkçe Cümle	İngilizce Cümle
iki sayı eklemek için bir Python programı yazın	write a python program to add two numbers
kullanıcı tarafından sağlanan iki sayıyı eklemek ve toplamı döndürmek için bir Python fonksiyonu yazın	write a python function to add two user provided numbers and return the sum
üç sayı arasında en büyüğünü bulmak ve yazdırmak için bir program yazın	write a program to find and print the largest among three numbers
üç sayı arasında en küçüğü bulmak ve yazdırmak için bir program yazın	write a program to find and print the smallest among three numbers
verilen iki listeyi bir araya getirmek için bir Python fonksiyonu yazın	write a python function to merge two given lists into one
bir sayının asal olup olmadığını kontrol etmek için bir program yazın	write a program to check whether a number is prime or not
belirli bir sayının faktörlerini basan bir Python fonksiyonu yazın	write a python function that prints the factors of a given number
bir sayının faktöryelini bulmak için bir program yazın	write a program to find the factorial of a number
bir sayının negatif, pozitif veya sıfır olup olmadığını yazdırmak için bir Python fonksiyonu yazın	write a python function to print whether a number is negative, positive or zero
belirli bir sayının çarpım tablosunu yazdırmak için bir program yazın	write a program to print the multiplication table of a given number
ikinin katlarını yazan bir Python fonksiyonu yazın.	write a python function to print powers of 2, for given number of terms

Çizelge 3.3'te oluşturulan veri kümesindeki Türkçe cümleler ve bu Türkçe cümlelere karşılık gelen İngilizce cümleler gösterilmiştir. Buradaki İngilizce cümleler, etiket olmaktadır. Yani model eğitilirken, girdi olarak Türkçe cümleleri alacak ve bu cümlelere karşılık gelen etiketler ile eşleştirme yapacaktır. Veri kümesindeki cümlelerin kaç kelimeden oluştuğu şekil 3.35'te gösterilmiştir.



Şekil 3.35. Veri Kümesi Kelime Sayı Dağılımları.

Şekil 3.35'e göre veri kümesinde bulunan her bir cümledeki kelime sayısı 7-15 arasındaki bir bölgede yığılmıştır. N-gram skorlarına bakıldığında, cümlelerin boyutu arttıkça MT'deki başarım oranının azaldığı görülmektedir. LSTM, GRU ve BiLSTM gibi yapılarda, cümledeki kelime miktarı arttıkça daha az başarılı çeviriler yapıldığı saptanmıştır. Örneğin, çizelge 3.4'te n-gramlara göre BLUE (Bilingual Evaluation Understudy) skorları hesaplanmıştır. Blue skorlarının gram sayısı arttıkça skor değerinin düştüğü gözlemlenmiştir. Bu gözleme istinaden cümledeki kelime sayısı arttıkça başarım oranının düştüğü sonucuna varılmıştır. Bu nedenle kelime sayıları, şekil 3.35'deki grafiğe bakılarak çok uzun tutulmamıştır.

Çizelge 3.4. LSTM İçin N-Gram BLUE Skorları.

Model	1-gram	2-gram	3-gram	4-gram
LSTM	0,76	0,73	0,73	0,69

3.9.2. Eğitim süreci

Modelin eğitimi, sağladığı GPU desteği nedeniyle, Google Colaboratory kullanılarak gerçekleştirilmiştir. Model, GPU desteği sayesinde, daha hızlı bir şekilde eğitilmiştir.

3.9.3. Lstm için eğitim süreci

Modeli eğitim sürecine getirebilmek için ilk başta yapılması gereken bir dizi işlem adımı bulunmaktadır. Bunlar aşağıda sıralanmıştır:

- Veri setinden verilerin okunması,
- Okunan verilerin tokenlerine ayrılması,
- Tokenlerine ayrılan cümleler için padding işlemlerinin yapılması,
- Embedding katmanının oluşturulması,
- Encoder-Decoder modelinin LSTM katmanları ile inşa edilmesi,

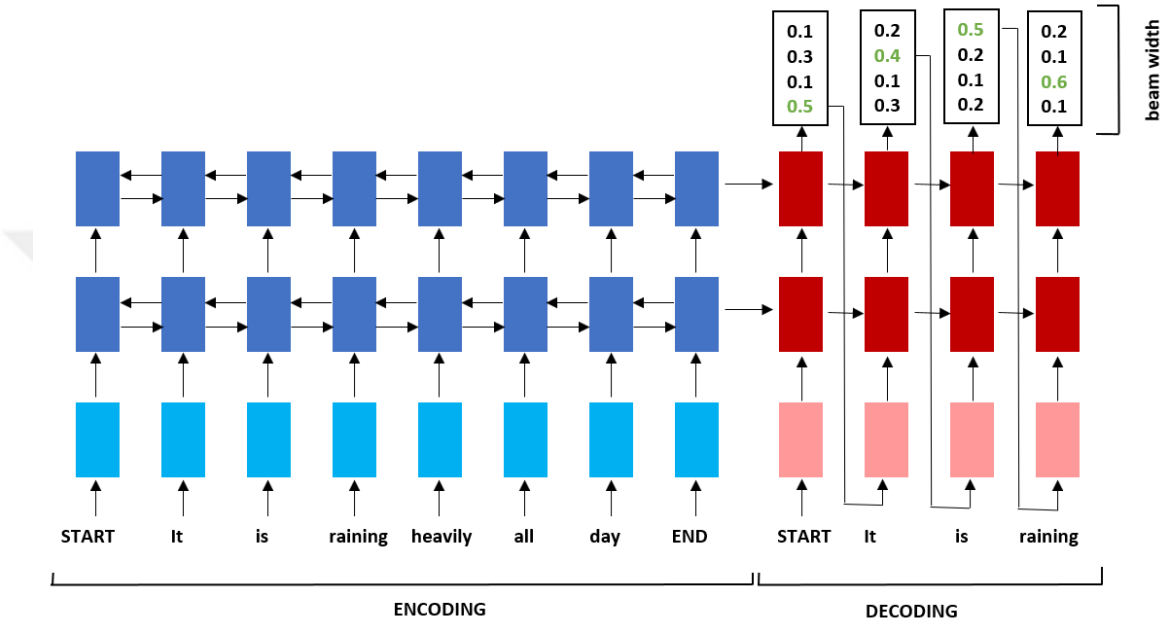
- Uçtan uca modelin birbirine bağlanması,
- Modelin compile edilmesi (Optimizasyon ve kayıp fonksiyonlarının belirlenmesi),
- Modelin fit edilmesi (Eğitim gerçekleştirilmesi),
- Modelin test edilmesi.

Modele veri setinden veriler okunduktan sonra, okunan verilerin tokenlerine ayırma işlemi gerçekleştirilmektedir. Modele girilen her bir kelime karakter katarlarından oluşmaktadır. Bilgisayar sayısal verilerle çalıştığı için bu string verilerin sayısallaştırılması gerekmektedir. Örneğin, eski bir yöntem olan "one hot encoding" ile bir cümle aşağıdaki gibi tokenize edilebilmektedir.

mehmet=[16,6,10,16,6,24]
mehmet=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]

Mehmet kelimesindeki "m" harfi, alfabede 16. sırada yer almaktadır. Bu şekilde sıra numarasına karşılık gelen yere "1" sayısı konularak [6x29]'luk bir matris oluşturulmuştur. Karakter katarları, bilgisayarların anlayabileceği formata bu şekilde dönüştürülmektedir. Tokenize işleminden sonra "padding" işlemi uygulanmıştır. "Padding" işlemi ile modele verilecek kelimelerin boyutları eşitlenmektedir. Çünkü her bir cümle farklı uzunluktaki kelimelerden oluşmaktadır. Encoder modeline giriş yapılacak cümleler, bu şekilde eşit bir uzunluğa getirilmektedir. Ardından "embeding" işlemi gerçekleştirilmektedir. Bu işlem ile "padding" işleminden geçirilen cümlelerin "embeding" matrislerinin çıkartılması sağlanmaktadır. "Embeding" matrisi, bir kelimenin uzayda diğerlerine ne kadar yakın olduğu ile alakalı bir durumdur. Birbirine benzer kelimeler uzayda vektörel olarak daha yakınken, benzer olmayan kelimeler ise

vektörel olarak birbirinden daha uzaktadırlar. Burada model için daha önceden eğitilmiş olan "embeding" matrisleri kullanılabilir. Ancak çalışmaya konu bu modelde ön eğitimli "embeding" matrisleri kullanılmamıştır. Daha sonra "Encoder-Decoder" modeli inşa edilmektedir. "Encoder-Decoder" modeli şekil 3.36'da görüleceği üzere kodlayıcı ve kod çözücü adı verilen iki bloktan oluşmaktadır (Szucs & Huszti, 2019).



Şekil 3.36. Encoder-Decoder Model İç Yapısı (Szucs & Huszti, 2019).

Encoderin içeriği, RNN hücrelerinin bir araya gelmesiyle oluşmuştur. Burada yalın bir şekilde RNN kullanmanın dışında, LSTM ve GRU hücreleri de kullanılarak "Encoder-Decoder" yapısı oluşturulmaktadır. "Encoder" girdi olarak aldığı cümlelerin bir özeti olan "context" (içerik vektörü) vektörünü üretmekte ve bu vektör, Decoder modeline gönderilerek bir tahmin işlemi gerçekleştirilmektedir. "Encoder-Decoder" yapılarının kullanıldığı bu mimariye "seq2seq" denilmektedir. Sonraki aşamada model uçtan uca birbirine bağlanmaktadır. Eğitim için Encodere giren Türkçe veri ile Decodere giren İngilizce veri ele alınmakta ve buna karşılık, etiket verisi için Decoderden çıkan veri kullanılmaktadır. Burada Decoder giriş verisi ile Decoder çıkış verisi aynıdır. Tek fark Decoder giriş verisinin bir "start" parametresi ile başlıyor olmasıdır.

```
[ ]  history=model.fit(x=[encoder_input_data,decoder_input_data],y=decoder_output_data,
                       batch_size=512,
                       epochs=500,
                       validation_split=0.2
                       )
```

Şekil 3.37. Encoder-Decoder Modelinin Uçtan Uca Eğitilmesi.

Şekil 3.37'ye göre;

"encoder_input_data": Encoder modeline verilen Türkçe cümleyi,
"decoder_input_data": "start" parametreli Decoder modeline verilen İngilizce cümleyi,
"decoder_output_data: Decoder modeline verilen İngilizce cümleyi, temsil etmektedir.

Bir sonraki aşamada model için gerekli olan hiper parametreleri ayarlanmaktadır. Şekil 3.38'de model için optimizer olarak "RMSprop" ve loss fonksiyonu olarak "sparse_categorical_crossentropy" ve metrik olarak "accuraccy" kullanılmıştır.

```
[ ]  model_train.compile(optimizer=RMSprop(learning_rate=1e-3),
                         loss='sparse_categorical_crossentropy',
                         metrics=['accuracy']
                         )
```

Şekil 3.38. Encoder-Decoder Modeli İçin Hiper Parametre Ayarı.

Optimizer algoritması ile öğrenme işlemi gerçekleştirilmektedir. Model geri yayılım yaparken, adım adım kayıp değeri azaltılarak, doğru sonuca ulaşılmaktadır. Buradaki "1e3" değeri "0,001" değerine karşılık gelmektedir. Yani model, "0,001" gibi küçük adımlarla doğru sonuca ulaşacaktır. Kayıp fonksiyonu ise modelin bir tahmin yaptıktan sonra tahmin değerini, gerçek değeri ile karşılaştırıp, ne kadar hata yapıldığını hesaplamaktadır. Metrik ile modelin doğruluğu tespit edilmektedir. Belirlenen hiper parametreler ve oluşturulan modele göre bir eğitim işlemi gerçekleştirilmektedir. LSTM ile oluşturulan modele ait parametre sayıları şekil 3.39'da gösterilmiştir.

```
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 encoder_input (InputLayer)     [(None, None)]       0           []

 encoder_embedding (Embedding)  (None, None, 128)    444416      ['encoder_input[0][0]']

 encoder_lstm1 (LSTM)           (None, None, 256)    394240      ['encoder_embedding[0][0]']

 decoder_input (InputLayer)     [(None, None)]       0           []

 encoder_lstm2 (LSTM)           (None, None, 256)    525312      ['encoder_lstm1[0][0]']

 decoder_embedding (Embedding)  (None, None, 128)    268800      ['decoder_input[0][0]']

 encoder_lstm3 (LSTM)           (None, 256)          525312      ['encoder_lstm2[0][0]']

 decoder_lstm1 (LSTM)           (None, None, 256)    394240      ['decoder_embedding[0][0]',
                                                                  'encoder_lstm3[0][0]',
                                                                  'encoder_lstm3[0][0]']

 decoder_lstm2 (LSTM)           (None, None, 256)    525312      ['decoder_lstm1[0][0]',
                                                                  'encoder_lstm3[0][0]',
                                                                  'encoder_lstm3[0][0]']

 decoder_lstm3 (LSTM)           (None, None, 256)    525312      ['decoder_lstm2[0][0]',
                                                                  'encoder_lstm3[0][0]',
                                                                  'encoder_lstm3[0][0]']

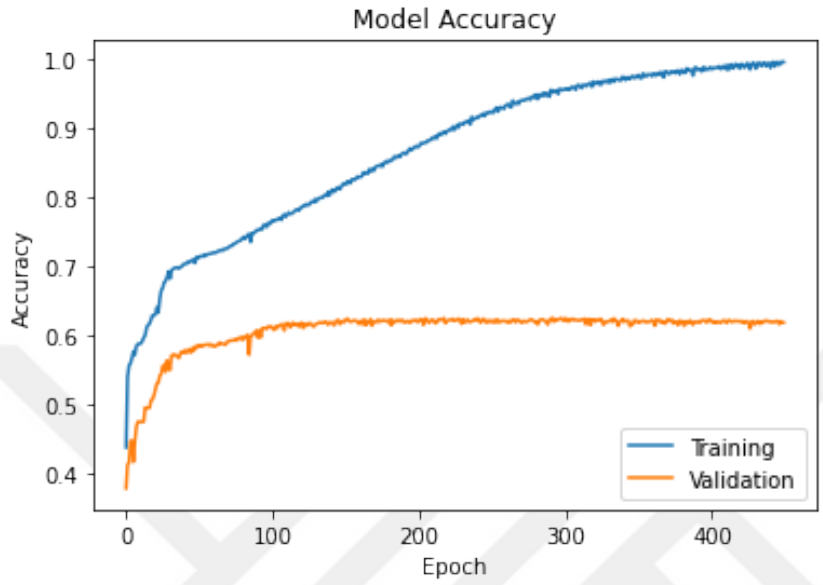
 decoder_output (Dense)         (None, None, 3505)   900785      ['decoder_lstm3[0][0]']

==================================================================================================
Total params: 4,503,729
Trainable params: 4,503,729
Non-trainable params: 0
```

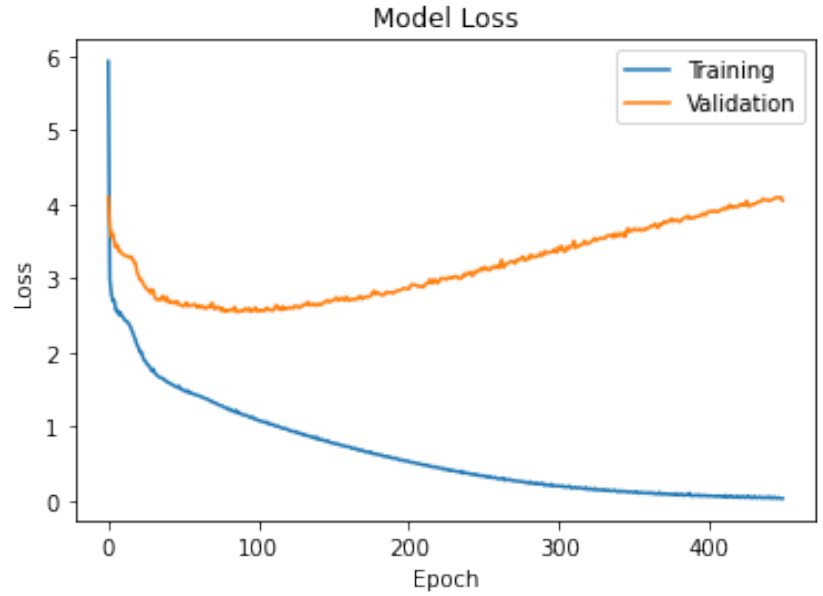
Şekil 3.39. LSTM Model Parametre Sayısı.

Şekil 3.39'daki modelin "embeding" katmanının boyutu 128'dir. Üç katmanlı bir LSTM yapısı kullanılmıştır. Her bir LSTM katmanı 256 birimlik bir çıktı vermektedir. Bu şekilde 4 503 729 parametreli bir model oluşturulmuştur. Son adımda ise fit edilen modelin doğruluk ve kayıp fonksiyonları sırayla şekil 3.40 ve şekil 3.41'de gösterilmiştir. Veri kümesi %20 oranında eğitim ve test için bölümlenmiştir. Eğitim verisi 2804 adet, test verisi ise 701 adet olarak belirlenmiştir. Şekil 3.40'ta model, 450 epok sonucunda eğitim verisi %99 gibi bir doğruluğa ulaşmaktadır. Buna karşın test verisi ise %60 gibi bir skor üretmiştir. Veri kümesinin küçük olmasından dolayı doğrulama verisinin eğitim verisi ile çok dengeli bir seyir izlemediği gözlemlenmiştir. Kayıp fonksiyonları "validation" veri kümesi için azalması beklenirken, bir noktadan sonra artışa geçtiği gözlemlenmiştir. Bunu çözmek için veri kümesindeki örnek sayısının arttırılması gerekmektedir. Şekil 3.41'de ise eğitim ve doğrulama aşamasındaki kayıp (loss) fonksiyonunun grafiği sunulmuştur. Kayıp değerinin azalarak 0,0279 gibi bir değere ulaştığı gözlemlenmiştir. Yine doğrulama veri setindeki kayıp değerinin bir noktadan

sonra artmaya başladığı gözlemlenmiştir. Bu durum, modelin "over fitting" (aşırı uydurma) olduğunu göstermektedir. Modeldeki veri setinin azlığı nedeniyle böyle bir yol izlenmiştir.



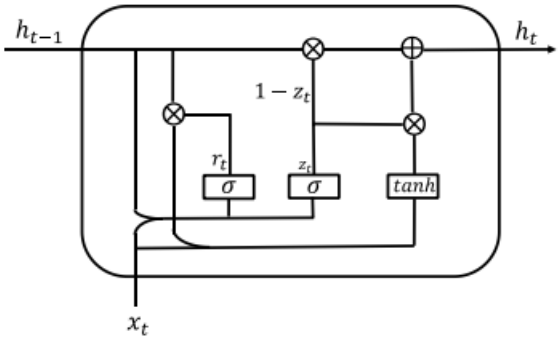
Şekil 3.40. LSTM Eğitim Doğruluğu Grafiği.



Şekil 3.41. LSTM Kayıp Fonksiyonu Grafiği.

3.9.4. Gru için eğitim süreci

LSTM modelinde olduğu gibi benzer işlem adımları GRU modeli için de gerçekleştirilmektedir. Encoder-Decoder yapısında LSTM hücreleri yerine GRU hücreleri kullanılmaktadır. GRU'nun LSTM'den farkı, modelin daha basit bir içerik sunmasıdır. Şekil 3.42'de GRU katmanının içyapısı gösterilmiştir. İçeriğinin daha basit olması, modelin daha hızlı bir şekilde eğitilmesine imkân vermektedir. Bunun yanı sıra oluşacak parametre sayısı da azalacaktır.



Şekil 3.42. GRU Hücresi İç Yapısı (Qian et al., 2021).

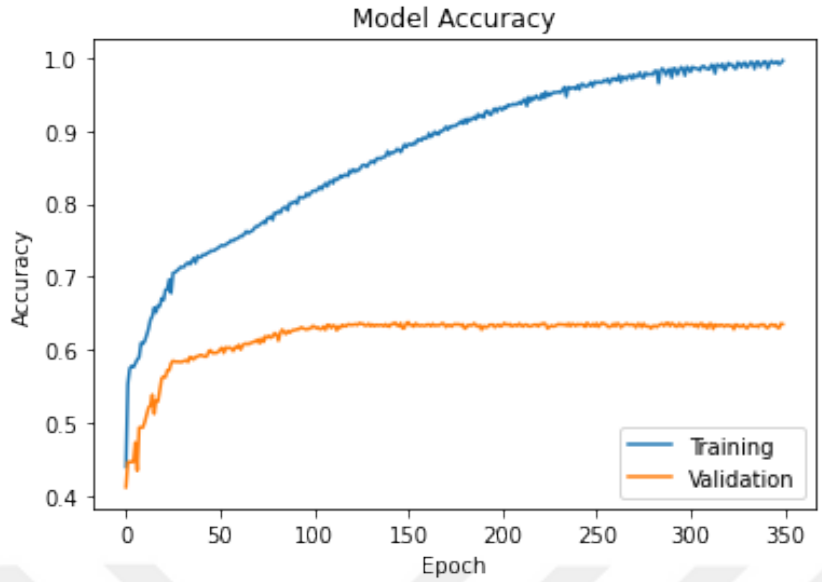
"Encoder-Decoder" yapısı bu sefer GRU katmanlarından oluşmaktadır. Modelde kullanılan bütün hiper parametreler LSTM katmanındaki model ile aynıdır. Şekil 3.43'te görüldüğü üzere embeding layer 128, GRU katmanları ise 256 birimden oluşmuştur. Üç katmanlı bir GRU yapısı oluşturulmuştur. "Dense layer" olarak üç boyutlu bir matris üretilmiştir. Bütün hiper parametreler aynı olmasına rağmen GRU hücresinde üretilen toplam parametre sayısının 3 785 905 olduğu görülmüştür. Bu değer LSTM hücrelerinde 4 503 858 olarak tespit edilmiştir. Üretilen parametre sayısının azalması modelin iç yapısının daha sade olmasından kaynaklanmaktadır.

Layer (type)	Output Shape	Param #	Connected to
encoder_input (InputLayer)	[(None, None)]	0	[]
encoder_embedding (Embedding)	(None, None, 128)	444416	['encoder_input[0][0]']
encoder_gru1 (GRU)	(None, None, 256)	296448	['encoder_embedding[0][0]']
decoder_input (InputLayer)	[(None, None)]	0	[]
encoder_gru2 (GRU)	(None, None, 256)	394752	['encoder_gru1[0][0]']
decoder_embedding (Embedding)	(None, None, 128)	268800	['decoder_input[0][0]']
encoder_gru3 (GRU)	(None, 256)	394752	['encoder_gru2[0][0]']
decoder_gru1 (GRU)	(None, None, 256)	296448	['decoder_embedding[0][0]', 'encoder_gru3[0][0]']
decoder_gru2 (GRU)	(None, None, 256)	394752	['decoder_gru1[0][0]', 'encoder_gru3[0][0]']
decoder_gru3 (GRU)	(None, None, 256)	394752	['decoder_gru2[0][0]', 'encoder_gru3[0][0]']
decoder_output (Dense)	(None, None, 3505)	900785	['decoder_gru3[0][0]']

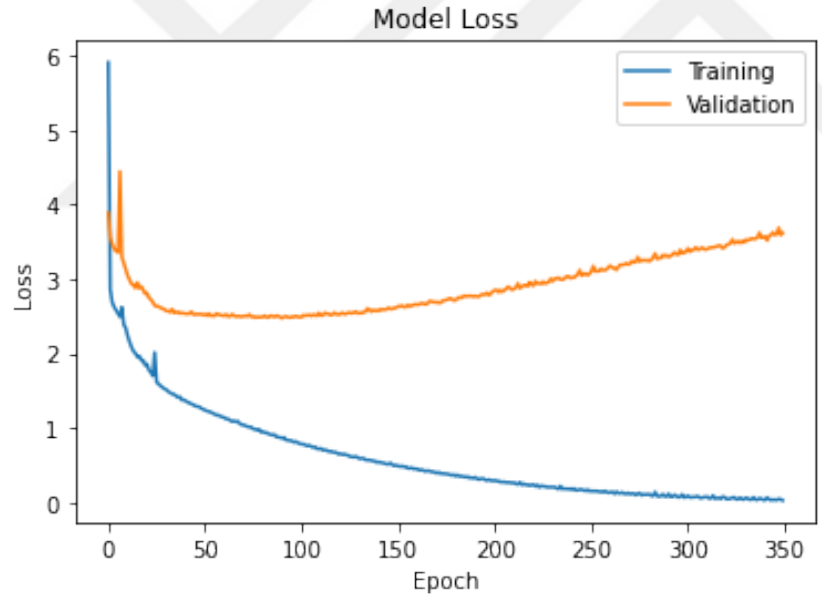
Total params: 3,785,905
Trainable params: 3,785,905
Non-trainable params: 0

Şekil 3.43. GRU Model Parametre Sayısı.

Veri kümesi %20 oranında eğitim ve test için bölümlenmiştir. Şekil 3.44'e bakıldığında, modelin 350 epok sonucunda eğitim verisinin %99 gibi bir doğruluğa ulaştığı, test verisinin ise %60 seviyelerinde kaldığı görülmektedir. Veri kümesinin küçük olmasından dolayı, doğrulama verisinin eğitim verisi ile çok dengeli bir seyir izlemediği bu modelde de görülmüştür. Şekil 3.45'te ise eğitim ve doğrulama aşamasındaki kayıp (loss) fonksiyonunun grafiği gösterilmiştir. Kayıp değerinin azalarak 0,0295 gibi bir değere ulaştığı gözlemlenmiştir. Yine doğrulama veri setindeki kayıp değerinin, bir noktadan sonra artmaya başladığı gözlemlenmiştir.



Şekil 3.44. GRU Eğitim Doğruluğu Grafiği.



Şekil 3.45. GRU Kayıp Fonksiyonu Grafiği.

3.9.5. Bidirectional lstm için eğitim süreci

BiLSTM model, çift yönlü LSTM olarak isimlendirilmektedir. Encoder modelden çıkan ve çift taraflı olan state (düşünce vektörü), Decoder modeline giriş verisi olarak verilmektedir. Bu yönüyle hem geçmiş hem de gelecekteki bilgiye bakılmış olmaktadır. İki yönlü olması sebebiyle, modelin ürettiği parametre sayısı da LSTM modele göre daha

fazla olmaktadır. Modelde kullanılan bütün hiper parametreler LSTM ve GRU ile aynıdır. Şekil 3.46'ya bakıldığında embeding layer 128, BİLSTM katmanları ise 256 birimden oluşmaktadır. Toplam parametre sayısının ise 5 307 137 olduğu görülmektedir. Parametre sayısının LSTM ve GRU göre yüksek olması, çift taraflı LSTM olmasından kaynaklanmaktadır.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 46)]              0

 embedding_2 (Embedding)     (None, 46, 128)           446976

 bidirectional_4 (Bidirectio  (None, 512)              788480
 nal)

 dense_4 (Dense)             (None, 256)               131328

 repeat_vector_2 (RepeatVect  (None, 51, 256)          0
 or)

 bidirectional_5 (Bidirectio  (None, 51, 512)          1050624
 nal)

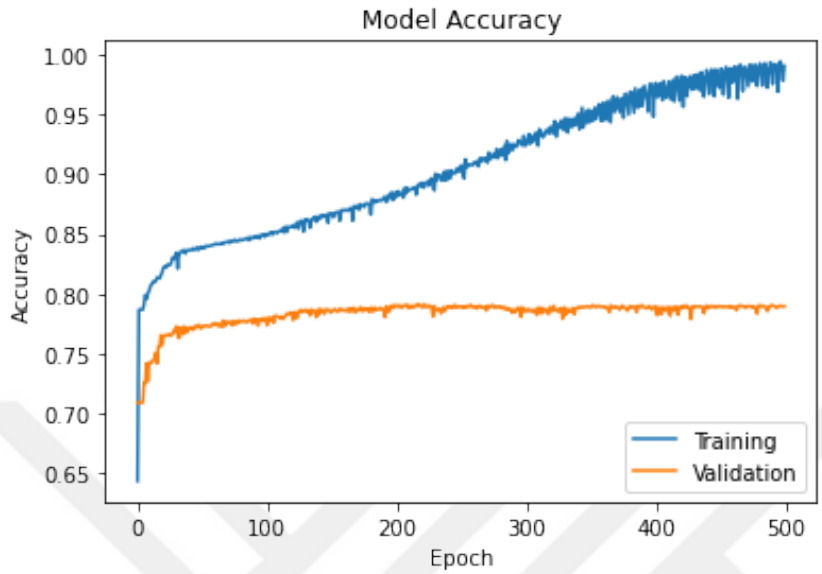
 time_distributed_2 (TimeDis  (None, 51, 5633)         2889729
 tributed)

=================================================================
Total params: 5,307,137
Trainable params: 5,307,137
Non-trainable params: 0
```

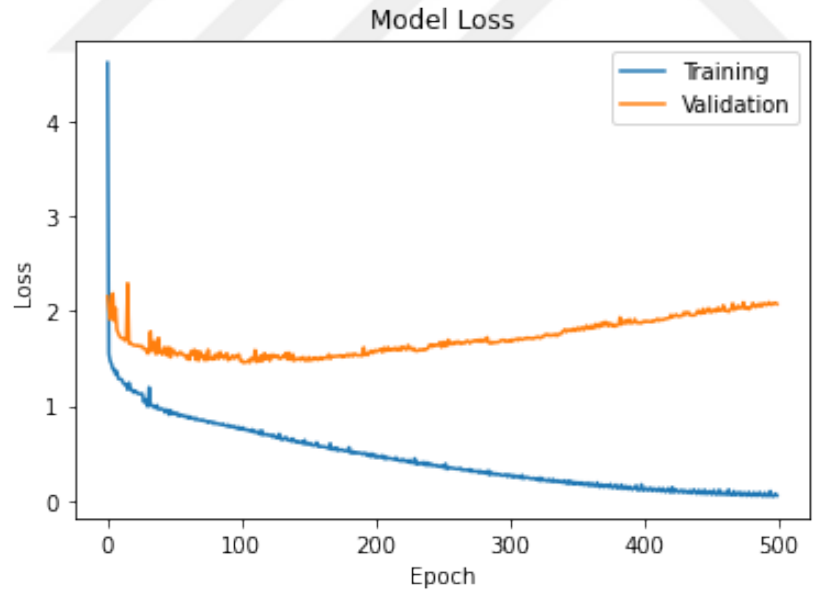
Şekil 3.46. BiLSTM Model Parametre Sayısı.

Veri kümesi %20 oranında eğitim ve test için bölümlenmiştir. Şekil 3.47'de modelin 500 epok sonucunda eğitim verisinin %99 gibi bir doğruluğa ulaştığı gözlemlenmiştir. Test verisi ise %78 seviyelerine ulaşmıştır. Bu noktada BiLSTM, LSTM ve GRU'ya göre test verisi üzerinde daha başarılı skor değeri üretmiştir. Veri kümesinin küçük olmasından dolayı doğrulama verisinin eğitim verisi ile çok dengeli bir seyir izlemediği bu modelde de görülmüştür. Şekil 3.48'de ise eğitim ve doğrulama aşamasındaki kayıp (loss) fonksiyonunun grafiği gösterilmektedir. Kayıp değerinin azalarak 0,0523 gibi bir değere

ulaştığı gözlemlenmiştir. Yine doğrulama veri setindeki kayıp değerinin bir noktadan sonra artmaya başladığı gözlemlenmiştir.



Şekil 3.47. BiLSTM Eğitim Doğruluğu Grafiği.



Şekil 3.48. BiLSTM Kayıp Fonksiyonu Grafiği.

4. BULGULAR

Yapılan makine çevirilerinin doğruluğunu tespit etmek için farklı metrikler kullanılmaktadır. Bu çalışmadaki modelin doğruluğu BLUE, METEOR (Metric For Evaluation of Translation with Explicit Ordering) ve TER (Translation Error Rate) ile değerlendirilmiştir. BLEU (Bilingual Evaluation Understudy) metriği ile yapılan çevirinin doğruluğu kontrol edilmektedir. Hipotez ve referans değerleri n-gramlara göre karşılaştırılarak bir skor değeri elde edilmiştir.

hypothesis = ['Create', 'a', 'return', 'function']
reference = ['Create', 'the', 'return', 'function']

Yukarıda verilen hipotez ve referans değerleri bi-gram'a göre değerlendirildiğinde;
Tahmin: (Create a), (a return), (return function)
Gerçek: (Create the), (the return), (return function)

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in \{Candidates\}} \sum_{n-gram' \in C'} Count \quad (n-gram')} \tag{3.15}$$

$p_{bi-gram}=\frac{0+0+1}{3}$=0,33 sonucu elde edilmektedir.

METEOR skor ile modelin doğruluğu hipotez ve referans değerleri arasındaki ilişkiye göre hesaplama yapılarak değerlendirilmektedir.

$$P = \frac{m}{w_t} \tag{3.16}$$

$$R = \frac{m}{w_r} \tag{3.17}$$

$$F_{mean} = \frac{10PR}{R + 9P} \tag{3.18}$$

$$Penalty = 0.5(\frac{c}{u_m})^3 \tag{3.19}$$

$$M = F_{mean}(1 - Penalty) \tag{3.20}$$

Meteor skoru hesaplayabilmek için Precision (Kesinlik) ve Recall (Duyarlılık) değerine göre F_{mean}'in hesaplanması gerekmektedir. Örneğin, aşağıdaki gibi referans ve hipotez değerine sahip olduğumuzu düşünelim.

reference=['the','cat','sat','on','the','mat']
hypothesis=['the','cat','was','sat','on','the','mat']

m=refererans cümle içinde geçen her bir kelimenin hipotez cümle içerisindeki sayısı,
w_t=hipotez cümle içindeki kelime sayısı,
w_r=referans cümle içindeki kelime sayısı,
c= chunks değeridir. Kaynak ve hedef arasındaki eşleşen sözcük öbeklerinin sayısıdır.

Denklem 3.16'ya göre $P = \frac{6}{7} = 0{,}8571$

Denklem 3.17'ye göre $R = \frac{6}{6} = 1$

Denklem 3.20'ye göre $F_{mean} = \frac{8{,}571}{8{,}7139} = 0{,}9836$

Uygulanacak ceza oranını hesaplayabilmek için hipotez ve referans arasında bir hizalama yapılması gerekmektedir. Aşağıdaki gibi hizalama yapıldığında;
c=2 olarak bulunur. Çünkü (the,cat) ifadesi (the,cat) ile eşleşmiştir. (sat,on,the,mat) ifadesi de aynı şekilde (sat,on,the,mat) ifadesi ile eşleşmiştir. Toplamda iki adet eşleşme olmuştur.

reference=['the','cat','sat','on','the','mat']
hypothesis=['the','cat','was','sat','on','the','mat']

$Penalty = 0{,}5(\frac{2}{6})^3 = 0{,}0185$

$M = F_{mean}(1 - Penalty) = 0{,}9836(1 - 0{,}0185) = 0{,}9654$

Gerekli hesaplamalar yapıldığında, METEOR skor değeri 0,9654 olarak bulunmuştur. METEOR skorun BLUE skordan farkı, BLUE skorun anlamsal benzerliğe bakmıyor olmasıdır. BLUE skor ile sade n-gram olarak kelimeler arasındaki benzerliğe bakılmaktadır. METEOR skor ile birlikte precision (kesinlik) ve recall (duyarlılık) gibi hesaplamalar yapılmaktadır. Ayrıca kelime eşleşmesi de yapılarak daha kapsamlı bir skor

değeri bulunmaktadır. Bundan ötürü METEOR skor insan çevirisine daha yakın bir skor değeri üretmektedir. TER skor ise hipotezin referans cümleye olan yakınlığını ölçmek için kullanılan bir metriktir. Hipotez değerini referans cümleye benzetebilmek için insertion (ekleme), deletion (silme), substitution (yer değiştirme) ve shift (kaydırma) gibi işlemler yapılmaktadır. Bu işlemler neticesinde çıkan sonuç ne kadar düşükse o kadar az düzenleme yapıldığını ve yapılan çevirinin iyi olduğunu; skor değerinin yüksek çıkması ise çok fazla düzenleme yapıldığını ve yapılan çevirinin iyi olmadığını ifade etmektedir.

reference="SAUDI ARABIA denied THIS WEEK information published in the AMERICAN new york times".
hypothesis ="THIS WEEK THE SAUDIS denied information published in the new york times" (Snover et al., 2006).

$$TER = \frac{number\ of\ edits}{words\ length\ of\ reference\ text} \tag{3.21}$$

Düzenlemelere bakıldığında 1 adet shift (kaydırma), 2 adet substitutions (yer değiştirme) ve 1 adet insertion (ekleme) olduğu görülmektedir. Sonuç olarak, toplamda 4 adet düzenleme yapılmaktadır. Referans cümledeki kelime sayısı ise 13 olduğundan;

$TER = \frac{4}{13} = 0{,}31$'dir.

Aşağıdaki iki adet cümleye bakıldığında ise sadece "was" kelimesinin hipotezden atılmasıyla istenilen cümleye ulaşılabilmektedir.

reference=['the','cat','sat','on','the','mat']
hypothesis =[the,'cat','was','sat','on','the','cat']

Bu durumda, sadece 1 adet deletion (silme) işlemi uygulandığından,

$TER = \frac{1}{6} = 0{,}16$'dır.

Sonucun 0,16 bulunması çok az bir değişiklik yapılacağını göstermektedir. Örneğe göre, sadece "was" kelimesi cümleden atılmıştır. Bu durum, hipotezin referans cümleye çok

yakın olduğunu göstermektedir. İnşa edilen modelde bulunan BLUE, METEOR ve TER skor değerleri, çizelge 4.1, 4.2 ve 4.3'te gösterilmiştir.

Çizelge 4.1. BLUE Skor.

Model	**1-gram**	**2-gram**	**3-gram**	**4-gram**
LSTM	0,74	0,71	0,71	0,66
GRU	0,68	0,75	0,75	0,71
BiLSTM	0,82	0,78	0,78	0,73

Çizelge 4.2. METEOR Skor.

Model	LSTM	GRU	BiLSTM
Skor	0,71	0,77	0,84

Çizelge 4.3. TER Skor.

Model	LSTM	GRU	BiLSTM
Skor	0,33	0,26	0,19

5. TARTIŞMA ve SONUÇ

Blue skor değerlerine bakıldığında, n-gram değeri arttıkça modellerin başarım oranlarının düştüğü gözlemlenmiştir. Bunun sebebi, LSTM ve GRU gibi modellerin de çok uzun cümlelerde etkili olamamasıdır. Bu yüzden derlemdeki kelimeler çok uzun tutulmamaya çalışılmıştır. Yapılan çalışmada, BiLSTM modelin diğer modellere göre daha başarılı sonuç verdiği tespit edilmiştir. Modelin başarısını değerlendirmek için BLUE skorlarının yanı sıra METEOR ve TER skorlara da bakılmıştır. METEOR skorlara bakıldığında da en başarılı modelin BİLSTM olduğu sonucuna varılmıştır. METEOR skor, kelime eşlemenin yanında precision ve recall gibi değerlere de baktığı için, insan çevirisine daha yakın sonuçlar vermektedir. Son olarak TER skorlara bakıldığında, yine BiLSTM modelin daha başarılı olduğu görülmektedir. TER skorda, oranın az çıkması daha az değişiklik yapıldığını, daha az değişiklik ise modelin daha doğru sonuç verdiğini göstermektedir. Netice itibari ile bütün skor değerleri BiLSTM modelin daha başarılı olduğu sonucunu ortaya çıkarmaktadır. LSTM ve GRU ise birbirlerine daha yakın skor değerleri üretmiştir. Modelin, verilen Türkçe cümleleri başarılı bir şekilde İngilizceye çevirebilmesi, veri setindeki veri miktarının çokluğuna bağlıdır. Oluşturulan örneklem veri kümesi 3505 adet cümleyi kapsamaktadır. Bu açıdan veri kümesi yeterli genişlikte değildir. Makine çevirisi üzerine yapılan çalışmalarda, bu rakamların 400-500 bin seviyelerinde olduğu görülmüştür. Önerilen model, mevcut haliyle bir prototip vazifesi görmekte ve elde edilen sonuçlar umut verici görülmektedir. Veri kümesinde kullanılan cümlelerin uzunlukları kısa tutularak başarı oranı artırılmaya çalışılmıştır. Geleceğe yönelik çalışmalarda attention (dikkat) mekanizması ya da transformer mimarisi kullanılarak başarım oranı üzerindeki etkisi araştırılacaktır (Bogdanchikov et al., 2013b).

KAYNAKLAR

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. https://doi.org/10.48550/arxiv.1603.04467

Abdul, S., Andrabi, B., & Wahid, A. (2021). A Review of Machine Translation for South Asian Low Resource Languages. In *Turkish Journal of Computer and Mathematics Education* (Vol. 12, Issue 5).

Al-Sabahi, K., Zuping, Z., & Kang, Y. (2018). *Bidirectional Attentional Encoder-Decoder Model and Bidirectional Beam Search for Abstractive Summarization*. https://doi.org/10.48550/arxiv.1809.06662

Bogdanchikov, A., Zhaparov, M., & Suliyev, R. (2013a). Python to learn programming. *Journal of Physics: Conference Series*, *423*(1). https://doi.org/10.1088/1742-6596/423/1/012027

Bogdanchikov, A., Zhaparov, M., & Suliyev, R. (2013b). Python to learn programming. *Journal of Physics: Conference Series*, *423*(1), 012027. https://doi.org/10.1088/1742-6596/423/1/012027

Bouguesmia, M. T. (2020). Using AI in Translation, a Technological Leap, or a Translator's Nightmare. *ALTRALANG*, 81–100.

ÇELİK, Ö., & ODABAS, A. (2020). Sign2Text: Konvolüsyonel Sinir Ağları Kullanarak Türk İşaret Dili Tanıma. *European Journal of Science and Technology*, *19*, 923–934. https://doi.org/10.31590/ejosat.747231

Chen, G. (2016). *A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation*. http://arxiv.org/abs/1610.02583

Demir, N. (2006). Türkiye'de Dil-Lehçe-Şive-Ağız Tartışmaları. *Bilgi Üniversitesi Yayınları*, 119–146.

Django introduction - Learn web development | MDN. (n.d.). Retrieved June 10, 2022, from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

En Popüler Programlama Dilleri • Turhost Blog. (n.d.). Retrieved June 9, 2022, from https://blog.turhost.com/2020nin-en-populer-programlama-dilleri/

Erhandı, B. (2020). *Derin öğrenme ile metin özetleme* [Yüksek lisans, Fen Bilimleri Enstitüsü]. https://acikerisim.sakarya.edu.tr/handle/20.500.12619/97078

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Ng, A. Y. (2014). *Deep Speech: Scaling up end-to-end speech recognition*. https://doi.org/10.48550/arxiv.1412.5567

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/SCIENCE.1127647/SUPPL_FILE/HINTON.SOM.PDF

Jeon, H., Jung, Y., Lee, S., & Jung, Y. (2020). Area-efficient short-time fourier transform processor for time–frequency analysis of non-stationary signals. *Applied Sciences (Switzerland)*, *10*(20), 1–10. https://doi.org/10.3390/app10207208

Keneshloo, Y., Shi, T., Ramakrishnan, N., & Reddy, C. K. (2020). Deep Reinforcement Learning for Sequence-to-Sequence Models. *IEEE Transactions on Neural*

Networks and Learning Systems, *31*(7), 2469–2489. https://doi.org/10.1109/TNNLS.2019.2929141

König, G. Ç. (1991). *TOPLUMDÎLBÎLÎM AÇISINDAN "Dil" VE "Dil Türleri "KAVRAMLARI ÜZERÎNE*.

Konuşmayı Metne Dönüştürme Nedir? - Yeni Başlayanlar İçin Transkripsiyon Kılavuzu - AWS. (n.d.). Amazon Web Server. Retrieved June 10, 2022, from https://aws.amazon.com/tr/what-is/speech-to-text/

Láng, B. (2018). *Real Life Cryptology*. Amsterdam University Press. https://www.aup.nl/en/book/9789462985544/real-life-cryptology

Mcculloch, W. S., & Pitts, W. (1990). A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*. In *Bulletin of Mathematical Biolooy* (Vol. 52, Issue 1).

Models - OpenAI API. (n.d.). Retrieved June 10, 2022, from https://beta.openai.com/docs/models/gpt-3

Moharm, K., Eltahan, M., & Elsaadany, E. (2020). Wind speed forecast using LSTM and Bi-LSTM algorithms over gabal el-zayt wind farm. *Proceedings - 2020 International Conference on Smart Grids and Energy Systems, SGES 2020*, 922–927. https://doi.org/10.1109/SGES51519.2020.00169

Onyedi, B., Üniversitesi, E., Ve Doğa, M., Fakültesi, B., Balıkesir, B. /, & Tarihleri, M. (2020). Uzun Kısa Dönem Bellek Ağlarını Kullanarak Erken Aşama Diyabet Tahmini Early-Stage Diabetes Prediction Using Long Short-Term Memory Networks. *Müh. Bil.ve Araş. Dergisi*, *2*(2), 50–57.

Panchbhai, A., & Pankanti, S. (2021). Exploring large language models in a limited resource scenario. *Proceedings of the Confluence 2021: 11th International Conference on Cloud Computing, Data Science and Engineering*, 147–152. https://doi.org/10.1109/Confluence51648.2021.9377081

Plekhanova, J. (2009). *Evaluating web development frameworks: Django, Ruby on Rails and CakePHP*. www.ibit.temple.edu

Qian, S., Yu, Y., Li, L., & Chang, Y. (2021). An attention-based GRU encoder decoder for hostload prediction in a data center. *2021 International Conference on Computer Communication and Artificial Intelligence, CCAI 2021*, 121–125. https://doi.org/10.1109/CCAI50917.2021.9447455

Sadıkov Taşpolat, S. K. (2021). MAKİNE ÇEVİRİ YÖNTEMLERİ VE MAKİNE ÇEVİRİSİNİN BUGÜNKÜ DURUMU. *Uluslararası Türkçe Edebiyat Kültür Eğitim Dergisi*, 192–205.

Salmanova, A. (2019). THE FINE LINE BETWEEN CRYPTOLOGY AND MACHINE TRANSLATION. *Baku Engineering University*, 392. https://www.academia.edu/60999972/THE_FINE_LINE_BETWEEN_CRYPTOLOGY_AND_MACHINE_TRANSLATION

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J. (2006). A Study of Translation Edit Rate with Targeted Human Annotation. *Association for Machine Translation in the Americas*, 2223–2231.

Sönmez, S. (1990). Dilbilim Araştırmaları Dergisi Sözlü Dil/Yazılı Dil. *Dilbilim Araştırmaları Dergisi*, 119–122. http://dad.boun.edu.tr/tr/pub/issue/29234/312972

Sutskever, I., Vinyals, O., & Le, Q. v. (2014). *Sequence to Sequence Learning with Neural Networks*. http://arxiv.org/abs/1409.3215

Szucs, G., & Huszti, D. (2019). Seq2seq deep learning method for summary generation by lstm with two-way encoder and beam search decoder. *SISY 2019 - IEEE 17th*

International Symposium on Intelligent Systems and Informatics, Proceedings, 221–225. https://doi.org/10.1109/SISY47553.2019.9111502

Terzi, M. B. (2021). Anomaly detection with deep long short term memory networks. *Proceedings - 6th International Conference on Computer Science and Engineering, UBMK 2021*, 129–132. https://doi.org/10.1109/UBMK52708.2021.9559034

The CMU Pronouncing Dictionary. (n.d.). Retrieved June 10, 2022, from http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Ünalan Saniye Uysal, P. N. T. Ö. M. T. (2013). *Diyalog Interkulturelle Zeitschrift Für Germanistik » Makale » Zwischen literarischen Welten und akademischen Generationen: Eine Ehrung des Literaturwissenschaftlers Kasım Eğits zum 65. Geburtstag* (2013th ed.). https://dergipark.org.tr/tr/pub/diyalog/issue/4738/395698

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, *2017-December*, 5999–6009. https://doi.org/10.48550/arxiv.1706.03762

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wu, K., Wu, J., Feng, L., Yang, B., Liang, R., Yang, S., & Zhao, R. (2021). An attention-based CNN-LSTM-BiLSTM model for short-term electric load forecasting in integrated energy system. *International Transactions on Electrical Energy Systems*, *31*(1). https://doi.org/10.1002/2050-7038.12637

Xiao Jianqiong, Z. (2020, June). Research Progress of RNN Language Model. *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*.

Yuan, Y., Lin, L., Huo, L. Z., Kong, Y. L., Zhou, Z. G., Wu, B., & Jia, Y. (2020). Using An Attention-Based LSTM Encoder-Decoder Network for near Real-Time Disturbance Detection. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 1819–1832. https://doi.org/10.1109/JSTARS.2020.2988324

Zivkovic Strahinja. (2020a, September 24). *RNN - Architecture, Mapping, and Propagation*. https://datahacker.rs/002-rnn-recurrent-neural-networks-architecture-mapping-and-propagation/

Zivkovic Strahinja. (2020b, September 24). *RNN - Tackling Vanishing Gradients with GRU and LSTM*. https://datahacker.rs/005-rnn-tackling-vanishing-gradients-with-gru-and-lstm/