Teknoloji Fakültesi

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

Otomatik Soru Cevap Sistemleri ve Örnek Bir Uygulaması

Ahmed Burhaneddin ÇOBAN

Can ÖZFUTTU

DANIŞMAN

Doç. Dr. Buket DOĞAN

İSTANBUL, 2024

Teknoloji Fakültesi

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

Otomatik Soru Cevap Sistemleri ve Örnek Bir Uygulaması

Ahmed Burhaneddin ÇOBAN – 170420041

Can ÖZFUTTU – 170420012

DANIŞMAN

Doç. Dr. Buket DOĞAN

İSTANBUL, 2024

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencisi Can ÖZFUTTU ve Ahmed Burhaneddin ÇOBAN'ın "Otomatik Soru Cevap Sistemleri ve Örnek Bir Uygulaması" başlıklı bitirme projesi çalışması, 05/07/2024 tarihinde sunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Doç. Dr. Buket DOĞAN (Danışman)

Marmara Üniversitesi ..... (İMZA) .....

Dr. Öğr. Üyesi Gözde KARATAŞ BAYDOĞMUŞ (Jüri)

Marmara Üniversitesi ..... (İMZA) .....

Prof. Dr. Ali BULDU (Jüri)

Marmara Üniversitesi ..... (İMZA) .....

TEŞEKKÜR

Bu tezi hazırlarken bize rehberlik eden, bilgi ve tecrübelerini bizimle paylaşan başta danışmanımız Doç. Dr. Buket DOĞAN olmak üzere, tüm Marmara Üniversitesi Bilgisayar Mühendisliği bölümü öğretim üyelerine en içten teşekkürlerimizi sunarız. Özellikle, dört yıllık lisans eğitim hayatımız boyunca bize her aşamada destek olan ve yol gösteren öğretmenimiz Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ'ye teşekkür ederiz. Ayrıca, bu çalışma süresince bizden desteğini esirgemeyen ve her zaman yanımızda olan annelerimiz Hatice DÖĞER ve Zeynep ÇOBAN'a ve arkadaşlarımıza teşekkür ederiz. Onların moral ve motivasyon desteği olmasaydı bu tez projemizi başarıyla tamamlayamazdık.

İÇİNDEKİLER

ÖZET

Gelişen dünyamızda her iş ve eyleme kadar otomatikleşen insana yardımcı veya insanın yerini alacak teknolojilerle donanmaktayız. Teknolojideki bu büyük gelişmeler yapay zekâ ile birçok işte insanın yerini alacak kadar yüksek doğruluk oranları ile iş kabiliyeti olan modellere evrilmiştir ve bu modeller giderek kendini eğitmekte ve geliştirmektedir. Teknolojideki gelişmelerin insanlara sayısız yararlarından birisi de bulunduğumuz yerde karşımızda insan gücüne dayalı bir muhatap olmadan da soru cevap sistemleri ile hayatımızı kolaylaştıracak uygulamalar geliştirilmesidir. Şu anki çoğu programda kullanıcıların, kullanacakları uygulamalarla tek başınayken yaşadıkları sorunları kendisi çözmesi beklenmektedir. Uygulamalarda SSS (Sık Sorulan Sorular) bölümü olsa da yetersiz kalmaktadır çünkü bazen kişi neyi araması gerektiğini de bilemeyebilir. Burada hizmet verebilecek ve her kullanıcıya hizmet sağlayabilecek Sohbet botu (Chat bot) tarzında kullanıcının sorduğu soruyu analiz ederek uygun cevabı otomatik veren sistemlere ihtiyaç vardır. Hem kullanıcılara hem de hizmet sağlayıcıya zaman, mekân ve maliyetten tasarruf sağlayacak ve aynı zamanda insan gücüne dayalı olarak verilemeyecek kadar sınırsız büyük ölçekte hizmetler sağlanabilir.

Bu tez, kullanıcıların doğal dilde sordukları sorulara otomatik olarak cevap verebilen bir soru cevaplama sisteminin tasarımı ve geliştirilmesini kapsamaktadır. Geliştirilen sistem, Türkçe dilinde eğitilmiş bir BERT modeli temel alarak, verilen metin bağlamında en uygun cevabı bulmayı amaçlamaktadır. Sistem, Flask kullanılarak oluşturulan bir backend ve React ile geliştirilen bir frontend'den oluşmaktadır. Bu çalışma, doğal dil işleme ve derin öğrenme tekniklerinin entegrasyonunu, özellikle Transformer tabanlı modellerin uygulamalı bir kullanımını sunmaktadır.

Tezin ana bölümleri, literatür taraması, materyal ve yöntem, sistemin uygulanması, bulgular ve tartışma şeklinde düzenlenmiştir. Literatür taraması, mevcut soru cevaplama sistemlerini ve kullanılan teknolojileri derinlemesine incelemekte, materyal ve yöntem bölümü ise sistemin nasıl geliştirildiğini detaylı bir şekilde açıklamaktadır. Sistemin uygulanması, kullanılan yazılım araçları ve kod yapısına dair teknik bilgiler sağlamakta, bulgular ve tartışma ise sistemin performansını değerlendirmekte ve gelecekteki çalışmalar için önerilerde bulunmaktadır.

Bu linkten Github üzerindeki repo’ya ulaşabilirsiniz.

ABSTRACT

In our developing world, with the effect of digitalization, we are equipped with technologies that will help or replace humans, automating every job and action. These great advances in technology have evolved into models that have the ability to replace humans in many jobs with artificial intelligence with high accuracy rates, and these models are gradually training and improving themselves. One of the countless benefits of technological advances is the development of applications that will make our lives easier with question and answer systems even without a human-powered interlocutor in front of us. In most current programs, users are expected to solve the problems they encounter when they are alone with the applications they will use. Even if there is a FAQ (Frequently Asked Questions) section in the applications, it is insufficient because sometimes the person may not know what to look for. Here, there is a need for chat-bot-style systems that can serve on a large scale and provide service to every user, analyzing the question asked by the user and automatically giving the appropriate answer. This will save time, space and cost for both the users and the service provider, and at the same time provide unlimited large-scale services that cannot be provided based on manpower.

This thesis covers the design and development of a question answering system that can automatically answer questions asked by users in natural language. The system is based on a BERT model trained in the Turkish language and aims to find the most appropriate answer in the context of the given text. The system consists of a backend built using Flask and a frontend developed with React. This work presents an integration of natural language processing and deep learning techniques, in particular an applied use of Transformer-based models.

The main sections of the thesis are organized as literature review, materials and methods, system implementation, findings and discussion. The literature review provides an in-depth review of existing question answering systems and the technologies used, while the materials and methods section describes in detail how the system was developed. The implementation of the system provides technical information about the software tools and code structure used, while the findings and discussion evaluate the performance of the system and make recommendations for future work.

You can access the repo on Github from this link.

SEMBOL LİSTESİ

- V_{dw} - Ağırlıkların momentum terimleri için hareketli ortalama.
- $V_{dw_{corr}}$ - Bias düzeltilmiş V_{dw} .
- β_1 - Momentum için sürtünme oranı.
- t - İterasyon sayısı.
- S_{dw} - Ağırlıkların uyarlanabilir öğrenme oranı için hareketli varyans.
- $S_{dw_{corr}}$ - Bias düzeltilmiş S_{dw} .
- β_2 - Uyarlanabilir öğrenme oranları için sürtünme oranı.

KISALTMA LİSTESİ

- ANN – Artificial Neural Network
- NLP –Natural Language Processing
- API –Application Programming Interface
- BERT – Bidirectional Encoder Representations from Transformers
- ELECTRA – Efficiently Learning an Encoder that Classifies Token Replacements Accurately
- ReLU – Rectified Linear Unit
- CORS – Cross-Origin Resource Sharing
- CI/CD – Continuous Integration/Continuous Deployment
- RESTful – Representational State Transfer
- JSON – JavaScript Object Notation
- SQL – Structured Query Language
- XSS – Cross-site Scripting
- HTTP – Hypertext Transfer Protocol
- QA – Question Answer
- GPU – Graphical Processing Unit

ŞEKİL LİSTESİ

TABLO LİSTESİ

1. GİRİŞ

NLP (Doğal Dil İşleme) alanında QA (Soru-Cevap) modellerinin uygulanması önemli ilerlemeler kaydetmiş olup metin verilerinin analizi için umut vaat eden çözümler sunmaktadır [1]. Bu proje raporu metin verilerini analiz etmek için QA modellerinin kullanımını derinlemesine inceleyerek, önişleme, model uygulaması ve performans değerlendirmesine değinmektedir.

1.1 Motivasyon

Bilgi Teknolojileri alanında hızla gelişen peyzajda her geçen gün artan veri havuzundan değerli sonuçlar çıkarma zorluğu kilit öneme sahip hale gelmiştir. Kuruluşlar maliyet etkinliği, üstün performans ve optimal sonuçlar sunan yöntemler arayışındayken odak noktası NLP gibi metin madenciliği için kritik bir disiplin haline gelmiştir. Özellikle NLP'nin bir alt kümesi olan QA sistemleri hem küresel hem de yerel düzeyde önemli bir ivme kazanmaktadır ve bu tür sistemler, eğitimden sağlığa, müşteri hizmetlerinden araştırma faaliyetlerine kadar geniş bir kullanım alanına sahiptir. Türkçe gibi doğal dil işleme kaynaklarının sınırlı olduğu dillerde, etkili bir soru cevaplama sistemi geliştirmek, bu alandaki boşluğu doldurabilir ve birçok kişiye fayda sağlayabilir.

1.2 Amaç

Bu çalışmanın amacı, kullanıcıların doğal dilde girdiği sorulara doğru ve anlamlı yanıtlar verebilen, derin öğrenme tekniklerini temel alan bir otomatik soru cevaplama sistemi geliştirmektir. Sistem, özellikle Türkçe metinler üzerinde çalışacak şekilde tasarlanmıştır ve derin öğrenme modeli olarak BERT (Bidirectional Encoder Representations from Transformers) kullanmaktadır. Projede geliştirilen sistem, hem teknik açıdan hem de kullanıcı deneyimi açısından, soru cevaplama alanında yüksek performans göstermeyi hedeflemektedir. Kullanıcı tarafından üretilen soruların titiz ön işlemesinin yanı sıra bağlamsal önemlerine göre anahtar kelimeleri önceliklendirerek amacımız en uygun yanıtları hassaslık ve verimlilikle ortaya çıkarmaktır [2]. Bu tez, sistem mimarisinin kurulumunu, kullanılan teknolojileri, veri işleme süreçlerini, ve elde edilen bulguları detaylı bir şekilde sunmaktadır.

2. LİTERATÜR TARAMASI

Öncelikle yapılacak olan otomatik cevap sistemi, web tabanlı bir uygulama şeklinde olacağı ve uygulama alanı olarak bir çeşit sohbet robotu olacağı için sohbet robotları hakkında yazılmış "Sohbet Robotları (Chatbots) ve Yabancı Dil Eğitimi" adlı bir makale incelendi [3]. Makalede sohbet robotlarının geçmişi, günümüzdeki yeri ve eğitimdeki önemine değinilmiştir.

Dünyadaki tüm değişim ve gelişimler insanların zekalarını kullanmalarıyla olmuştur. İnsan zekâsı teknolojideki ilerlemelerle beraber taklit edilebilmesi ve taklidin ihtiyaçlar ölçüsünde kullanılabilmesine açısından değerlendirilmiştir. Günlük hayatta artık adını sıkça duyduğumuz yapay zekâ ile güçlendirilmiş sistemler ve otomatikleştirilmiş modeller tıpkı insan gibi akıl yürütme, tahmin etme, karmaşık ve önceden belirli olmayan problemlere cevap verme gibi görevleri yerine getirebilir. Bankacılık, finans, sosyal medya, reklamlar, yüz okuma, parmak okuma ve birçok güvenlik sistemleri başta olmak üzere neredeyse bilişimle ilişkili her alanda kullanıma girmiştir. Günümüzde gittikçe içine girdiğimiz bu yapay zekâ devri artık "ikinci makine çağı" olarak adlandırılmaktadır [3]

Yapay zekanın uygulamalarından birisi ve makalenin de konusu olan chat-bot İngilizce anlamından anlaşılacağı üzere karşılıklı sohbeti sağlayan bir tarafın yapay zekâ olduğu bir diyalog sistemidir. Sohbet robotu, kullanıcıdan aldığı doğal dil girişini işleyerek kullanıcıya akıllı ve göreceli yanıtlar üreten ve bunu gerçek zamanlı olarak yapabilen birebir sohbet sistemidir. Bu sohbet robotlarının tarihi Turing'in robotun insan tarafından ayırt edip edilemeyeceğini sınamak için yaptığı "Turing Testi" ile başlamıştır. Birçok denemeler yapılarak gelişen sohbet robotları halkın genelinin yararına kullanılabilecek kadar gelişince ve teknolojinin elverdiği haliyle de Siri ve Cortana gibi kişisel asistanlar haline gelebilmiştir. Özellikle de son zamanlarda en gelişmiş örneği olan OPEN AI şirketinin geliştirdiği ChatGPT sohbet robotudur. Türkiye bazında ise Türkçe diyalog için geliştirilmiş Gökbilge sohbet robotu bulunmaktadır. Konuşmaların metinsel ve işitsel yöntemlerle yürütüldüğü sohbet robotları kullanıcı tarafından alınan içerikleri analiz edip ve bunları olası yanıtları içeren kendi veri tabanına bağlar. İnsan sözlü iletişimini taklit etmeye programlanan sohbet robotları, kural temelli olanlar veri tabanında önceden hazırlanmış hazır kelime ve cevaplardan oluşurken yapay zekâ alt yapısı olan sohbet

robotları ise amaç ve bağlam doğrultusunda makine öğrenimi yaparak sürekli öğrenmektedir [3].

Sohbet robotları her alanda ve sektörde daha iyi ve daha hızlı hizmet verebilmesi onu eğitimde de önemli roller üstlenecek hale ulaştırmıştır. Örnek vermek gerekirse bir öğrenci okul hayatında veya önceden pratik etmek istediği bir alanda bunu kullanabilir. Mesela bir hemşirelik öğrencisi muhtemel hastalarla nasıl bir diyalog içerisine girebileceğini sohbet robotuyla önceden deneyimleyerek kendini buna hazırlayabilir. Okulda arkadaşlarıyla veya öğretmeniyle uyum sorunu yaşayan bir öğrenci sohbet robotuyla rahat rahat konuşarak kendisini iletişimde geliştirebilir. Öğrenci bunu bilgi öğrenmek için veya yabancı dil eğitimi gibi ikinci bir dil öğrenmek için zamandan ve mekândan bağımsız bir şekilde hata yapmaktan çekinmeden kullanabilir. Sohbet robotları belirli bir konu veya genel amaçlı sohbet için tasarlanmış olabilirler. Kullanıcı karmaşık sorulara cevap verme yeteneği güçlü olan genel amaçlı sohbet robotlarında konu dışına çıksa bile tutarlı ve mantıklı cevaplarla karşılaşacağı için iyi bir deneyim sağlar ancak karşısında bir insan olmadığını bildiği için zamanla usanabilir.

Sonuç olarak eğitimle ilgili her şeyi sohbet robotlarına bırakmak pek mümkün gözükmese de eğitimdeki yardımcı rolü ve dil eğitimi gibi okul dışı yapılabilecek eğitimlerde sohbet robotlarının sayısız faydalarına değinilmiştir. Sohbet robotlarının eğitimde kullanımında yanlış sonuçlarına karşı kullanıcıların bilinçlendirilmesi gereklidir çünkü sohbet robotlarıyla ilgili bazı sorunlar vardır bunlar kullanıcıların sohbet robotlarına duygusal olarak bağlanması, yaşı küçük kullanıcıların yanlış yönlendirilmesi veya kullanıcıların karşılarında insan olmadığı için aşırı rahatlık ve argo kelime kullanımıyla sohbet robotunu olumsuz yönde eğitmesi gibi sıkıntılardan doğan durumlardır.

Bir diğer incelenen makale “AVA: an Automatic eValuation Approach to Question Answering Systems” [1]’dir.

Makalede sorular, cevapları ve referans metinlerin transformers tabanlı modeller kullanarak referans ve otomatik cevaplar arasındaki benzerliği soru semantiğine bir eğilim ile cevap oluşturulmasına değinilmiştir. QA (Soru-Cevap) sistemleri günümüzde geleneksel kıyaslama yöntemleriyle oldukça yetersiz kalmıştır. QA sistemlerin doğruluğunu sağlamak için transformer tabanlı modeller önerilir. AVA için genelde iki tür model önerilmektedir; doğrusal sınıflayıcı ve çeşitli transformer tabanlı yöntemler.

Doğrusal sınıflandırıcı, tokenler ve metinler arasındaki benzerlik özelliklerini kullanırken, Transformer modeller soru, referans ve sistem cevap metinleri üçlüleri için yerleştirmeler kullanır [1].

Deneyler, AVA'nın bireysel cevap doğruluğunu ve genel sistem doğruluğunu tahmin etme performansını değerlendirmek için yapılmıştır. Transformer tabanlı modeller, nokta bazlı tahminde yüksek doğruluk göstererek doğrusal sınıflandırıcıyı geride bırakmış ve sistem bazlı tahminde insan değerlendirmelerini etkili bir şekilde taklit etmiştir. AVA, QA sistemlerinin otomatik değerlendirilmesi için güvenilir bir yöntem sunmakta olup, manuel anotasyon maliyetlerini azaltma ve sistem geliştirme verimliliğini artırma potansiyeli göstermektedir.

Bir diğer incelenen makale Türkçe otomatik cevap sistemleri üzerine “Metin Madenciliği ile Soru Cevaplama Sistemi” [2] Türkçenin sondan eklemeli bir dil olmasının özel sorunlarına değinilmiştir.

Bilgi teknolojileri alanındaki hızlı büyüme ile artan verilerden kullanışlı bilgi elde etme ihtiyacı önemli bir konu haline gelmiştir. Doğal dil işleme ve metin madenciliği teknikleri bu konuda etkin ve başarılı yöntemler sunmaktadır. Türkçe’nin sondan eklemeli bir dil olması doğal dil işlemede zorluklar yaratmakla birlikte bu çalışma kapsamında kullanıcıdan alınan sorulara en uygun cevabı içeren metinleri keşfetmek amacıyla vektör uzay modeli ve kosinüs benzerliği gibi yöntemler kullanılmıştır. Anahtar sözcük seçimi metinlerin ön işleme aşamalarından geçirilmesi ve uygun cevapların bulunması üzerine odaklanılmıştır.

İncelenen diğer makaleler, kitaplar ve web sayfaları referans bölümünde belirtilmiştir. İncelenen veri seti ve modeller de referans bölümünde belirtilmiştir.

3. MATERYAL VE YÖNTEM

Bu bölüm, projenin temelini oluşturan metodolojileri, kullanılan araçları ve materyalleri detaylı bir şekilde açıklamaktadır. Otomatik soru cevaplama sisteminin geliştirilmesi sürecinde kullanılan doğal dil işleme teknikleri, metin madenciliği yöntemleri ve derin öğrenme modelleri ele alınmaktadır. Ayrıca, veri toplama, işleme ve model eğitimi için kullanılan yöntemler ve teknolojik araçlar tanıtılmaktadır. Bu metodolojilerin etkin bir şekilde nasıl entegre edildiği ve sistemin nasıl uygulandığına dair bilgiler verilmektedir.

3.1. Doğal Dil İşleme

NLP (Doğal Dil İşleme), bilgisayarların insan dilini anlamasını ve işlemesini sağlayan bir bilgisayar bilimi dalıdır. NLP, metin ve ses verilerinden anlam çıkararak, bu verileri daha işlenebilir ve anlamlı hale getirme sürecidir. Otomatik soru cevaplama sistemlerinde NLP, soruların ve metin bağlamlarının doğru şekilde analiz edilmesi ve cevapların üretilmesi için kritik bir rol oynamaktadır.

Bu doğrultuda bilgisayarlara, iletişim dilinin aktarılması ve dilin doğru bir şekilde çözümlenmesi ihtiyacı oluşmuştur. Bunun sonucunda da bilgisayar bilimi, yapay zekâ ve bilişimsel dil biliminin birleşiminin bir araya gelmesiyle NLP doğmuş olup bu alanın asıl kaynağını bilgisayar kullanarak iki farklı dili birbirine çevirme işlemi oluşturmaktadır [1, 2]

Son yıllarda oldukça önem kazanmış bir çalışma alanı ve bilgisayar bilimi olan NLP, esasen 20. yüzyılın ikinci yarısının başlarında yapay zekânın küçük bir alt dalı olarak meydana gelmiştir [4]. 1954 yılında Georgetown deneyi olarak adlandırılan Georgetown Üniversitesi ve IBM‘in birlikte geliştirdiği bilgisayarlı çeviri deneyinde 60‘dan fazla Rusça cümlenin başarılı bir şekilde İngilizce’ye çevrilebildiği görülmüştür. 1980‘li yıllarda ise hesaplama gücünün hızla artması ile bu alandaki gelişmeler hızlanmıştır [5]. Zamanla yapılan araştırmalar ve gerçekleştirilen uygulamalar sonucunda elde edilen başarılar ile de NLP nihayetinde bilgisayar bilimlerinin temel bir disiplini olarak kabul edilmeye başlanmıştır. NLP kısaca; birçok kaynaktan elde edilen doğal dilde yazılmış metinler ile ilgilenen bir alan olarak tanımlanabilir [6]. NLP, doğal dillerin kurallı olan yapısını detaylıca incelerken aynı zamanda çözümleyerek işlemekte ve anlaşılması yahut yeniden üretilmesi amacını taşıyarak otomatik çeviri, konuşma, ses tanıma, üretme ve duygu analizi gibi pek çok konuyu içeren çalışmada kullanılmaktadır. [7]. Genellikle

metin tabanlı çalışmalarda kullanılmakta olup istatistiksel olarak metnin üzerinden sonuçlar üretmeyi kapsarken çoğunlukla yapay zekâ altındaki dil bilim bilgisine dayalı çalışmaları içermektedir [8].

NLP yöntemleri, insanların el ile yapmak zorunda olduğu bazı metin sınıflandırmaların otomatik olarak yapılabilmesine olanak vermekte ve birçok spesifik uygulamada ciddi kolaylıklar sağlamaktadır. NLP'nin metinsel verinin yoğun olduğu soru cevaplama, bilginin çıkartılması, duygu analizi, makine diline çeviri, sözcük anlamı açıklaştırma ve özetleme alanlarında kullanılabilecek tekniklerden oluştuğu söylenebilir. Bilgisayarların doğal dili anlamlandırabilmesi için, dilin modellenebilen matematiksel bir çerçeveye dönüştürülmesi gerekir. Bu amaçla kullanılan başlıca teknikler şunlardır:

- **Dizgelere ayırma (Tokenization):** Bir metni kelimelere bölme işlemidir. Herhangi bir ayraç kullanarak bölme yapılabileceği gibi genelde boşluk karakterine göre bölme yapılır. Metin bölme işleminden sonra, bütün metin çok öğeli bir sözcük dizisine dönüşür.
- **Durak ifadelerin (stop words) kaldırılması:** Metin içerisindeki bağlaçların, niteleyicilerin, tek başına anlamsal katkısı olmayan sözcüklerin temizlenmesi işlemidir. Bu işlem esnasında kullanılmayan rakamlar, noktalama işaretleri, yeni satır (\n), tab (\t), başa dönme (\r) gibi özel karakterlerin temizliği de gerçekleştirilir.
- **N-Gram:** Belirli bir N sayısına bağlı olarak sıralı sözcükleri gruplamak için kullanılır.
- **Kelime vektörü, ağırlıklandırma ve skorlama:** Metin içerisinde n-gram yöntemleri ile oluşturulan vektörlerin kodlanması için belirli skorlama ve ağırlıklandırma teknikleri kullanılır. Bu tekniklerden olan Terim sıklığı (Term Frequency –TF), bir doküman içerisinde belirli terimlerin ne kadar sık yer aldığını hesaplamak için kullanılan yöntemdir. Çeşitli frekans ve normalizasyon formülleri kullanılarak hesaplama yapılır.

Otomatik soru cevap sistemleri bağlamında NLP, soru cevap çiftlerinin bulunduğu veri seti oluşturulduktan sonra anlamsız noktalama işaretleri, gereksiz karakterler (örneğin fazladan boşluklar, yanlış basılan karakterler vb.) gibi istenmeyen sembollerin

filtrelenmesi kısmında kullanılmıştır. Bu filtreleme işlemlerinin nasıl yapıldığı, tezin ilerleyen başlıklarında belirtilmiştir.

3.2. Metin Madenciliği

Son yıllarda donanım ve yazılım teknolojisindeki büyük gelişmelere bağlı olarak hızlı ilerlemeler gösteren veri madenciliği, farklı veri türlerine uygulanabilen bir alan haline gelmiştir. Donanım ve yazılım platformlarının geliştirilmesiyle bu durum, özellikle web ve sosyal ağlar için büyük miktardaki metin türündeki veriler için de gerçekleşmiştir. Farklı uygulamalar sonucu elde edilen metin verilerinin miktarının artmasıyla verilerin dinamik ve ölçülebilir bir şekilde öğrenilebilmesini sağlayan algoritmik tasarımların geliştirilmesine ihtiyaç duyulmaya başlanmıştır [9].

Metin madenciliği; yapılanmamış ya da yarı yapılı metinlerden anlamlı sonuçlar çıkarabilmek amacıyla metinlerin belirli süreçlerden geçirilip yapılandırılmış hale getirilerek çıktıların analiz edilmesini ifade etmektedir [10]. Metin madenciliği uygulaması temelde insan beynindeki en karmaşık analitik işleme sistemine sahip anlayışları, yazı dilinde analiz etmeyi amaçlamaktadır. Metin türündeki verilerde kelime ve cümlelerin her zaman doğal sıralanışında olmamasından dolayı metin madenciliğinde kullanılacak istatistiksel yöntemlerin doğru sonuçlar verebilmesi, NLP teknikleriyle kelime ve cümlelerin ön işlemesinin dikkatli bir şekilde yapılması ile mümkündür [11]. İşte bu noktada metin madenciliği devreye girmektedir. Otomatik soru cevaplama sistemlerinde ise geniş ve çeşitlendirilmiş veri kümeleri, modelin genel başarımını ve uyarlanabilirliğini artırmada kritik rol oynar. Bu bölüm, projede kullanılan metin madenciliği tekniklerinin uygulanışını detaylı bir şekilde açıklamaktadır.

Tablo 3.1 Metin Madenciliği ile Veri Madenciliğinin Karşılaştırılması

Metin Madenciliği	Veri Madenciliği
Önceden bilinmeyen bilgilerin keşfi	Nedensel ilişkiyi tanımlama
Dil işleme / NLP	Doğrudan işlem
Birden fazla tipte bulunan belirli bir alanda bulunmayan veri seti	Veri tabanında bulunan, formatlanmış veri
Yarı yapılandırılmış ve yapılandırılmamış veri seti	Yapılandırılmış veri seti

Tablo 3.1'de metin madenciliği ile veri madenciliğinin karşılaştırılması yapılmıştır. Mevcut verinin türüne göre metin madenciliği ya da veri madenciliği tercih edilebilir. Genel olarak, veri madenciliğinde veri hazır bir veri tabanından alınır ve formatı bellidir. Yapılandırılmamış veri seti kullanılarak nedensel ilişkiyi tanımlama amacı güdülür. NLP çalışmaları ile birlikte yürütülen metin madenciliğinde ise, metin kaynaklı literatürdeki diğer bir çalışma alanı olan birçok veri türü aynı anda daha dağınık veri kaynaklarında bulunmaktadır. Ayrıca yarı yapılandırılmış ve yapılandırılmamış veri seti kullanılarak önceden bilinmeyen bilgilerin keşfedilmesi amaçlanır. Metin madenciliğinin bilgisayar bilimi, istatistik, yapay zekâ, makine öğrenmesi, yönetim bilimi ve diğer bilimler ile iç içe olduğu söylenebilir [12]. Esasında metin madenciliği kavram çıkarma, bilgi çıkarma, bilgi geri getirme, web madenciliği, sınıflandırma, kümeleme ve NLP alanlarında kullanılır.

Metin madenciliği teknikleri, temelde dört kategoriye ayrılmaktadır [13]:

- **Sınıflandırma (Classification):** Nesnelerin önceden bilinen sınıflara yahut kategorilere dahil edilmesi işlemini ifade eder.
- **Birliktelik Analizi (Association Analysis):** Sıklıkla birlikte yer alan yahut gelişen sözcük veya kavramların belirlenerek doküman içeriğinin veya doküman kümelerinin anlaşılmasını sağlar.
- **Bilgi Çıkarımı (Information Extraction):** Dokümanların içerisindeki yararlı veri ya da ifadelerin bulunması işlemini ifade eder.
- **Kümeleme (Clustering):** Doküman kümelerinin temelini oluşturan yapıların keşfedilmesi işlemini ifade eder.

Veri kaynağı olarak metinlerin ele alındığı metin madenciliğinde, ilk olarak metin kaynaklarından amaca uygun olan veri seçimi yapılırken sonrasında tekrar eden ve etkisiz kelimeler filtrelenmekte ve metinler istenilen özelliklere göre parçalara ayrılmaktadır. Bunu yaparken de aynı zamanda metinler noktalama işaretlerinden, sayısal verilerden arındırılarak metinlerdeki büyük harfler küçük harflere de dönüştürülmektedir. Böylece metin analizi için bir ön işleme yapılmaktadır. Metin verilerinin ön işleme aşamasını geçmesinin ardından gösterim aşamasına geçilir ve bu aşamada bir kelimenin metin içerisinde ne kadar önemli olduğunun istatistiki olarak değerlendirilmesi için metnin

sayısal olarak ifade edilmesi işlemi gerçekleştirilmektedir [14]. Son olarak da pek çok aşamadan geçerek metin madenciliği için kullanıma hazır hale getirilen veriler metin kaynaklarından bilgi keşfi için sınıflandırma, birliktelik analizi, bilgi çıkarımı ve kümeleme teknikleri uygulanarak analiz edilmektedir.

Otomatik soru cevap sistemleri bağlamında toplanacak veri formatı, tek bir sözlük listesi (veya aynısını içeren bir JSON dosyasına giden yol) olmalıdır. Bir sözlük tek bir bağlamı ve ilişkili soruları temsil eder. Bu tür sözlüklerin her biri "context" ve "qas" olmak üzere iki öznitelik içerir.

- **context:** Sorunun sorulduğu paragraf veya metin.
- **qas:** Bir soru ve cevap listesi (aşağıdaki formatta verilmiştir).

Sorular ve cevaplar sözlükler olarak temsil edilir. “qas” içindeki her sözlük aşağıdaki biçime sahiptir.

- **id:** Soru için benzersiz bir ID. Tüm veri kümesinde benzersiz olmalıdır.
- **question:** Bir soru.
- **answers:** Soruya verilen doğru yanıtların listesi.

Tek bir cevap, aşağıdaki niteliklere sahip bir sözlük ile temsil edilir.

- **text:** Sorunun cevabı. Bağlamın bir alt dizesi olmalıdır.
- **answer_start:** Bağlamdaki cevabın başlangıç indeksi. [15]

Aşağıda **Şekil 3.1**’de örnek bir yapı görülmektedir:


```
  {
    "context": "Mustafa Kemal Atatürk[d] (1881,[e] Selanik, Osmanlı İmparatorluğu - 10 Kasım 1938, İstanbul, Türkiye), Türk mareşal, devlet adamı, yazar, Türk Kurtuluş Savaşı'nın başkomutanı, Türkiye Cumhuriyeti'nin kurucusu ve ilk cumhurbaşkanıdır. Türkiye'yi laik, sanayileşen bir ulusa dönüştüren kapsamlı ilerici reformlar üstlenmiştir. İdeolojik olarak sekülarist ve milliyetçi politikaları ve sosyo-politik teorileri Kemalizm olarak tanınmıştır.",
    "qas": [
      {
        "id": "00001",
        "question": "Mustafa Kemal Atatürk ne zaman doğmuştur?",
        "answers": [
          {
            "text": "1881",
            "answer_start": 26
          }
        ]
      },
      {
        "id": "00002",
        "question": "Atatürk'ün doğum yeri neresidir?",
        "answers": [
          {
            "text": "Selanik, Osmanlı İmparatorluğu",
```

```
            "answer_start": 35
          }
        ]},
```

Şekil 3.1 Örnek Veri Yapısı

3.2.1. Web Scraping

Web scraping, internet üzerinden programatik yollarla veri toplama işlemidir. Bu projede, özellikle Türkçe içerikli metinlerin toplanması için web scraping yöntemleri kullanılmıştır. Aşağıdaki Python kod blokları, belirli bir web sayfasından metin verilerini çekmek için kullanılmıştır:

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
url = "https://tr.wikipedia.org/wiki/Mustafa_Kemal_Atatürk"
driver.get(url)
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')

# Paragrafların bulunduğu bölümü seçme
paragraphs = soup.select("#bodyContent > #mw-content-text > div > p")

# Paragrafları temizleyip 'sup' etiketlerini filtreleyerek contexts array'ini oluşturma
contexts = []
for para in paragraphs:
    for sup in para.find_all('sup'):
        sup.decompose()
    text = para.get_text().strip()
    if text:
        contexts.append(f'`{text}`')

driver.quit()

# contexts array'ini istenilen formatta bir dosyaya yazdırma
with open("contexts.txt", "w", encoding="utf-8") as file:
    file.write("[")
    file.write(",\n".join(contexts))
    file.write("]")

print("contexts.txt dosyası oluşturuldu.")
```

Şekil 3.2 Web Scraping Python Script

Bu script, Selenium WebDriver kullanarak web sayfasının dinamik içeriğini işler ve BeautifulSoup ile HTML içeriğinden gerekli metinleri çıkarır. Script, sayfada bulunan paragrafları dolaşarak, referans etiketlerini (örneğin 'sup' etiketleri) kaldırır ve metinleri temiz bir şekilde contexts listesine ekler. Sonuç olarak, elde edilen metin verileri, daha sonra NLP işlemleri ve model eğitimi için kullanılmak üzere dosyaya kaydedilir.

3.2.2. Sentetik Veri Üretimi

Veri setlerinin çeşitliliği ve büyüklüğü, derin öğrenme modellerinin başarısında önemli bir rol oynar. Ancak, özellikle Türkçe gibi daha az kaynağa sahip dillerde yeterli ve çeşitli doğal veri bulmak zor olabilir. Bu nedenle, projede modelin daha sağlam (robust) olmasını sağlamak ve veri setindeki çeşitliliği artırmak için sentetik veri üretimi yöntemlerinden yararlanılmıştır. Sentetik veriler, bilgisayar simülasyonlarının veya algoritmalarının gerçek dünya verilerine alternatif olarak oluşturduğu açıklamalı bilgilerdir [15]. Başka bir deyişle, sentetik veriler gerçek dünyadan toplanmak veya gerçek dünyada ölçülmek yerine dijital dünyalarda oluşturulur. Bu tür veriler, sistemin çalışma sırasında işleyeceği gerçek verilerle aynı istatistiksel özelliklere sahip olmalıdır [16]. İki ana sentetik veri türü vardır: kısmi ve tam [6].

- **Kısmi sentetik veri:** Kısmen sentetik veri, gerçek bir veri kümesinin küçük bir bölümünü sentetik bilgilerle değiştirir. Bir veri kümesinin hassas kısımlarını korumak için bunu kullanabilirsiniz. Örneğin, müşteriye özel verileri analiz etmeniz gerekiyorsa ad, iletişim bilgileri ve belirli bir kişiye kadar izlenebilecek diğer gerçek dünya bilgileri gibi öznitelikleri sentezleyebilirsiniz.
- **Tamamen sentetik veri:** Tamamen sentetik veriler, tamamen yeni veriler üretilmesini içerir. Tamamen sentetik veri kümesi herhangi bir gerçek dünya verisi içermez. Bununla birlikte, gerçek verilerle aynı ilişkileri, grafik dağılımlarını ve istatistiksel özellikleri kullanır. Bu veriler gerçek kaydedilen verilerden gelmese de aynı sonuçları çıkarmanıza olanak tanır.

Bu projede, Türkçe bir şekilde sentetik soru cevap çiftleri oluşturmak için GPT-3.5 API kullanılmıştır. Aşağıda, Javascript programlama dilinde yazılan sentetik soru cevap çifti oluşturma kod blokları görülmektedir:

```
const fs = require("fs");
const { OpenAI } = require("openai");
require("dotenv").config();

const openai = new OpenAI({
  apiKey: process.env.OPENAI_API_KEY,
});

// 10 words - 1 question
const contexts = [
  // contextler buraya girilir.
];
```

```
// Mevcut JSON dosyasını okuma fonksiyonu
async function readExistingJson(filePath) {
  if (!fs.existsSync(filePath)) {
    return [];
  }

  const data = fs.readFileSync(filePath);
  return JSON.parse(data);
}

// OpenAI API'yi kullanarak Atatürk hakkında sorular ve cevaplar oluşturma fonksiyonu
async function generateQA(context, questionCount) {
  try {
    const response = await openai.chat.completions.create({
      model: "gpt-3.5-turbo",
      messages: [
        {
          role: "system",
          content: `Sen bir asistansın. Yeni sorular oluşturup cevap vereceksin. Kullanıcı,
sana soru ve cevap oluşturabilmen için 'context' verecek. Cevapları JSON yapısında
veriyorsun. JSON yapısında key olarak 'qas' olacak. 'qas' bir array olacak ve array içinde
de 'id', 'question' ve 'answers' key'lerinden oluşan bir obje olacak. answers, bir array
olacak, ve her bir elemanında 'text' ve 'answer_start' olacak. text'in içine yazacağın
cevap, 'answer_start' key'inin değerinden başlayacak. text'in içeriğindeki cevabın,
context'in içinde birebir aynı bir şekilde bulunuyor olmasına dikkat et. JSON yapısının
içine 'context' i koymayacaksın. JSON yapısını örnek olarak aşağıda veriyorum. JSON
yapısını oluşturduktan sonra, bu yapının içindeki verileri JSON şeklinde göndereceksin.
Örnek JSON yapısı: {
          "qas": [
              {
                  "id": "00002",
                  "is_impossible": False,
                  "question": "When was the series published?",
                  "answers": [
                      {
                          "text": "between 2006 and 2008",
                          "answer_start": 28,
                      }
                  ],
              },
              {
                  "id": "00003",
                  "is_impossible": False,
                  "question": "What are the three books in the series?",
                  "answers": [
                      {
                           "text": "The Final Empire, The Well of Ascension, and The Hero
of Ages",
                          "answer_start": 63,
                      }
                  ],
              },
          ],
      },`,
        },
        {
          role: "user",
          content: `Yapay zeka modelimi (Otomatik soru cevap sistemi) eğitmek için veri
seti oluşturmak istiyorum. Bana, vereceğim context için ${questionCount} adet soru ve
cevap oluşturur musun?. Context, \`${context}\` olsun.`,
        },
      ],
      max_tokens: 4000,
      n: 1,
      stop: null,
    });

    const qaPairs = JSON.parse(response.choices[0].message.content);
    return qaPairs.qas; // Sadece 'qas' arrayini döndür
  } catch (error) {
    console.error("Error generating QA:", error);
    return [];
  }
}
```

```
// Verileri JSON dosyasına yazma fonksiyonu
function writeJson(filePath, data) {
  fs.writeFileSync(filePath, JSON.stringify(data, null, 2));
  console.log("qa.json dosyasına veriler eklendi.");
}

// Gruplar halinde işlem yapma fonksiyonu
async function processGroup(group, filePath) {
  const qaData = await readExistingJson(filePath);

  const promises = group.map(async (context, i) => {
     console.log(`Generating QA for context ${context.slice(0, 50)}... (${i +
1}/${group.length})...`);
    const questionCount = Math.ceil(context.split(" ").length / 10); // Soru sayısını
belirle
    const newQAData = await generateQA(context, questionCount);
    qaData.push({ context, qas: newQAData }); // Her context ile birlikte soruları ekle
  });

  await Promise.all(promises);

  // Tüm verileri JSON dosyasına yaz
  writeJson(filePath, qaData);
}

// Array'i belirli büyüklükteki parçalara bölme fonksiyonu
function chunkArray(array, chunkSize) {
  const chunks = [];
  for (let i = 0; i < array.length; i += chunkSize) {
    chunks.push(array.slice(i, chunkSize + i));
  }
  return chunks;
}

// Ana fonksiyon
async function main() {
  const filePath = "qa.json";

  // contexts array'ini 5'li gruplara böl
  const contextGroups = chunkArray(contexts, 5);

  // Her bir grubu işleme al
  for (const group of contextGroups) {
    await processGroup(group, filePath);
  }
}

// Ana fonksiyonu çalıştır
main();
```

Şekil 3.3 Sentetik Veri Oluşturma Scripti

Bu JavaScript script'i, OpenAI'nin GPT-3.5 modelini kullanarak otomatik olarak soru-cevap çiftleri üretmek için tasarlanmıştır. Script, öncelikle OpenAI Uygulama Programlama Arabirimi (API) ile bağlantı kurar. Kullanıcı tarafından sağlanan bağlam (context) bilgisine dayanarak, belirtilen sayıda soru ve cevap üretir. Bu süreç, OpenAI'nin sohbet API'ni kullanarak, sistem rolüne özel talimatlar ve kullanıcı rolüne context ile ilgili soruları içeren mesajlar gönderilmesiyle gerçekleştirilir. Üretilen cevaplar, JSON formatında alınır ve bu JSON dosyası daha sonra veri işleme adımlarına iletilmek üzere kaydedilir.

3.2.3. Veri Ön İşleme

Veri seti oluşturma aşamasında elde edilen metinleri kullanabilmek için öncelikle metinlerin düzenlenmesi gerekmektedir. Bu düzenleme ile metinlerde bulunan yazım yanlışlarını, kullanılmayacak olan internet sayfası linkleri, kullanıcı adı gibi verileri kısaltma olarak kullanılan kelimelerde düzenlemeler yaptıktan sonra kullanıma hazır hale getirilir [17].

Bu projede veri ön işleme olarak sentetik bir şekilde oluşturulan veri seti içinde tekrar eden verileri filtrelemeye yarayan bir script hazırlanmıştır. Aşağıda bu filtrelemeyi yapan script görülmektedir:

```
const fs = require('fs');

fs.readFile('qa.json', 'utf8', (err, data) => {
    if (err) {
        console.error('Error reading qa.json:', err);
        return;
    }

    let jsonData = JSON.parse(data);
    let currentId = 1;
    const uniqueQuestions = new Set();

    jsonData.forEach(item => {
        const context = item.context;
        const filteredQas = [];

        item.qas.forEach(qa => {
            if (!uniqueQuestions.has(qa.question)) {
                uniqueQuestions.add(qa.question);

                qa.id = String(currentId).padStart(3, '0');
                currentId++;

                qa.answers.forEach(answer => {
                    const answerText = answer.text;
                    const answerStart = context.indexOf(answerText);
                    if (answerStart !== -1) {
                        answer.answer_start = answerStart;
                    } else {
                        console.log(`Answer '${answerText}' not found in context.`);
                    }
                });

                filteredQas.push(qa);
            }
        });

        item.qas = filteredQas;
    });

    fs.writeFile('filtered_qa.json', JSON.stringify(jsonData, null, 2), (err) => {
        if (err) throw err;
        console.log('Filtered data has been saved to filtered_qa.json');
    });
```

Şekil 3.4 Answer Indeks Düzeltme

Bunun dışında cevabın direkt context'te bulunmadığı verileri de çıkarmak gerekmektedir. Bunu yapan script ise aşağıda bulunmaktadır:

```
const fs = require("fs");
const fileName = "qa";
const data = JSON.parse(fs.readFileSync(`${fileName}.json`));

function filterQuestions(dataArray) {
  return dataArray.map((item) => {
    const context = item.context;
    const filteredQAs = item.qas.filter((qa) => {
      return qa.answers.every((answer) => {
        return context.includes(answer.text);
      });
    });
    return {
      ...item,
      qas: filteredQAs,
    };
  });
}
const filteredData = filterQuestions(data);
fs.writeFile(`filtered_${fileName}.json`, JSON.stringify(filteredData, null, 2), (err) =>
{
  if (err) throw err;
  console.log(`Filtered data has been saved to filtered_${fileName}.json`);
```

Şekil 3.5 Filtreleme Scripti

Bir de sorulara verilen cevapların, context'te kaçıncı karakterden itibaren başladığını belirten "answer_start" değeri, GPT-3.5 modeli tarafından yanlış bir şekilde oluşturulabilmektedir. Bu durum, GPT-3.5 modelinin matematiksel görevleri doğru bir şekilde tamamlayamasından kaynaklanmaktadır. Aşağıda, answer_start değerini düzelten script bulunmaktadır:

```
const fs = require("fs");
const fileName = "qa_final";
fs.readFile(`${fileName}.json`, "utf8", (err, data) => {
  if (err) {
    console.error("Error reading qa.json:", err);
    return;}
  let jsonData = JSON.parse(data);
  let currentId = 1;
  jsonData.forEach((item) => {
    const context = item.context;
    item.qas.forEach((qa) => {
      qa.id = String(currentId).padStart(5, "0");
      currentId++;

      qa.answers.forEach((answer) => {
        const answerText = answer.text;
        const answerStart = context.indexOf(answerText);
        if (answerStart !== -1) {
          answer.answer_start = answerStart;
        } else {
          console.log(`Answer '${answerText}' not found in context.`);
        }
      });
    });});
  fs.writeFile(`${fileName}.json`, JSON.stringify(jsonData, null, 2), (err) => {
    if (err) throw err;
    console.log(`Filtered data has been saved to ${fileName}.json`);
  });
```

Şekil 3.6 Soru ID Düzenleme Scripti

Ön işleme sürecinden sonra elde edilen toplam veri sayısı ve kullanım alanları aşağıda **Tablo 3.2**'de gösterilmektedir.

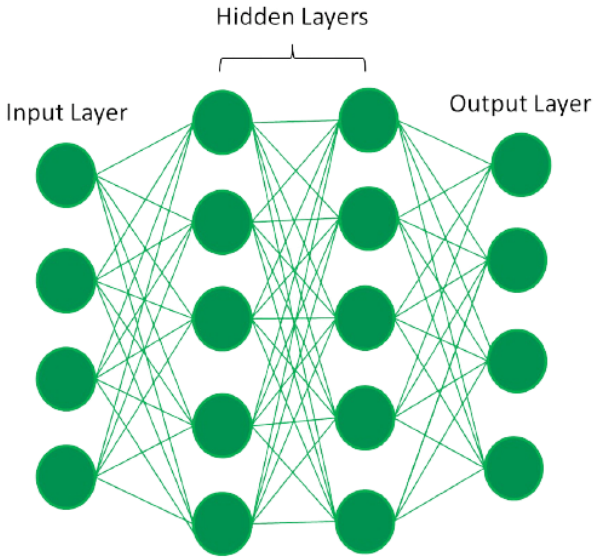
Tablo 3.2 Veri Seti Eğitim Test Dağılımı

	Bağlam (Context)	**Soru-Cevap (QA)**
Eğitim (Train)	914	8226
Test	228	2056

3.3. Derin Öğrenme

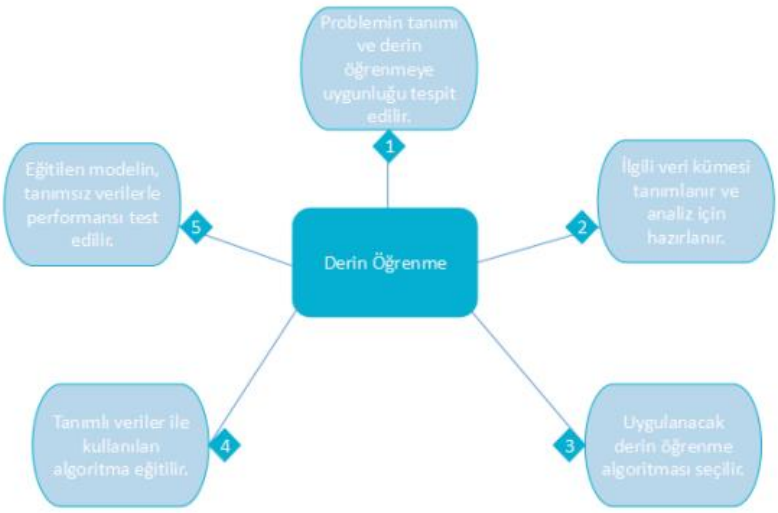
Derin Öğrenme son yılların en önemli konuları arasındadır. Derin öğrenme, doğrusal olmayan işlem birimlerinin birçok katmanını kullandığı, birbirini izleyen her katmanın girdi olarak bir önceki katmanların çıktısını aldığı bir makine öğrenmesidir. Derin öğrenme algoritmaları, yapay sinir ağlarının (ANN) daha karmaşık bir yapısal versiyonu olarak kabul edilebilir. Derin öğrenmenin tek katmanlı yapay sinir ağlarından farkı, lineer olmayan problemlerin çözülmesine olanak tanımasıdır [18]. Derin öğrenme yaklaşımında gizli katmanlar bulunmaktadır.

Derin öğrenmenin kullanım alanları her geçen gün artmaktadır. Günümüzde GPU hızlarının artmasıyla birlikte, derin ağlar çok daha hızlı ve etkin olarak eğitilebilmekte ve bu sayede artan başarı oranları ile hemen hemen her alanda yaygınlaşarak kullanılmaktadır. Derin öğrenme modeli, makine öğreniminin birçok uygulamasını sağlamış ve yapay zeka alanındaki başarıyı artırmıştır. Derin sinir ağının hücre yapısı aşağıdaki **Şekil 3.7Şekil 3.7**'de gösterilmektedir [19].



Şekil 3.7 Derin Sinir Ağı

Geleneksel makine öğrenmesi algoritmaları doğrusal yapıdayken, derin öğrenme algoritmaları problemin karmaşıklığına göre değişen bir hiyerarşi modeline sahiptir. Derin öğrenme süreci başarı sonucu belirli bir seviyeye gelene kadar devam eder. Bu süreçte verilerin geçmesi gereken genel adımlar **Şekil 3.8**'de gösterilmektedir [20].



Şekil 3.8 Derin Öğrenme Adımları

3.3.1. Derin Öğrenme Modelleri

Derin öğrenme modelleri, dil modellemesi ve metin anlama konusunda son yıllarda önemli ilerlemeler kaydetmiştir. Bu bölüm, projede kullanılan derin öğrenme modellerinin ayrıntılarını sunar. Özellikle, Transformer mimarisine dayanan modeller, dil işleme görevlerinde yeni standartlar belirlemiştir. Projede kullanılan BERT ve ELECTRA gibi modeller, metinler üzerinde karmaşık dil işleme görevlerini yerine getirebilmek için önceden eğitilmiş modelleri ince ayar (fine-tuning) yöntemi ile uyarlamaktadır. Bu modellerin seçimi, performansları ve projedeki uygulamaları detaylı bir şekilde incelenmiştir.

Günümüz derin öğrenme modellerinin temeli insan beynindeki bir nöronun bilgisayar benzetmesi yapılmaya çalışılmasıyla ortaya çıkmış bir süreçtir. Donald Hebb'in sinir hücrelerinin incelenmesiyle başlayan süreç, daha sonra bilgisayar ortamında matematiksel olarak modelinin oluşturulmasıyla yapay sinir ağlarının temelini oluşturmuştur [21]. Yapay sinir ağlarının gelişimi ve ilerleyişi sürecinde yıllar geçmiş ve son yıllarda derin öğrenme terimi ortaya atılmış, araştırmalar bu alanda yoğunlaşmıştır.

Derin öğrenme, özellikle sınıflandırma, tanıma ve tespit gibi alanlarda kullanılmaktadır [18]. Yapılan çalışmalar bu yönde ağırlık kazanmıştır. Son zamanlarda, derin öğrenme algoritmaları çeşitli alanlarda yaygın olarak kullanılmakta ve günlük yaşamımızda birçok teknolojik yapı içerisinde yer almaktadır. Bu algoritmaların başarılı sonuçlar elde etmesi, bu alana olan ilgiyi artırmıştır. Yapılan pek çok sınıflandırma, tanıma çalışmaları mevcuttur.

3.3.1.1. Yapay Sinir Ağları (ANN)

Yapay Sinir Ağları (ANN); girdi, çıktı ve ağırlık değerleri bir algoritmaya göre kendi kendine öğrenebilen, çıkarımlar yapabilen ve ayrıca kararlar alabilen doğrusal olmayan problemlerin de çözülebilmesine olanak sağlayan insan beynindeki biyolojik sinir yapısını modellemektedir [22, 23]. Tek katmanlı algılayıcıların ilk sinir ağlarından biri olduğu söylenebilir.Bu yapı, birden fazla girdi alıp bu girdilerden çıktı üreten bir sinir hücresine dayanır. Ve (and), Veya (or), Değil (not) durumları gibi doğrusal fonksiyonlarla ilgili problemlerde kullanılabilirler.

Bu ağ yapısında öğrenmeyi sağlamak için eğitim seti adı verilen ve örneklerden oluşan bir veri seti kullanılmaktadır. Tek bir nöron için birçok girdi yeterli olmayacağından, paralel işlem için birden fazla nörona ihtiyaç vardır. Veriler giriş katmanından alınır ve gizli katmana aktarılır. Ara katman, en az bir katman olarak farklılık gösterebilir. Her katmanın çıkışı, bir sonraki katmanın girişi olmaktadır. Her nöron, bir sonraki katmandaki nöronlara bağlanır. Çıktı katmanı, önceki katmandan alınan verileri işleyerek ağın çıktısını belirler. Çıktıların sayısı çıktı katmanındaki elemanların sayısına eşittir. Ağın öğrenmesi için örnek girdi ve çıktılardan oluşan bir eğitim seti gereklidir. Aktivasyon fonksiyonuna girecek girdi, ağırlıklar, transfer fonksiyonu ve net girdidir. Yani ağın giriş değerleri, ağırlıkları ve ağda uygulanan aktivasyon işlemi kullanılarak ağın çıktısı elde edilir [19] . Bir çıktı katmanı, önceki katmandaki işlem öğelerine bağlı birden fazla işlem öğesi ve işlem öğesi içerir. Çok katmanlı ağlarda çeşitli aktivasyon fonksiyonları kullanılabilir. Sigmoid işlevi bunlardan birisidir. Etkinleştirme işlevi, girdi katmanları ve çıktı katmanları tarafından oluşturulan eğri eşitlemesini etkinleştirir. Ağın performansını etkilediği için uygun aktivasyon işlevinin seçilmesi önemlidir. Aktivasyon fonksiyonunu doğrusal olarak seçmek de mümkündür. Sigmoid aktivasyon fonksiyonu 0 ile 1 arasındaki değerleri hesapladığından, bir olayın meydana gelme olasılığını bulan modellerde kullanılır. Yapay sinir ağlarında, girdiyi işleyerek bu girdi sonucunda üreteceği çıktıyı belirleyen aktivasyon fonksiyonları kullanılmaktadır. Derin öğrenme modelinde, hata değerini minimize etmek için kullanılan geriye besleme işleminde türev hesabı kullanılmaktadır. Bu sebeple genellikle türevi rahatlıkla alınabilen aktivasyon fonksiyonları seçilmektedir. Bir aktivasyon fonksiyonu, yapay bir nörondaki, girdilere dayalı bir çıkış sağlayan fonksiyondur [24].

3.3.1.2. Transformers

Transformer mimarisi, Google tarafından geliştirilen ve 2017 tarihli "Attention Is All You Need" makalesinde önerilen multi-head dikkat mekanizmasına dayanan bir derin öğrenme mimarisidir. Metin, token adı verilen sayısal temsillere dönüştürülür ve her bir token, bir kelime gömme tablosundan bakılarak bir vektöre dönüştürülür. Her katmanda, her bir belirteç daha sonra bağlam penceresi kapsamında diğer (maskelenmemiş) belirteçlerle paralel birçok kafalı dikkat mekanizması aracılığıyla bağlamsallaştırılır ve önemli belirteçler için sinyalin güçlendirilmesine ve daha az önemli belirteçlerin azaltılmasına olanak tanır [25].

Transformer modelleri, özellikle paralel işleme kabiliyetleri sayesinde, büyük veri kümeleri üzerinde hızlı ve etkili bir şekilde çalışabilir. Bu modeller, kelime öbeklerinin anlamları arasındaki ilişkileri daha iyi anlamak ve çeşitli dil işleme görevlerinde üstün performans sergilemek için geniş bir kontekst penceresini değerlendirebilir. Bu yönleriyle Transformer, QA sistemleri, metin özetleme, makine çevirisi ve duygu analizi gibi birçok NLP uygulamasında tercih edilen bir model haline gelmiştir [7].

3.3.1.2.1. BERT

Bidirectional Encoder Representations from Transformers (BERT), transformatör mimarisine dayanan bir dil modelidir ve önceki son teknoloji modellere göre çarpıcı bir şekilde iyileştirilmesiyle dikkat çekmektedir. Ekim 2018'de Google'daki araştırmacılar tarafından tanıtılmıştır. 2020 yılında yapılan bir literatür araştırması, "bir yıldan biraz daha uzun bir süre içinde, BERT'in Doğal Dil İşleme (NLP) deneylerinde her yerde bulunan bir temel haline geldiği ve modeli analiz eden ve geliştiren 150'den fazla araştırma yayınının sayıldığı" sonucuna varmıştır [8, 25].

BERT modeli, öncelikle büyük bir metin korpusu üzerinde denetimsiz (unsupervised) bir şekilde ön eğitime tabi tutulur. Bu ön eğitim sırasında, model rastgele maskeleme yaparak metin içindeki eksik kelimeleri tahmin etmeye çalışır ve ardından next sentence sonraki cümleyi tahmin etme (prediction) görevi ile cümleler arası ilişkileri anlamayı öğrenir. Bu süreçler, modelin geniş bir bağlamda dil yapısını ve anlamını kavramasını sağlar [26].

Projede BERT, özellikle Türkçe diline uyarlanmış bir versiyonu kullanılmaktadır. Bu model, verilen sorulara ve metinlere dayanarak doğru cevapları üretebilmek için ince ayar (fine-tuning) yoluyla özelleştirilmiştir. BERT'in çift yönlü yapısı sayesinde, verilen metin bağlamında soruların nüanslarını ve ince anlam katmanlarını başarıyla çözümleyebilmekte ve böylece daha doğru ve anlamlı cevaplar üretebilmektedir.

BERT'in gücü iki aşamalı sürecinden gelmektedir [9]:

- **Ön Eğitim (Pre Training):** Ön eğitim, BERT'in büyük miktarda veri üzerinde eğitildiği aşamadır. Sonuç olarak, bir cümledeki maskelenmiş kelimeleri tahmin etmeyi (MLM görevi) ve bir cümlenin başka bir cümleyi takip edip etmediğini tahmin etmeyi (NSP) öğrenir. Bu aşamanın çıktısı, dilin genel amaçlı bir "anlayışına" sahip önceden eğitilmiş bir NLP modelidir

- **İnce ayar (Fine Tuning):** İnce ayar, önceden eğitilmiş BERT modelinin belirli bir görev üzerinde daha fazla eğitildiği yerdir. Model önceden eğitilmiş parametrelerle başlatılır ve modelin tamamı bir sonraki görev üzerinde eğitilerek BERT'in dil anlayışını eldeki görevin özelliklerine göre ince ayar yapmasına olanak tanır.

3.3.1.2.2. ELECTRA

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately, iki transformer modelini eğiten yeni bir ön eğitim yaklaşımına sahip bir transformer modelidir: üreteç (generator) ve diskriminatör. Üreteç, maskelenmiş bir dil modeli olarak eğitilen dizideki belirteçlerin yerini alır ve diskriminatör, dizide hangi belirteçlerin üreteç tarafından değiştirildiğini belirlemeye çalışır. Bu ön eğitim görevine değiştirilen belirteç algılama adı verilir ve girdiyi maskelemenin yerine geçer [10].

ELECTRA, geleneksel BERT modelinin yerine geçen ve bazı yönlerden daha verimli olan bir yaklaşımdır. Bu model, metin içindeki yanlış yerleştirilmiş kelimeleri tanıma üzerine kurulmuş bir yapıdadır ve bu sayede, daha az kaynakla daha hızlı ve daha etkin bir şekilde eğitilebilir [27].

Projemizde, ELECTRA modeli özellikle fine-tuning aşamasında kullanılmıştır. Bu model, verilen metin bağlamında sorulara ve cevaplara yönelik detaylı dil işleme yetenekleri sayesinde, özellikle ince anlam ayrıntılarını çözümlemede büyük başarı göstermesi beklenmiştir, ancak Türkçe karakterlere özel bir şekilde uyarlanmış versiyonu bulunamadığından BERT'e göre daha düşük performans göstermiştir.

3.3.2. Derin Öğrenme Algoritmaları

Derin öğrenme algoritmaları, yapay sinir ağlarının verimli bir şekilde eğitilmesi ve optimizasyonu için kullanılan matematiksel yöntemlerdir. Bu algoritmalar, büyük veri kümesi üzerinde eğitilerek başarılı tahminler yapabilirler [11].

3.3.2.1. Geri Yayılım (Backpropagation)

Geri yayılım, yapay sinir ağlarını eğitmek için kullanılan temel bir algoritmadır. Bu yöntem, çıktı katmanından başlayarak her katmanda hata oranını hesaplar ve bu hataları ağın giriş katmanına doğru geri yayarak ağın ağırlıklarını günceller. Bu algoritma denklemin local minimuna yakınsamak için birinci türev kullanır [12]. Bu süreç, modelin

eğitim verileri üzerindeki performansını sürekli olarak iyileştirir ve sonuçta daha etkin öğrenme sağlar.

3.3.2.2. Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları, bir nöronun çıktısını belirlemek için kullanılır ve non-lineer problemlerin çözülmesini sağlar. Yapay sinir ağlarına doğrusal olmayan gerçek dünya özelliklerini tanıtmak için aktivasyon fonksiyonuna ihtiyaç duyulmaktadır. Temel olarak basit bir yapay sinir ağında x girdiler, w ağırlıklar olarak tanımlanır ve ağın çıkışına aktarılan değere f(x) yani aktivasyon işlemi uygulanır. Daha sonra bu, nihai çıkış ya da bir başka katmanın girişi olacaktır [13].

En yaygın kullanılan aktivasyon fonksiyonları arasında ReLU (Rectified Linear Unit), sigmoid ve tanh (hiperbolik tanjant) bulunur [28]. ReLU, genellikle gizli katmanlarda kullanılırken, sigmoid ve tanh fonksiyonları çıktı katmanında tercih edilir. Bu fonksiyonlar, modelin karmaşık veri yapılarını modellemesine ve daha geniş bir fonksiyon uzayında etkili bir şekilde öğrenmesine olanak tanır.

3.3.2.3. Adam Optimizasyonu

Adam optimizasyon algoritması, stokastik gradyan inişinin bir varyasyonudur ve özellikle büyük veri setleri ve parametre sayıları ile çalışırken etkili bir şekilde çalışır. Adam, her bir parametre için öğrenme hızını adaptif olarak ayarlar, bu da modelin daha hızlı ve daha etkin bir şekilde yakınsamasını sağlar. Adam algoritması, son zamanlarda doğal dil işleme konularında geniş kabul görmüştür. Bu algoritma deep learning konularında çok kullanılmaktadır, çünkü iyi sonuçları hızlı şekilde vermektedir [14].

Moment ve RmsProb algoritmaları başlangıç değeri olarak sıfırdan başlamakta ve bu hız bakımında algoritmaları dezavantajlı duruma getirmektedir [14]. RMSProp ve Momentum algoritmalarında doğrudan türeve uygulanan yön düzeltmesi Adam algoritmasında önce iterasyon sayısına bağlı bir düzeltmeye tabi tutulur:

$$V_{dw_{corr}} = \frac{V_{dw}}{1 - {\beta_1}^t} \qquad S_{dw_{corr}} = \frac{S_{dw}}{1 - {\beta^t}_2}$$

Şekil 3.9 Adam Algoritması Düzeltmesi

Bu düzeltme, momentum ve RMSProp algoritmalarının aksine Adam algoritmasının ilk iterasyondan itibaren etkin bir şekilde yakınsamaya başlamasını sağlar [14].

3.4. Modelin Eğitilmesi

Bu bölüm, tez çalışmamızın temelini oluşturan otomatik soru cevaplama sistemi için kullanılan modelin eğitim sürecini detaylı bir şekilde ele alır. Modelin eğitimi, derin öğrenme tekniklerinin ve doğal dil işleme algoritmalarının bir arada kullanıldığı kapsamlı bir süreçtir. Bu süreç, donanım ortamının seçimi, model mimarilerinin karşılaştırılması, en uygun modelin belirlenmesi ve modelin parametre ayarlarının optimizasyonu gibi adımları içerir. Ayrıca, modelin eğitimi sırasında modelin doğrulanması ve test süreçleri gibi aşamalar, "BULGULAR ve TARTIŞMA" başlığı altında detaylandırılacaktır.

Bu bölümde, kullanılan tekniklerin seçim gerekçeleri, uygulanan yöntemler ve bu yöntemlerin sonuçları üzerinde durulacak, böylece modelin eğitim sürecinin tez çalışmamızdaki rolü ve önemi ortaya konulacaktır.

3.4.1. Donanım Ortamı

Modelin eğitimi, Google Colab platformu kullanılarak gerçekleştirilmiştir. Bu platform, bulut tabanlı ve yüksek performanslı hesaplama kaynakları sunar, böylece geniş veri setleri ile derin öğrenme modelleri üzerinde çalışmak için ideal bir ortam sağlar. Kullanılan donanım ortamı aşağıda liste şeklinde verilmiştir:

- **Grafik İşlem Birimi (GPU):** L4 GPU kullanılmıştır, bu GPU yüksek işlem kapasitesi ile derin öğrenme algoritmalarının hızlı ve verimli bir şekilde eğitilmesini sağlar.
- **Depolama ve Bellek:** Colab, dinamik bellek yönetimi sayesinde büyük veri setlerinin işlenmesine olanak tanırken, kullanıcıların depolama ihtiyaçlarına göre esneklik sunar.

Bu donanım ortamı, özellikle hesaplama yoğun işlemler ve büyük ölçekli veri analizleri için mükemmeldir. Modelin eğitim sürecinde karşılaşılan performans gereksinimleri, bu güçlü donanım sayesinde kolaylıkla karşılanmıştır.

3.4.2. Modelin Seçilmesi

Model seçimi, otomatik soru cevaplama sisteminin başarısında kritik bir rol oynar. Bu projede, doğal dil işleme görevlerinde öne çıkan BERT ve ELECTRA gibi Transformer tabanlı modeller üzerinde durulmuştur. Her iki model de dilin anlamını derinlemesine

anlayabilme kapasiteleri ve metin üzerindeki karmaşık ilişkileri modelleyebilme yetenekleri nedeniyle seçilmiştir.

Otomatik soru cevaplama sistemi için model olarak BERT tabanlı bir model olan "dbmdz/bert-base-turkish-cased" modeli seçilmiştir. Bu seçim, modelin Türkçe karakterlere ve dil yapısına olan uyumluluğu göz önünde bulundurularak yapılmıştır. Modelin mevcut versiyonu, Türkçe OSCAR derleminin (Vikipedi dökümü) filtrelenmiş ve cümle bölümlerine ayrılmış bir versiyonu, çeşitli OPUS derlemleri ve Kemal Oflazer tarafından sağlanan özel bir derlem üzerinde eğitilmiştir. Nihai eğitim derlemi 35 GB boyutunda ve 44.040.976.662 tokenden oluşmaktadır. Google'ın TensorFlow Araştırma Bulutu (TFRC) sayesinde bir TPU v3-8 üzerinde 2M adım için bir cased modeli eğitilmiştir. Model, BERT tabanlı bir model olduğundan 110M adet parametre ile hesaplanmıştır [29].

ELECTRA tabanlı Türkçe karakterlere uyumlu bir modelin bulunmaması, araştırma kapsamında BERT modelinin tercih edilmesinde belirleyici bir faktör olmuştur. Ayrıca, yapılan denemelerde "dbmdz/bert-base-turkish-cased" modeli, F1 Skoru ve Tam Eşleşme (Exact Match) metriklerinde diğer modellere göre daha yüksek performans göstermiştir. Bu metriklerin bulunduğu tablo, **Tablo 5.1** ve **Tablo 5.2**'de daha detaylı bir şekilde incelenebilir. Bu metrikler, modelin Türkçe dilindeki doğruluk ve bağlam anlama kapasitesinin üstünlüğünü kanıtlar niteliktedir.

3.4.3. Modelin Parametreleri

Bu projede kullanılan "dbmdz/bert-base-turkish-cased" BERT modeli, optimal sonuçlar elde etmek için çeşitli parametre ayarlamalarıyla eğitilmiştir. Bu bölüm, modelin parametrelerini ve eğitim sürecinde yapılan ayarlamaları detaylandırmaktadır.

Model eğitimi sırasında dikkate alınan başlıca parametreler şunlardır:

- **Maksimum Sekans Uzunluğu (max_seq_length):** Modelin kabul edebileceği maksimum token sayısı olarak belirlenen bu parametre, 512 olarak ayarlanmıştır. Bu değer, modelin daha geniş bağlamları işleyebilmesini sağlayarak, soru ve paragraflar arasındaki ilişkileri daha iyi anlamasına yardımcı olur.

- **Batch Boyutu (batch_size):** Model eğitimi sırasında her bir işlemde işlenen veri miktarı. Eğitim verimliliğini artırmak ve bellek kullanımını optimize etmek için 16 olarak ayarlanmıştır.
- **Öğrenme Oranı (learning_rate):** Model ağırlıklarının her bir eğitim adımında ne kadar güncelleneceğini belirleyen faktör. Bu projede kullanılan öğrenme oranı 2e-5 olarak belirlenmiştir, bu düşük oran modelin daha stabil ve kontrollü bir şekilde öğrenmesini sağlar.
- **Epoch Sayısı:** Modelin eğitim veri seti üzerinde kaç kez tam bir geçiş yapacağı. Bu proje için belirlenen epoch sayısı 4 olarak ayarlanmıştır, bu sayede model yeterince eğitilmiş ve aynı zamanda aşırı uyumdan (overfitting) kaçınılmıştır.

Belirtilen parametreler ve ayarlamalar sayesinde, model, özellikle Türkçe dilindeki metinler üzerinde yüksek doğruluk ve bağlam anlama kapasitesi sergilemiştir. Bu, F1 Skoru ve Tam Eşleşme (Exact Match) değerleri ile ölçülmüş ve modelin üstün performansını kanıtlamıştır. Bu ölçüm değerleri arasındaki bağıntı, **Şekil 5.1** ve **Şekil 5.2**'den incelenebilir.

4. SİSTEMİN UYGULANMASI

Bu bölüm, otomatik soru cevaplama sisteminin pratik uygulamalarını ve teknik detaylarını kapsamlı bir şekilde ele alır. Projede geliştirilen sistem, modern web teknolojileri ve ileri düzey doğal dil işleme modelleri kullanılarak tasarlanmıştır. Sistem, Flask web framework üzerinde kurulan bir backend ve React.js kullanılarak geliştirilen bir frontend'den oluşmaktadır. Bu bölümde, backend ve frontend bileşenlerinin nasıl entegre edildiği, sistem mimarisinin detayları ve kullanılan teknolojilerin her birinin rolü açıklanacaktır.

Özellikle, Flask backend'in nasıl bir RESTful API servisi olarak konfigüre edildiği, veri alışverişi için hangi "/predict" endpoint'inin nasıl tasarlandığı ve bu servislerin nasıl güvenli bir şekilde kullanıldığı üzerinde durulacaktır. Ayrıca, React.js ile geliştirilen frontend'in, kullanıcı arayüzü tasarımı, etkileşimli bileşenlerin nasıl oluşturulduğu ve kullanıcı deneyimini nasıl iyileştirdiği detayları ile ele alınır. İki bileşenin entegrasyonu ve veri akışının yönetimi teknik kısımlar da bu bölümde tartışılacaktır.

4.1. Flask ile Backend Geliştirme

Projede, otomatik soru cevaplama sisteminin backend kısmı, Python tabanlı bir mikro web framework olan Flask kullanılarak geliştirilmiştir. Flask, hafifliği ve modüler yapısıyla bilinir, bu da onu küçük ölçekli projelerden büyük, karmaşık uygulamalara kadar geniş bir yelpazede kullanım için ideal kılar.

Backend, sistemin iş mantığını ve veri yönetimini kapsar. Flask ile geliştirilen API, farklı frontend uygulamalarından gelen HTTP isteklerini kabul eder ve cevaplar üretir. Bu projede, API'nin temel fonksiyonları şunları içerir:

- Soru Alma ve İşleme: Kullanıcılardan gelen soruları alır, gerekli ön işleme adımlarından geçirir ve doğru formatta işlemek üzere hazırlar.
- Modelle Etkileşim: Ön işlenmiş soruları, cevapları üretmek üzere eğitilmiş derin öğrenme modeline ileterek gerekli tahminlerin yapılmasını sağlar.
- Cevapların Dönüşü: Model tarafından üretilen cevapları kullanıcıya geri döndürür, bu süreç JSON formatında veri transferini içerir.

Aşağıda, Flask ile yazılan backend kodunun açıklaması bulunmaktadır:

```
from flask import Flask, request, jsonify
import torch
from transformers import BertTokenizerFast, BertForQuestionAnswering

app = Flask(__name__)
```

Şekil 4.1 Flask Backend Bölüm 1

- Flask: Flask uygulamasını oluşturmak için kullanılır.
- request: Gelen istekleri işlemek için kullanılır.
- jsonify: JSON cevapları oluşturmak için kullanılır.
- torch: PyTorch kütüphanesi, derin öğrenme işlemleri için.
- BertTokenizerFast ve BertForQuestionAnswering: BERT modelini yüklemek ve kullanmak için transformer kütüphanesinden modüller.

```
model = BertForQuestionAnswering.from_pretrained('cozfuttu/results')
tokenizer = BertTokenizerFast.from_pretrained('dbmdz/bert-base-turkish-cased')
```

Şekil 4.2 Flask Backend Bölüm 2

- Model ve tokenizer, önceden eğitilmiş parametrelerle yüklenir. Bu örnekte, BERT modeli Türkçe için özel olarak eğitilmiş ve bir dizine kaydedilmiş.

```
@app.route('/predict', methods=['POST'])
def predict():
```

```
  data = request.get_json(force=True)
  question = data['question']
  context = data['context']

  inputs     =     tokenizer.encode_plus(question,     context,     return_tensors="pt",
add_special_tokens=True)
  input_ids = inputs["input_ids"].tolist()[0]

  outputs = model(**inputs)
  answer_start_scores = outputs.start_logits
  answer_end_scores = outputs.end_logits

  answer_start = torch.argmax(answer_start_scores)
  answer_end = torch.argmax(answer_end_scores) + 1

  answer                                                                          =
tokenizer.convert_tokens_to_string(tokenizer.convert_ids_to_tokens(input_ids[answer_star
t:answer_end]))`

  response = jsonify(answer=answer)
  response.headers.add('Access-Control-Allow-Origin', '*')
  return response
```

Şekil 4.3 Flask Backend Bölüm 3

- /predict endpoint'i, HTTP POST isteklerini kabul eder. Bu endpoint, kullanıcıdan bir question ve context alır.
- get_json(force=True): Gelen JSON verisini zorunlu olarak alır.
- encode_plus: Soru ve bağlamı BERT modelinin işleyebileceği biçime dönüştürür.
- model(**inputs): Modeli kullanarak sorunun cevabını tahmin eder.
- convert_tokens_to_string: İndekslerden elde edilen token'ları string'e çevirir.
- En son JSON formatında cevap hazırlanır ve CORS (Cross-Origin Resource Sharing) başlıklarını ekleyerek herhangi bir domainden isteğe izin verir. Bu, farklı domainlerdeki frontend uygulamalarının API'ye erişebilmesini sağlar.

```
if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

Şekil 4.4 Flask Backend Bölüm 4

- Uygulama, geliştirme modunda ve 5000 portunda çalıştırılır.

4.2. React ile Frontend Geliştirme

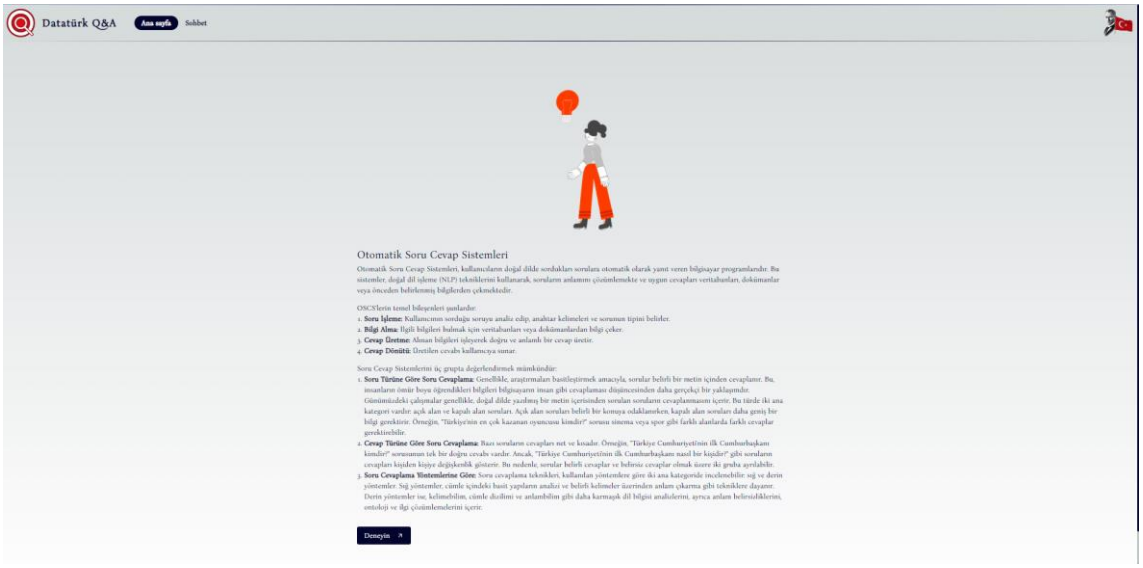
Projede frontend geliştirme süreci, modern JavaScript kütüphanesi olan React ve özellikle Next.js framework'ü kullanılarak gerçekleştirilmiştir. Next.js, React tabanlı uygulamalar için sunucu tarafı işlemler (Server-Side Rendering) ve statik site üretimi (Static Site Generation) gibi çeşitli optimizasyonları kolaylıkla uygulama imkanı sunar. Bu projede, Next.js'in App Router'ı kullanılarak uygulamanın rotalama (routing) sistemi etkin bir şekilde yönetilmiştir.

Datatürk arayüzü, kullanıcı dostu bir arayüz oluşturmak ve kullanıcı etkileşimlerini yönetmek için tasarlanmıştır. React, komponent bazlı yapısı sayesinde uygulamanın modüler ve bakımı kolay bölümlere ayrılmasını sağlar. Kullanıcı arayüzünde, soru sorma ve cevap görüntüleme işlevleri ana komponentler olarak planlanmıştır.

- **Soru Sorma Komponenti:** Kullanıcıların sorularını girebilecekleri bir metin kutusu içerir. Bu komponent, kullanıcıdan alınan girdiyi doğrudan Flask backend'ine POST isteği olarak gönderir.
- **Cevap Görüntüleme Komponenti:** Backend'den alınan cevapları gösterir. Kullanıcılar soru sorduklarında, cevap dinamik olarak bu bölümde güncellenir ve kullanıcılara sunulur.

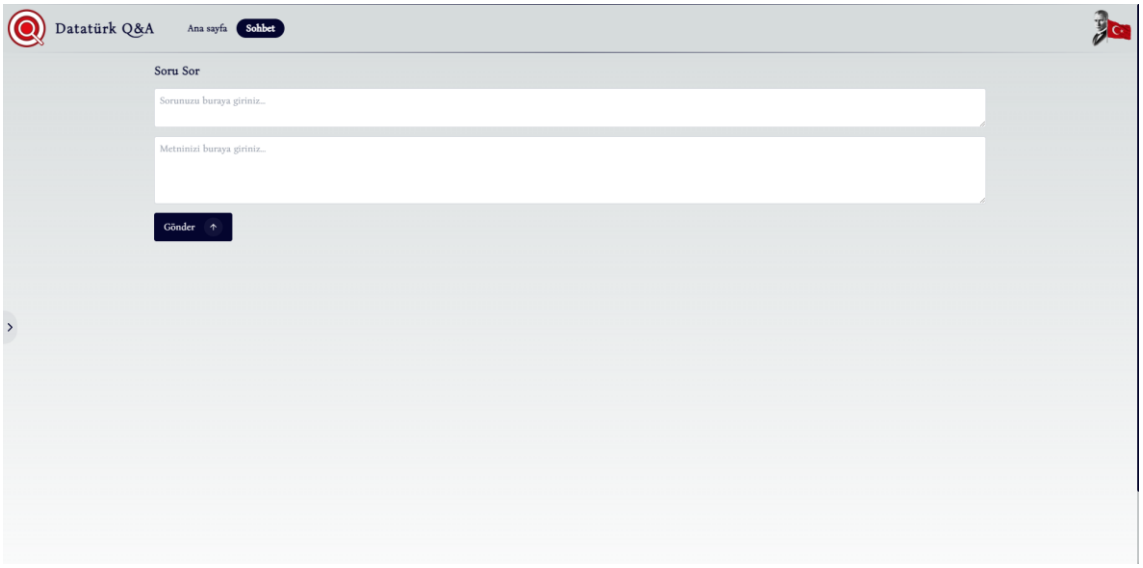
Next.js App Router, uygulamanın sayfa bazlı rotalarını yönetmek için kullanılır. Bu projede, App Router, kullanıcıların farklı bölümler arasında sorunsuz bir şekilde geçiş yapmalarını sağlamak ve URL bazlı navigasyon sunmak için etkin bir şekilde kullanılmıştır. Örneğin, ana sayfa, hakkında sayfası, yardım sayfası gibi farklı rotalar App Router ile tanımlanır. Bu yapı, SEO (Arama Motoru Optimizasyonu) dostu uygulamalar geliştirmeyi ve sayfa yüklemelerini hızlandırmayı mümkün kılar.

Aşağıdaki ekran görüntüsü, ana sayfamızı göstermektedir. Ana sayfa, kullanıcı siteye girdiği zaman karşılacağı ilk sayfadır. Bu sayfada, otomatik soru cevap sistemlerinin nasıl çalıştığı hakkında bilgi verilmiş ve sistemi kullanmaya teşvik etme amacıyla bir adet “Call to Action” butonu yerleştirilmiştir.



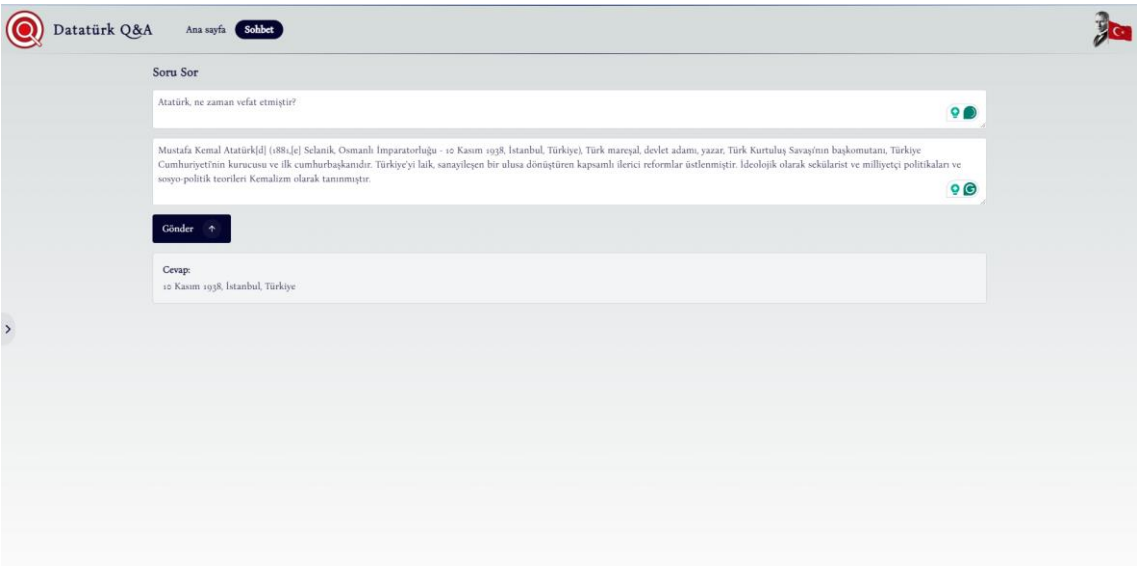
Şekil 4.5 Arayüz - Ana Sayfa

Kullanıcının sayfanın en altında bulunan Call to Action butonuna basması durumunda, kullanıcı "sohbet" sayfasına yönlendirilmektedir. Sohbet sayfasında ise kullanıcıdan almak üzere soru için ve bağlam (context) için birer girdi yerleştirilmiştir. Soru girdisine kullanıcı cevap almak istediği soruyu sorar, bağlam girdisine ise içinde sorduğu sorunun cevabının bulunduğu metni yapıştırır.



Şekil 4.6 Arayüz - Sohbet Ekranı

Soru girdisi ve metin girdisi doldurulduktan sonra, "Gönder" butonuna basılır. Bu butona basılması durumunda ön uç üzerinden arka uca bir sorgu atılmakta ve arka ucun model ile etkileşimi sonucu çıkan cevap, arayüzde gösterilmektedir.



Şekil 4.7 Arayüz - Cevap Ekranı

Aşağıda, React ile yazılan ön uç kodunun açıklaması bulunmaktadır:

- Home komponenti, uygulamanın ana sayfasını temsil eder ve kullanıcı arayüzünün temelini oluşturur. Bu komponent içerisinde, kullanıcı tarafından daha önce sorulmuş sorular yerel depolama alanından çekilir ve arayüze yüklenir. Kullanıcı, yeni sorular ekleyebilir ve mevcut soruları silebilir. İlgili kod parçası aşağıdaki gibidir:

```
const Home: React.FC = () => {
  const [questions, setQuestions] = useState<Question[]>([]);

  useEffect(() => {
    const savedQuestions = localStorage.getItem("questions");
    if (savedQuestions) {
      setQuestions(JSON.parse(savedQuestions));
    }
  }, []);
  ...
```

Şekil 4.8 Ana Sayfa Komponenti

Bu kod, kullanıcının geçmişteki soru-cevap kayıtlarını yönetmek için yerel depolama mekanizmasını kullanır. useEffect kancası (react-hook) ile uygulama yüklendiğinde mevcut sorular yüklenir.

- Sidebar komponenti, kullanıcının geçmiş sorularını listeler ve soruları tek tek veya topluca silme işlevlerini sağlar. Kullanıcı arayüzü dinamik olarak genişleyip daralabilir, bu da kullanıcıya ekran alanını daha verimli kullanma imkanı sunar.

```
const Sidebar: React.FC<SidebarProps> = ({ questions, deleteQuestion, deleteAllQuestions
}) => {
```

```
  ...
```

Şekil 4.9 Sidebar Komponenti

- ChatBox komponenti, kullanıcıların yeni sorularını sisteme girmelerine olanak tanır. Kullanıcılar soru ve bağlam bilgilerini girer ve bu bilgiler, API aracılığıyla işlenmek üzere sunucuya gönderilir.

```
const ChatBox = ({ addQuestion }: ChatBoxProps) => {
  ...
  const handleSubmit = async (e: any) => {
    ...
    const generatedAnswer = await getAnswer(question, context);
    addQuestion(question, generatedAnswer, context);
    setAnswer(generatedAnswer);
  };
```

Şekil 4.10 ChatBox Komponenti

Bu komponentte, kullanıcıdan alınan girdiler ile backend'e POST isteği yapılarak cevap alınır ve bu cevap arayüzde gösterilir.

- Backend API ile etkileşim axios kütüphanesi kullanılarak sağlanır. Kullanıcılar tarafından girilen soru ve bağlam bilgileri, API aracılığıyla işlenir ve cevap olarak geri döndürülür.

```
const getAnswer = async (question: string, context: string) => {
  const response = await axios.post("/api/predict", {
    question,
    context,
  });
  return response.data.answer;
```

Şekil 4.11 getAnswer Metodu

4.3. Sistem Entegrasyonu

Projede, frontend ve backend arasındaki entegrasyon, sistemin genel performansı ve kullanıcı deneyimi üzerinde büyük bir etkiye sahiptir. Bu entegrasyon, kullanıcıların sistemle etkileşimini sağlayan sorunsuz bir veri akışı ve işlevsellik sağlamak için kritik öneme sahiptir.

Projedeki entegrasyon, esas olarak React frontend ve Flask backend arasında RESTful API üzerinden gerçekleşir. Frontend, kullanıcı arayüzünde toplanan verileri JSON formatında backend'e POST istekleri olarak gönderir. Backend bu istekleri işler ve uygun cevapları hesapladıktan sonra yine JSON formatında frontend'e geri döndürür. Bu süreç, kullanıcıların sistemle etkileşimde bulunurken karşılaştığı cevap sürelerinin ve sistem yanıtlarının doğruluğunun optimizasyonunu içerir.

```
// Frontend'den POST isteği
axios.post("/api/predict", { question, context });
```

```
// Backend'de Flask route işlemcisi
@app.route('/predict', methods=['POST'])
def predict():
    ...
    return jsonify(answer=answer)
```

Şekil 4.12 Backend ve Frontend İletişimi

Bu kod parçaları, frontend'in kullanıcıdan aldığı soru ve bağlam bilgilerini backend'e gönderdiğini ve backend'in bu bilgileri işleyip bir cevap ürettiğini gösterir. Flask backend, soru ve bağlamı BERT modeline besler ve elde edilen cevabı frontend'e geri yollar.

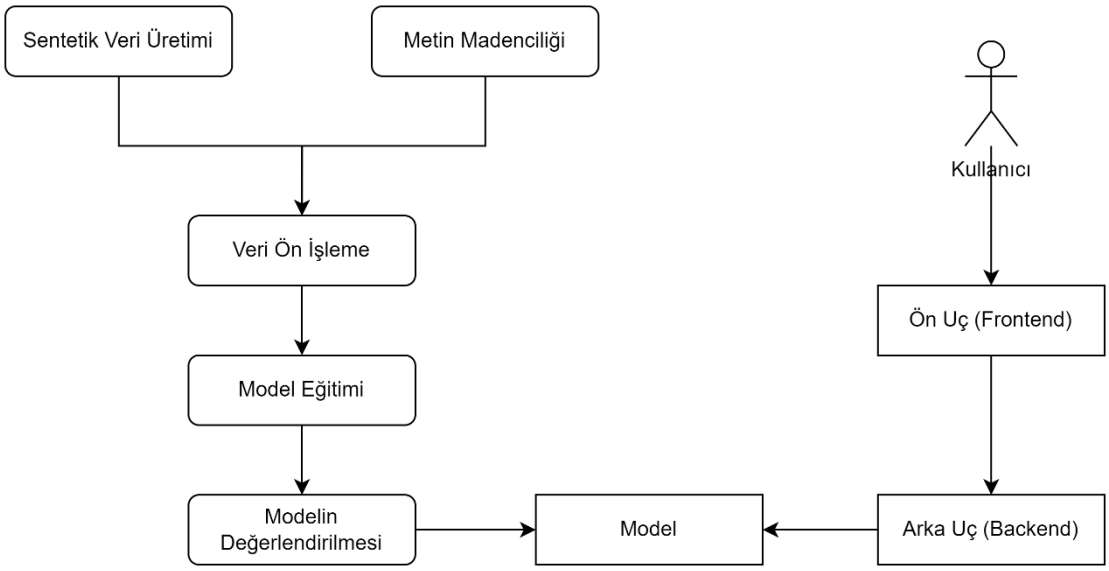
Sistem entegrasyonunda güvenlik de büyük bir rol oynar. CORS (Cross-Origin Resource Sharing) politikaları, farklı kaynaklardan gelen isteklerin güvenli bir şekilde kabul edilmesini sağlar. Bu, özellikle bir web uygulamasının birden fazla kaynaktan erişilebilir olduğu durumlarda önemlidir.

```
// Flask backend'de CORS ayarları
response.headers.add('Access-Control-Allow-Origin', '*');
```

Şekil 4.13 CORS Ayarı

Bu ayar, tüm kaynaklardan gelen isteklerin kabul edilmesine izin verir, bu da geliştirme ve test aşamalarında frontend'in farklı konumlardan backend'e sorunsuz bir şekilde erişim sağlamasına olanak tanır.

Sonuç olarak siteye giren bir kullanıcı, ön uç ile etkileşime geçerek cevabını almak istediği soruyu ve cevabın içinde bulunduğu bağlamı arka uca gönderir. Arka uç, kullanıcıdan gelen soru ve bağlamı tokenize ederek modele verir ve modelden bir cevap bekler. Modelden cevap dönünce bunu arayüze iletir ve kullanıcı, sorduğu sorunun cevabını almış olur.



Şekil 4.14 İşlem Şeması

5. BULGULAR ve TARTIŞMA

Bu bölüm, otomatik soru cevaplama sistemimizin geliştirilmesi ve test edilmesi sırasında elde edilen bulguları ve bu bulguların daha geniş bağlamda ne anlama geldiğini tartışır. Sistemin performansı, kullanılan tekniklerin etkinliği ve karşılaşılan zorluklar üzerine yoğunlaşır.

Projede kullanılan BERT ve ELECTRA gibi derin öğrenme modelleri, veri setleri üzerinde çeşitli denemelerle değerlendirilmiştir. Model performansının ölçülmesi için kullanılan metrikler arasında doğruluk, F1 skoru ve yanıt süreleri bulunur. Özellikle, sistem Türkçe dilindeki doğal dil işleme görevlerinde yüksek bir doğruluk oranı göstermiştir.

- BERT, kelimeleri teker teker değerlendirmek yerine, önündeki ve arkasındaki kelimeler ile veya birbirine benzeyen ve anlamdaş kelimeler ile beraber bir çıkarım oluşturmaktadır. Bunun sonucu olarak ise karmaşık sorular çok daha iyi anlaşılabilir ve daha yüksek çözümlenme olanağı sağlanmaktadır.
- ELECTRA, BERT'in avantajlarını sunarken, aynı hesaplama bütçesiyle mevcut tekniklerden daha iyi performans sergileyen yeni bir ön eğitim yöntemidir. Çalışmada bu algoritma kullanılarak ve parametreleri değiştirilerek modelin eğitilmesi ve en iyi sonuç elde edilmesi amaçlanmıştır.

İlk başta BERT modeli üzerinde belli parametreler değiştirilerek farklı sonuçlar karşılaştırılmıştır. Değiştirilen parametreler epoch, max_seq_length ve learning_rate parametreleridir.

Tablo 5.1 BERT Farklı Hiper Parametrelerin Kıyaslanması

Model	Epoch	max_seq_length	learning_rate	F1 Score	Exact Match
BERT #1	3	256	2e-5	76,65	60,12
BERT #2	3	256	4e-5	74,67	60,81
BERT #3	3	512	2e-5	77,1	61,24
BERT #4	3	512	4e-5	75,91	61,21
BERT #5	5	256	2e-5	77,31	61,45
BERT #6	5	256	4e-5	77,22	61,41
BERT #7	5	512	2e-5	77,78	61,8
BERT #8	5	512	4e-5	76,95	62,09

Tablo 5.1'e bakarak aşağıdaki çıkarımlar yapılabilir:

- **Eğitim Epoch Sayısı:** Epoch sayısının artması genel olarak modelin performansını iyileştirmiştir. Örneğin, 3 epoch ile eğitilmiş modellerle kıyaslandığında, 5 epoch ile eğitilmiş modeller (BERT #5, #6, #7, #8) daha yüksek F1 ve Exact Match skorları elde etmişlerdir. Bu, modelin daha fazla eğitim iterasyonundan geçmesinin, öğrenme sürecini ve sonuçta elde edilen performansı olumlu yönde etkilediğini göstermektedir.
- **Maksimum Sekans Uzunluğu:** Maksimum sekans uzunluğunun artırılması (256'dan 512'ye) genellikle modelin performansını artırmıştır. Özellikle, 512 maksimum sekans uzunluğunda eğitilmiş modeller, 256 uzunluğundakilere göre genellikle daha iyi performans göstermiştir. Bu, daha uzun sekansların, modelin bağlamı daha iyi anlamasına ve dolayısıyla soruları daha doğru cevaplamasına olanak tanıdığını düşündürmektedir.

- **Öğrenme Oranı:** Farklı öğrenme oranları (learning rate), model performansı üzerinde önemli bir etkiye sahiptir. Daha düşük bir öğrenme oranı (2e-5), genellikle daha yüksek bir F1 skoru ve Exact Match oranı ile sonuçlanmıştır. Bu, daha düşük öğrenme oranlarının, modelin eğitim sürecinde daha stabil bir şekilde ilerlemesine ve aşırı uydurma (overfitting) riskini azaltmasına yardımcı olabileceğini gösterir.
- **En İyi Model:** BERT #7 modeli, 5 epoch, 512 maksimum sekans uzunluğu ve 2e-5 öğrenme oranı ile eğitildiğinde en yüksek F1 Skorunu (77.78) ve BERT #8 ise en yüksek Exact Match oranını (62.09) elde etmiştir. Bu, yüksek epoch sayısı, uzun maksimum sekans uzunluğu ve düşük öğrenme oranının birleşiminin, en iyi sonuçları verdiğini göstermektedir.

Tablo 5.2 ELECTRA Farklı Hiper Parametrelerin Kıyaslanması

Model	Epoch	max_seq_length	F1 Score	Exact Match
ELECTRA #1	3	256	58,1	44,12
ELECTRA #2	3	512	59,67	44,81
ELECTRA #3	5	256	65,1	50,24
ELECTRA #4	5	512	62,26	49,8

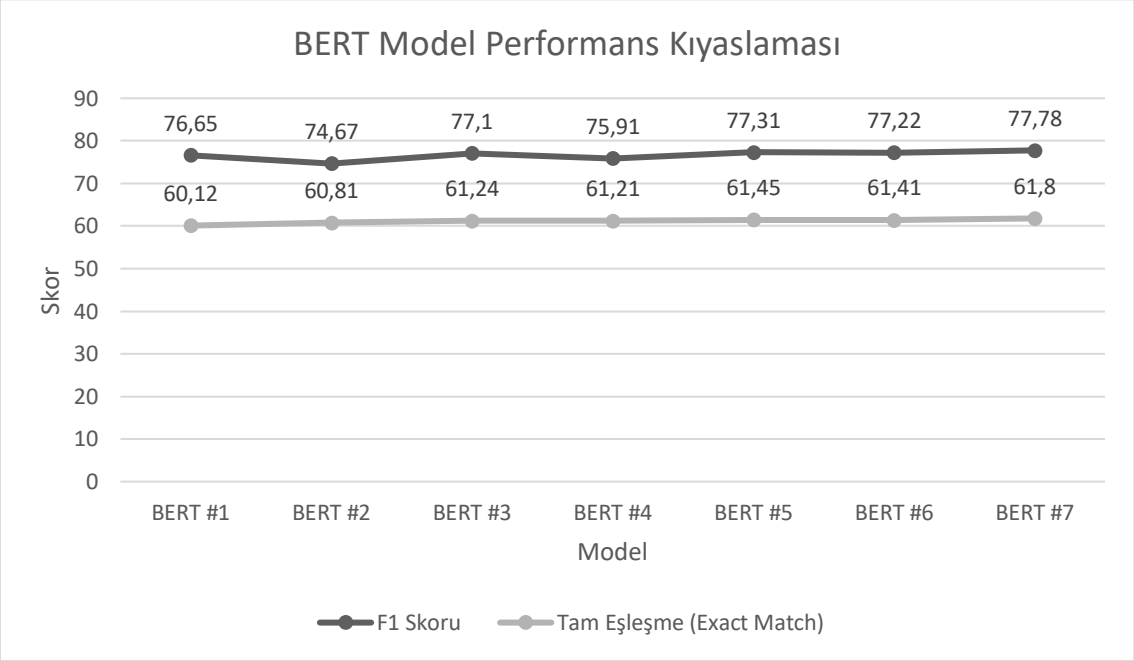
Tablo 5.2'ye bakarak aşağıdaki çıkarımlar yapılabilir:

- **Eğitim Epoch Sayısı:** ELECTRA modellerinde, epoch sayısının artırılması (3'ten 5'e) modelin performansında belirgin bir iyileşme sağlamıştır. Hem F1 skoru hem de Exact Match değerleri, 5 epoch ile eğitilen modellerde daha yüksek çıkmıştır (ELECTRA #3 ve #4). Bu durum, modelin daha fazla eğitim almasının, öğrenme kapasitesini ve genel performansını artırdığını göstermektedir.
- **Maksimum Sekans Uzunluğu:** Maksimum sekans uzunluğu artırıldığında (256'dan 512'ye), 3 epoch ile eğitilen modellerde (ELECTRA #1 ve #2) F1 ve Exact Match skorlarında küçük bir artış gözlemlenmiştir. Ancak, 5 epoch ile eğitilen modeller arasında (ELECTRA #3 ve #4), maksimum sekans uzunluğunun artışı F1 skorunu hafifçe artırırken, Exact Match değerinde küçük bir düşüşe neden olmuştur. Bu, daha uzun sekansların bazı durumlarda

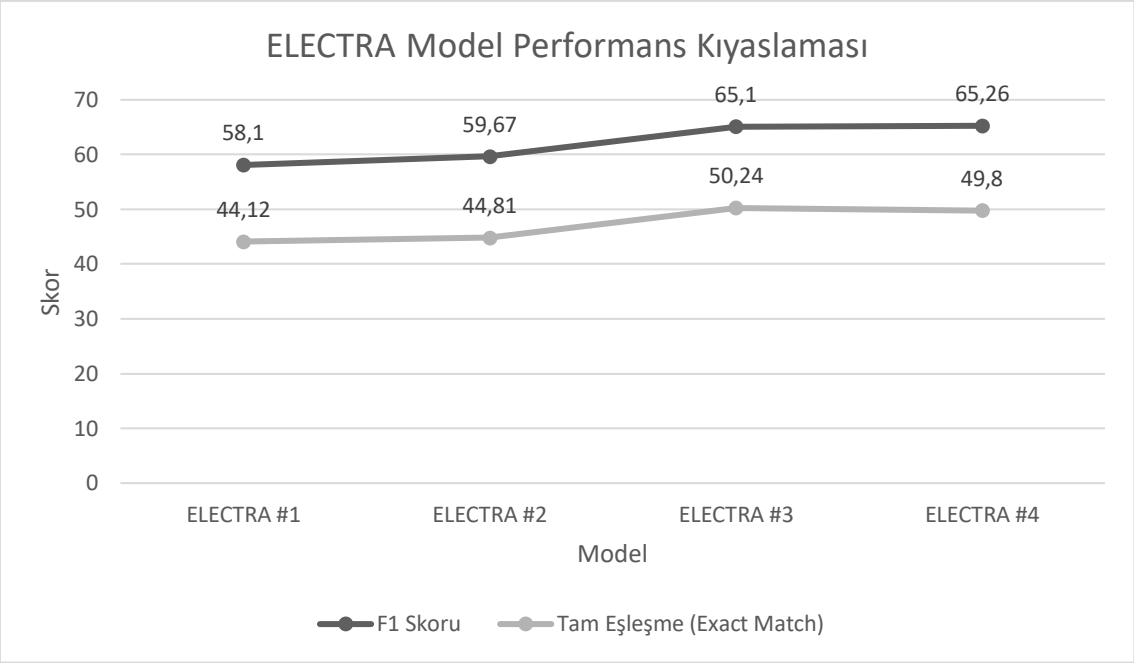
modelin performansını artırabileceğini, ancak dengeli bir şekilde optimize edilmesi gerektiğini gösterir.

- **Genel Performans:** Genel olarak, ELECTRA #4 (5 epoch ve 512 maksimum sekans uzunluğu) en yüksek F1 skorunu elde etmiştir. Ancak, Tam Eşleşme oranı açısından en iyi performans gösteren model ELECTRA #3'tür. Bu, farklı performans metrikleri arasında bazen trade-offlar olabileceğini göstermektedir.

Elde edilen sonuçlara göre BERT ve ELECTRA modellerinin çizgi grafiği aşağıda sırasıyla **Şekil 5.1** ve **Şekil 5.2**'de çizilmiştir:



Şekil 5.1 BERT Modeli Çizgi Grafiği



Şekil 5.2 ELECTRA Modeli Çizgi Grafiği

6. SONUÇLAR

Bu tez çalışması kapsamında, derin öğrenme tabanlı bir otomatik soru cevaplama sistemi geliştirilmiştir. Bu sistem, BERT ve ELECTRA gibi modern doğal dil işleme modellerini kullanarak, Türkçe dilindeki metinler üzerinde etkili bir şekilde soru cevaplama yeteneğini göstermiştir. Çalışma boyunca elde edilen sonuçlar, bu modellerin dil işleme konusunda nasıl optimize edilebileceğini ve gerçek dünya uygulamalarında nasıl performans gösterebileceğini ortaya koymuştur.

Ana Bulgular

- **Model Performansı:** BERT ve ELECTRA modelleri, çeşitli hiper parametre ayarları altında test edilmiştir. Sonuçlar, eğitim sürecinin uzunluğu, maksimum sekans uzunluğu ve öğrenme oranının model performansı üzerinde belirgin etkileri olduğunu göstermiştir. Özellikle, daha uzun eğitim süreleri ve uygun şekilde ayarlanmış sekans uzunlukları modellerin doğruluk ve tam eşleşme skorlarını iyileştirmiştir.

- **Teknik Zorluklar ve Çözümler:** Projede karşılaşılan teknik zorluklar arasında modelin eğitim süresi ve aşırı uydurma (overfitting) sorunları yer almaktadır. Bu

sorunlar, veri artırma teknikleri ve düzenlileştirme yöntemleri kullanılarak başarıyla ele alınmıştır.

- **Kullanıcı Deneyimi:** Sistem, kullanıcı dostu bir arayüz üzerinden erişilebilir kılınmıştır.

Öneriler ve Gelecek Çalışmalar

- **Model Çeşitliliği:** Gelecekte, daha fazla dil modeli ve dil çifti ekleyerek sistemin çok dilli ve kültürlerarası uyarlanabilirliğini artırmak önerilmektedir.
- **Algoritma Optimizasyonu:** Yapay sinir ağları için alternatif optimizasyon algoritmalarının araştırılması, modelin daha hızlı ve verimli eğitilmesine olanak tanıyabilir.
- **Kullanıcı Geri Bildirimleri:** Sistem, sürekli kullanıcı geri bildirimleri ile güncellenmeli ve iyileştirilmelidir. Bu, sistemin daha geniş bir kullanıcı tabanı tarafından kabul görüp kullanılmasını sağlayacaktır.

Sonuç

Geliştirilen otomatik soru cevaplama sistemi, Türkçe metinler üzerinde yüksek performans göstermiş ve kullanıcılar tarafından olumlu değerlendirmeler almıştır. Bu tez, yapay zeka ve doğal dil işleme teknolojilerinin eğitim, araştırma ve gerçek zamanlı uygulamalar için potansiyelini vurgulamaktadır. Yapılan çalışma, bu alanda yapılacak daha fazla araştırma ve geliştirme için sağlam bir temel oluşturmuştur.

KAYNAKÇA

[1] A. M. Thuy Vu, «AVA: an Automatic eValuation Approach to Question Answering Systems,» 2020.

[2] S. İlhan, N. Duru, Ş. Karagöz ve M. Sağır, «Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü Kocaeli Üniversitesi,» Kocaeli, 2008.

[3] N. Akkaya ve L. Şengül, «Sohbet Robotları (Chatbots) ve Yabancı Dil Eğitimi,» 2023.

[4] K. . Oflazer, «Türkçe ve Doğal Dil İşleme,» , 2016. [Çevrimiçi]. Available: https://dergipark.org.tr/tr/download/article-file/207207. [Erişildi: 3 7 2024].

[5] G. Soltana, M. Sabetzadeh ve L. C. Briand, «Synthetic data generation for statistical testing,» IEEE, 2017.

[6] Amazon Web Services (AWS), «What is Synthetic Data?,» Amazon Web Services (AWS), [Çevrimiçi]. Available: https://aws.amazon.com/tr/what-is/synthetic-data/. [Erişildi: 26 Haziran 2024].

[7] Wikipedia, «Transformer (deep learning architecture),» Wikipedia, Şubat 2023. [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture). [Erişildi: 26 Haziran 2024].

[8] R. Merritt, «What Is a Transformer Model?,» NVIDIA, 25 Mart 2022. [Çevrimiçi]. Available: https://blogs.nvidia.com/blog/what-is-a-transformer-model/. [Erişildi: 26 Haziran 2024].

[9] Wikipedia, «BERT (language model),» Wikipedia, 21 Haziran 2024. [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/BERT_(language_model). [Erişildi: 26 Haziran 2024].

[10] S. Kapadia, Towards Data Science, 17 Temmuz 2023. [Çevrimiçi]. Available: https://towardsdatascience.com/practical-introduction-to-transformer-models-bert-4715ed0deede. [Erişildi: 26 Haziran 2024].

[11] K. Clark, M.-T. Luong, Q. V. Le ve C. D. Manning, «ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators,» 2020.

[12] Wikipedia, «Derin öğrenme,» Wikipedia, 11 Mayıs 2024. [Çevrimiçi]. Available: https://tr.wikipedia.org/wiki/Derin_öğrenme. [Erişildi: 26 Haziran 2024].

[13] K. Güzel, «Geri Yayılımlı Çok Katmanlı Yapay Sinir Ağları,» Medium, 20 Ağustos 2018. [Çevrimiçi]. Available: https://kadirguzel.medium.com/geri-yayılımlı-çok-katmanlı-yapay-sinir-ağları-2-6a47b4f3a6c. [Erişildi: 26 Haziran 2024].

[14] A. Kızrak, «Derin Öğrenme İçin Aktivasyon Fonksiyonlarının Karşılaştırılması,» Medium, 4 Şubat 2019. [Çevrimiçi]. Available: https://ayyucekizrak.medium.com/derin-öğrenme-için-aktivasyon-fonksiyonlarının-karşılaştırılması-cee17fd1d9cd. [Erişildi: 27 Haziran 2024].

[15] A. Atcılı, «Yapay Sinir Aglarinda Kullanilan Optimizasyon Algoritmalari,» Medium, 6 Kasım 2020. [Çevrimiçi]. Available: https://medium.com/machine-learning-türkiye/yapay-sinir-ağlarında-kullanılan-optimizasyon-algoritmaları-3e87cd738cb5. [Erişildi: 27 Haziran 2024].

[16] OpenZeka, «Sentetik Veri Nedir?,» OpenZeka, [Çevrimiçi]. Available: https://blog.openzeka.com/ai/sentetik-veri-nedir/. [Erişildi: 25 Haziran 2024].

[17] «Data Preprocessing Techniques for Data Mining,» , . [Çevrimiçi]. Available: http://iasri.res.in/ebook/win_school_aa/notes/Data_Preprocessing.pdf. [Erişildi: 3 7 2024].

[18] L. . Deng ve D. . Yu, «Deep Learning: Methods and Applications,» *Foundations and Trends in Signal Processing,* cilt 7, no. 3–4, p. , 2014.

[19] M. A. Nielsen, Neural Networks and Deep Learning, dü., cilt , , : Determination Press, 2015, p. .

[20] İ. F. Pilavcılar, «Metin Madenciliği ile Metin Sınıflandırma,» Yıldız Teknik Üniversitesi. FBE, İstanbul, 2007.

[21] S. J. Cooper, «Donald O. Hebb's synapse and learning rule: a history and commentary,» *Neuroscience & Biobehavioral Reviews,* cilt 28, no. 8, pp. 851-874, 2005.

[22] A. . Uršič, «Evolution of neural networks,» , 2012. [Çevrimiçi]. Available: http://eprints.fri.uni-lj.si/1736. [Erişildi: 3 7 2024].

[23] H. . Singh, «ARTIFICAL NEURAL NETWORKS: The New Era Of I.T,» *International Journal of Research in Education Methodology,* cilt 2, no. 1, pp. 86-88, 2012.

[24] H. K. Kwan, «Simple sigmoid-like activation function suitable for digital hardware implementation,» *Electronics Letters,* cilt 28, no. 15, pp. 1379-1380, 1992.

[25] Simple Transformers, «Documentation — Simple Transformers 0.61.1 documentation,» Simple Transformers, [Çevrimiçi]. Available: https://simpletransformers.ai/docs/qa-data-formats/. [Erişildi: 25 Haziran 2024].

[26] B. v. Aken, B. . Winter, A. . Löser ve F. A. Gers, «How Does BERT Answer Questions?: A Layer-Wise Analysis of Transformer Representations.,» *arXiv: Computation and Language,* cilt , no. , pp. 1823-1832, 2019.

[27] K. . Clark, M.-T. . Luong, Q. V. Le ve C. D. Manning, «ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators,» , 2020. [Çevrimiçi]. Available: https://openreview.net/forum?id=r1xmh1btvb. [Erişildi: 3 7 2024].

[28] Q. . Meng, S. . Zheng, H. . Zhang, W. . Chen, Q. . Ye, Z.-M. . Ma, N. . Yu ve T.-Y. . Liu, «G-SGD: Optimizing ReLU Neural Networks in its Positively Scale-Invariant Space,» , 2019. [Çevrimiçi]. Available: https://iclr.cc/conferences/2019/schedule. [Erişildi: 3 7 2024].

[29] Bayerische Staatsbibliothek, «Hugging Face,» [Çevrimiçi]. Available: https://huggingface.co/dbmdz/bert-base-turkish-cased. [Erişildi: 4 Temmuz 2024].

[30] S. Savaş, N. Topaloğlu ve M. Yılmaz, «Veri Madenciliği ve Türkiye'deki Uygulama Örnekleri,» 2012.

[31] Yrd.Doç.Dr. Kadriye ERGÜN, «VERİ MADENCİLİĞİ (Data Mining)».

[32] M. F. Amasyalı ve B. Diri, «Bir Soru Cevaplama Sistemi: BayBilmiş,» 2005.

[33] G. Yiğit ve M. F. Amasyalı, «Soru Cevaplama Sistemleri Üzerine Detaylı Bir Çalışma: Veri Kümeleri, Yöntemler ve Açık Araştırma Alanları,» 2021.

[34] M. F. Amasyalı ve A. Beken, «Türkçe Kelimelerin Anlamsal Benzerliklerinin Ölçülmesi ve Metin Sınıflandırmada Kullanılması,» 2009.

[35] Ö. Özyurt ve H. Özyurt, «İNSAN-BİLGİSAYAR ETKİLEŞİMİ ARACI OLARAK METİN TABANLI DİYALOG SİSTEMİ İÇİN TÜRKÇENİN MORFOLOJİK VE ANLAMSAL ANALİZİ,» 2010.

[36] T. W. Smith ve S. A. Colby, «Teaching for Deep Learning.,» *The Clearing House,* cilt 80, no. 5, pp. 205-210, 2007.

[37] Y. . Sun, X. . Wang ve X. . Tang, «Deep Learning Face Representation from Predicting 10,000 Classes,» , 2014. [Çevrimiçi]. Available: http://mmlab.ie.cuhk.edu.hk/pdf/yisun_cvpr14.pdf. [Erişildi: 3 7 2024].

ÖZGEÇMİŞ

Adı Soyadı	Can Özfuttu
Doğum Yeri ve Tarihi	İstanbul, 6 Nisan 2002
Ana Dili	Türkçe
Yabancı Dili	İngilizce – Japonca – Almanca – İspanyolca
E-posta	canozfuttu@marun.edu.tr
Github	https://github.com/cozfuttu
Eğitim	Lisans – Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği 2020 – 2024
	Lisans – Marmara Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği 2021 – 2025 (pasif)
	Yan Dal – Marmara Üniversitesi Teknoloji Fakültesi Elektrik-Elektronik Mühendisliği 2022 – 2025
	Lisans – İstanbul Üniversitesi İktisadi ve İdari Bilimler Fakültesi İşletme Bölümü 2023 – 2027
Deneyim	Gesk: Embedded Systems Programming, Full Stack Developer
	Clade Club: ETHGlobal 2023 Hackathon winner, Full Stack Developer, Web Designer
	Dappad: Chief Technology Officer & Full Stack Developer

Adı Soyadı	Ahmed Burhaneddin Çoban
Doğum Yeri ve Tarihi	İstanbul, 25 Kasım 2001
Ana Dili	Türkçe
Yabancı Dil	İngilizce
E-posta	ahmedcoban20@marun.edu.tr
Github	https://github.com/ahmedcoban20
Eğitim	Lisans – Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği 2020 – 2024
	Lisans – Marmara Üniversitesi Teknoloji Fakültesi Elektrik-Elektronik Mühendisliği 2021 – 2025
	Lisans – İstanbul Üniversitesi İktisadi ve İdari Bilimler Fakültesi İşletme Bölümü 2023 – 2027
Deneyim	Florawise: Embedded System Programming, Web Front-end, Mobile App Tester (IOS)
	Web Solution Home: Web Design & Programming Intern (Front-end)
	Anel Group: IT Intern (Network & Security, System & Server Administration)