T.C.
KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

TWITTER BÜYÜK VERİ ANALİZİ

Hamza PEKDOĞAN

ŞUBAT 2021

Bilgisayar Mühendisliği Ana Bilim Dalında Hamza PEKDOĞAN tarafından hazırlanan "TWITTER BÜYÜK VERİ ANALİZİ" adlı Yüksek Lisans Tezinin Anabilim Dalı standartlarına uygun olduğunu onaylarım.

Doç. Dr. Atilla ERGÜZEN

Anabilim Dalı Başkanı

Bu tezi okuduğumu ve tezin **Yüksek Lisans Tezi** olarak bütün gereklilikleri yerine getirdiğini onaylarım.

Doç. Dr. Atilla ERGÜZEN

Danışman

Jüri Üyeleri

Başkan : Dr. Öğr. Üyesi Abdül Kadir GÖRÜR _____

Üye (Danışman): Doç. Dr. Atilla ERGÜZEN _____

Üye : Dr. Öğr. Üyesi Erdal ERDAL _____

...../...../.....

Bu tez ile Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onaylamıştır.

Prof. Dr. Recep ÇALIN

Fen Bilimleri Enstitüsü Müdürü

ÖZET

TWITTER BÜYÜK VERİ ANALİZİ

PEKDOĞAN, Hamza
Kırıkkale Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi
Danışman: Doç. Dr. Atilla ERGÜZEN
Şubat 2021, 61 sayfa

Günümüzde internet kullanımı artmaktadır. İnternet kullanımı yaşam tarzlarını değiştirmektedir. Alışveriş, tanışma, sosyal çevre gibi birçok faaliyet internette gerçekleşmektedir. Bunun sonucunda büyük miktarda veri oluşmaktadır. Veri miktarı arttıkça bu veriyi depolama amaçlı çalışmalar gerçekleştirilmiştir. Depolama sonrasında verilerin analizi konusunda çalışmalar yapılmaktadır. Kuruluşlar reklamlarını ve yatırımlarını fiziksel araştırmalar üzerine gerçekleştirmektedir. Büyük veri sonrası ise bu araştırmalar büyük veriler üzerinden yapılmaya başlanmıştır. Büyük veri analizleri birçok alanda kendine yer edinmektedir. Sağlık, ticari, siyasi gibi birçok alanı bulunmaktadır. Veriler sadece alanlara özgü bilgi vermek dışında kişisel olarak insanları analiz etmektedir. Bunun sonucunda analiz edilen insanlar üzerinde istenilen derecede etki oluşturulmaktadır. Bu yüzden en çok kullanılan alanlardan biri siyasi alandır. Siyasi alanda en çok kullanılan platform ise Twitter'dır. Bu yüzden amacım twitter üzerinden büyük veri elde etmek,bu veriyi depolamak ve analizini gerçekleştirmektir

Anahtar kelimeler: Büyük Veri, Twitter, Büyük Veri Analizi

ABSTRACT

TWITTER BIG DATA ANALYSIS

PEKDOĞAN, Hamza
Kırıkkale University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering, Master Thesis
Supervisor Doç. Dr. Atilla ERGÜZEN
February 2021, 61 Pages

Nowadays, internet usage is increasing. Using the Internet changes their lifestyles. Many activities such as shopping, dating and social environment take place on the internet. This results in large amounts of data. As the amount of data increased, studies were carried out to store this data. After storage, studies are conducted on data analysis. Organizations carry out their advertisements and investments on physical research. After the big data, these researches started to be done on big data. Big data analysis takes place in many areas. There are many fields such as health, commercial and political. The data analyzes people personally as well as providing information specific to the fields. As a result, the desired degree of impact on the analyzed people is created. Therefore, one of the most used fields is the political field. The most widely used political platform is Twitter. So my goal is to get big data via twitter, to store and analyze this data.

Keywords: Big Data, Twitter, Big Data Analytics

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

1.GİRİŞ

Günümüze kadar gelen zaman ile birlikte bilgi miktarı artarak devam etmiş ve bulunduğumuz son yıllarda artış miktarı büyük bir ivmelenme ile artış göstererek bugün içinde bulunduğumuz veri miktarı ve veri ortamı sağlanmıştır. Bugün ortaya çıkan bu veri ortamı da gitgide büyümeye devam etmektedir. Çeşitliliği çok fazla olan bu veri ortamının büyümesi, miktarı ile de toplumsal hayatta önemli bir yer tutmaktadır. Özellikle son dönemde teknolojinin gelişim sağlaması, internetin daha geniş anlamda kullanımı ve gelişen sosyal medya araçları ile beraber ortaya çıkan bilgi miktarı tahmin edilemeyecek kadar çoğalmıştır.

Veri miktarının artış göstermesi ve verinin çeşitlilik kavramı alanında da sağlanan farklılıklar sonrasında ivmelenerek artması sonucunda büyük veri kavramını oluşturmaya başlamıştır. Büyük veri olarak nitelendirilen kavram; medya paylaşımları, blog yazıları, sosyal medya fotoğrafları, sosyal medya videoları, web siteleri ve web uygulamaları gibi uygulamaların log dosyaları gibi farklı türlerden derlenen yapısal olan veya olmayan bütün verinin anlamlı hale getirilmesi ve bu verinin işlenebilir hale hazır olarak sunulmasına verilen isimdir.

Günümüzde yalnız bilgi miktarının artması değil aynı zamanda bilgiye erişim hızlanmış ve bilgi sayısında da artış yaşanmıştır. Verinin bütün halinde bir anlam kazanacak şekilde toplanması ve bir düzende tutulması ilk olarak astronomi ve genetik alanında olmuştur. Büyük veri kavramı da ilk olarak bu alanlarda kullanılmış, ancak daha sonra birçok alanda verilerin artması ile bu kavram daha sık kullanılmaya başlanmıştır. Elde edilen veriler incelendiğinde büyük veri neredeyse artık hayatımızın her alanında oluşmaya ve sonuçlar elde etmeye başlamıştır.

Son birkaç yılda bilişim alanında çok devasa atılımlar yapılmıştır ve bu atılımlar iletişimin hızla artmasına neden olmuştur. Bununla birlikte internetin

kullanımı da büyük kitlelere ulaşmış ve daha da yaygınlaşmaya başlamıştır. Bu nedenlerin toplamı sonucunda ortaya iletilen bilgi türlerinin çeşitliliği artmıştır. Ses formatında, görüntü formatında, grafik ve metin tabanı formatında yapısal her türlü veriler ve yapısal olmayan veriler yüksek bir miktarda ivmesel olarak artmıştır.

Büyük veri sadece artan veri miktarı ile oluşmaya başlamamıştır. 20.yy'ın başından itibaren kademe kademe artış göstermiştir. Büyük verinin kullanılabilecek hale gelmesi için işlenmesi ve anlamlandırılması gerekmektedir. Bunun için bileşenlerinin modelleme kısmında ayrıştırılması gerekir. Bileşen kısmına baktığımızda verilerin büyüklüğü, hızı ve çeşitlilik kavramlarının önemli olduğu ortaya çıkmaktadır.

Büyük veri, farklı veri kaynaklarından gelen verilerin birleştirilmesi ile oluşan büyük verinin sonrasında elde edilen verilerin analiz işlemlerinin uygulanması, analiz sonuçlarının işlenmesi ve bütün bu işlemlerin uygulanması sonrasında depolanması ile ortaya çıkan bir alandır. Büyük veri alanında yapılan analiz ve sonuçların işlenmesi uygulamalarının kendine özgü olmasına ihtiyaç duyulmaktadır. Büyük veri alanında elde edilen veri miktarı artış gösterdikçe geleneksel yani normal veriler için kullanılan veri analizi işlem yöntemleri ve depolama çözümleri yetmemeye başlamıştır. Özellikle büyük veri, ilişkisiz ve çoklu veri setlerinin birleştirilmesi, büyük miktarda bulunan verilerin işlenmesi, kısıtlı zaman içinde yani hızlı bir şekilde toplanması gibi farklı gereksinimleri ortaya çıkarmaktadır.

2000'li yılların başında üç bileşenli olarak tanımlanan büyük veri kavramı günümüzde kullanım çeşitliliğine göre beş bileşenli olarak da tanımlanabilmektedir. Mevcut verilerin büyük veri alanı içerisinde kabul edilmesi için aşağıda bulunan beş etmenden en az birine sahip olması gerekmektedir (Açıklanacak olan özelliklere 5V kuralı da denilmektedir).

Variety(Çeşitlilik) : Verilerin herhangi bir türde belirtilmesine gerek duyulmamaktadır. Resimler, metin dosyaları, log dosyaları, ses dosyaları gibi

birçok veri türü oluşturulmaktadır. Verilerin bir bütün halinde olması ve birbirlerine dönüştürülebilir olmalarına ihtiyaç duyulmaktadır.

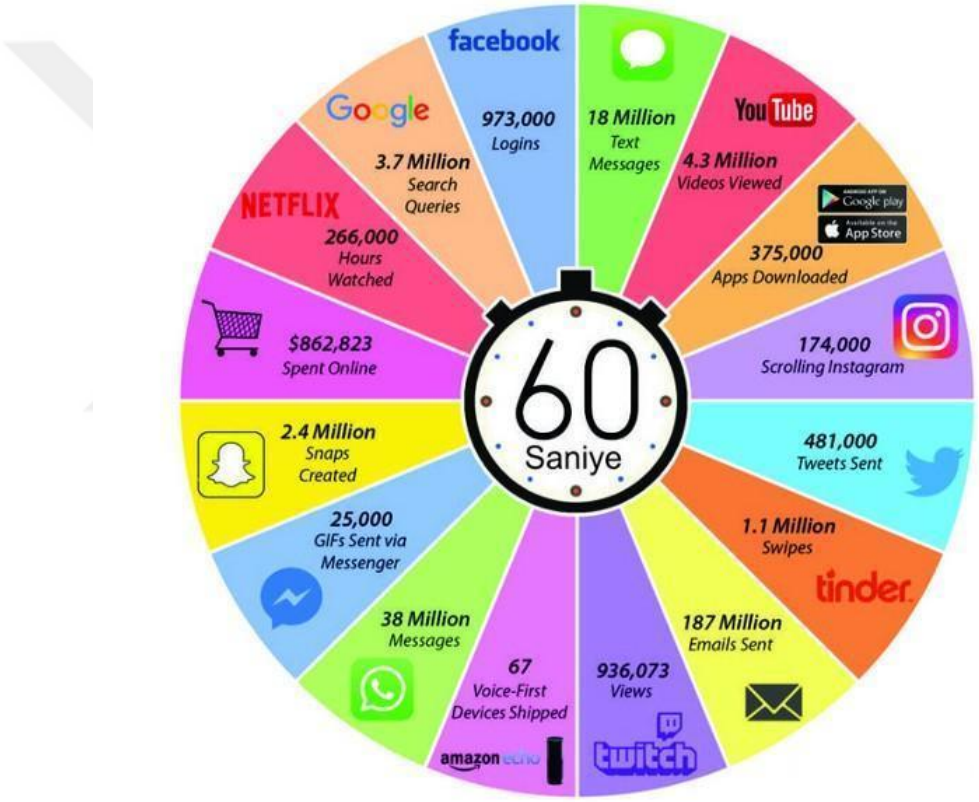
Velocity(Hız): Verilerin bir akış halinde hızlı olarak elde edilmesi gerekmektedir. Veriler elde edilirken bütünlüğü bozulmamasına dikkat edilmelidir. Sosyal medya hızı gibi sürekli hız ön planda tutulmalıdır.

Volume(Hacim): Bir verinin 'büyük veri' kategorisinde değerlendirilme şartı verinin yüksek boyutlarda olmasıdır. Verinin boyutu, değeri ile doğru orantılıdır. Veri birçok kaynaktan üretilebilir. IDC'nin istatistiklerine göre yaklaşık 44 farklı kaynaktan veri üretilmektedir. Yine IDC'nin istatistiklerine göre dijital evrenin yüzde 40 oranında bölümünü internet ortamı oluşturacaktır.

Verification(Doğruluk): Verilerin akışı esnasında gelen bilgilerin güvenilir olup olmadığı kontrol edilmelidir. Çünkü eksik veya bozulan verinin depolanması ve daha sonra analiz edilmesi vakit almaktadır ve istenmeyen sonuçlar doğurabilir. Dijital evrene bakıldığı zaman yüzde 70-80 oranında düzensiz veriler bulunmaktadır. Bu verilerin düzenlenmesi, toparlanması; makine öğrenmesi, veri madenciliği gibi alanların yönelimi ile olmaktadır.

Value(Değer): Değerler katmanı veri düzeninin önemli parçalarından biridir. Bugün elde edilen veriler daha önce elde edilen verilere göre daha fazla veri miktarının ve daha fazla veri kaynağının varlığı söz konusudur. Akıllı telefon kullanım oranının artması, tablet bilgisayarların yaygınlaşması, sağlık sektöründe teknolojik gelişmelerin yaşanması, web trafiği artması, sosyal ağlardaki etkileşimlerin daha kolay olması ve eczacılık alanında teknolojinin kullanımı gibi birçok olay ve gelişme büyük veriyi desteklemektedir. Bununla birlikte web ortamının artışı, web sayfaları üzerinde farklı işlemlerin yapılabilmesi gibi konular veri içeriğinin genişlemesine ve büyük veri alanına katkı sağlamasına sebep olmaktadır.

Büyük veri çeşitliliği artışının en önemli nedenlerinden birisi çok sayıda veri kaynağının olmasıdır. Bu kaynaklardan söz etmek gerekirse bazıları yeni veri kaynağı olmaktaydı. Bazı veri kaynakları ise elde bulunan verinin ayrışması, diğer bir deyişle elde bulunan kaynakların sayısal ortama aktarılması sonucu oluşmaktadır. Çoğu endüstriyel alanda yapılan teknolojik gelişmeler ile birlikte yeni veri üretimi oluşmaktadır. Bazı endüstriyel alanda ise mevcut verinin sayısallaştırılması gibi uygulamalar artmaktadır. Bu sonuçların toplamında yeni büyük veri kaynakları oluştuğu görülmektedir [1]



Şekil 1.1. Bir dakikada internet veri akışı [2]

Şekil 1.1'de internet âleminde bir dakikada oluşan veri sayısı gösterilmektedir. Bu gösterimden de anlaşılacağı gibi veri trafiği oldukça sıkışık gözükmektedir. Burada depolanacak olan verinin her saniyede devasa şekilde artması büyük veri teknolojilerinin önlem almasını gerektirmektedir.

Sosyal medya ortamlarının yanı sıra büyük veri uygulama alanları çeşitliliği ile ön plana çıkmaktadır. Bankacılık alanında, sağlık sektöründe, iletişim, medya, eğlence sektöründe, eğitim sektöründe, üretim sektöründe, devlet hizmet sektöründe, ulaşım sektöründe, sigortacılık sektöründe, enerji sektöründe ve daha birçok alanda büyük veri uygulamaları kullanılmaktadır.

Büyük veride fazla veriye sahip olmaktan ziyade elde bulunan verilerle neler yapılabileceği önem arz etmektedir. Yeni yeni birçok şirket, bu amaç doğrultusunda ellerindeki veriyi çoğaltmaya çalışmazlar. Daha çok veriyi analiz etmek doğrultusunda kullanmayı amaçlarlar. Birden fazla kaynaktan toplanan verinin daha başarılı şekilde analizi sonucunda iş gücü tasarrufu etmek, üretim maliyetini azaltmak veya ürün geliştirme süreçlerini iyileştirmek gibi birçok alanda kazanç sağlamaya çabalamaktadırlar. Gerçek ve büyük veri sayesinde elde edilen analizler sonucunda şirketler daha yararlı ve güçlü kararlar alabilmektedirler. Büyük veri sayesinde ve verimli bir iş analitiği uygulaması ile işlenen veriler, hemen hemen her sektörde fayda sağlayacak hale gelmektedir. Ayrıca rekabet alanında avantaja dönüşebilmekte ve faydalar elde etmek mümkün olmaktadır.

Daha önceden yapılan Twitter büyük veri analizi çalışmalarından “rk kargo hizmet sağlayıcılar ile ilgili Twitter'da üretilen içeriğin analizi ” çalışması incelenmektedir. Çalışmada kargo firmaları hakkında bir çalışma yapılmıştır. ARTUN TOMATIR 2018 “Türk kargo hizmet sağlayıcılar ile ilgili Twitter'da üretilen içeriğin analizi” adlı çalışmasını şöyle açıklamaktadır;

“Twitter' ın işlemek ve saklamak durumunda olduğu veri miktarı pek çok araştırmacı tarafından büyük veri olarak nitelendirilmektedir. Büyük veri yeni bir kavram olmasına rağmen günümüzde pek çok disiplinde tartışılmaktadır. Artan veri büyüklüğü ve karmaşıklığı yeni teknoloji ve yaklaşımları gerektirmekte ve yeni tekniklerle işletmelere pratik faydalar sağlayabilmektedir. Büyük veri üzerine yapılan araştırmaların temel olarak iki problem üzerine odaklandığı görülmektedir. İlk problem büyük veriyi işleme ve analiz etmek için metot, teknoloji ve algoritmaların geliştirilmesidir. İkinci

problem büyük veri analizinin farklı disiplinlere ve problemlere uyarlanması ve kullanılmasıdır. Bu araştırmada Twitter tarafından sağlanan kullanıcı tarafından üretilen içerik değerlendirilmiş ve büyük veri analizinin lojistik yönetiminde kullanımına ilişkin öneriler geliştirilmeye çalışılmıştır. Kimi araştırmacılar büyük veriyi değişkenlerin kapsamı ve verinin güvenilirliği konusunda eleştirmektedir. İfadelerin kapsamı bu araştırmanın ilk problemidir. Bu nedenle daha önce yapılmış araştırmalardan birinin değişkenlerine ilişkin ifadeler tweetlerin içinde aranmıştır. İkinci problem kullanıcı tarafından üretilen içeriğin lojistik yönetimi için nasıl kullanılabileceğidir. Farklı Twitter kullanıcılarının Twitter'ı kullanma amaçları ve çıkarılabilecek kullanışlı bilgiler keşfedilmeye çalışılmıştır. Araştırmanın güçlü yönü büyük veri analizi tekniklerinin lojistik yönetiminde kullanımına ilişkin Türkiye'de yapılan ilk yüksek lisans tezi olmasıdır. Aynı zamanda Türkiye'de kargo firmaları üzerine yapılan birkaç çalışma bulunmaktadır. Araştırmanın zayıf yönü ise sınırlı sayıda tweet üzerinde yapılmış olması ve sadece kullanıcı tarafından oluşturulan içeriğin değerlendirilmiş olmasıdır. Pek çok tanıma göre ilişkisel veri tabanlarında saklanabilen ve elle ya da konvansiyonel metotlarla değerlendirilebilecek miktarda veri, örneklem büyük veri kaynağından alınmış olsa bile, büyük veri olarak tanımlanmamaktadır. Bu nedenle bu çalışma büyük veri analizinin lojistik yönetiminde kullanımına ilişkin bir ön çalışmadır. Daha önce yapılan araştırmada 31 değişken çalışılmış ve 29' una ilişkin ifadeye örneklemde rastlanmıştır. Müşterilerin kullanım amacına ilişkin en yüksek frekans şikâyet, konu ise teslimattadır. Firmaların kullanım amacına ilişkin en yüksek frekans yardımdır ve ilk yaptıkları takip numarası, iletişim bilgilerini istemektir. Araştırmacılara göre büyük veri analizinin lojistikte kullanımının üç boyutu bulunmaktadır. Bunlar operasyonel verimlilik, müşteri deneyimi ve yeni iş modelleri geliştirilmesidir. Kimi araştırmacılara göre müşteri hizmetleri lojistik firmalarının fonksiyonlarından biri olarak görülmektedir. Bu çalışma sonunda büyük veri tekniklerinin sosyal medyayı şikâyetleri bulma amacıyla tarama, iletişim bilgileri ve takip numaralarını isteyip ilgili taraflara konu ile birlikte gönderilmesi amacıyla kullanılabileceği düşünülmektedir. Araştırmacılar tarafından Twitter konum bilgisi sağlayabilen geo sosyal bir ağ olarak

tanımlanmaktadır. Kimi kullanıcılar da rakip analizi ve tedarikçi seçiminde kullanılabilecek konum bilgisi sağlamakta ve kargo firmalarını birbiri ile karşılaştırmaktadır. Bu önerilerin uygulanabilirliği ve işe yararlılığı başka araştırmaların konusunu oluşturmaktadır.”

2.BÜYÜK VERİ DEPOLAMA TEKNOLOJİLERİ

Büyük veri genel olarak kullanılan programların yaptığı saklama işlemleri, yönetme işlemleri ve bu işlemlerin kapasite miktarının daha ötesini kapsayan, bu olguları kontrol eden bir araçtır. Verilerin saklanması işlemi de yıllar geçtikçe daha kolay hale gelmeye başlamıştır. Bu bölümde veri depolama alanları hakkında bilgi verilecektir.

Verilerin gitgide artmasından kaynaklanan en önemli problemlerden birisi bu verilerin depolama işlemlerinin sorunsuz bir şekilde yapılmasıdır. Bu depolama işlemlerinin çoğu genel olarak veri tabanı ile yapılmaktadır. Verilerin boyutları şu aşamada petabyte, exabyte veya zettabyte seviyelerinde değerlendirilmektedir. Zira 2020 yılına gelindiğinde depolanacak veri miktarının 35 zettabyte olacağı tahmin edilmektedir.

Bu depolama işlemlerine öncülük eden ve büyük şirketlerin de kullanımında olan birçok uygulama mevcuttur. Bunlardan en öne çıkanı BigQuery uygulamasıdır. BigQuery uygulaması sunucusuz çalışan büyük verilerin analitiğini kendine konu edinen bir web hizmetidir. Bunun yanı sıra asıl bahsedilecek olan konu veritabanı sistemleri olacaktır.

Veri tabanı sistemlerinden ön plana çıkan mysql, nosql, postgresql, mongoDb veri depolama alanlarının kullanıldığı başlıca elemanlardır. SQL ifadesi yapılanmış sorgulama dili olarak geçmektedir. Veri tabanları bir program tarafından okunur ve komutların çalışması ile depolama alanı çalışmaktadır. Aradaki iletişimi kuran dil sorgulama dilidir.

Mysql altı milyon gibi rakamlardan fazla uygulamada kullanılmaktadır. Mysql çoklu iş parçacıklıdır. Ayrıca çok kullanıcılı desteği vardır. Mysql hızlıdır ve sağlam bir veri tabanı yönetim yapısıdır.[3] 1994 yılında geliştirilmeye başlanmıştır. 1995 yılında ilk versiyonu yayınlanmıştır. “UNIX, Windows ve OS/2 gibi platformlar için ücretsiz elde edilmektedir. Bununla

birlikte istenirse ticari lisans gibi kullanımlar içinde ücretli bir lisans seçeneğine sahiptir. Linux ile birlikte daha hızlı ve kullanışlı bir performans sergilemektedir. MySQL açık kaynak kodludur ve pek çok platform için çalıştırılabilir ayrıca ikilik kod halindeki indirilebilir versiyonları da vardır. Ayrıca ODBC sürücülerine sahiptir ve ODBC sürücüleri bulunduğu için birçok geliştirme platformunda rahatlıkla uygulanabilir.

Geliştiricilerin verilerine göre, 500'den fazlası 7.000.000 kayıt içeren tablolar kullanmakta ayrıca 10.000 tablodan fazla oluşan kendi veritabanlarını (100 gigabyte veri miktarından fazla) MySQL'de tuttuklarını söylüyorlar.

Web sunucu dünyasında en çok kullanılan veri tabanı olduğu görülmektedir.

MySQL, verileri tuttuğu tablolarla çoklu kullanıcılı sistemlerde genelde söz konusu olan erişim ve yetki durumları gibi sorunlara başarılı bir şekilde çözümler üretmektedir. MySQL'in 4.0 sürümü ile birlikte "transaction" desteği eklenmiştir. MySQL'in 4.1 sürümüyle birlikte de gelen çok kullanışlı alt sorgu desteği eklenmiştir.

Ayrıca veri tutarlılığını sağlama kısmı programcıya bırakılmıştır ancak bu tercih bir dezavantaj olarak görülmeyebilir. Çünkü birçok veri tabanı programcısı veri tabanı yönetim sistemindeki veri tutarlılığının esneklik desteklemeyen bir yapı olduğunu düşünmektedir. [3]”

NoSQL, klasik ilişkisel veri tabanı yönetim sistemlerinden (İVTYS) bir kaç şekilde farklılık gösteren veri tabanı yönetim sistemleri için kullanılan bir yapı kavramıdır. Bu veri depolarının standart tabela düzenlerine gereksinimleri olmayabilir, genelde kullanılan join sorgu işlemleri tercih edilmez, genel olarak yatay ölçeklemeye gidilmektedir. Akademisyenlerce ve makalelerde genel olarak bu şekilde veri depolama yapılarına bellek (structured storage) adı verilmektedir [4][5]. Bu yapı klasik ilişkisel veri tabanlarını bir alt kavram olarak değerlendirilmektedir. Bu kavrama uyan

yapılara aynı zamanda SQL ve Daha Fazlası (Not Only SQL) olarak da kabul edilir ve ilk başta 1998 yıllarında piyasaya sürülmüştür. Modern ilişkisel veri tabanları veri indeksleme işlemlerinde ve web trafiğinde yoğun veri uygulamalarında yetersiz kalmaktadır. NoSql yapısı ise tutarlılığı ve dağıtık çalışma sistemini kendine temel alarak bu açıdan modern ilişkisel veri tabanlarına göre daha hızlı yapıdadır. Farklı sunucularda yedekleme işlemi yapan dağıtık mimari teknolojisi kullanarak birden fazla sunucunun kullanılmasına öncülük etmektedir.

İlişkisel veri tabanı sistemlerinde oluşturulacak yapı iyi planlanmalıdır. Sonradan tablo ve kolonlar eklenmesi maliyeti artırabilir. Fakat NoSQL veritabanı sistemlerinde, bilgilerin kayıt altına alınması adına yapı geliştirilirken tablo ve sütunların bulunmaması nedeniyle maliyet aşaması en az kayıpla atlatılmaktadır. NoSQL sistemlerinde yeni bir alana gerek duyulduğu zamanlarda kayıt direkt olarak eklenebilir bunun nedeni NoSQL sisteminin alanı kendi oluşturması ve değeri anında kaydetmesidir. Kayıtlar maliyet olmadan gerçekleşmektedir, bu durumun asıl nedeni ise verilerin tablo ve sütunlarda saklanmamasıdır. Bunun yerine JSON ve XML formatına benzer yapıda saklayarak yapmaktadır.

PostgreSQL, ilişkisel veri tabanı modeli kullanan ve SQL standart sorgu dili ile beraber çalışabilen bir veri tabanı yönetim sistemidir. PostgreSQL iyi bir performans sağlar, güvenlidir ve çeşitli özellikleri ile kullanışlı olabilmektedir. Unix ile birlikte benzer işletim sistemlerinde çalışmaktadır. 1986 yılında Berkeley Üniversitesi'nde yapılan çalışmalar sonucunda ortaya çıkmıştır. Son halini 1994 yılında alan bu sistem ücretsiz ve açık kaynak kodlu olmasıyla ön plana çıkmaktadır.

"PostgreSQL'i rakipleri karşısında güçlü kılan bazı özellikleri vardır. Örneğin; transaction, subselect, trigger gibi. Ayrıca kilit çakışmalarını düşürmek için multi-version uyumluluğu özelliklerine sahip olması gibi özellikleri mevcuttur. Performans bakımından ise PostgreSQL başka ticari veya açık kaynak kodlu veri tabanları ile karşılaştırıldığında başarılı veriler ve

sonuçlar elde etmektedir. Bazı veri tabanı sistemleri ile karşılaştırıldığında birkaç açıdan hızlı, bazı açılardan ise yavaştır [6]".

PostgreSQL güvenli bir sistemdir. Güvenli olması kanıtlanmıştır. Her bir sürümü birden fazla olacak şekilde kontrollerden geçirilmiş ve her bir beta sürümü sürekli testlerle incelenmiştir. Geniş kullanıcı grubu ve kaynak koduna ulaşım konusunda oldukça kullanışlı olması sayesinde olası hatalar en kısa süre de çözülmektedir.

MongoDb'ye bakıldığı zaman bir NoSql uygulaması olduğu görülmektedir. MongoDb C++ programlama dili ile geliştirilmiştir. MongoDb büyüyen veriler sonucunda gerektiğinde ölçeklenebilir ve ayrıca doküman tabanlıdır. Verileri belge olarak saklamaktadır ve bu belgeleri de JSON şeklinde tutmaktadır. Bu sayede verilerde değişiklik olsa dahi kaydetme konusunda sıkıntı çıkarmaz. Ayrıca verilerin kopyası alınabilir ve bununla birlikte veri kaybı önlenmiş olmaktadır.

Ölçeklenebilir olması sayesinde verilere ulaşmakta hız kavramı ön plana çıkmaktadır. Diğer yapılara göre daha hızlı ve kullanışlıdır. Hız konusunda öncü olması büyük verilerin depolanması ve onlara ulaşılabilirlik açısından önemlidir.

2.1 Büyük Veri Analiz Teknolojileri

Dünyanın verileştirilmesi bilgi birikimi bakımından da büyük bir patlamaya neden olmaktadır. Bu bilgilerin kullanımı kişi açısından sınırlı kalacaktır. Çünkü çok büyük verilerden bahsedilmektedir. Toplanan bu bilgilerin yani verilerin bir zaman sonra ayrıştırılması gerekmektedir. Bu işlem esnasında verilerin analiz edilmesi şarttır. Büyük veri söz konusu olunca verilerin analizi de önemli bir problem olarak kullanıcıların karşısına çıkacaktır.

Veriyi analiz etmede başlıca yöntemler mevcuttur. Bunlardan en çok veri madenciliği ve doğal dil işleme metotları ön plana çıkmaktadır.

Veri madenciliği büyük veri analiz yöntemlerinin başlıcalarındandır. Veri madenciliği, verinin bilgiye ulaşımı şeklinde tanımlanabilir. Veri madenciliğinde genelde hem otomatik hem de bazı kısımları otomatik yapılar kullanılarak veri elde edinimi hedeflenir. Veri madenciliği veri üzerinden modeller üretmek, tema kısmında yaratıcı olmak gibi bu şekilde amaçlar için istatistik, yapay zekâ, bilgisayar bilimi gibi farklı birçok bilim dallarından algoritmalardan yararlanmaktadır.

Veri madenciliği bir anlamda büyük veride gizli olarak sınıflandırılan verilerin güvenilir istatistiksel yöntemlerle ortaya çıkarılma işlemidir.

Web üzerinde gerçekleştirilen işlemler sonucunda verileri elde etmeye yarayan bir diğer yöntem metin madenciliğidir. Özellikle sosyal medya üzerinde çalışılan verileri sosyal medya madenciliği adı altında toplamaktadır. Veri madenciliğinin bir tekniği anlamına da gelebilir.

Son yıllarda büyük veri için depolama işlemleri yapan bir teknoloji daha geliştirilmeye başlanmıştır. Bu teknolojinin adı Spark'tır. Spark teknolojisi 2009 yılında geliştirilmeye başlanmış Scala dili ile programlanan açık kaynak kod kütüphanesine sahip bir sistemdir. Büyük veri üzerinde paralel işlem yapmaktadır. Makine öğrenmesi içerisinde SQL, akan veri üzerinde çalışma (streaming) ve grafik işleme gibi bölümler vardır. Spark temelde veri mühendisleri tarafından talep görmektedir. Ayrıca veri bilimcileri keşif amaçlı veri analizleri için, özellik çıkarımı için ve tabiki ML yapmak için Spark'tan faydalanırlar.

Hadoop, sıradan sunucuları (commodity hardware) veya sıradan bilgisayarları birleştirerek kümeler (cluster) oluşturmaktadır ve üzerinden büyük verileri işlemektedir. Veriyi tek bir sunucuda veya bilgisayarda tutmak yerine dağıtık olarak tutan bir dosya sistemi Hadoop Distributed File System (HDFS) olarak adlandırılmaktadır. Hadoop ayrıca MapReduce özelliklerini bir araya getirmektedir. Java ile geliştirilmiştir ve açık kaynaklı bir kütüphanedir.

Daha basit ve sade bir şekilde anlatılırsa Hadoop, HDFS ve MapReduce teknoloji bileşenlerinden oluşan bir yazılımdır.

HDFS ile sıradan sunucuların disklerini birlikte çalıştırmasını sağlar ve bu diskleri bir araya getirerek büyük tek bir sanal disk oluşturur. Bu şekilde genişleyebilir bir yapı elde edilir ve tek diske sığmayacak kadar büyük boyutta birçok dosyalar HDFS sistemlerinde saklanabilir. Bu dosyalar parçalar halinde blok yapılarında (ayarlanabilir varsayılan boyutlar) birden fazla ve birden fazla sunucu üzerine (varsayılan 3 kopya) dağıtılarak RAID sistemleri benzeri bir yapıya benzer şekilde yedeklenmektedir. Bu şekilde veri yedekli şekilde tutulur ve veri kaybı önlenmiş olmaktadır. Ayrıca HDFS tek diske sığmayacak kadar büyük boyutlu dosyalarda hızlı ve güvenilir okuma işlemi (streaming) sunmaktadır. Ancak HDFS yapısında rastlantısal erişim (random access) benzeri özellikler bulunmaz. HDFS yapı olarak NameNode ve DataNode gibi bazı süreçlerden (process) oluşmaktadır.

NameNode ana (master) süreç genel yapıdan sorumludur. Genel yapı olarak blokların sunucular üzerinden oluşması örnek gösterilebilir. Başka görevleri ise blokların dağıtılması, silinmesi, bir blokta sorun meydana geldiğinde düzeltmesi ve her türlü yetki yada dosya erişiminden sorumludur. Kısaca anlatmak gerekirse HDFS üzerindeki tüm dosyalar hakkındaki işlemler NameNode tarafından saklanmakta ve yönetilmektedir. Bu görevler sonucunda çakışma sorunları olmaması nedeni ile her kümede yalnızca bir adet NameNode olabilir.

DataNode, NameNode ‘a düzenli olarak kod listesi sağlayan, veriyi saklayan işçi süreçtir. Her bir DataNode kendi yerel diskindeki veriden yetki alanı olarak sorumludur. Ancak veri kaybını önlemek amaçlı diğer DataNode’lardaki verilerin yedeklerini de kendi içlerinde barındırırlar. Genel yapıya bakıldığında DataNode’lar küme içerisinde birden fazla olabildiği görülmektedir.

Hadoop MapReduce ise NameNode ve DataNode yapılarından oluşan HDFS üzerindeki birden fazla parçadan oluşan büyük dosyaları işleme amacıyla kullanılan bir yöntemdir. Büyük verileri filtrelemek için Map fonksiyonu kullanılır ve bu filtrelenen verilerden sonuç elde etmek için ise Reduce fonksiyonlarından destek alınır. Bu fonksiyonlar sonucunda oluşan program ise Hadoop üzerinde çalıştırılır. Hadoop Map ve Reduce'lerden oluşan işleri paralel işlemek için kümelere dağıtmaktadır ve sonrasında kendi içlerinde bulunan verilerle aynı anda işlenmesini sağlamaktadır. Bu işlemler sonucunda oluşan sonuç verilerini tekrar bir araya getirilmesini sağlamaktadır. Hadoop'un bu alanda başarılı olmasının sebebi; işlenen dosyaların okunması her zaman ilgili düğümün yerel diskidir ve bu da ağ trafiğini meşgul etmez. Yani Hadoop kümesindeki düğüm sayısı ne kadar fazla ise performansı da doğrusal olarak artmaktadır [7]. Spark teknolojisi Hadoop teknolojisi ile karşılaştırıldığında Spark'ın daha hızlı olduğu iddia edilmektedir.

3.DOĞAL DİL İŞLEME

Doğal dil işleme, doğal dildeki metinlerin veya seslerin bilgisayar ortamında işlenmesi alanında çalışmalar yapılan bilgisayar ve dil biliminin bir arada çalışmasına öncü olan bilim dallarından biridir. Bu kapsamda sesin dalgalarının tanınması ve bilgisayar ortamında metne aktarılması, metin seslendirme, biçimbilimsel çözümleme/üretme, sözdizim çözümlemesi, anlamsal çözümleme gibi yöntemler kullanılmaktadır. Ayrıca kullanılan bu yöntemler üzerinde çalışılmaya devam edilmekte ve geliştirilmektedir. Yazım hatalarının denetimi ve buna bağlı olarak düzeltilmesi, bilgisayar destekli çeviri işlemi, bilgi çıkarımı, özet çıkartma gibi uygulamalar doğal dil işleme alanının çalışma alanları içerisinde yer almaktadır.

Doğal diller birbirlerinden farklı özellikler taşımaktadırlar bu yüzden doğal dil işlemenin sonuçlara ulaşması için belli bir miktarda veri tabanına ve belirli bir biçimsel metotları bulunan dil modelinin yapılarını dikkate alması gerekmektedir. Örnek verilmesi durumunda bir cümleyi ciddi anlamda çözümlemek ve sonrasında anlayabilmek için, sözcüklere analiz işlemi uygulanmalı, cümle yapıları, kelimelerin bağlamları ve birbirleriyle ilişkileri ortaya konulmalı ve cümlenin gerçek anlamına erişmeden önce, birçok olası anlam göz önünde bulundurulmalıdır. Bu yapıya bakıldığında doğal dil işlemenin ne kadar zor bir yapı olduğu görülmektedir.

Doğal dil işleme girdilere göre iki çeşittir. Bunlar yazılı ve sesli metinlere göre yapılan sistemlerdir. Doğal diller çoğunlukla iki kaynaktan oluşmaktadır. Birinin konuşması sonrasında ona sesli yanıt verilmesi olabilir veya yazılı bir metnin analiz edilmesi gibi düşünülebilir. Doğal dil işleme çalışmaları başlıca şu şekildedir; ses üzerinden çok fazla işlem ve çalışma yapılmamaktadır. Bununla birlikte daha çok sesli anlatımların yazıya dökülmesi işlemleri yapılmaktadır. Daha sonra yazılı hale gelen metinler üzerinden işleme dâhil edilmektedir. Doğal dil işlemenin çalışma durumlarından yola çıkılarak dört ana madde görülmektedir. Kelime bilimi, sözdizimsel, anlamsal ve söylevdir. Kelime bilimi kelimelerin anlamlarının

anlaşılmasıyla ilgilenmektedir. Kelime bilimi kelimelerin kökünün hangi ekler sonrasında hangi anlamlara dönüştüğünü incelemektedir ve bu incelemeler sayesinde kelimenin ne olduğu anlaşılmaktadır. Sözdizimsel kısmı ise kelimelerin cümledeki dizilim şeklini incelemektedir. Genelde bir cümlede bulunan kelimelerin nasıl dizildiği ile ilgilenir. Anlamsal kısmı ise kullanılan cümlelerin anlamlarını incelemektedir. Çünkü bilgisayar tarafından doğal dilin doğru işlenebilmesi bu cümlenin doğru bir şekilde anlaşılmasına bağlıdır. Söylev ise bir konuşma sırasında kullanılan kelimeler ve anlamlarıyla ilgilenmektedir. İlk önce bilgisayar tarafından konuşma anlaşılmalı ve sonrasında nelerden bahsedildiği anlaşılan cümleler doğrultusunda kelimeler kullanılarak cevap verilmesi durumudur [8].

Doğal Dil Araç Seti olarak bilinen daha genel olarak NLTK, Python programlama dili kullanılarak yazılmıştır. İngilizce dili için sembolik ve istatistiksel doğal dil işleme (NLP) yapmaktadır. NLTK, grafiksel gösterimleri içermekte ve örnek verileri sunmaktadır.

NLTK, deneysel dilbilim, bilişsel bilim, yapay zekâ, bilgi edinme ve makine öğrenimi dâhil olmak üzere NLP'de veya yakından ilgili alanlarda araştırma ve öğretmeyi desteklemeyi amaçlamaktadır. NLTK başarılı bir şekilde öğretim aracı, bireysel çalışma aracı ve araştırma sistemlerinin hazır hale getirilmesi ve inşa edilmesi için bir platform olarak kullanılmıştır. ABD'de 32 üniversite ve kurslarında NLTK kullanan 25 ülke vardır. NLTK, sınıflandırma, belirtme, köklendirme, etiketleme, ayrıştırma ve anlamsal akıl yürütme işlevlerini desteklemektedir.

Bu araç seti, makinelerin insan dilini anlamalarını ve uygun cevaplarla cevap vermelerini sağlayan paketler içeren en güçlü NLP kütüphanelerinden biridir. Tokenizasyon, Stemming, Lemmatizasyon, Noktalama, Karakter sayımı, Kelime sayımı bu araç setinde çalışılan bazı konulardır.

Lemmatizasyon gömme işlemidir. Gömme işlemine çok benzer bir işleme lemmatizing denir. Tamamen gerçek kelimelerden oluşur.

Lemmatizasyon, bir kelimenin farklı yöndeki biçimlerini bir araya getirerek, tek bir madde olarak analiz edilmelerini sağlayan bir süreçtir. Lemmatizasyon, kök salmaya benzer, ancak kelimelere bağlam getirir. Bu yüzden benzer anlamı olan kelimeleri bir kelimeye bağlar.

Stemming bir kelimenin kök parçasını tutmaktadır. Örnek olarak yazı, yazlık, yazar gibi kelimeler veriliyor olsun. Bu kelimelerin kökü yaz kelimesi olacağından stemming adlı çalışmada bu “yaz” kelimesini hafızasında saklayacaktır.

Metin ön işlemleri genelde Stemming ve Lemmatizasyon olaylarını birlikte içerir. Çoğu zaman insanlar bu iki terimi kafa karıştırıcı bulurlar. Bazıları bu ikisine aynı şekilde davranır. Aslında, Lemmatizasyon Stemming yerine tercih edilir çünkü Lemmatization kelimelerin morfolojik analizini yapar.

NLP'de dizi etiketlemesi, n-gram modelleri, geri çekilme ve değerlendirme gibi bazı temel teknikleri ele alırken kategori ve etiketleme kelimeleri kullanılır. Bu teknikler birçok alanda kullanışlıdır ve etiketleme yapabilmek için basit bir bağlam sunmaktadır. Etiketlemenin ardından tipik NLP boru hattında ikinci adımın oluşturulmasını sağlamaktadır.

Kelimeleri konuşma bölümlerine ayırma ve bunları etiketleme süreci, konuşmanın bir kısmı etiketleme, POS etiketleme veya basitçe etiketleme olarak bilinmektedir. Konuşmanın bölümleri, kelime sınıfları veya sözcük kategorileri olarak da bilinmektedir. Belirli bir görev için kullanılan etiket koleksiyonu, etiket kümesi olarak adlandırılabilir.

Herhangi bir soru için, birinin cevabı bir yere yazmış olması muhtemeldir. Elektronik formda mevcut olan doğal dil metni miktarı gerçekten şaşırtıcı boyuttadır ve her geçen gün artmaya devam etmektedir. Ancak, doğal dilin karmaşıklığı o metindeki bilgilere ulaşmayı çok zorlaştırabilir. NLP'deki sanatın durumu, sınırlandırılmamış metinden genel amaçlı anlam temsilleri inşa etmekten çok uzaktır. Bunun yerine, çabalarımızı "farklı

bağlamların nerede bulunduğu" veya "hangi şirket tarafından kimin işe alındığı" gibi sınırlı bir dizi soruya veya "varlık ilişkilerine" odaklarsak önemli ilerleme kaydedebiliriz. Bu ilerlemeyi yapabilmek için metinden bilgi çıkarma yöntemi kullanılmaktadır.

Doğal dil işlemede NLTK metodu kullanılırken istenmeyen kelimelerin durdurulması sağlanabilir. Bu durdurmayı sağlamak için durdurucu kelimeler yöntemi kullanılmaktadır. Durdurucu kelimeler için işlenecek metinden filtreleme işlemi yapılabilir. NLP araştırmasında evrensel bir durma kelimesi listesi yoktu, ancak NLTK modülü durma kelimelerinin bir listesini içerir.

Word2Vec, kelimeler arasındaki uzaklık durumunu vektörel uzayda tahmin tabanlı anlamamızı sağlayan algoritmalar bütünüdür. Kelimelerin birbirine olan uzaklığını yorumlayarak yeni ilişkileri anlamlandırılmasını sağlar. Araştırmacı Tomas Mikolov ve çalışma arkadaşlarının çabası ile 2013 yılında kullanılmaya başlanmıştır. İki adet yöntem üzerinde oluşmaktadır, bunlardan birisi CBOW(Continuous Bag of Words) yöntemi ve diğeri Skip-Gram yöntemidir. İki yöntem de yapı olarak birbirine benzerlik taşımaktadır. Bu sistemin üzerine yazılmış yapılar ile bir kelimeye en yakın kelimelerin liste şeklinde gösterilmesi mümkündür.

“distance” metodu kullanıldığı takdirde verilen kelimeye yakın uygun kelimeler listelenmektedir. Kelimelerin yapısını incelemek için “word analogy “ tekniği kullanılmaktadır. Bu teknik kullanılırken CBOW algoritması ve SkipGram algoritmalarından faydanılmaktadır. CBOW ve SkipGram yöntemlerinde aslında kelimelerin arka arkaya farklı şekilde gruplanmasından oluşmaktadır.

CBOW modelinde verilen pencere ekranının çevresinde olan kelimeler giriş olarak alınıp, verilen pencere ekranının merkezinde olan kelimeler çıkış olarak tahmin edilmeye çalışılmaktadır. Merkezde olmayan kelimeler çıkış olarak tahmin edilmeye çalışan ve tam tersi merkezdeki kelime giriş olarak alan modelin adı ise Skip-Gram modelidir.

Derin öğrenme makine öğrenmesinin sınıfıdır. Derin öğrenme, özellikleri çıkarmak için ve özellikleri dönüştürmek için birçok doğrusal olmayan işlem birim katmanını kullanmaktadır. Her ardışık katman bir önceki katmandaki çıktı verisini girdi olarak almaktadır [9]. Algoritmalar sınıflandırma gibi yani denetimli veya desen analizi gibi denetimsiz olabilmektedir. Derin öğrenmenin temel yapısı verilerin birden farklı özellik seviyesinin veya temsillerinin öğrenilmesine dayanmaktadır. Alt düzey özelliklerden üst düzey özellikler türetilerek hiyerarşik bir temsil oluşmaktadır. Bu temsil, soyutlamanın değişken seviyelerine karşılık gelen birden farklı temsil seviyesini öğrenmektedir [10]. Derin öğrenme temelinde verinin temsilinden öğrenmeye dayalıdır. Bir görüntüleri temsil etmek için yoğunluk değerlerin piksel başına bir vektörü düşünülmektedir. Ek olarak görüntünün kenar kümeleri ve özel şekiller gibi özellikleri düşünülmektedir. Avantaj sağlamak için algoritmalar kullanılmaktadır ve bu kullanılan algoritmalar derin öğrenme yöntemlerinden birisi olan elle çıkarılan özellikleri kullanmak yerine veriyi en iyi temsil eden hiyerarşik özellik çıkarımlarını kullanmaktadır [11].

Bilgisayar teknolojilerinin gelişmesine istinaden toplum neredeyse tüm işlemlerini bu yeni sistemler üzerinde yapmakta ve yeni yöntemlerin bulunmasına katkıda bulunmaktadır. Bu sebepten dolayı, 1980'li yıllarda ortaya bir fikir sunulmuştur. Bu fikir de makinanın insan gibi düşünebilmesini göz önüne almaktadır. 1990'lı yıllarda Yapay Sinir Ağları teknolojisine ilgi artmış ve kısa sürede gelişimi ile ön plana çıkmıştır.

Konvolüsyonel Sinir Ağları (Convolution Neural Network-CNN) algılayıcı türüdür ve çok katmanlı algılayıcılara (Multi Layer Perceptron-MLP) bir örnektir. Görme merkezindeki hücreler alt bölgelere ayrılmıştır çünkü tüm görseli kapsayacağı düşünülmüştür. Basit hücrelerin yoğunlaştığı kenar benzeri özelliklere yoğunlaştığı ve karmaşık hücrelerin ise daha geniş alıcılarla, tüm görsele yoğunlaştığı düşünülmektedir. Hayvanların görme merkezinden esinlenilerek ortaya atıldığı ileri yönlü bir sinir ağı olan CNN algoritmasıdır. Bir nöronun kendi alanından uyaranlara verdiği cevap olarak burada matematiksel konvolüsyon işlemi kullanılmaktadır. Bir veya daha

fazla konvolüsyonel katmandan oluşmaktadır. Katmanlardan alt örnekleme (subsampling) katmanı bulunmaktadır. Sonrasında standart çok katmanlı bir sinir ağı gibi bir veya daha fazla tamamen bağlı katmandan oluşmaktadır.

Yapay sinir ağlarının özellikleri, doğrusal olmama, paralel çalışma olanağı sağlama, öğrenme, genelleme, hata konusunda esnek olma, birden fazla değişkene ve parametre kullanımına olanak sağlama, uyarlanabilirlik olarak verilebilir. Yapay sinir ağlarının temel faktörleri incelendiğinde 3 sistematik yapı ortaya çıkacaktır. Öngörü performansını doğrudan etkileyen bu yapılar mimari yapı, öğrenme algoritması ve aktivasyon fonksiyonudur. Bu yüzden karar aşamasında veri uygunluğuna dikkat edilmelidir. İleri beslemeli bir sinir ağında 3 katmanlı girdi, gizli ve çıktı katmanları bulunmaktadır.

Tek yönlü bilgi akışı söz konusu olan ileri beslemeli model YSA modelidir. YSA ağ modelinde gizli katmana iletilmek için girdi tabakasından bilgiler alınır. Çıkış değeri gizli ve çıktı tabakalarından bilginin işlenmesi ile belirlenir.

Geri beslemeli YSA modeli; geri beslemenin çıkış ya da ara katmandaki çıkışların; giriş birimlerine ve ya önceki ara katmanlara aktarıldığı ağ yapısıdır. Bu sebeple giriş birimleri ileri ve geri yönde aktarılabilir.

Bu çeşit YSA modellerinin statik değil dinamik hafızaları vardır. YSA modelleri anlık çıkışları yansıtır. Ayrıca hem anlık hem de önceki girişleri YSA modelleri yansıtır. Bu sebepten ve bu özelliklerle önceden tahmin işlemleri için uygun hale gelmektedir. Geri beslemeli YSA modeli içeren ağlar değişik zaman serilerinde tahmininde oldukça başarı sağlamaktadır.

YSA'nın en önemli özelliklerinden biri öğrenme algoritmasına bakıldığında bir bilgi kaynağından öğrenebilme yeteneğidir. Yapay sinir ağlarında bilgi, ağdaki sinirlerin bağlantılarının ağırlıklarında tutulmaktadır. Ağdaki sinirlerin bağlantılarının ağırlıkları tutulduğu için ağırlıkların nasıl

belirleneceği önemli hale gelmektedir. Bilgi tek ağda elde edilememektedir tüm ağda bulunmaktadır ve bu durumdan dolayı bir düğümün sahip olduğu ağırlık değeri tek başına sonuç vermemektedir. Tüm ağdaki ağırlıklar için en uygun değerler alınmalıdır. Ağın eğitilmesi; ağırlıklara ulaşılabilmesi için yapılan işleme denir. Ağırlık değerlerinin dinamik olarak değiştirilebilir olması ağın eğitilebilir olabilmesini sağlamaktadır.

Girdi ve çıktı katmanları arasındaki eğrisel eşleşmeyi sağlamak için Aktivasyon fonksiyonu kullanılmaktadır. Ağın performansını önemli derecede etkileyen durumlardan birisi aktivasyon fonksiyonunun doğru seçilmesidir. Aktivasyon fonksiyonu genelde 0 ve 1 olmak üzere tek kutuplu, -1 ve +1 olmak üzere çift kutuplu ve doğrusal olarak seçilebilirler. Aktivasyon fonksiyonu ağın yapıyı öğrenmesini sağlayan kısmıdır.

Bazı Aktivasyon fonksiyonları adımsal aktivasyon fonksiyonu, sigmoid aktivasyon fonksiyonu, parçalı doğrusal aktivasyon fonksiyonu, gaussian aktivasyon fonksiyonu, doğrusal aktivasyon fonksiyonu ve hiperbolik tanjant aktivasyon fonksiyonudur.

Yapay Sinir Ağları finans, bankacılık, otomotiv, güvenlik ve sağlık gibi geniş bir kullanım alanına sahiptir.

Tekrarlayan bir sinir ağı (RNN), düğümler arasındaki bağlantıların zamansal bir dizi boyunca yönlendirilmiş bir grafik oluşturduğu bir yapay sinir ağları sınıfıdır. Bu geçici dinamik davranış sergilemesini sağlar. İleri beslemeli sinir ağlarından farklı olarak, RNN'ler girdi sıralarını işlemek için iç durumlarını (hafızasını) kullanabilir. Bu onları ayrılmamış, bağlı el yazısı tanıma [12] veya konuşma tanıma gibi görevlere uygulanabilir kılmaktadır [13-14].

Tekrarlayan sinir ağı terimi, biri sonlu ve diğeri sonsuz dürtü olmak üzere benzer bir yapıya sahip iki geniş ağ sınıfını belirlemek için ayırt edici olmayan bir şekilde kullanılmaktadır. Geçici dinamik davranış sergilemek her

iki ağ sınıfı içinde geçerlidir [15]. Sonlu bir dürtüsel tekrarlayan ağ, kontrol edilemeyen ve katı bir şekilde ileriye dönük bir sinir ağı ile değiştirilebilen yönlendirilmiş bir asiklik grafiktir; sonsuz bir dürtüsel tekrarlayan ağ, kontrol edilemeyen bir yönlendirilmiş döngüsel grafiktir.

Sonlu dürtü ve sonsuz dürtü olmak üzere tekrarlayan ağların her ikisi de ek depolanma durumuna sahip olabilir. Depolama durumu doğrudan sinir ağı tarafından kontrol altına alınabilir. Depolama durumunda geri bildirim döngüleri bulunuyorsa veya zaman gecikmeleri içeriyorsa bu durumlarda başka bir grafik veya başka bir ağ ile değiştirilebilir. Geçişli durum veya girişli bellek gibi durumlar kontrol edilen durumlardır. Bu durumlar uzun kısa süreli bellek ağlarının ve girişli tekrarlanan birimlerin bir kısmıdır.

LSTM uzun kısa süreli bellek anlamına gelmektedir ve 1997 yılında ortaya çıkmıştır. Büyük kelime konuşma tanıma sisteminde kullanılmış ve Google bünyesine katılmıştır. Metin konuşma sentezini gerçekleştirme işlemi sağlamıştır.

Uzun ve kısa süreli hafıza olan LSTM, kaybolma gradyan problemini önleyen derin bir öğrenme sistemidir. LSTM normalde “unut” kapıları adı verilen tekrarlayan kapılarla artırılmaktadır. Geri yayılan hataların kaybolmasını veya patlamasını önlemek için LSTM kullanılmaktadır. Bunun yerine, uzayda katlanamayan sınırsız sayıda sanal katman boyunca hatalar geriye doğru akabilir. Yani LSTM ile milyonlarca ayrık zaman adımının yaşadığı olayların anılarını gerektiren görevleri öğrenebilir. LSTM uzun gecikmeler bulunan önemli olaylar arasında bile çalışmaktadır. Bu nedenden istenildiğinde her probleme özgü LSTM benzeri topolojiler üretilebilir. Ayrıca düşük ve yüksek frekanslı bileşenleri karıştıran sinyaller yönetilmektedir.

TensorFlow bir kütüphanedir. Açık kaynak kodlu olan bu kütüphane derin öğrenme kütüphanelerinden biridir. Esnek yapıya sahiptir bu nedenden dolayı herhangi bir API ile platform fark etmez ve hesaplamaları bir veya birden fazla CPU, GPU kullanarak işlemlerin gerçekleşmesini

desteklemektedir. Python kullanılarak geliştirilen bu framework kullanışlı bir yapı olduğunu Java, C++, C#, Javascript ve R gibi birçok dili desteklemesiyle kanıtlamaktadır.

Çoğunlukla kullanılan web ve mobil uygulamalara özel geliştirilen TensorFlow paketlerinin olması TensorFlow'u çeşitli alanlarda kullanılabilir kılmaktadır. Farklı platformlara göre hazırlanmış TensorFlow kütüphanelerinin olması yaygın olarak kullanılmasındaki önemli faktörlerden biridir. Python, R, Java ve C++ gibi dillerde TensorFlow kullanarak uygulama geliştirmek için bu dillere ait olan kendi TensorFlow kütüphaneleri kullanılabilir. TensorFlow yapay zekânın gelişimi ve kullanılması için özel olarak hazırlanan bir sistemdir. İki çeşit kullanım alanına sahiptir. Bunlar GPU ve CPU seçenekleridir. GPU seçeneği grafik özelliğinin ön planda olduğu sistemdir ve bu sistemde TensorFlow'un çalıştırılması önerilmemektedir. Bunun sebebi yavaş kalmasından kaynaklanmaktadır. CPU , GPU'ya göre daha hızlı olup daha sistematik çalışmaktadır. TensorFlow'un sadece kendine has çalışabildiği TPU metodu da sık sık kullanılmaktadır.

Sentiment Analysis yani duygu analizi yapay zekâda doğal dil işleme içinde kullanılan en büyük yapılarda biridir. Bu tez çalışmasında da önemli yer kaplamaktadır. Duygu analizi, duygusal durumları ve subjektif bilgileri sistematik olarak tanımlamak , ölçmek, çıkarmak, ve incelemek için doğal dil işleme, metin analizi, hesaplama dilbilimi ve biyometri kullanımı anlamına gelmektedir.

Verilen metinin duygusal olarak ifade etmek istediği sınıfı belirleme işlemine duygu analizi denir ve temel olarak bir metin işleme (text processing) işlemidir. Verilen metni olumlu, olumsuz ve tarafsız olduğunu belirlemeyi amaçlamakta olan duygu analizinin ilk çalışmaları duygusal kutupsallık olarak adlandırılmaktadır. Daha sonra yapılan çalışmalar sonucunda farklı duygu durumları belirten analizlerde elde edilmiştir. Duygu durumlarını kodlamak için veri kümesi üzerinde çalışılmaktadır ve her metnin tek bir duygu ile etiketlenmesi veya metinlerde birden fazla duygunun

etiketlenmesi gerekmektedir bu şekilde sınıf etiketlemesi yaklaşımına benzer yaklaşımlar görülmektedir [16].

Literatürde genel çalışmalar enformasyon getirme (information retrieval) alanı altında çalışmaktadır. Duygu analizindeki metin ve etiket bağlantısının dışına çıkan çalışmaların yerini belirli olmayan sonuçlara dönüştüğü ve fikir madenciliği (opinion mining) çalışmalarına bıraktığı gözlemlenmiştir. Bu çalışma kapsamında duygu analizinin etraflıca anlaşılması ve yaşanan problemlerin ayrıştırılması gerekmektedir bunun için bilgi (knowledge) seviyesi problemlerden bahsedilecek olsa da sonuçta projenin kapsamı enformasyon (information) seviyesi problemlerin çözümü ile kısıtlıdır. Sosyal ağlardaki problemlerden birisi de fikir veya kanaat madenciliği olarak literatürde geçen problemdir. Problem literatürdeki konumu itibariyle duygu analizi (sentimental analysis) altında geçmektedir. Buna göre mesaj, paylaşım, duvar yazısı ve haber gibi bir sosyal medya bilgisinin taşımış olduğu fikri anlambilimsel olarak göstermek için yapılan çalışmaya fikir madenciliği denir. Fikir veya kanaat oluşumunun bir topluluk üzerinde inceleniyor olması fikir madenciliği için en önemli kısımlarından birisidir. Özel olarak seçilmiş uzmanların fikirleri alınmadığı sürece fikir madenciliğinin ulaşmak istediği sonuç, bir topluluktaki bütün bireylerin fikirlerini anlayabilmektir. Ne yazık ki bütün bireylere ulaşmanın imkânsızlığı yüzünden genel de bu işlem örneklemelerle yapılmaktadır. Örneğin anket çalışmaları bu örneklemelerden birisidir [16].

Duyarlılık analiz sistemlerini uygulamak için aşağıdaki gibi sınıflandırılabilecek birçok yöntem ve algoritma vardır:

- Bir dizi elle hazırlanmış kurallara dayanarak duyarlılık analizi yapan kural tabanlı sistemler.
- Verilerden öğrenmek için makine öğrenme tekniklerine dayanan otomatik sistemler.
- Hem kural tabanlı hem de otomatik yaklaşımları birleştiren hibrit sistemler.

Genellikle kurala dayalı yaklaşımlar, öznelliği, kutupsallığı veya bir fikrin konusunu tanımlayan bir tür kodlama dilinde bir dizi kural tanımlar.

Kurallar, aşağıdakiler gibi çeşitli girdiler kullanabilir:

- Stemming, tokenization, konuşma etiketleme ve ayrıştırmanın bir parçası gibi klasik NLP teknikleri.
- Sözlükler gibi diğer kaynaklar (örneğin, kelime ve ifadelerin listeleri).

Kural tabanlı bir uygulamanın temel bir örneği şöyle olacaktır:

İki polarize kelime listesi tanımlayın (örneğin, kötü, en kötü, çirkin vb. Negatif kelimeler ve iyi, en iyi, güzel vb. Pozitif kelimeler).

Bir metin verildi:

“Metinde görünen pozitif kelimelerin sayısını”.

“Metinde görünen negatif kelimelerin sayısını.”

Olumlu kelime görünme sayısı, olumsuz kelime görünme sayısından büyükse, olumlu bir duyarlılık, tersine olumsuz bir duyarlılık döndürür. Aksi takdirde, tarafsız döndürür.

Bu sistem çok saftır çünkü kelimelerin sırayla nasıl birleştirildiğini dikkate almaz. Daha gelişmiş bir işlem yapılabilir, ancak bu sistemler hızla çok karmaşık hale gelir. Yeni ifadeler ve kelimeler için destek eklemek için yeni kurallar gerekebileceği için bakımı çok zor olabilir. Ayrıca, yeni kuralların eklenmesi, önceki kurallarla etkileşimin bir sonucu olarak istenmeyen sonuçlar doğurabilir. Sonuç olarak, bu sistemler kuralları manuel olarak ayarlamak ve sürdürmek için önemli yatırımlar gerektirir [17].

Bu tez çalışmasında bu ana kadar belirtilen yapılar kullanılarak bir yöntem önerilecektir. Bu aşamada programlama dili olarak çalışmanın içinde

de çok kullanıldığı gözlenen Python dili kullanılacaktır. Python nesne yönelimli programlama desteği olan yorumlamalı ve etkileşimli yüksek seviyeli bir programlama dilidir. Bir çok programlama dilinde kullanılan parantezler dışında farklı olarak girintilere dayalı basit sözdizimi vardır. Bu girintiler Python programlama dilinin hızlı öğrenilmesini ve akılda kalmasını kolaylaştırmasını sağlamaktadır. Bu da python Programlama diline söz diziminin ayrıntıları ile vakit kaybetmekten kurtarır ve asıl odaklanılması gereken programlama aşamasına geçilen bir dil olma özelliği kazandırır.

Bu tez çalışmasında Python programlama dili ile beraber Anaconda tool paketi de beraberinde kullanılacaktır. Anaconda Python ve R programlama dillerinin paketlerinin yönetimini kolaylaştırmayı sağlayan ücretsiz ve açık kaynaklı bir yapıdır. Anaconda içerisinde kod yazılırken kullanılacak yapılardan birisi de Jupyter yapısıdır. Bu yazılım da Python programlama dilinin oluşturduğu alt yazılım çeşididir.

Jupyter özellikle Python olmak üzere çeşitli programlama dilleri için bir geliştirme ortamı sağlayan yazılımdır. Orijinal olarak IPython (interactive python) adıyla Python programlama dili için geliştirildi. Ancak daha sonra kurucuları Jupyter projesini başlatmıştır ve IPython'ın birçok tarafını Jupyter'e kaydırmışlardır. IPython, sadece Jupyter'in kerneli olarak devam etmektedir.

Jupyter'in özellikleri;

- Etkileşimli bir shell sunmaktadır. Komut istemi terminalden Jupyter console komutu ile başlatılır ve sonrasında orijinal Python shell'ine göre otomatik tamamlama gibi kullanıcı dostu özellikleri barındırır.

- Tarayıcı tabanlı defter sunması. Komut İstemi terminalden Jupyter notebook komutu ile başlatılır ve açılan tarayıcı penceresinden yeni defter oluşturularak çeşitli programlama dillerinde kodlar yazılabilir. Ve sonrasında bu kodlar çalıştırılarak çıktıları metin ve grafik gibi etkileşimli olarak direkt

tarayıcıda görüntülenebilir. Bu özelliği ile programlama dili öğretmek ya da bir programlama dilinde kod çalıştırıp sonuçlarını analiz etmek çok pratikleşir.

- Paralel hesaplama için araçlar barındırır.

Jupyter sadece Python programlama dili için değil ekstra yüklemeler yapılarak R, Ruby ve Haskell gibi programlama dilleri için de kullanılmaktadır [18].

4. YAZILIM GELİŞTİRMESİNDE KULLANILAN TEKNOLOJİLER

Bu kısımda tez çalışmasında yararlanılan ve kullanılan teknolojilerin açıklamaları ve kullanım amaçları anlatılmaktadır. Uygulamanın oluşturulduğu bu alanların detaylı incelenmesi yapılacaktır.

Tez çalışmasında Twitter ile veri iletişimi sağlayacak olan uygulama Java programlama dili ile yazılmıştır. Twitter'ın ücretsiz olarak sunduğu Stream API (Application Programming Interface) kullanılmıştır. Verileri depolamak için PostgreSQL veritabanı kullanılmıştır. Twitter anlık veri akışı sırasında veri tabanına depolama işlemini yapmak için Kafka kullanılmıştır.

Kafka microservis mimarisi tarzında çalıştığından ayrı bir sanal makinede çalışması gerekmektedir. Bunun maliyet ve yönetim açısından bulunan zorlukları nedeni ile Docker teknolojisi kullanılması tercih edilmiştir. Docker alternatifleri bulunan konteyner çalışma prensibi sağlayan bir teknolojidir. Konteyner yapısı işletim sisteminden izole çalışmaktadır. Yani bir işletim sistemi çalışır birden fazla sanal makine oluşabilmektedir. Bunun avantajları şunlardır;

- Bir konteyner çalıştırmak saniyeler sürmektedir. Çok hızlı bir şekilde çalışmaktadır.

- Her sanal makine üzerinde işletim sistemi çalıştığı için boşta olsa bile bir miktar ram ve hafıza harcamaktadır. Konteyner yapısında ram ve hafıza için tek bir yerden harcanmaktadır. Diğerleri bu harcanan kısmı kullanmaktadır.

Bu tez çalışmasında konteyner yapısı kullanmak için Docker tercih edildi. Docker avantajları şunlardır;

- Docker, Hypervisor kullanmamaktadır, tamamen bir işletim sistemi gibi çalışmamaktadır, kurulum aşamaları olmadığından saniyeler içinde çalışır ve kullanıma hazır hale gelebilmektedir.

- Docker, yazılımlara gerekli olan işletim sistemleri yml kodları olarak bulunmaktadır. İstenildiği zaman istenilen işletim sistemine dönüş sağlanabilmektedir. Her koda bir numara atanmakta ve bu yapıya versiyonlama yapısı denilmektedir. Bu özellik ile yazılımlarınızı ve işletim sisteminizin imajını farklı servis sağlayıcılarında kullanabilmekte, gerektiğinde çoğaltabilmekte ya da kullanacak olan başka kişilerce paylaşılabilmektedir.

- Docker, çok az kaynak kullanmakta ve bu kaynağa oranla büyük işler gerçekleştirilebilmektedir. Bunu konteyner teknolojisi sağlamaktadır. Hypervisor gibi sanal makine oluşturucular ile sanal oluşturulan işletim sisteminin sistem gereksinimleri 1 işlemci çekirdeği, 10 gigabyte depolama alanı ve 1 gigabyte bellek gibi yüksek kaynaklarla ölçülmektedir, 5 sunucu olduğu düşünülürse 5 sunucu için gerekli olan işletim sisteminin sorunsuz çalışması için 5 işlemci çekirdeği , 5 gigabyte bellek , 50 gigabyte depolama alanı gerekmektedir. Bu işlemleri Docker teknolojisi ile yapıldığında 1 işlemci çekirdeği, gigabyte bellek, 10 gigabyte depolama alanı yeterli olmaktadır.

- Çalışmakta olan sisteme plansız olarak yüksek giriş olmaya başladığında saniyeler gibi kısa süre içerisinde yüzlerce konteyner çalışmakta ve hazır hale gelmektedir, hazır hale gelen konteyner'lar oluşan yüksek girişe kesintisiz bir şekilde sistemin devam etmesini sağlamaktadır.

- Kullanılan teknolojiye göre geliştirme ortamları değişmektedir. Docker, yazılımların çalışacağı ortamları oluşturmakta ve her platformda çalışmasını sağlamaktadır. Geliştiricilerin genel sıkıntısı olan platform farklılıkları docker ile giderilmektedir. Her platformda aynı sonuçları vermektedir.

- Saniyeler gibi kısa sürelerde standart işletim sistemleri olan Centos, Debian ve Ubuntu gibi sistemleri çalıştırabilmektedir. Java, PHP ve ASP gibi teknolojilerinde içinde kurulu olduğu imajlarda bulunmaktadır. Docker Hub mağaza benzeri ücretsiz docker imajlarının bulunduğu bir oluşumdur. Buraya isterseniz kendi gizli veya dışarıya açık imajlarınızı yayınlayabilmektedir.

Java Sun Microsystems ile piyasada yer almıştır. Yayınlanma 1995 yılında gerçekleşmiştir. Java masaüstü, web ve mobil gibi platformlarda uygulama geliştirebileceğiniz bir programlama dilidir. Java ücretsizdir ve açık kaynak kodludur. Yapısal olarak bakıldığında nesne yönelimli bir programlama dilidir. Gün geçtikçe destekleri artmaktadır. Yeni kütüphaneleri çıkmaktadır.

PostgreSQL ücretsizdir ve açık kaynak kodludur. Yapısal olarak nesne yönelimlidir. Büyük verilerde rahatlıkla çalışan bir veri tabanı sistemidir. Geliştirme yapılırken SQL dili kullanılmaktadır. SQL dilinin en önemli özelliklerinden olan güvenlik özelliklerinden faydalanmaktadır. SQL'in diğer bir özelliği olan depolanabilirlik özelliğini de kullanmaktadır. Birçok alanda veri tabanı olarak kullanılmaktadır. Birçok veri tabanı işlemlerini (depolama, sorgulama vb.) yapmaktadır. 1986 yılında ilk olarak piyasaya sürülen bu sistem hala gelişmeye devam etmektedir. En büyük avantajlarından birisi birçok programlama dili ile çalışabilmesidir. PostgreSQL'i diğer veri tabanları ile kıyasladığımızda ön plana çıkaran ve en güçlü kılan özelliği işlemler arası bağlantı kopukluklarında yapılan işlemleri alma yönetim özelliğidir. Ayrıca tetikleyici destekleri sunmaktadır. Hız, depolama alanı kullanımı ve performans konusunda birçok veri tabanı ile kıyaslandığında başarılı istatistik sonuçları bulunmaktadır.

Java'nın birçok framework yapısı bulunmaktadır. Kütüphaneleri birleştirerek ve kendi standartlarını koyarak Framework yapıları oluşturulmaktadır. Yazılım geliştiricilerin ihtiyaçları olan kütüphanelerin hazır bulunduğu ve bu framework yapısını daha fazla geliştirme imkanı sağlayan yapılardır. Frameworklerde genellikle form kontrolü vardır, veri tabanı türüne göre, desteklerine göre değişen veri tabanı bağlantısı bulunmaktadır. Kullanıcı session yani giriş çıkış işlerini gerçekleştiren yapılar bulunmaktadır. E-posta ve sms gönderme gibi çok kullanılan yapılar bulunmaktadır. Bazı framework yapılarında bulunan tema altyapıları bulunmaktadır. Frameworklerin bazılarında bulunan yapısal farklardan sayılan MVC (Model,

View, Controller) yapıları geliştirilen yazılımların okunurluğu geliştirilebilirliğini artırmaktadır.

Bu framework yapılarının bazıları şunlardır; Java Server Faces, Struts, Hibernate, Vaadin, Spring, Spring Boot vb… Bu tez çalışmasında framework yapılarında son zamanlarda en çok kullanılan Spring Boot framework yapısı kullanılmıştır. Spring Boot, Spring framework yapısı üzerine geliştirilmiştir. Spring framework ve Spring Boot framework yapısı arasında bulunan başlıca farklar şunlardır;

- Spring Boot, birçok geliştirme süresini azaltır ve üretkenliği artırır.

- Spring Boot çok sayıda kodu, ek açıklama ve XML(Extensible Markup Language) yapılandırması yazmaktan kaçınır.

- Spring Boot, geliştirici çabasını azaltmak için "Önerilen Varsayılan Yapılandırma" yaklaşımını takip eder.

- Web uygulamalarımızı kolayca geliştirmek ve test etmek için Tomcat, Jetty vb. Gömülü HTTP sunucularını sağlar.

- Birleştirilmiş ve bellek içi veri tabanlarıyla çok kolay çalışabilmesi için pek çok eklenti sağlar.

Veri tabanı ara katmanı olarak ORM(Object Relational Mapping) frameworklerinden olan Hibernate kullanılmıştır. ORM frameworklerine veri tabanı seçimini sonradan değiştirmek zorunda kalındığı zamanlarda ihtiyaç duyulmaktadır. Veri tabanlarının kendi artıları ve eksileri vardır. Uygulamanın gelişmesi aşamasında veya ihtiyaçların tam belirlenememesinden kaynaklı yanlış veri tabanı seçimleri yapılmaktadır. Bu durum sonucunda veritabanı değişikliğini kolay bir şekilde yapılması için ORM framework yapıları geliştirilmiştir. Veri tabanlarının bazıları SQL dil desteği sağlaması ve bazı veri tabanlarının bu desteği sağlamaması sonucunda ORM frameworkleri esneklik sağlamıştır.

Doğrudan SQL komutları yerine bir fonksiyon üzerinden çalışmaktadır. Fonksiyonun çalışması sonrasında komutların hangi veri tabanı kullanılıyorsa ona uygun şekle getirilmesi sağlanmıştır. Bu tez aşamasında yapılan çalışmanın geliştirmeye açık olmasını sağlamak amaçlı ORM kullanılmıştır. ORM framework çeşitleri bulunmaktadır. Artıları ve eksileri incelenerek Hibernate ORM frameworku tercih edilmiştir. Java Spring Boot uyumluluğu ve kaynak bakımından zengin olması büyük avantaj sağlamıştır.

ORM Framework kullanmanın avantajları;

- Veri tabanı tablolarını oluşturmak ve sınıflar arası bağlantıları sağlamak amaçlı tasarlanmıştır.

- Verilerin genel işlemlerini; silme, güncelleme, ekleme ve okuma gibi genel işlemleri gerçekleştirme aşamasında yardımcı olmaktadır.

- Verilerin genel işlemleri yapılma aşamasında karşılaşılan sorunların çözümlerini barındırmakta ve böylelikle hata oranını azaltmaktadır.

- SQL dilinin fazla bilinmemesi nedeni ile SQL bilmeyen yazılımcıların geliştirme yapabilmesine olanak sağlamaktadır.

Veri tabanı ile ORM framework arasındaki bağlantıyı sağlamak gerekmektedir bunun için JPA (Java Persistence API) teknolojisi kullanılmıştır. ORM framework yapısının değişmesi gerektiği zaman kolay bir geçiş sağlamaktadır. Kullanılan Hibernate ORM Framework yapısı zamanla desteğini kaybeder ve geliştirmesi durdurulursa yeni güncel bir ORM Framework yapısına geçiş kolaylığı sağlayacaktır. JPA ayrıca JAVA ile uyumludur. Kolay bir kullanım sağlar.

JPA avantajları;

- JPA kullanarak veri tabanında bağımsız uygulamalar geliştirme yapılabilir.

- JPA ile hızlı bir şekilde farklı bir veritabanına geçiş sağlanmaktadır.

- Veri yapısı kalıcılığı sağlanmaktadır.

- Veri işlemleri sırasında oluşan performans kayıplarını önlemektedir.

- Ölçeği büyük ve uzun yıllar kullanımda kalacak olan projelerde geliştirme ,yükseltme, yeni teknolojilere geçiş ve bakım işlemlerinin yapılmasını kolaylaştırır.

Bu teknolojiler Twitter'dan verileri çekip depolama işlemlerini yapmaktadır. Çekilecek veriye göre teknolojilerde değişiklik yapılabilir. Sonrasında bu verileri analiz edecek yapay zekâ teknolojileri gereklidir.

Yapay zekâ alanında en çok kullanılmakta olan teknolojilerin başında Python programlama dili gelmektedir.

Python nesne yüksek seviyeli bir dildir, yorumsal ve modülerdir. Programlama dillerinin asıl amacı makine ile insan arasında bulunan bağlantıyı sağlamaktır. Bir dilin düşük seviye yani makine diline yakın olması bu dil ile yazılan yazılımların daha hızlı çalışmasını sağlamaktadır. Makine diline yakınlaşmak geliştiricilerin daha fazla zaman harcaması daha fazla geliştirme yapmasına neden olmaktadır, ayrıca geliştirilen yazılımın kodlarının anlaşılmasının zorlaşmasına neden olmaktadır. Bir programlama dili; insanların kolay anlayabileceği bir hale gelirse yüksek seviye, daha karmaşık bir halde olursa yani makine diline yaklaşırsa düşük seviye olarak anılmaktadır. Python

programlama dili ise makine dilinden uzak ancak insan mantığına yakın olmasından dolayı yüksek seviye programlama dili olarak adlandırılmaktadır.

Python dünyada popüler dillerden biridir. Akademik çalışmalarda Python dilinin tercih edilmesi sonucunda yeni oluşan teknolojilerde öncülük etmektedir, bu yüzden Yapay zeka ve makine öğrenmesi gibi güncel teknolojilerde tercih edilmektedir. Bunun en önemli nedeni yapay zekâ alanında çok kütüphaneye sahip olması ve ayrıca makine öğrenmesi gibi alanlarda geliştirmelerin Python dili ile yapılması sonucunda Python programlama dilinin tercih sebebi artmaktadır. Bundan dolayı bu tez çalışmasında Python programlama dili tercih edilmiştir.

Python programlama dili avantajları;

- Python diğer dillerden farklı olarak daha kolay yazım şekline sahip bulunmaktadır.

- Çalışma sırasında debug dediğimiz çalışan nesneleri inceleme anlık değerlerini görme imkanlarını barındırmaktadır.

- Nesne yönelimli programlama yapısını desteklemektedir.

- Hata yönetimi kısmında exception tabanlı bir yapıya sahip bulunmaktadır.

- Dinamik veri yapıları konusunda yüksek seviye destek sunmaktadır.

- Akademik çalışmalarda tercih edilmesinden dolayı yeni teknolojilerde çok fazla kütüphaneleri bulunmaktadır.

- Diğer diller kullanılarak ek çalışma yapılabilmektedir yani C veya C++ ile ek modül yazılmaktadır ve bu işlemlerde kolaylıklar sağlamaktadır.

- Diğer programlama dillerinde kullanılan kodlama arayüzlerini dahil edebilmektedir.

Derin öğrenme alanında kullanılan teknolojilere bakıldığında en çok kullanılan kütüphanelerden birisi TensowFlow kütüphanesi bulunmaktadır. Son zamanlarda yapay zeka alanına yaptığı yatırımlarla bilinen Google tarafından geliştirilmektedir ve TensorFlow açık kaynaklıdır. TensorFlow ile derin öğrenme uygulamaları kolaylıkla geliştirilmekte ve yapay zekâ gibi uygulamaları desteklemektedir. Esnek yapıya sahip olmaktadır ve bu esnek yapı sayesinde, tek bir API ile çalışan platform fark etmeden hesaplama işlemlerini, bir veya birden fazla CPU yada GPU kullanarak yapmanıza olanak sağlamaktadır. İlk başlarda ve Python kullanılarak geliştirilmektedir ancak bu yapı, günümüzde Python'ın yanı sıra başka programlama dillerini de desteklemektedir. Bunlardan bazıları C++, Java, Javascript, C# ve R gibi programlama dilleridir. Bu tez çalışmasında TensowFlow Python programlama dili ile kullanılmaktadır. TensowFlow CPU yavaş çalışmaktadır. Bu yüzden TensowFlow GPU kullanılmaktadır. Tensorflow makine öğrenmesi sırasında büyük matris verileri ile çalışmaktadır. Bu yüzden çok çekirdekli donanımlarda daha hızlı çalışmaktadır. GPU CPU'a göre daha çok daha fazla çekirdekli olmasından dolayı hızlı çalışmaktadır.

Gensim; konu ve uzay modelleme amaçlı geliştirilmiş açık kaynaklı bir kütüphanedir. Doğal dil işleme için ham ve yapısal denetlenemeyen metinler için kullanılmaktadır. Türkçe dil desteği bulunmaktadır.Kullanımı ve öğrenilmesi basittir. Cümle içerisinde özne, nesne ve yüklem gibi ayıklama işlemleri yapmaktadır. Gereksiz kelimeleri anlayıp çıkarmaktadır. Derin öğrenme işlemi başlamadan önce gereksiz kelimeleri çıkarmak ve veri setini hazırlamak için kullanılmaktadır.

Kelimelerin bağlantıları arasındaki uzaklıkları hesaplamanızı sağlayan ve vektörel olarak işleyen bir algoritma barındıran araç kiti olarak adlandırılan Word2Vec kullanılmaktadır. Bu vektörlerin amacı kelimeler arasında yakınlığı hesaplamak ve en yakın kelimeleri belirlemektir. Kelimeler arası bağlantılar bu teknoloji ile kurulmaktadır.

4.1 Yazılım Geliştirme Ortamları

Bu bölümde tez çalışmasında kullanılan uygulamaların hazırlanmasında kullanılan uygulamalar hakkında bilgi verilecektir. Uygulamanın oluşturulduğu bu alanların detaylı incelenmesi yapılacaktır.

4.1.1 Jetbrains Uygulaması

JetBrains (resmi adı ile IntelliJ) yazılım geliştirme ve proje yönetimi ile ilgili araçlar geliştiren bir yazılım şirketidir [19].

Hızlı büyüyen bir oluşumdur.

Ofisleri, Prag, Sankt-Peterburg, Münih ve Boston şehirlerindedir [20].

Şirketin ilk uygulaması Intellij Renamer olarak kabul edilmektedir. Bu uygulamanın amacı Java kodlarını hızlı geliştirme ortamı sağlamaktır.

Şirket kullanıcılar için birçok programlama dilinin IDE'si bulunmaktadır. Bunlardan bazıları Javascript programlama dili için Webstorm, Java için IntelliJ Idea, PHP için PHPStorm, Ruby için RubyMine, Python için PyCharm, C ve C++ için CLion, veri tabanı işlemleri için DataGrip yazılımları bulunmaktadır. 2011 yılında ise Kotlin adında bir programlama dili de geliştirilmiş ve şu anda birçok firma tarafından kullanılmaktadır.

2010 yılında Android geliştirme desteği başlatılmıştır. Google 2012 yılında EclipseADT'nin yanı sıra IntelliJ IDEA'ya da destek vermeye başlamış ve Android Studio çıkarılmıştır. 2015 yılında EclipseADT üzerinde destek

çekilince Android konusunda Android Studio programı daha işlevsel hale gelmiştir.

Halen çalışmalarına devam etmektedir. Yazılım şirketi çeşitli anketler ile de yazılım dünyasına ışık tutmaktadır. Teknolojinin hızla ilerlemesiyle ortaya çıkan yeni ürünlere de çabuk uyum sağlamış olması ile kullanıcılara büyük kolaylık sağlamaktadır.

4.1.2 Intellij IDEA

Java programlama dilinde bilgisayar yazılımı geliştirmek için 2016 yılında sunulmaktadır. Ticari ve kişisel kullanımları lisans karşılığı mevcuttur. Yeni teknolojinin ve güncellemelerin sunulması kullanımı açısından kullanıcıyı hızlandırmakta ve rahatlatmaktadır. İlk kullanımı Ocak 2001'dir fakat bu kullanımda sadece Java kodlarının düzenlenmesi yapılmaktaydı.

2010 da InfoWorld tarafından yapılan test sonucunda IntelliJ; Eclipse, IntelliJ IDEA, NetBeans ve JDeveloper IDE'leri arasında en yüksek sonucu almıştır [17].

4.1.3 Jupyter Notebook

Jupyter Notebook, birçok programlama diline destek veren açık kaynaklı ve etkileşimleri sağlayan bir programdır. Bir defter gibidir. Aynı word içerisine yazı yazar gibi Jupyter Notebook'a Python kodları yazılabilmektedir. Yazılan kodları saklamakta ve istenildiği zaman çalıştırmaktadır. Tezlerde kullanılmasında göze çarpan iki önemli özelliği şunlardır;

- Öğrendiği ve alıştırma yapmak istediği kodlarla kendine göre notlar alabilir ve nasıl çalıştığını gösterebilir bir yapıya sahiptir.

- Rapor hazırlaması gerektiğinde bunu etkili bir şekilde kullanıcıya sağlayabilmektedir.

4.1.4 Data Grip

Jetbrain tarafından geliştirilen birçok yazılım bulunmaktadır, Data Grip bunlardan biri olan ve veri tabanı yönetim sistemlerinde kullanılan bir programdır. Veri tabanı yönetim işlemlerini yapar. Desteklediği veri tabanlarında bulunan komutları çalıştırma, tablo oluşturma, tablo silme, yedek alma, yedek yükleme, veri tabanı aktarımı gibi tüm veri tabanı işlemlerini desteklemektedir. Data Grip'in diğer avantajları birçok veri tabanı desteğinin bulunmasıdır. Aynı ekranda birden fazla farklı veri tabanı ile çalışma imkânı sunmaktadır.

Desteklediği veri tabanları;

- MySQL
- PostgreSQL
- AWS Redshift
- DB2
- Derby
- Exasol
- Sybase ASE
- ClickHouse
- SQL Server
- Azure SQL Database
- Oracle

- H2
- HSQL
- SQLite
- MariaDB
- Cassandra

Bu tez çalışmasında PostgreSQL kullanılmaktadır. Data Grip'in PostgreSQL desteği sağlamasından dolayı veri tabanı yönetim kısmında Data Grip kullanılmaktadır.

5. DENEYSEL ÇALIŞMA

Bu bölümde tezin amacında da belirtildiği gibi yapay zekâ ile Twitter'dan elde edilen verilerin siyasi olarak olumlu veya olumsuz olduğunu belirleyip, istatistik elde edilme çalışması incelenmektedir.

5.1 Yapay Zeka

Bu bölümde tezin yorumları olumlu ve olumsuz olduğunu anlamasını sağlayacak olan yapay zekâ yapısı incelenecektir.

Diller makine dili ve doğal dil olarak ikiye ayrılır. Makine dili C++,Java,Python vb. programlama dilleri ile yazılan algoritmalar ile bilgisayarların bunu işlemesi sonucu oluşur. Doğal diller ise Türkçe,İngilizce vb. insanlar arasında anlaşılan dillerdir. Doğal dillerde bir kelime birden fazla anlamı karşılayabilmektedir. Bunun aksine makine dillerinde tek bir doğru vardır. Burada bulunan problem doğal dilleri makine diline çevirmektir. Doğal dil işleme yapay zekânın bir alt koludur ve bilgisayarların insanların arasındaki dilleri kavramasını ve insanlarla iletişim halinde olmasına yarayacak çalışmaları içeren bir mühendislik bölümüdür. Geçmişte doğal dil işleme için tamamen kurallara dayalı bir dil işleme yapılmaktaydı ve kurallar istatistiksel tabanlıydı. Ancak günümüzde derin öğrenme ile geliştirmelere devam edilmektedir. Bu çalışmada da derin öğrenme ile geliştirmeler yapılmaktadır.

```
import …

import …

import …
```

Yukarıda verilen kod parçacığında kütüphanelerin nasıl eklendiği yer almaktadır.

Numpy vektörlerle ve matrislerle işlemler yapmayı kolaylaştıran bir kütüphanedir.

Pandas dataset okuma ve düzenleme işlemlerini yapmaktadır.

Öncelikle yapay zekâyı eğitmek için bir dataseti'ne ihtiyacımız vardır. Bunu yapmanın iki yöntemi bulunmaktadır. Birinci yöntemi Twitter'dan çekilen verilerin hangilerinin olumlu hangilerinin olumsuz olduğunun tek tek insan gücü ile belirlenmesidir. İkinci yöntemi hazır datasetleri veren kurumlardan veya sitelerden bu datasetleri almaktır. Hepsiburada sitesinin paylaştığı bir çalışma dataseti araştırma sonucunda bulunmaktadır. Sonrasında dataseti yapay zekâya öğretmek için ekleme çalışması yapılmaktadır.

```
dataset = pd.read_csv('hepsiburada.csv')
printf(dataset)
```

Yukarıda hepsiburada.csv dosyası içerisinde bulunan dataseti yüklemektedir. Sonrasında datasetin yüklendiğini onaylamak amaçlı ekrana yazdırmaktadır.

Çıktı şu şekilde olacaktır;

	Rating	Review
0	1	3 yıldır tık demedi. :)
1	1	3 yıldır kullanıyorum müthiş
2	1	Ürün bugün elime geçti çok fazla inceleme fırs...
3	1	Almaya karar verdim. Hemencecik geldi. Keyifle...
4	1	Günlük kullanımınızı çok çok iyi karsılıyor kı...
5	1	gayet güzel
6	1	Çok kaliteli bir ürün ve fiyatı da uygun. Uzun...
7	1	yaklaşık 5 senedir kullanıyorum. defalarca düş...
8	1	Ürün günlük kullanım için çok uygun. Ürünle bi...
9	1	Ürünü bir arkadaşım 2009 yılında hediye olarak...
10	1	1 gün gibi kısa bir sürede elime geçti. Ve bu ...

Şekil 5.1. Read_csv çıktısı

Dataseti incelersek olumlu yorumların karşısında 1 olumsuz yorumların karşısında 0 verildiği görülmektedir. Bunu öğrenen yapay zekâ sonrasında bir yorum üzerinde çalıştığında aynı şekilde davranacak ve olumlu ise 1'e yakın bir değer olumsuz ise 0'a yakın bir değer verecektir.

```
target = dataset['Rating'].values.tolist()

data = dataset['Review'].values.tolist()
```

Yukarıda verilen sayılar ve yorumlar ayrı dizilere alınmaktadır. Dataset değişkeni dizi olduğundan rating kolonu verilerek kolon seçilmektedir. Sonrasında values ile değerleri çekilmektedir. Tolist() ile listeye dönüştürülmektedir. Aynı işlemler review kolunu içinde yapılmaktadır. Datasete göre bu çalışmalar değişiklik göstermektedir.

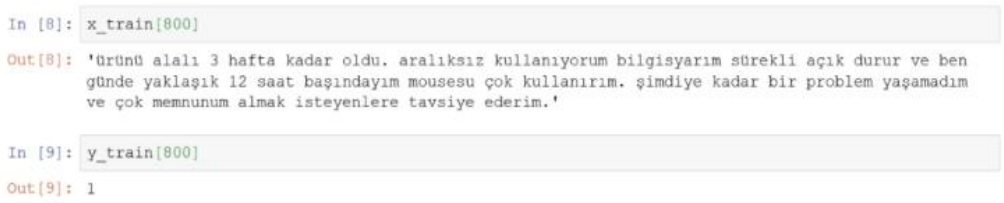
Bir yapay zekâ eğitilirken veriler test verisi olarak verilmektedir. Eğitim tamamlandıktan sonra eğitim test set üzerinden yapay zekâ test edilmektedir. Böylelikle yapay zekânın gerçekten doğru sonuçlar verip vermediği görülmektedir. Verilerin sonuçları yanlış çıktığı durumlarda eğitim başarısız olmuştur. Bunu test etmek için bulunan veri setinin bir kısmını test bir kısmını öğretmek amaçlı kullanılmaktadır. Verilerin %80'lik kısmı eğitime %20'lik kısmı teste ayrılmaktadır.

```
cutoff = int(len(data) * 0.80)

x_training, testing_x = content[:cutoff], content[cutoff:]

y_training, testing_y = result[:cutoff], result[cutoff:]

printf(x_ training[800])

printf(y_ training[800])
```

Yukarıda bulunan kod parçacığında verileri %80 ve %20 olarak iki parçaya ayrılmaktadır. Cutoff değişkenine len(data) ile datasetin uzunluğunu alıp 0.80 ile çarpıp %80 kısmının kaç veri olduğunu bulmaktadır. %80'lik kısmını x_training ve testing_x olarak iki değişkene aktarmaktadır.

Sonrasında aynı işlemleri etiketler içinde uygulamaktadır. Bu kod parçacığı sonrasında veri %80 ve %20 olmak üzere iki parçaya ayrılmaktadır. Printf ile verilerin doğru bölünüp bölünmediğini ekrana verileri yazdırarak test edilmektedir.

Çıktı şu şekildedir;



Şekil 5.2. Eğitim sonuçları

Şekil 5.2'de kelimelere birer sayı atanması işlemi yapılmaktadır. O yüzden kelime haznesinde kaç kelime var bu belirtilmektedir. general_words değişkeni bu değeri tutmaktadır ve 10000 belirtilmektedir. Burada en sık geçen 10000 kelime alınacaktır.

Kelimelere bölme işlemi Tokenizer ile yapılmaktadır.

general_token = Tokenizer(general_words = general_words) ile kelimelere ayrılıp general_token değişkenine atanmaktadır. Sonrasında general_token.fit_on_texts(data) yaparak data içerisinde bulunan bütün kelimelere sayı atama işlemleri yapılmaktadır. Bu işlem biraz zaman almaktadır. printf(general_token.word_index) ile test edilmektedir. Çıktı da ekrana kelimeler ve kelimelere atanan değerler görüntülenmektedir.

Şekil 5.3. word_index çıktıları

Burada en çok bulunan sayı en başa gelecektir. Burada 10000 kelime sınırı bulunmaktadır.

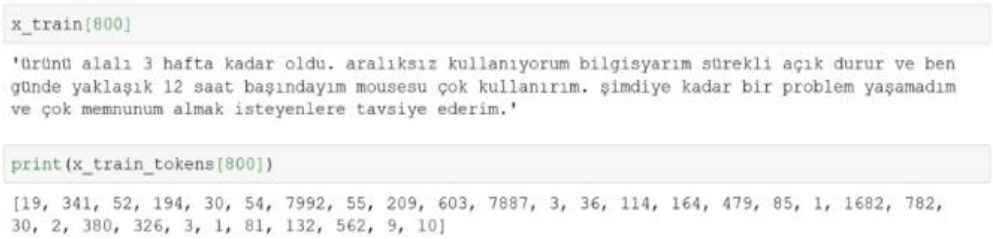
x_training_tokens = general_token.texts_to_sequences(x_training)

x_test_tokens = general_token.texts_to_sequences(testing_x)

printf(x_training [800])

printf(x_training_tokens[800])

Kelimeleri tokenleştirme işlemi uygulanmaktadır. Tokenleştirme işlemi; kelimeler yerine belirlenen sayıları tüm cümle içerisinde atama işlemi yapmaktır. “x_training_tokens = tokenizer.texts_to_sequences(x_training)” ile verilerin %80’inin atandığı verileri tokenleştirme işlemi yapılmaktadır. “x_test_tokens = general_token.texts_to_sequences(testing_x)” ile verilerin %20’sinin atandığı verileri tokenleştirme işlemi yapılmaktadır. “printf(x_training [800])” ve “printf(x_training_tokens[800])” ile tokenleştirme işleminin doğru çalıştığı test edilmektedir.



Şekil 5.4. x_train çıktıları

Kelime limiti 10000 olarak belirtildiği için bazı kelimelerin sayısal karşılığı olmayabilmektedir. Bunun nedeni kelimenin fazla tekrarının olmamasıdır. Burada “bilgisyarım” kelimesinin yazım hatasından dolayı karşılığı bulunmayacaktır.

Sinir ağlarında belirli boyutlarda bulunan verilerle çalışılmaktadır. Dataset içerisinde bulunan verilerin boyutları sabit değildir. Bu yüzden dataset içerisinde bulunan verilerin boyutları sabitlenmektedir.

num_tokens = [len(tokens) for tokens in x_training_tokens + x_test_tokens]

num_tokens = np.array(num_tokens)

max_tokens = np.mean(num_tokens) + 2 * np.std(num_tokens)

max_tokens = int(max_tokens)

np.sum(num_tokens < max_tokens) / len(num_tokens)

x_train_pad = pad_sequences(x_training_tokens, maxlen=max_tokens)

x_test_pad = pad_sequences(x_test_tokens, maxlen=max_tokens)

Veriler belirli bir boyuta sabitlenmektedir.

“max_tokens = np.mean(num_tokens) + 2 * np.std(num_tokens)” ile standart sapmasını almaktayız.

“max_tokens = int(max_tokens)” ile tam sayıya yuvarlamaktayız.

”x_train_pad=pad_sequences(x_training_tokens,maxlen=max_tokens” ile verileri sabit boyuta çekmekteyiz. Bütün verileri belirlenen boyuta sabitlemektedir.

Aynı şekilde test verileri içinde uygulamaktadır.

“x_test_pad = pad_sequences(x_test_tokens, maxlen=max_tokens)” kodu ile test verileri de sabitlenmektedir.

Tokenleri görmekteyiz ancak kelimelerin sayısal olarak tokenleri bulunmaktadır. Bunu tam tersine çeviren bir fonksiyon geliştirilmektedir.

```
idx = tokenizer.word_index

inverse_map = dict(zip(idx.values(),idx.keys()))
```

“idx = tokenizer.word_index” burada

“inverse_map = dict(zip(idx.values(),idx.keys()))” kod parçası ile ters çevirme işlemi yapılmaktadır. Bir sayı verildiğinde kelime karşılığı elde edilmektedir.

```
def tokens_to_string(tokens):
    words = [inverse_map[token] for token in tokens if token != 0]
    text = ' '.join(words)
    return text
```

def tokens_to_string(tokens) fonksiyonu oluşturulmaktadır. Bu fonksiyon ile gönderilen sayı dizisini aralarında boşluk olan kelime dizisine çevirmektedir.

words = [inverse_map[token] for token in tokens if token != 0] kod parçacığı ile sayıların hepsi sırası ile karşılığı olan kelimeye eşlenmektedir. text = ' '.join(words) ile kelimeler birleştirilmekte cümle oluşturulmaktadır.

“return text” ile oluşturulan kelime geri döndürülmektedir.

```
generate_model_object = Sequential()
embedding_size = 50
generate_model_object.add(Embedding(input_dim=num_words,output_dim=
embedding_size,input_length=max_tokens,name='embedding_layer'))
generate_model_object.add(CuDNNGRU(units=16,  return_sequences=True))
generate_model_object.add(CuDNNGRU(units=8, return_sequences=True))
generate_model_object.add(CuDNNGRU(units=4))
generate_model_object.add(Dense(1,activation='sigmoid'))
optimizer_work = Adam(lr=1e-3)
generate_model_object.compile(loss='binary_crossentropy',optimizer_work=
optimizer_work,metrics=['accuracy'])
generate_model_object.summary()
generate_model_object.fit(x_train_pad, y_training, epochs=5,
batch_size=256)
```

Öncelikle eğitim modeli oluşturulmaktadır. Ardışık bir model oluşturmak için “generate_model_object = Sequential()” kod parçacığı kullanılmaktadır. “embedding_size = 50” matrisin her kelimeye karşılık gelen uzunluğunu belirtmektedir.
“generate_model_object.add(Embedding(input_dim=num_words,output_dim =embedding_size,input_length=max_tokens,name='embedding_layer'))” modele ekleme yapmaktadır. Rastgeleme ekleme yapmaktadır. Kelime sayıları, kelime vektör uzunluğu verilmektedir. Böylelikle 10.000 x 50 uzunluğunda bir matris oluşturulmaktadır. 3. parametre gelen giriş değerlerinin uzunluğunu belirtmektedir. Sonrasında kullanmak için son parametre kısmında bir isim verilmektedir.

Sonrasında yinelenen sinir ağı oluşturulmaktadır. Yinelenen sinir ağı oluşturmak için

“generate_model_object.add(CuDNNGRU(units=16,return_sequences=True))” kod parçacığı kullanılmaktadır. GRU fonksiyonu CPU ile işlem yapmaktadır. CuDNNGRU fonksiyonu Nvidia ekran kartı ile

desteklenmektedir. CPU ile yavaş çalışmaktadır. Bu yüzden CuDNNGRU fonksiyonu kullanılmaktadır. “units=16” kod parçacığı ile kaç nöron oluşturulacağı belirtilmektedir. “return_sequences=True” kod parçacığı kaç çıkış değeri olacağını belirtmektedir. “TRUE” olur ise tek bir çıkış değerine sahip olmaktadır. “generate_model_object.add(CuDNNGRU(units=8, return_sequences=True))” kod parçacığı ile 16 lık sinir ağında gelen değeri 8 nörona indirgemektedir. Sonrasında “generate_model_object.add(CuDNNGRU(units=4))” kod parçacığı ile 4 nöron oluşturmakta ve tek çıkış değerine indirgemektedir. “generate_model_object.add(Dense(1,activation='sigmoid'))” kod parçacığı ile dense katmanı oluşmakta ve tek nörondan oluşmaktadır.

“sigmoid” 1 veya 0 değerini döndüreceğini belirten parametredir.

Optimize etmek için optimizasyon algoritması kullanılmaktadır. “optimizer_work = Adam(lr=1e-3)” kod parçacığı ile optimizasyon algoritması eklenmektedir.
“generate_model_object.compile(loss='binary_crossentropy',optimizer_work= optimizer_work,metrics=['accuracy'])” kod parçacığı ile derlemeye hazır hale gelmektedir.

Sonrasında “generate_model_object.summary()” kod parçacığı ile eğitime hazır hale gelmektedir.

“generate_model_object.fit(x_train_pad, y_training, epochs=5, batch_size=256)” kod parçacığı ile eğitim başlatılmaktadır. Sonrasında model hazır hale gelmektedir.

result = generate_model.evaluate(x_test_pad, testing_y)

y_pred = generate_model.predict(x=x_test_pad[0:1000])

y_pred = y_pred.T[0]

cls_pred = np.array([1.0 if p>0.5 else 0.0 for p in y_pred])

```
cls_true = np.array(y_test[0:1000])

incorrect = np.where(cls_pred != cls_true)

incorrect = incorrect[0]
```

“result = generate_model_object.evaluate(x_test_pad, y_test)” kod parçacığı oluşturulan modeli test etmektedir. Sonuçları “result” değişkenine aktarmaktadır.

“y_pred = generate_model_object.predict(x=x_test_pad[0:1000])“ kod parçacığı ile modelin ilk 1000 elemanını test etmektedir.

“y_pred = y_pred.T[0]” kod parçacığı ile sütun elemanlarını satır bazında çevirme yapmaktadır.

“cls_pred = np.array([1.0 if p>0.5 else 0.0 for p in y_pred])” kod parçacığı ile küsürlü değerleri 1 ve 0 lardan oluşan değerlere yuvarlamaktadır.

“cls_true = np.array(y_test[0:1000])” kod parçacığı ile test etmek için ilk 1000 elemanın sonuçlarını almaktadır.

“incorrect = np.where(cls_pred != cls_true)” kod parçacığı ile ilk değerleri ve model içerisinde işlendikten sonra elde edilen değerleri karşılaştırmaktadır.

“incorrect = incorrect[0]” yanlış sonuçları dönmektedir. Test sonuçlarında olumlu bir sonuç alınmaktadır. Şimdi Twitter’dan elde edilen verilerden örnekleri test edilmektedir.

```
texts = ["mükemmel","bu parti çok iyi","böyle parti mi olur"]

tokensCreate = tokenizer.texts_to_sequences(texts)

tokens_pad = pad_sequnces(tokensCreate, maxlen=max_tokens)

tokens_pad.shape
```

print(generate_model.predict(tokens_pad))

“texts = ["mükemmel","bu parti çok iyi","böyle parti mi olur"]” kod parçacığı ile Twitter’dan gelen verileri bir diziye aktarmaktadır.

“tokensCreate = tokenizer.texts_to_sequences(texts)” kelimelerin token değerlerini yani sayı değerlerini atamaktadır.

“tokens_pad = pad_sequnces(tokensCreate , maxlen=max_tokens)” cümleleri limitler arasına almaktadır.

“tokens_pad.shape” kod parçacığı ile model işlemesine hazır hale getirmektedir.

“print(generate_model_object.predict(tokens_pad))” kod parçacığı ile cümleleri işlemekte ve değerleri dönmektedir. “mükemmel” kelimesi için 0.955432, “bu parti çok iyi” cümlesi için 0.93443 değerini, “böyle parti mi olur” cümlesi için 0.43233 değerlerini dönmektedir. 0.5 üzeri için olumlu 0.5 değerinin altında kalan değerler için yapılan yorumların olumsuz olduğu belirtilmektedir. Sonuçlara göre yapay zekâ doğru çalışmaktadır.

Yapay zekâ hazır bulunmaktadır. Şimdi Twitter Stream API ile birlikte Twitter’dan anlık verilen elde edilip yapay zekâ ile işlenmektedir. Bunun için öncelikle verilerin çekilmesi gerekmektedir.

5.2 Twitter Yorumların Elde Edilmesi

Twitter verilerini elde etmek için Twitter’ın sunduğu Twitter Stream API kullanılmaktadır. Belirli sayıda bulunan tweetleri ücretsiz sunmaktadır. Daha fazlası için ücretlendirmesi bulunmaktadır. Ücretsiz kısmı yeterli olmaktadır.

Elde edilen verilerin kaydedilmesi gerekmektedir. Bu kaydetme işlemi sırasında hızlı bir işlem sağlanmayacağından kuyruk yapılarının kullanılması ve veri kaybının olmaması gerektiği belirtilmektedir. Kuyruk yapısı olarak bu

tez çalışmasında Kafka kullanılmaktadır. Kafka'nın çalışması için Zookeper gereklidir. Zookeper ve Kafka'yı sanal makine yerine konteyner yapısında çalıştırmak daha az kaynak gerektirdiğinden konteyner yapısı seçilmektedir. Konteyner yapısında çalışması için Docker teknolojisi kullanılmaktadır. Öncelikle docker file yazılmaktadır.

```
 version: '2'
services:
  kafka:
    image: wurstmeister/kafka
    ports:
      - "90922:90922"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: 192.168.20.103
      KAFKA_ADVERTISED_PORT: "90922"
      KAFKA_CREATE_TOPICS: "tweets:1:1,tweets_test:1:1"
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:21811
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "21811:21811"
```

Yukarıda bulunan kod parçacığı Kafka ve Zookeper konteyner'larını çalıştırmaktadır. "version: '2' " kod parçacığı kod yapısı versiyonunu belirtmektedir. Versiyon çeşitliliklerine göre farklı yazım şekilleri bulunmaktadır. Bu tez çalışmasında versiyon 2 seçilmektedir. "services:" kod parçacığı servisleri belirtmektedir. Bu tez çalışmasında iki servis yeterli gelmektedir. "zookeeper:" ve "kafka:" iki servis tanımlanmaktadır. "image: wurstmeister/zookeeper" zookeper için bir imaj seçilmektedir. Hazırlanan imaj yapıları DockerHub sitesine gönderilmektedir. Sonrasında "image:

wurstmeister/zookeeper” gibi yazarak “wurstmeister/zookeeper” imaj yapısını indirmekte ve çalıştırmaktadır.

“ ports:

- "21811:21811" ”

Kod parçacığı Zookeper servisinin hangi portu kullanacağını belirtmektedir.

Aynı şekilde Kafka servisinin özellikleri tanımlanmaktadır. Kafka servisi “image: wurstmeister/kafka” imaj yapısını kullanmaktadır.

“ports:

- "90922:90922" ”

Kod parçacığı Zookeper servisinin hangi portu kullanacağını belirtmektedir.

“environment:

KAFKA_ADVERTISED_HOST_NAME: 192.168.20.103

KAFKA_ADVERTISED_PORT: "90922"

KAFKA_CREATE_TOPICS: "tweets:1:1,tweets_test:1:1"

KAFKA_ZOOKEEPER_CONNECT: zookeeper:21811”

Kod parçacığı Kafka'nın kullanacağı Zookeeper gibi Kafka'nın kullanacağı parametreler tanımlanmaktadır.

Docker file tamamlandıktan sonra Docker ile çalıştırılmaktadır. Console ekranında

“docker-compose –f {docker file adı} up” komutu girilerek konteyner’ların çalışması sağlanmaktadır.

Kafka ve Zookeeper hazır bulunduktan sonra Twitter Stream API ile verileri çekmeye hazır bulunmaktadır.

Kullanılan kütüphaneler şunlardır;

İmport …

import …

import …

Yukarıda görünen kod parçacığı kullanılan kütüphaneleri eklemektedir. “İmport java.util.Properties;” kütüphanesi Java’da sistem bilgilerini alma işlemlerini yapmaktadır.

“import java.util.concurrent” kütüphanesi threadleri kontrol etme işlemlerini yapmaktadır. Genellikle paralel programlama işlemlerinde kullanılmaktadır.

“import com.google.common.collect” kütüphanesi yukarıda belirtilen eylemleri veri koleksiyonları üzerinde yapmaya yaramaktadır. Java Collection Framework sayesinde API1 arayüzleri öğrenmeden istediğimiz işlemleri gerçekleştirmektedir.

“import com.twitter.hbc” kütüphanesi Twitter API yapılarını kullanmak için gerekmektedir.

“import kafka” kütüphanesi Kafka ile haberleşme işlemlerini yapmaktadır.

Twitter uygulaması için Twitter API Key kullanılmaktadır.

Örnek key;

```
String consumerMyKey = "9GkEMR8SEzmDndbKFgUSLGPDU";
String consumerMySecret =
"pcQrh8T3F5y2vBhjsKXS7cuEqdBCGH9ADha8QvK19NAzRBL0SA";
String myToken = "790611908973985792-
9TJs07YhBy2qz0b2llLwZpSBzPGz0C2";
String mySecret =
"wNVQqLTBwuhPreNCIucayqrmQlyrXyP5QKAjdfxN8UE7Y";
```

Twitter API kaydı sonucunda Twitter (5.14) de bulunan keyleri göndermektedir.

```
KeyedMessage<String, String> message = null;
BlockingQueue<String> myQueue = new
LinkedBlockingQueue<String>(9000);
StatusesFilterEndpoint myEndpoint = new StatusesFilterEndpoint();
```

“KeyedMessage” Kafka ile haberleşme için kullanılmaktadır.

“BlockingQueue” arayüzü kuyruk işlemleri yapmaktadır, kuyruk doluysa veya boşsa beklememizi sağlayan metotlar sağlamaktadır. Yani producer/consumer işlemleri için kullanılmaktadır. Klasik bir producer/consumer örüntüsü için put() ve take() metotları kullanılmaktadır.

“StatusesFilterEndpoint” sınıfı gelen Twitter verilerinden gerekli tagları ayırma işlemlerini yapmaktadır.

```
myEndpoint.trackTerms(Lists.newArrayList("twitterapi", "#turkey"));
Authentication myAuth = new OAuth1(consumerMyKey, consumerMySecret,
myToken, mySecret);
```

“endpoint” içerisine çekilen yorumlarda hangi etiketleri ve hangi kelimeleri içermesini belirtmektedir.

“Authentication” sınıfı Twitter API bağlantısını sağlamak için kullanılmaktadır. Twitter API den alınan keyler içerisine tanımlanmaktadır.

```
Client myClient = new ClientBuilder()
        .hosts(Constants.STREAM_HOST)
        .endpoint(myEndpoint)
        .authentication(myAuth)
        .processor(new StringDelimitedProcessor(myAueue))
        .build();
myClient.connect();
```

Yukarıda bulunan kod parçacığı auth ile Twitter’a bağlantı sağlamakta, endpoint içerisinde bulunan kelime ve etiketler ile filtreleme yapmaktadır. Sonrasında gelen verileri queue içerisine aktarmaktadır.

“client.connect();” ile client çalışmaktadır.

```
for (int read = 0; read < 1000; read++) {
   try {
       String msg = myQueue.take();
       System.out.println(msg);
       myMessage = new KeyedMessage<String, String>(topic,
myQueue.take());
   } catch (Exception a) {
      System.out.println(“Error ”+a);
   }
   myProducer.send(myMessage);
```

```
}
myProducer.close();
myClient.stop();
```

Yukarıda da bulunan kod parçacığında en uzun gelecek veri tahmin edilemeyeceğinden 1000 e kadar örnek döngü sağlanmaktadır.

“String msg = myQueue.take();” kod parçacığı ile gelen verinin bir parçası alınmaktadır.

“System.out.println(msg);” kod parçacığı ile gelen veriyi console kısmına yazmaktadır.

“message = new KeyedMessage<String, String>(topic, myQueue.take());” kod parçacığı Kafka ile haberleşme için gerekmektedir. “myProducer.send(message);” kod parçacığı Kafka’ya veriyi göndermektedir. “myProducer.close();” ile açılan haberleşmeyi kapatmaktadır. “myClient.stop();” ile açılan Twitter haberleşmesini kapatmaktadır.

```
Properties myProps = new Properties();
myProps.put("metadata.broker.list", "192.168.20.103:90922");
myProps.put("serializer.class", "kafka.serializer.StringEncoder");
ProducerConfig myProducerConfig = new ProducerConfig(props);
Producer<String, String> myProducer = new Producer<String,
String>( myProducerConfig);
try {
   TwitterStreamProducer.PushTwittermessage(myProducer);
} catch (Exception a) {
```

```
    System.out.println((“Error ”+a);
}
```

“Properties myProps = new Properties();” ile gerekli veriler ayarlanmaktadır.

“myProps.put("metadata.broker.list", "192.168.1.103:9092");” Kafka ile haberleşmek için gerekli ayarlar ayarlanmaktadır. “myProps.put("serializer.class", "kafka.serializer.StringEncoder");” veriyi hangi sınıf kullanılarak parçalanacağı belirlenmektedir.

“ProducerConfig myProducerConfig = new ProducerConfig(myProps);” belirlenen ayarlar eklenmektedir.

“Producer<String, String> myProducer = new Producer<String, String>(myProducerConfig);” Kafka ile iletişim ayarları yapılandırılmaktadır. “TwitterStreamProducer.PushTwittermessage(myProducer);” ile Twitter veri çekme işlemini başlatmaktadır.

“try” ve “catch (Exception e)” kod parçacığı bir hata olduğu zaman alınan hatayı belirtmek için kullanılmaktadır.

Bu uygulama ile veriler elde edilmektedir. Sonrasında yapay zekâya gönderilmekte ve tespit edilen tweetler olumlu veya olumsuz olmak üzere istatistiksel veriler elde edilmektedir.

Yapı tamamlandıktan sonra deneklerden yorum alınmaktadır. Bu yorumların geliştirilen yapay zeka ile analizleri gerçekleştirilmiş ve doğruluk oranları belirtilmiştir.

Denek Yorumları;

- x partisi kadrolaşma yaratmak için sadece kendi yandaşlarını işe aldı. (0)
- x partisi ülkenin sağlık sorunlarını göz ardı ediyor. (0)
- x partisi belediye giderlerini ve fuzuli masrafları azalttı. (1)
- x partisi seçim öncesinde bulunduğu vaadlerin hiç birini tutmadı. (0)
- x partisi ülkenin savunma sorunlarını göz ardı ediyor. (0)
- x partisi milli gorus dışında bir şey demiyor. (0)
- x partisinin basindaki adami vasifsiz goruyorlar. (0)
- x partisi ülkenin işsizlik sorunlarını göz ardı ediyor. (0)

Benzer yorumlardan 50 adet yorum alınmıştır. Yanlarında olumlu ise 1 olumsuz ise 0 verileri bulunmaktadır. Yapay zeka analizleri sonucu doğruluk oranları belirlenmiştir.

Yorumların analizleri sonucunda %82 oranında doğruluk belirlenmiştir. Yapay zekanın yanlış yorumladığı yorumlar öğretildiğinde ve tekrar analiz edildiğinde %100 oranına çıkmıştır. Yanlış bildiği yorumlarda öğretilerek çok yüksek doğruluk oranlarına ulaşılmaktadır.

6. SONUÇ

Teknolojinin geldiği son noktaya bakıldığında sosyal medya kullanımı artmaktadır. Kullanımın artması sonucunda kullanıcıların düşünceleri rahat bir şekilde internet üzerinden erişilebilir olmaktadır.

Kullanıcıların rahatlıkla düşüncelerini yazdığı Twitter gibi sosyal medyalar oluşmaktadır. Twitter'da kullanıcıların her konu hakkında birçok yorumları bulunmaktadır. Bu yorumları analiz ederek birçok gerekli bilgi elde edilmektedir.

Bu tez çalışmasında istatistik ve anket firmalarının yaptığı işlemleri internet ortamına taşınması işlenmiştir. Kullanıcıların yorumlarını filtreleyip sonrasında verileri yapay zekâ ile olumlu olumsuz olarak ayırma işlemleri yaptıktan sonra bir istatistik elde etmek için kullanılmaktadır. Twitter'dan siyasi yorumları belirli anahtar kelime ve etiketler vererek elde edilmektedir. Sonrasında bu yorumların olumlu mu yoksa olumsuz mu olduğunu elde etmek için yapay zekâ ile analiz işlemleri yapılmaktadır. Sonrasında hangi partilerin veya insanların siyasal açıdan daha önde olduğu görülmektedir.

Bir siyaset tartışması olan programın başlangıcı ve sonuna kadar Twitter'dan veriler elde edilmektedir. Sonrasında bu elde edilen veriler yapay zekâ ile analiz edilmektedir. Elde edilen analiz sonuçlarına bakılarak konuşmacının konuştuğu konuları belirli bir kitlenin olumlu veya olumsuz gördüğü elde edilmektedir.

Gelecekte sosyal medya kullanımı çok fazla olduğunda daha fazla veri elde edilir olacaktır. Fazla veri daha iyi sonuçları elde etmeyi sağlamaktadır. Siyasi partilerin anket firmalarına ödedikleri ücret ve ayırdıkları zamana baktığımızda bu çalışmanın önemini göstermektedir. Bu çalışma siyasi alanda kullanıcılara çok yarar sağlayacaktır.

KAYNAKLAR

[1] AKTAN, E., Büyük Veri: Uygulama Alanları, Analitiği ve Güvenlik Boyutu. Bilgi Yönetimi, 1(1), 1–22, 2018.

[2] Talha KILIÇ, Big Data Kaç Gigabyte ?. Günümüzde son derece değerli ve popüler… | by Talha KILIÇ | Deep Learning Türkiye | Medium. https://medium.com/deep-learning-turkiye/big-data-kaç-gigabyte-d664cb78d582 (Erişim tarihi: 01.01.2021)

[3] MySQL - Vikipedi, https://tr.wikipedia.org/wiki/MySQL (Erişim tarihi: 01.01.2021)

[4] James Hamilton, One Size Does Not Fit All – Perspectives. https://perspectives.mvdirona.com/2009/11/one-size-does-not-fit-all (Erişim tarihi: 01.01.2021)

[5] Azzam, A. Z., Hegab, B., Khalaf, H., Al Bahili, H., Mohammed, H., Kamel, Y., Khail, F. A., Al-Hamoudi, W., Al Sofayan, M., & Al Sebayel, M. Liver transplantation in patients with hepatocellular carcinoma: A single-center experience. Experimental and Clinical Transplantation, 9(5), 323–328, 2011.

[6] PostgreSQL Nedir? Özellikleri ve Avantajları | WM Aracı, https://wmaraci.com/nedir/postgresql (Erişim tarihi: 01.01.2021)

[7] HadoopDevVeri.com, http://devveri.com/hadoop-nedir (Erişim tarihi: 01.01.2021)

[8] Seker, S. E., YBS Ansiklopedi Doğal Dil İşleme(Natural Language Processing) 2015.

[9] Deng, L., & Yu, D., Deep learning: Methods and applications. Foundations and Trends in Signal Processing, 7(3–4), 197–387, 2013.

[10] Bengio, Y. Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1), 1–27, 2009.

[11] Song, H. A., & Lee, S. Y., Hierarchical representation using NMF. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8226 LNCS(PART 1), 466–473 2013.

[12] Yeh, C. W., & Chen, J. Der., Role of ligand conformation in the structural diversity of divalent complexes containing phosphinic amide ligand. Inorganic Chemistry Communications, 14(8), 1212–1216 2011.

[13] Lee, D., Lim, M., Park, H., Kang, Y., Park, J. S., Jang, G. J., & Kim, J. H., Long short-term memory recurrent neural network-based acoustic model using connectionist temporal classification on a large-scale training corpus. China Communications, 14(9), 23–31 2017.

[14] Li, X., & Wu, X., Long short-term memory based convolutional recurrent neural networks for large vocabulary speech recognition. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2015-Janua, 3219–3223 2015.

[15] Miljanovic, M., Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction. Indian Journal of Computer Science and Engineering (IJCSE), 3(1), 180–191 2012.

[16] Seker, S. E., Duygu Analizi (Sentimental Analysis). YBS Ansiklopedi, 3(3), 21–36 2016.

[17] Everything There Is to Know about Sentiment Analysis, https://monkeylearn.com/sentiment-analysis (Erişim tarihi: 01.01.2021)

InfoWorld review: Top Java programming tools | InfoWorld, https://www.infoworld.com/article/2683534/infoworld-review--top-java-programming-tools.html (Erişim tarihi: 01.01.2021)

[18] Jupyter / Python Nedir, Nasıl Kurulur? · Güngör Budak, https://www.gungorbudak.com/blog/2018/03/31/jupyter-python-nedir-nasil-kurulur (Erişim tarihi: 01.01.2021)

[19] JetBrains Developer Tools, https://www.infoq.com/articles/jetbrains-developer-tools (Erişim tarihi: 01.01.2021)

[20] Company - JetBrains, https://www.jetbrains.com/company (Erişim tarihi: 01.01.2021)