T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞİM SİSTEMLERİNDE AKILLI OLAY ANALİZİ VE YÖNETİMİ

ATAHAN DUMAN
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
SİBER GÜVENLİK PROGRAMI

GEBZE
2021

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞİM SİSTEMLERİNDE AKILLI OLAY ANALİZİ VE YÖNETİMİ

ATAHAN DUMAN
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
SİBER GÜVENLİK PROGRAMI

DANIŞMANI
PROF. DR. İBRAHİM SOĞUKPINAR

GEBZE
2021

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SMART EVENT ANALYSIS AND MANAGEMENT IN INFORMATION SYSTEMS

ATAHAN DUMAN

A THESIS SUBMITTED FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER ENGINEERING

CYBER SECURITY PROGRAM

THESIS SUPERVISOR

PROF. DR. İBRAHİM SOĞUKPINAR

GEBZE

2021

GEBZE TEKNİK ÜNİVERSİTESİ	**YÜKSEK LİSANS JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 08/07/2021 tarih ve 2021/32 sayılı kararıyla oluşturulan jüri tarafından 29/07/2021 tarihinde tez savunma sınavı yapılan Atahan Duman'ın tez çalışması Bilgisayar Mühendisliği Anabilim Dalı Siber Güvenlik Programında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) : İbrahim SOĞUKPINAR

ÜYE : Yakup GENÇ

ÜYE : A. Gökhan YAVUZ

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
..…/……/…… tarih ve ……/…..... sayılı kararı.

ÖZET

Bilişim sistemleri ve uygulamaları hayatın her aşamasında vazgeçilmez hizmetler sunarak faaliyetlerimizi daha etkin ve verimli olarak gerçekleştirmemizi sağlamaktadır. Günümüzde bilgi teknolojileri sistemleri birçok alarm ve olay kayıtları üretmektedir. Üretilen bu kayıtlar çoğu zaman birbirleriyle bir ilişkiye sahiptir ve bu ilişki doğru bir şekilde yakalandığı zaman kurumları zarara uğratacak birçok kesinti henüz oluşmadan engellenebilecektir. Örneğin bir sunucunun disk G/Ç hızında ortaya gelen artış ya da sorun o sunucu üzerinde çalışan iş yazılımlarının yavaşlamasına bu yavaşlıkta farklı birtakım sonuçlara neden olabilecektir. İşte bir kurumun tüm olay kayıtlarını doğru bir şekilde analiz ederek yönetmesi, oluşan kayıtların belirli zaman dilimlerinde ve belirli kurallara bağlı olarak analizlerini kural bazlı yapmak ortaya çıkan milyonlarca alarmın verimli ve etkin bir şekilde yönetilmesini sağlayacaktır. Ayrıca olayların birbirleriyle ilişkilerini çıkartarak olası problemlerin önlenmesi mümkün olabilecektir. BT sistemlerinde meydana gelen olaylar bir çeşit ayak izi özelliği taşımaktadır. Söz konusu olayların kaydının tutulması da hayati önem taşımakta ve gerektiğinde bu olay kayıtlarının analizi yapılarak sistemlerin verimliliği, zararlı birtakım girişimler, sistem arıza eğilimi vs. gibi istenmeyen durumların anlaşılarak gerekli önlemlerin alınması ile olası kayıpların önlenmesi sağlanabilmektedir. Bu çalışmada bilişim sistemlerinde olay kaydı analizi yaparak sistemlerde arıza tahmini için geliştirilen model açıklanarak elde edilen deneysel sonuçlar verilmiştir.

Anahtar Kelimeler: Derin Öğrenme, Olay Kaydı, Siber Güvenlik, Sinir Ağları, LSTM, GRU, RNN, Dikkat Mekanizması.

SUMMARY

Information systems and applications provide indispensable services at every stage of life, enabling us to carry out our activities more effectively and efficiently. Today, information technology systems produce many alarm and event records. These produced records often have a relationship with each other, and when this relationship is captured correctly, many interruptions that will harm institutions can be prevented before they occur. For example, an increase in the disk I/O speed of a server or a problem may cause the business software running on that server to slow down and cause different results in this slowness. Here, an institution's accurate analysis and management of all event records, and rule-based analysis of the resulting records in certain time periods and depending on certain rules will ensure efficient and effective management of millions of alarms. In addition, it will be possible to prevent possible problems by removing the relationships between events. Events that occur in IT systems are a kind of footprint. It is also vital to keep a record of the events in question, and when necessary, these event records can be analyzed to analyze the efficiency of the systems, harmful interferences, system failure tendency, etc. By understanding the undesirable situations such as taking the necessary precautions, possible losses can be prevented. In this study, the model developed for fault prediction in systems by performing event log analysis in information systems is explained and the experimental results obtained are given.

Key Words: Deep Learning, Event Log, Cyber Security, Neural Networks, LSTM, GRU, RNN, Attention Mechanism.

TEŞEKKÜR

Başta, yüksek lisans eğitimimde ve akademik hayatımda desteğini ve yardımlarını hiçbir zaman esirgemeyip bilgisi ile bu çalışmanın oluşmasının yolunu açan danışmanım Prof. Dr. İbrahim Soğukpınar'a,

tüm çalışmada kullandığım ve araç takımları, programlama dilleri ve kaynak kodları paylaşan açık kaynak kod savunucusu herkese,

bu çalışmamın her anında yanımda olan ve beni destekleyen sevgili eşim Şeydanur Ahi Duman'a

ve göstermiş olduğu desteklerinden dolayı sevgili aileme en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

	Sayfa
ÖZET	v
SUMMARY	vi
TEŞEKKÜR	vii
İÇİNDEKİLER	viii
SİMGELER ve KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	x
TABLOLAR DİZİNİ	xi
1. GİRİŞ	1
1.1. Tezin Amacı, Katkısı ve İçeriği	2
2. KURAMSAL TEMELLER VE İLGİLİ ÇALIŞMALAR	3
2.1. Olay Kaydı	3
2.1.1. Olay Kaydı Toplama	4
2.1.2. Olay Kaydı Ayrıştırma	5
2.1.2.1. Olay Kaydı Kümeleme	6
2.1.2.2. Örüntü Tanıma	7
2.1.3. Olay Kaydı Özellik Çıkarımı	8
2.1.3.1. Sabit Pencere Yöntemi	9
2.1.3.2. Kayan Pencere Yöntemi	10
2.1.3.3. Anahtar Penceresi Yöntemi	10
2.1.4. Olay Kaydı Analizi	11
2.1.5. Olay Kaydı Korelasyon Yapısı	12
2.2. Yapay Zekâ	14
2.3. Makine Öğrenimi	15
2.3.1. Denetimli Öğrenme	15
2.3.1.1. Naive Bayes	16
2.3.1.2. Destek Vektör Makinesi	16
2.3.1.3. Doğrusal Regresyon	17
2.3.1.4. Lojistik Regresyon	17
2.3.1.5. Rastgele Orman	18

2.3.1.6. Karar Ağacı 18
2.3.2. Denetimsiz Öğrenme 18
2.3.2.1. K-Means Algoritması 19
2.3.2.2. Kümeleme 19
2.3.2.3. Temel Bileşen Analizi 20
2.3.3. Yarı Denetimli Öğrenme 21
2.3.4. Takviyeli Öğrenme 21
2.4. Derin Öğrenme 22
2.4.1. Yapay Sinir Ağları 22
2.4.2. Evrişimsel Sinir Ağları 23
2.4.3. Tekrarlayan Sinir Ağları 24
2.4.3.1. Geçitli Tekrarlayan Ünite 25
2.4.3.2. Uzun Kısa Süreli Bellek 26
2.4.4. Dikkat Mekanizması 27
2.5. İlgili Çalışmalar 28
3. GELİŞTİRİLEN YÖNTEM 31
4. DENEY SONUÇLARI VE DEĞERLENDİRME 34
4.1. Olay Kaydı Veri Kümesi 34
4.2. Deney Ortamı 34
4.3. Değerlendirme Ölçütleri 35
4.4. Test Sonuçları ve Değerlendirme 37
5. SONUÇLAR VE GELECEK ÇALIŞMALAR 38

KAYNAKLAR 39
ÖZGEÇMİŞ 42

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		**Açıklamalar**
CNN	:	Evrişimsel Sinir Ağı
FC	:	Tamamen bağlı
FP	:	Yanlış Pozitif
GBDT	:	Gradyan Artırıcı Karar Ağaçları
GPU	:	Grafik İşlemci Birimi
GRU	:	Geçitli Tekrarlayan Ünite
GTÜ	:	Gebze Teknik Üniversitesi
HDFS	:	Olay Kaydı Kümeleme Ağacı
LCS	:	En Uzun Ortak Sıra
LR	:	Lojistik regresyon
LSTM	:	Uzun Kısa Süreli Bellek
MLP	:	Çok katmanlı algılayıcı
OPTICS	:	Sıralama Noktaları Kümeleme Yapısı
PCA	:	Temel Bileşen Analizi
RNN	:	Tekrarlayan Sinir Ağı
SVM	:	Destek Vektör Makinesi
TF-IDF	:	Terim frekansı-ters metin frekansı
TN	:	Gerçek Negatif
TP	:	Gerçek Pozitif
UPGMA	:	Ağırlıksız Çift Grup Yöntem
YSA	:	Yapay Sinir Ağı

ŞEKİLLER DİZİNİ

Şekil No:		Sayfa
2.1:	Örnek bir olay kaydının yapısı.	3
2.2:	HDFS olay kaydı örnekleri.	4
2.3	Hiyerarşik kümelemede dendrogram gösterimi.	7
2.4	Örüntü tanıma işleminin görselleştirilmesi.	8
2.5	Pencere yöntemlerinin gösterimi.	9
2.6	Yaklaşan bir arızanın zaman çizelgesinde gösterimi.	11
2.7	Korelasyon motorunun genel çalışma prensibi.	13
2.8	Temel Bileşen Analizi yöntemi gösterimi.	20
2.9	Yapay Sinir Ağının görselleştirilmesi.	23
2.10	Tekrarlayan sinir ağının yapısı.	24
2.11	Uzun Kısa Süreli Bellek ağının mimarisi.	26
3.1	Spell aracılığıyla çıkarılan şablonların gösterimi.	32
3.2	Geliştirilen yöntemlerin genel mimarisi.	32

TABLOLAR DİZİNİ

Tablo No:		**Sayfa**
4.1:	Veri Kümesinin Dağılımı.	34
4.2:	Sınıflandırma karmaşıklık matrisi.	35
4.3:	Değerlendirme ve karşılaştırma tablosu.	36

1. GİRİŞ

Günümüzde bilgi teknolojileri sistemleri birçok alarm ve olay kayıtları üretmektedir. Olay kayıtları bu sistemler üzerinde gerçekleştirilen hesaplamalar hakkında ayrıntılı bilgileri kaydetmektedir. Üretilen bu kayıtlar çoğu zaman birbirleriyle bir ilişkiye sahiptir ve bu ilişki doğru bir şekilde yakalandığı zaman kurumları zarara uğratacak birçok kesinti veya saldırı henüz oluşmadan engellenebilecektir. Bu nedenle şirket ve devlet kurumlarının büyük bir yüzdesi bu kayıtların toplanmasına ve doğru bir şekilde analiz edilmesine özen göstermeye çalışmaktadırlar. Söz konusu olayların kaydının tutulması da hayati önem taşımaktadır. Gerektiğinde bu olay kayıtlarının analizi yapılarak sistemlerin verimliliği, zararlı birtakım girişimler, sistem arıza eğilimi vs. gibi istenmeyen durumların anlaşılarak gerekli önlemlerin alınması ile olası kayıpların önlenmesi sağlanabilmektedir. Örneğin, akıllı ev ve şehir projelerine başlanmasıyla birlikte oluşan ihtiyaç, nesnelerin interneti ve artan sayıda kentsel hizmetler (örneğin ulaşım, acil durum ve acil durum dışı müdahale vb.) çeşitli bilgi teknolojileri sistemlerine daha bağımlı hale gelmektedir. Açıkça görülmelidir ki bu sistemlerin çalışmama süresinin en aza indirilmesi, toplumun işlevi için son derece önemlidir (örneğin, acil durum telefon hatlarının kontrolü ve yönetimi sisteminin veya metro yolunu ve planlamasını kontrol eden sistemin çökmesi). Bir veya birden fazla sistem arızasından kaynaklanan bir kesintinin büyük ölçekli etkilerini gösteren çok sayıda benzer örnek vardır. Bu bilişim sistemleri, siber uzayda çeşitli kötü niyetli siber saldırılara karşı da savunmasızdır. Bu nedenle bu sistemlerde anormal olayları zamanında tespit etmek, sistemleri korumanın ilk adımıdır. Devamında bu anormal olayların önlenmesi ve tekrar gerçekleşmemesi için gerekli koruyucu önlemlerin alınması gerekmektedir. Günümüzde yapılan çalışmalarda, olay kayıtlarından anormal olayları tanımlamak için birçok geleneksel makine öğrenimi ve derin öğrenme modelleri önerilmektedir. Bu yaklaşımlar, olay kaydı mesajlarından anomalileri tespit edebilmek için önemli özellikler çıkararak, olay kaydı verilerini analiz etmek ve anormal olayları tespit etmek için çeşitli algoritmalar kullanmaktadır. Olay kayıtları birçok farklı sistemden çok çeşitli yapılarda toplandığı için olay kayıtlarının tutulduğu veri kümelerinin heterojen ve dengesiz olmalarına sebep olmaktadır. Bu nedenle, anormal olay kaydı dizilerini tespit etmek için dengesiz veri kümesiyle bir ikili sınıflandırıcı geliştirmek yanıltıcı

sonuçlar verebilir. Tek sınıflı destek vektör makineleri gibi denetimli veya temel bileşen analizi gibi birçok denetimsiz öğrenme modeli anormallikleri tespit etmek için yaygın olarak kullanılmaktadır. Ancak, tek sınıflı destek vektör makineleri gibi geleneksel makine öğrenimi modellerinin, ayrı olay kaydı mesajlarının zamansal bilgilerini yakalaması zordur. Bu zamansal bilgileri yakalamak için son zamanlarda, derin öğrenme modelleri, özellikle tekrarlayan sinir ağları, sıralı verileri modelleyebilme kabiliyetleri sebebiyle olay kayıtlarında anomali tespiti amacıyla yaygın olarak kullanılmaktadır. Kurumsal organizasyonlarda kullanımı giderek aratan bilişim sistemlerinin ürettikleri olay kaydı ve alarmlar arasındaki ilişkileri çıkararak sistemlerin etkin yönetimi hayati önem kazanmaktadır. Bu konuda yapılan birçok çalışma olmasına rağmen önemini ve güncelliğini korumaktadır.

1.1. Tezin Amacı, Katkısı ve İçeriği

Bilişim ve teknoloji sistemleri gündelik yaşantının hemen hemen her alanında akışın vazgeçilmez bir parçası haline gelmiştir. Bu bilişim sistemlerin ürettiği olay kayıtları gerçekleşen bütün olayların bir nevi ayak izleridir. Bu olay kayıtları üzerinden yapılan analizler hayati önem taşımaktadır. Bu tezin amacı bilişim sistemlerinde olay kayıtlarının analizini temel alarak, çeşitli derin öğrenme yöntemlerinin bu olay kayıtlarında anomali tespitinde kullanılması için geliştirilmesidir. Bu anomali tespiti için LSTM ve GRU tekrarlayan sinir ağları kullanarak farklı tespit modelleri geliştirilerek performans karşılaştırması yapılmıştır.

Tezin geri kalan bölümleri şu şekilde düzenlenmiştir: 2. Bölüm, bilişim sistemlerinde olay kayıtlarının toplanması, saklanması ve analizi gibi konulara giriş ve kuramsal temellerden bahsedildikten sonra bu alanda yapılan ilgili çalışmaları özetlemektedir. Sonraki 3. Bölüm bilgi teknolojilerinde olay kayıtlarında anormallik tespiti yaparak sistemin arıza veya saldırı eğilimini tespit etmek için geliştirilmiş derin öğrenme modelleri hakkında detaylı bilgi verirken, ardından gelen Bölüm 4'te değerlendirme ve deney sonuçları vardır. Son olarak, Bölüm 5 sonuç ve önerilerdir.

2. KURAMSAL TEMELLER VE İLGİLİ ÇALIŞMALAR

Olay kayıtları artık bilişim ve teknoloji sistemlerinde yaygın olarak kullanılmaktadır. Bu sistemlerde her gün milyonlarca olay kaydı oluşturulduğundan bu olay kayıtlarının sınıflandırılması analizi, toplanması ve işlenmesi çok önemlidir. Olay kayıtlarının çoğu yapılandırılmamış bir şablondadır ve bu da olay kayıtlarının genel bir biçimde sınıflandırılmasını zorlaştırır. Bu çalışmada, olay kayıtlarında anomali tespiti için Uzun Kısa Süreli Bellek (LSTM) ve Geçitli Tekrarlayan Ünite (GRU) tekrarlayan sinir ağı modelleri ve Dikkat mekanizması ekleyerek oluşturulan çeşitli varyasyonlarıyla deneyler gerçekleştirilmiştir. Bu deneyler sonucunda oluşturulan tekrarlayan sinir ağı modellerinin performanslarının analizi ve karşılaştırılması yapılmıştır. Bu bölümde bu alandaki kuramsal temellerin tanımı ve bilgilendirmesi yapıldıktan sonra daha önce olay kayıtlarının analizi, toplanması ve sınıflandırılması gibi alanlarda yapılan çalışmaların literatür incelemesi yapılmıştır. Bu çalışma boyunca gerekli olacak tüm arka plan bilgisi bu bölümde verilmiştir.

2.1. Olay Kaydı

Olay Kaydı bilişim sistemlerinde gerçekleşen bir aksiyonun bilgilerini içeren kayıt dizisidir. Bu olay kayıtları gerçekleşen olayın zaman damgası, eğer aksiyon iki alıcı arasındaysa hedef ve kaynak bilgileri, üretici kimliği, alarm mesajı gibi çeşitli alanları bünyesinde barındırmaktadır. Şekil 2.1'de örnek bir olay kaydının yapısı gösterilmiştir.

```
< Event xmlns = \" http : / / schemas . microsoft . com / win / 2004 / 08 / events / event \" > < System > <
Provider Name = \" MSExchange ADAccess \" / > < EventID Qualifiers = \" 16388 \" > 2070 < / EventID > <
Level > 4 < / Level > < Task > 3 < / Task > < Keywords > 0x0080000000000000 < / Keywords > <
TimeCreated SystemTime = 2014 / 08 / 18 08 : 22 : 55 / > < EventRecordID > 148040 < / EventRecordID > <
Channel > Application < / Channel > < Computer > C1 .zzzz. xxx . co . jp < / Computer > < Security / > < /
System > < EventData > < Data > [ 0 ] w3wp . exe ( ) [ 1 ] 6708 [ 2 ] g1. zzzz. xxx. co . jp [ 3 ] 70 [ 4 ] Active
directory response Since the limitation of the transmission timeout is exceeded, the operation was
aborted [ 5 ] SendTimeOut < / Data > < Binary > < / Binary > < / EventData > < / Event >
```

Şekil 2.1: Örnek bir olay kaydının yapısı.

2.1.1. Olay Kaydı Toplama

Büyük ölçekli sistemler, her biri bir zaman damgası ve ne olduğunu gösteren bir olay kaydı mesajı içeren sistem durumlarını ve çalışma zamanı bilgilerini kaydetmek için rutin olarak olay kayıtları oluşturur. Bu değerli bilgiler birden fazla amaç için kullanılabilir (örneğin, anormallik tespiti) ve böylece daha sonraki kullanımlar için önce olay kayıtları toplanır. Örneğin, Şekil 2.2, Amazon EC2 platformundaki [1] HDFS olay kayıtlarından alınan 8 olay kaydı satırını gösterirken, sunum kolaylığı için burada bazı alanlar gösterilmemiştir.

1	**2008-11-09 20:55:54** PacketResponder 0 for block blk_321 terminating
2	**2008-11-09 20:55:54** Received block blk_321 of size 67108864 from /10.251.195.70
3	**2008-11-09 20:55:54** PacketResponder 2 for block blk_321 terminating
4	**2008-11-09 20:55:54** Received block blk_321 of size 67108864 from /10.251.126.5
5	**2008-11-09 21:56:50** 10.251.126.5:50010:Got exception while serving blk_321 to /10.251.127.243:
6	**2008-11-10 03:58:04** Verification succeeded for blk_321
7	**2008-11-10 10:36:37** Deleting block blk_321 file /mnt/hadoop/dfs/data/current/subdir1/blk_321
8	**2008-11-10 10:36:50** Deleting block blk_321 file /mnt/hadoop/dfs/data/current/subdir51/blk_321

Şekil 2.2: HDFS olay kaydı örnekleri.

Günümüzde büyük veri ve endüstri 4.0 gibi gelişmelerden sonra ortaya çok fazla teknolojik ürün ve gelişme çıkmıştır. Bu ürünlerin çoğunun olay kayıtlarının toplanması ve saklanması için birçok yöntem geliştirilmiştir. Hadoop gibi birçok dağıtık dosya sistemi de bu olay kayıtlarını saklamak ve gruplandırmak için kendi içlerinde yeni yöntemler geliştirmişlerdir. Bu dosya sistemi dağıtık ve bloklar halinde çalıştığı için her blok kendi içerisinde eşsiz blk_id değerlerine sahiptir. Bu değerler olay kayıtlarında analiz çalışmalarında çoğunlukla ayrıştırma işleminde kullanılmıştır. Olay kayıtlarını ayrıştırma analiz işlemlerinde en zorlu ön işlemlerdendir. Her sistemin kendine özgü farklı olay kaydı şablonları olduğu için bunları genel bir şekilde ayrıştırmak mümkün değildir.

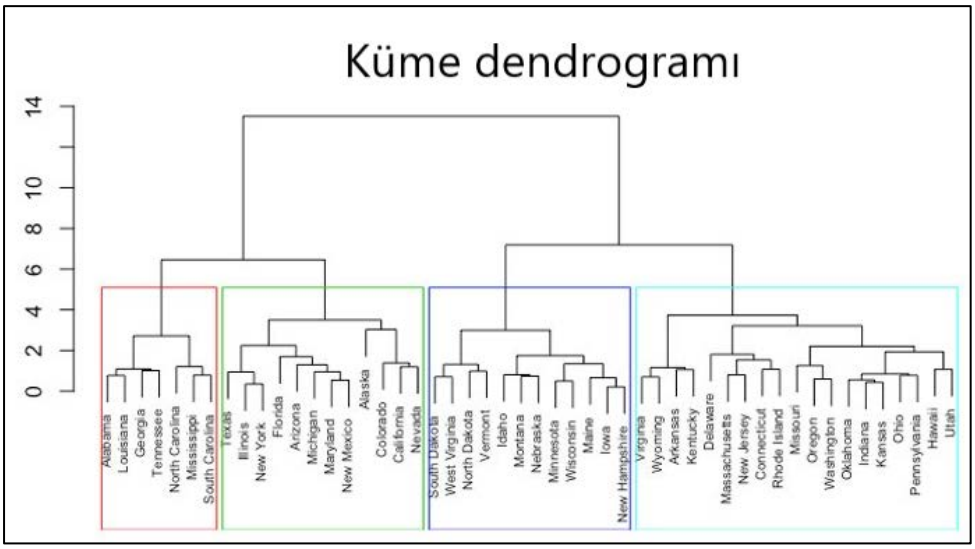
2.1.2. Olay Kaydı Ayrıştırma

Olay kayıtlarının geneli, serbest biçimde ve yapılandırılmamıştır. Olay kayıtlarını ayrıştırmanın amacı, ham olay kayıtlarının yapılandırılabileceği bir grup olay kaydı şablonu çıkarmaktır. Daha spesifik olarak, her bir olay kaydı mesajı, bazı spesifik parametrelerle (değişken kısım) bir olay kaydı şablonuna (sabit kısım) ayrıştırılabilir. Olay kayıtları genellikle çok sayıda uygulama veya işlevsellikten üretilir, bu nedenle son derece çeşitli ve çarpık bir kelime dağılımına sahiptir ve heterojen yapıdadır. Bu durum, geleneksel metin madenciliği yöntemlerini (ör. Konu modelleme, kelime torbası vb.) kullanarak olay kayıtlarından anlamlı özellikler elde etmeyi zorlaştırmaktadır. Ancak, olay kayıtları genellikle uygulamaların kaynak kodunda uygulamanın geliştiricilerinin belirlediği yapıda şablonlardan oluşturulur ve bu nedenle yapısı önceden tanımlanmıştır. Ayrıca, olay kayıtları genellikle uygulamaların temizlik amacıyla oluşturduğu veya bilgilendirme amaçlı oluşturduğu mesajlardan oluştuğu için anlamlı veri çıkarılamamaktadır. Benzer olay kayıtlarının bir kümesini temsil etmek için genel bir şablon bulmak, gereksiz olay kayıtlarının azaltılmasına aynı zamanda önemli olan olay kaydı bilgilerinin yakalanabilmesine olanak sağlamaktadır. Bu sayede olay kaydı mesajlarının anlamları daha kolay özetlenebilmektedir. Olay kayıtları öncelikle gruplara ayrılır ve her olay kaydı grubu için düzenli ifadeye dayalı bir şablon çıkarılır. Heterojen olay kayıtlarında ilk veri ön işleme adımı, olay kaydı verilerini, her olay kaydının mesajlarının sözcüklerinden veya tümceciklerinden daha düşük düzeyli bilgilerin tanımlanabileceği ve alınabileceği şekilde sayısal diziler haline getirilmesidir. Bununla birlikte, farklı uygulamalardan veya sistemlerden gelen heterojen olay kayıtlarının farklı formatları ve dolayısıyla farklı belirteçleri vardır. Spesifik bilgi veya insan incelemesi olmadan, heterojen olay kayıtlarının tamamı için önceden tanımlanmış herhangi bir belirteç kullanmak tüm olay kayıtlarında başarılı bir sonuç vermemektedir. Bu nedenle, çok genel bir belirteç kullanılmalıdır. Fakat bu tür bir sınırlayıcının kullanımı, olay kaydı verilerinden hiçbir bölümü değiştirmemeli ve olay kaydının taşıdığı önemli bilgiye müdahale etmemelidir. Genel bir sınırlayıcının kullanımı aynı zaman da düşük eşleştirme veya önem düzeyinin bağlamının yakalanmasında zorluklar ortaya çıkarmaktadır. Sayılar dışındaki tüm kelimeler ve özel semboller genelde tek boşluk karakteri ile ayrılmaktadır. Fakat güncel yöntemlerde bu ayrıştırma işleminde sıklıkla alt sekansların bulunması, kümeleme algoritmaları, karar ağaçları veya ikili ağaçlar ile

ayrıştırma, graflar kullanarak ayrıştırma gibi yöntemler bulunmaktadır. Bu ayrıştırma algoritmaları üzerine çeşitli performans karşılaştırma çalışmaları yapılmıştır. Bu olay kayıtları içerisinde, IP adresleri ve URL'ler gibi bazı birimler birden çok boş alana sahip olabilmektedir. Heterojen olay kayıtları, olay kaydı kümelemesini ve örüntü tanımayı zorlaştıracak birçok farklı zaman damgası biçimine sahip olabilir. Bu nedenle, olay kayıtlarında tüm zaman damgalarını algılamak ve bunları standart bir formata dönüştürmek ayrıştırma algoritmalarının başarılarını artırmaktadır.

2.1.2.1. Olay Kaydı Kümeleme

Kümeleme yöntemlerinde daha spesifik olarak, heterojen olay kayıtlarının hiyerarşik bir yapısını oluşturmak için hiyerarşik kümeleme algoritmaları kullanılır. Dendrogramlarda bunlardan biridir. Dendrogram bir ağacı temsil eden diyagramdır. Bu şekilde gösterim farklı bağlamlarda sıklıkla kullanılabilmektedir. Hiyerarşik kümelemenin tercih edilmesinin sebebi olay kayıtları dengesiz şekilde dağılmış olabilir. Verimlilik amacıyla hiyerarşik ağaç yapısı daha fazla tercih edilir. Genel olarak kullanılan hiyerarşik ağaç yapısı ve özellikle olay kayıtları için geliştirilmiş olan ağaç yapısı Olay Kaydı Kümeleme Ağacı (LCT) olarak isimlendirilir. LCT'de, hiyerarşik kümeleme algoritması Sıralama Noktaları Kümeleme Yapısını (OPTICS) [2] tanımlamak için uygulanmaktadır. OPTICS, belirli bir veri noktasından önceden tanımlanmış bir eşiğin altında yeterince yakın olan tüm komşu veri noktalarına doğru genişleyerek yoğun veri bölgelerini aramaktadır. Kümeleme algoritması ayrıca, veri noktası sıralamasından hiyerarşik bir kümeleme yapısı oluşturur. Burada daha gevşek bir veri bölgesi içindeki daha yoğun veri bölgesi, küçük bir küme olarak nitelendirilir ve daha gevşek bölgenin (yani kümelenen kısmın) daha düşük seviyeli bir alt öğesi haline gelmektedir. Kümeleme tabanlı olay kaydı ayrıştırıcılarda, önce olay kayıtları arasındaki mesafeler hesaplanır ve bir sonraki adımda olay kayıtlarını farklı kümelerde gruplamak için genellikle kümeleme teknikleri kullanılır. Son olarak, her kümeden olay şablonu oluşturulur. Sezgisel tabanlı yaklaşımlar için, her bir olay kaydı pozisyonundaki her kelimenin oluşumları sayılır. Ardından, etkinlik adayları olarak sık kullanılan kelimeler seçilir ve oluşturulur. Son olarak, günlük olayları olarak bazı adaylar seçilir.

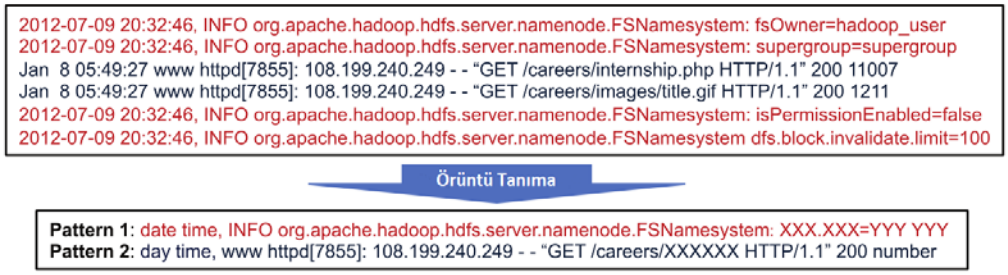


Şekil 2.3: Hiyerarşik kümelemede dendrogram gösterimi.

2.1.2.2. Örüntü Tanıma

Olay kaydı verileri kümelendikten sonra, tüm heterojen olay kayıtlarının genel yapısı oluşturulmaktadır. Fakat, yine de her bir kümeleme içerisinde daha ayrıntılı şablonlar elde edilmesi gerekmektedir. Her küme içerisinde, olay kayıtları çok benzer biçimlere sahip olduğundan gerçekleştirilen örüntü tanıma işlemi, biyoinformatik alanında yapılan araştırmalarda sıklıkla kullanılan dizi hizalama fikri kullanılarak seçilen olay kaydı kümesi içerisinde gerçekleştirilmektedir. Özellikle, bir kümedeki tüm olay kayıtları, en sık tekrar eden kısmı dolayısıyla olay kayıtlarından gelen ortak desenleri tanımlamak için birlikte hizalanmaktadır. Genellikle bu hizalama işleminde Smith-Waterman algoritması [3] ikili hizalama gerçekleştirmek için kullanılmaktadır. Bu işlemin ardından çoklu hizalama gerçekleştirebilmek için Aritmetik Ortalama (UPGMA) [11] stratejisi ile Ağırlıksız Çift Grup Yöntemi sıklıkla kullanılmaktadır. Örüntü tanıma işlemi öncelikle küme saflığının yüksek olduğu kısımlarda yapılmaktadır. Bu kısımlarda hizalama daha net bir şekilde görülebildiği yaprak düğümlerde yapılmaktadır. Bunun sebebi en çok benzeyen olay kayıtları yaprak düğümlerde daha sık bir şekilde hizalanmasıdır. Daha sonra örüntü bilgisinin yaprak düğümlerden kök düğüme doğru geri çoğaltma (backpropagation) işlemi yapılmaktadır. Olay kayıtlarından örüntüler oluşturulduktan sonra ayrıştırma aşamaları genelde tamamlanmaktadır. Ayrıştırma amacı analize gitmeden önce olay kayıtlarının kümelenebilir, kolaylıkla parçalanabilir veya anlamlandırılabilir duruma

getirilmesidir. Ayrıştırma işlemi tamamlandıktan sonra girdi olarak verilen heterojen bir olay kaydı bahsedilen ayrıştırma yöntemleriyle örüntülere ayrıştırılır. Oluşan örüntü yapısına göre olay kaydı daha önce bulunan örüntüler ile eşleştirilmektedir. Eğer eşleşmezse farklı bir grupta temsil edilmelidir. Olay kayıtlarında ayrıştırma ve örüntü tanıma işlemlerinde oluşan zorluklardan bir tanesi de hiçbir gruba veya kümeye uymayan, örüntü yakalanamayan olay kayıtlarıdır. Bu olay kayıtlarının da temsil edildiği harici bir küme veya örüntü grubu oluşturulmalıdır. Çünkü sistemlerde bulunan ve hiçbir kalıba uymayan olay kayıtlarının anomali taşıma ihtimali yüksektir. Bu çalışmada da olay kaydı analizi ile anomali tespit etmeye yönelik modeller geliştirildiği için bu durumun ele alınması önemli hale gelmektedir.



Şekil 2.4: Örüntü tanıma işleminin görselleştirilmesi.

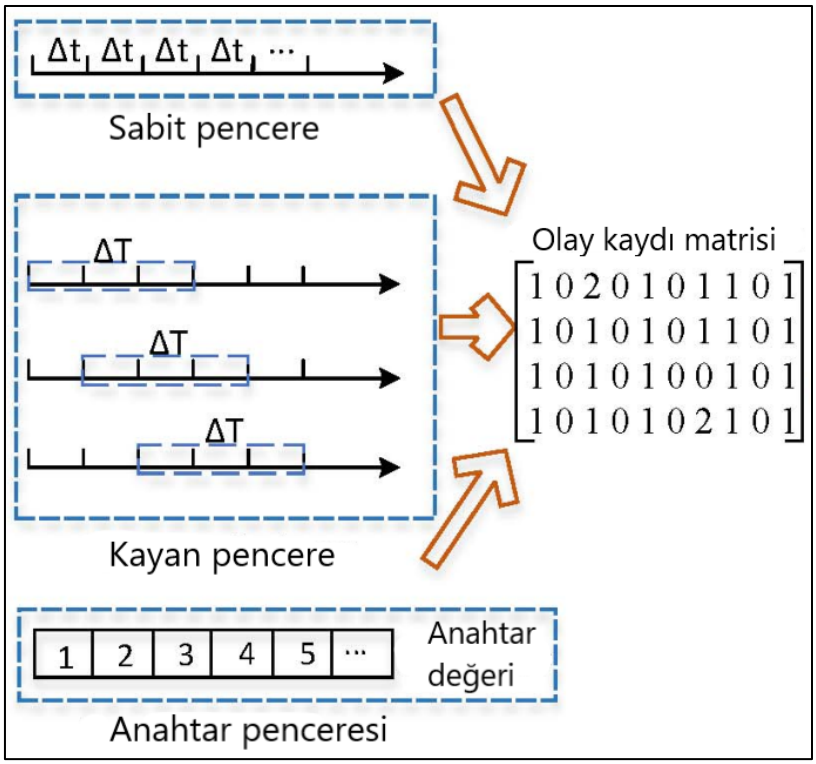
2.1.3. Olay Kaydı Özellik Çıkarımı

Özellik çıkarımının temel amacı, anomali tespit modellerine beslenebilecek olay kayıtlarından değerli özellikler çıkarmaktır. Özellik çıkarmanın girişi, olay kaydı ayrıştırma adımında oluşturulan olay kayıtlarıdır ve çıktı, bir olay kaydı anahtar dizisi matrisidir. Öznitelikleri çıkarmak için öncelikle olay kayıtlarını, her grubun bir olay kaydı dizisini temsil ettiği çeşitli gruplara ayırmamız gerekir Olay kayıtlarını ayrıştırdıktan sonra, bunları makine öğrenimi veya derin öğrenme modellerinin uygulanabileceği sayısal özellik vektörlerine kodlamamız gerekir. Bunu yapmak için birçok yöntem geliştirilmiştir. Bu kısımda bu yöntemlerden kısaca bahsedilecektir. Sabit pencereler, kayan pencereler ve oturum pencereleri gibi farklı gruplama teknikleri kullanarak ham olay kayıtları diziler haline dönüştürülebilmektedir. Ardından, her bir olay kaydı dizisi için, her olay kaydı için belirlenmiş özel anahtar değerleri temsil eden bir özellik vektörü (olay kaydı anahtar vektörü) oluşturulmaktadır. Tüm özellik vektörleri birlikte bir özellik matrisi, yani bir olay

kaydı anahtar matrisi oluşturabilmektedir. Bu yapısal yöntemlerle olay kayıtlarından özellik çıkarımı işlemleri yapılmaktadır. Bazı yöntemlerde zamansal bilgiler ve taşıdığı değerler göz önüne alınsa da bu değerlerin çeşitliliği ve öngörülemez olmaları nedeniyle kullanımı zorlaştırmaktadır. Karakter olarak ele alan yöntemler olduğu gibi kelime olarak ele alan yöntemlerde bulunmaktadır.

2.1.3.1.Sabit Pencere Yöntemi

Sabit pencere yöntemleri, her olay kaydının oluşum zamanını taşıyan zaman damgasını özelliğini temel almaktadır. Her sabit pencerenin kendi boyutu vardır, bu da zaman aralığı veya zaman süresi anlamına gelmektedir. Şekil 2.4' de gösterildiği gibi, pencere boyutu, bir saat veya bir gün gibi sabit bir değer olan Δt'dir. Bu değer zaman değişimini tutmaktadır.



Şekil 2.5: Pencere yöntemlerinin gösterimi.

Sabit pencerelerin sayısı önceden tanımlanmış pencere boyutuna bağlıdır. Aynı pencerede gerçekleşen olay kayıtları bir olay kaydı dizisi olarak kabul edilir. Sıra, sabit bir boyut penceresi tarafından dilimlenir. Bu yaklaşım, sabit bir zaman aralığında

kaydedilen olay kayıtları için uygundur, örn. saatlik veya günlük. Eğer veriler periyodik ise, sabit bir pencere kullanmak, her aralık zamanındaki davranışı modelleyebilir ve periyotlar arasındaki farkı karşılaştırabilir.

2.1.3.2.Kayan Pencere Yöntemi

Sabit pencerelerden farklı olarak, kayan pencereler iki özellikten oluşur: pencere boyutu ve adım boyutu, örneğin her beş dakikada bir kayan saatlik pencereler. Genel olarak, adım boyutu pencere boyutundan daha küçüktür, bu nedenle farklı pencerelerin çakışmasına neden olmaktadır. Şekil 2.4, pencere boyutunun ΔT olduğunu, adım boyutunun ise aktarım mesafesi olduğunu göstermektedir. Genellikle sabit pencerelerden daha büyük olan kayan pencerelerin sayısı, esas olarak hem pencere boyutuna hem de adım boyutuna bağlıdır. Aynı kayan pencerede meydana gelen olay kayıtları da bir olay kaydı dizisi olarak gruplanır, ancak olay kayıtları çakışma nedeniyle birden çok kayan pencerede tekrar edebilir. Sıra, sabit pencere boyutu ile dilimlenir ve sabit adım uzunluğu ile başka bir pencereye kaydırılır. Adım uzunluğu, pencere uzunluğundan daha kısa olabilir, bu, pencerelerin üst üste geldiği anlamına gelir. Her pencerenin sınırındaki olayların bilgisi, bitişik kaydırılan pencerelerinde tutulur.

2.1.3.3.Anahtar Penceresi Yöntemi

Diğer iki pencere yöntemiyle karşılaştırıldığında, anahtar pencereleri zaman damgası yerine özel anahtar değerlere dayanır. Bu anahtar değerler, bazı olay kaydı kümelerinde farklı yürütme yollarını işaretlemek için kullanılır. Örneğin, HDFS olay kayıtlarını blk_id değeri ile saklar, belirli bir bloğun tahsis edilmesini, yazılmasını, çoğaltılmasını, silinmesini kaydeder. Böylece olay kayıtları, her anahtar penceresinin benzersiz bir olay kaydı anahtarına sahip olduğu anahtarlara göre gruplayabiliriz.

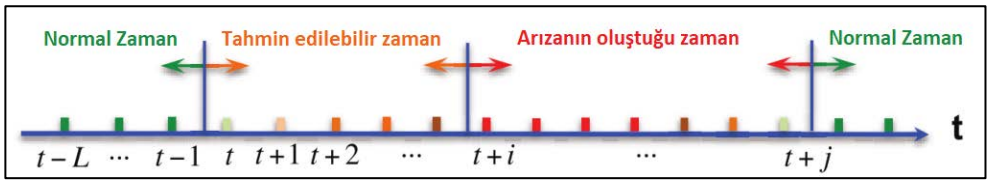
Bu pencere yöntemleri ile olay kaydı dizileri oluşturulduktan sonra, bir olay kaydı matrisi oluşturulur. Her bir olay kaydı dizisinde, olay kaydı anahtar dizi vektörünü oluşturmak için her bir olay kaydının oluşturulan anahtar değeri kullanılmaktadır. Örneğin, olay kaydı vektörü [0, 0, 2, 3, 0, 1, 0] ise, bu olay kaydı dizisinde olay kayı anahtar değeri 0 olan olay kaydı 1, 2, 5 ve 7. sırada gerçekleşmiş demektir.

2.1.4. Olay Kaydı Analizi

Bütün bu işlemler tamamlandıktan sonra olay kayıtlarının analizi yapılmaktadır. Bazı çalışmalarda olay kaydı madenciliği gibi farklı isimler verilse de aslında mevcut olay kayıtlarının istenilen duruma göre analiz edilmesidir. Bu durum bir anomali tespiti olabilir, saldırı tespiti olabilir veya daha önce gerçekleşmiş problemlerin analizi olabilir. Olay kayıtlarının analizi 3 ana başlığa odaklanmaktadır. Bu ana başlıklar;

- Adli Analiz
- Arıza Tespiti
- Sistem Hata Tahmini

Adli analiz, sistem olay kayıtlarının bir son durum analizidir ve başarısızlığın temel nedenini belirlemek amacıyla çalışır [5], [6]. Arıza tespiti kritik arıza bulgularını ortaya çıktıklarında hızlı bir şekilde tespit etmeyi amaçlamaktadır ve anomali tespiti olay kaydı analizi açısından ortak bir yöntemdir [7], [8]. Aksine, başarısızlık tahmini, potansiyel başarısızlıklar için erken uyarılar sağlamayı amaçlayan proaktif (yalnızca tepki vermek yerine gelecekteki bir durumdan önce hareket etmeyi içerir) bir yaklaşımdır. Şekil 2.5' de verilen örnek tahmin modeli tarihsel öznitelik vektörlerinin girdi dizisi göz önüne alındığında, yaklaşan bir arızanın olasılığını verir. Hesaplanan olasılık önceden tanımlanmış bir eşiği aşarsa, yakın gelecekte olası bir arızayı işaret etmek için erken uyarı verilir. Erken uyarı işareti tipik olarak zayıftır ve bu nedenle basit modeller kullanılarak yakalanması zordur.



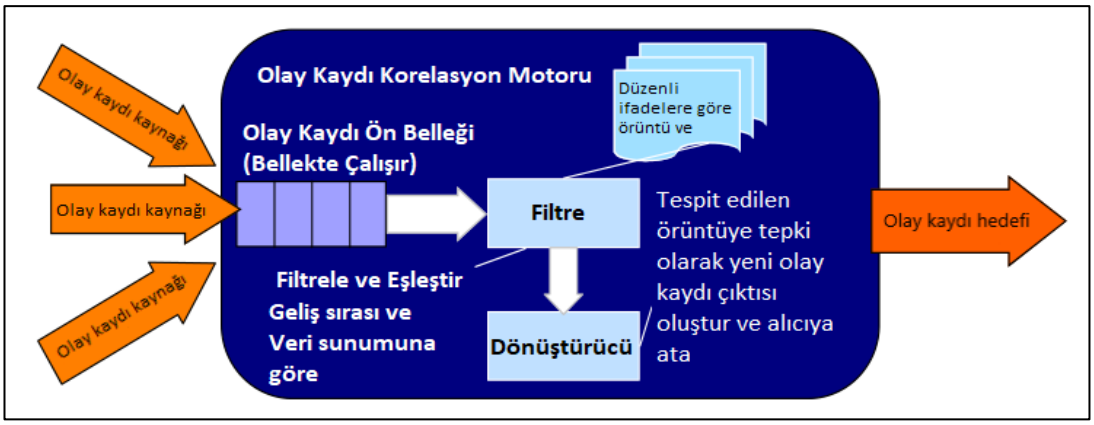
Şekil 2.6: Yaklaşan bir arızanın zaman çizelgesinde gösterimi.

Bir bilgisayar sisteminin zamansal dinamikleri göz önüne alındığında, ikincisinin durumları muhtemelen uzun bir tarihsel eğilime bağlıdır. Lojistik regresyon, SVM ve ağaç tabanlı sınıflandırıcılar gibi geleneksel denetimli öğrenme yöntemleri, bir girdi dizisini bağımsız özellikler olarak kabul eder ve bunlar arasındaki

zaman bağımlılıklarını yakalayamaz. Buna benzer arıza tahminindeki güçlü zamansal bağımlılıklar göz önüne alındığında tekrarlayan sinir ağları veya zamansal bağımlılıkları daha iyi yakalayabilen yöntemler kullanılması gerekmektedir.

2.1.5. Olay Kaydı Korelasyon Yapısı

Bugün sistem yöneticilerinin karşılaştığı sorun, dağıtılmış ve heterojen olay kayıtlarının entegrasyonu ve korelasyonudur. Tüm bu olay kayıtları bir tehdidi tanımlayabilir ve her gün yüzlerce veya binlerce kez tekrar edebilir. Bu olay kayıtlarının incelenmesi genellikle personel tarafından zaman ve gerekli bilgiden yoksun bir şekilde sınırlı kaynaklarla yapılmaktadır. Kaynak sınırlamalarına rağmen, hata ayıklamayı ve izinsiz giriş tespitini iyileştirdiği için olay kayıtlarının korelasyonunun yapılması birçok sorunu önceden görebilme veya otomatikleştirilmek istenen ve insan gücünden tasarruf edilmek istenen kısımlarda kullanılabilir. Bazı durumlar personelin gözünden kaçsa bile oluşturulmuş korelasyon kuralları bu bağıntıları otomatik olarak yakalayarak hata payını en aza indirgemektedir. Bugünün sistem yöneticilerinin karşılaştığı sorun, merkezi olmayan olay kayıtları platformlarına gelen milyonlarca olay kayıtlarının korelasyonudur [9]. Yorumlama süreci genellikle manuel, zaman alıcı, maliyetli ve yanlıştır. Bu dezavantajlara rağmen, şirketler olay kayıtlarını daha temiz bir şekilde yorumlayabilen yeni sistemlere daha fazla yatırım yapmaktadırlar. Sürekli sorgularla veri akışı işlemenin potansiyel faydaları ve uygulamaları, olay kaydı korelasyonu için ortaya çıkan bir ihtiyaç haline gelmektedir. Sürekli sorgular, gerçek zamanlı bir olay kaydı korelasyon sisteminde kullanmak için yararlı olabilmektedir [10]. Keskin (ve klasik) sürekli sorgu motorları çok güçlüdür ve akışları geçici disk alanlarına depoladığı için fazla esneklik sunmamaktadır. Hatta çok büyük bir olay kaydı dosyanın var olduğu (büyük olay akışları) varsayılabilir. Sonuç olarak, korelasyon tekniklerinde bu büyük olay kaydı kümeleri kolay bir şekilde kullanılamamaktadır. Şekil 2.7’ de genel bir korelasyon motorunun çalışma prensibi gösterilmektedir.



Şekil 2.7: Korelasyon motorunun genel çalışma prensibi.

Bir olay kaydı korelasyon sistemi oluşturmak veya mevcut bir sistemi kullanabilmek için aşağıdaki kuralların göz ardı edilmemesi gerekmektedir.

- Olay kaydı dosyaları genellikle dağıtık bir durumdadırlar ve çok büyük boyuttadırlar.
- Olay kaydı etkinlikleri yapısal olarak zayıftırlar ve yoğunlukla metin içerirler. Harici olarak birkaç alan içerirler (kimlik, kaynak, zaman damgası, açıklama, ...).
- Olay kaydı dosyaları genellikle heterojendir; bir olay belirlenirse, diğer ağ bileşenlerindeki olaylarla olası ilişkileri kontrol edebilmenin kolay bir yolu bulunmamaktadır.
- Olay kaydı dosyaları sürekli bir akış halindedir ve sürekli olarak olay kayıtları oluşmaktadır.
- Olay kaydı dosyaları genellikle farklı yapılara ve modellere sahip olaylar içerir.
- Olay kaydı dosyası girişleri farklı akış hızlarına sahip olabilir. 2 farklı uygulama veya donanım (örneğin bir ters proxy veya bir web uygulaması) tamamen farklı ekipler tarafından geliştirildiğinde, eski versiyon olan uygulamadan veya web servisten gelen olay kayıtlarının aşırı fazla olması veya yeni teknolojilerle 2. ekip tarafından geliştirilen ve kendi bünyesinde barındırdığı güçlü filtreleme yapısıyla daha az tekrar eden ve anlamlı olay kayıtları oluşturması olay kayıtlarının takibi ve ayrıştırma işlemlerini zorlaştırabilmektedir.
- Olay kaydı korelasyonu, genellikle uzmanlar veya bu alanı kontrol eden üçüncü bir departman tarafından oluşturulan kurallara dayanmaktadır [10].

Günümüzde olay kaydı toplama, izleme ve analiz faaliyetleri yapmak için kullanılan mevcut teknikler genellikle iki kategoriye ayrılmaktadır: merkezi ve dağıtılmış teknikler. Merkezi bir modelde, olay kaydı üreten makineler tipik olarak olay kaydı mesajlarını ağ üzerinden merkezi bir olay kaydı sunucusuna (veya daha sonra bunları ileten bir alt ağ seviyesi olay kaydı sunucusuna) göndermektedir. Olay kaydı korelasyonu veya olay kaydı analizi faaliyetleri gerçekleştirilirse, bunlar genellikle bir veri tabanı kullanılarak merkezi olay kaydı sunucusunda yapılmaktadır. Bu yaklaşımın olumlu ve olumsuz özellikleri vardır. Merkezi olarak yönetilmeleri kolaydır ve potansiyel bir saldırgan için muhtemelen daha küçük bir güvenlik açığı alanı sunmaktadır. Bununla birlikte, olay kaydı üreten makinelerin sayısı ve olay kaydı hacmi, olay kaydı sunucusu ve olay kaydı korelasyon motorlarının merkezi etkinlik ağ etki alanlarına dağıtıldığında bile olaylarla dolması riskini taşımaktadır [11]. Ek olarak, olayların hacmi, önemli hesaplama kaynaklarının (işlemci süresi, bellek, disk alanı) olay kaydı analizi ve olay kaydı korelasyonu faaliyetlerine ayrılmasını gerektirmektedir.

2.2. Yapay Zekâ

Yapay zekâ, insan zekâsının belli bir parçasını taklit etmeye çalışan ve insan zekâsının gerçekleştirmeye çalıştırdığı işlemleri mümkün olduğunda başarmaya çalışan sistemlerin genel bir tabiri olarak kullanılmaktadır. Buna rağmen, yapay zekâ terimi kullanıldığında insanın zekâsını tamamen taklit edebilen veya amacı bu olan sistemleri tam olarak ifade etmemektedir. Yapay zekâ, farklı alanlarda kullanılabilmektedir:

- Gündelik hayatta kullanılan ve bilişim sistemlerinde yazılanları tahmin ederek tamamlayan sistemler,
- Shazam isimli müzik arama motorunun girdi olarak verilen müziği bulması,
- Spotify müzik paylaşım platformunda bulunan müzik önerme sistemi,
- Facebook’ta bulunan arkadaş öneri sistemi sık karşılaşılan örneklerden birkaçıdır.

Fakat bu uygulamalar her zaman karmaşık ve derin olmak zorunda değildir: Rastgele oluşturulmuş bir sayı dizisini büyükten küçüğe sıralayabilen, X noktasından

Y noktasına giden en kısa yolu bulabilen, bulmaca çözen basit denilebilecek algoritmaları çözen veya üretilmesini sağlayan yapılara da yapay zekâ denilebilmektedir. Dikkat çeken nokta, otomatikleştirme yöntemi belirli kurallara veya örüntülere dayanmayan ve zekâ sahibi olmayan varlıkların/sistemlerin çözmelerinin beklenmediği problemleri çözebilmeleridir. Günümüzde yapay zekâ terimi, akademik mecrada kendi başına pek bir şey ifade etmemektedir; zira uğraşılan birçok problem, halihazırda yapay zekâ içermektedir.

2.3. Makine Öğrenimi

Makine öğrenmesi; çıktı değerlerini kabul edilebilir bir aralıkta tahmin etmek için girdi verilerini alan ve analiz eden programlanmış algoritmalar bütününe denilmektedir. Bu algoritmalar gönderilen yeni veriler ile performansı iyileştirmek ve zamanla 'zekâ' kavramını geliştirmek için operasyonları öğrenip, optimize etmektedirler. Diğer bir deyişle makine öğrenmesi, matematiksel ve istatistiksel işlemler ile veriler üzerinden çıkarımlar yaparak tahminlerde bulunan sistemlerin bilgisayarlar ile modellenmesidir. Makine öğrenmesi, diğer yapay zekâ uygulamalarından farklı olarak, bir yandan insan zekâsını taklit ederken, diğer yandan uzmanların yorumlayıp elle gireceği kurallara ihtiyaç duymayan algoritmalar bütünüdür. Nasıl ki bir insan zihni gördükleri ve duydukları ile kavramları kendi kendine öğreniyor, yorumluyor ve kalıcı hale getiriyorsa, harici olarak dış etkenler tarafından bu kuralların zihnine işlemesine ihtiyaç duymuyorsa, makine öğrenmesi uygulamaları da benzer bir şekilde, kendisine sunulan veri kümelerini özümseyerek yapılması istenen görevi öğrenmektedir. Makine öğrenmesi, her ne kadar yapay zekânın alt dalı olsa bile, içerisinde birçok farklı algoritma ve yöntem bulundurmaktadır.

2.3.1. Denetimli Öğrenme

Denetimli öğrenme, etiketlenmiş eğitim verilerinden bir model türetmeye yönelik bir makine öğrenimi görevi olarak tanımlanmaktadır. Etiketlerle normal veya anormal durumu gösteren etiketlenmiş eğitim verileri, denetimli anormallik tespitinin ön koşuludur. Eğitim verileri ne kadar doğru bir şekilde etiketlenirse, model o kadar kesin olmaktadır. Eğitim verisi hem girdilerden hem de çıktılardan oluşur. Bu

fonksiyon, sınıflandırma veya regresyon (eğri uydurma) algoritmaları ile belirlenebilmektedir. Veri kümesindeki çıkışlar kategorik olarak ayrılabiliyor ise sınıflandırma, nümerik ise regresyon, (eğri uydurma) algoritmaları kullanılmaktadır.

Denetimli öğrenme, sonuçları bilinen veri noktalarını kullanarak tahmine dayalı bir model belirleyen bir sistem ve algoritma sınıfını ifade eder. Model, bir kayıp veya hata fonksiyonunu en aza indirmek için tipik olarak bazı optimizasyon rutinleri aracılığıyla çalışan uygun bir öğrenme algoritması (doğrusal regresyon, rastgele ormanlar veya sinir ağları gibi) aracılığıyla eğitilerek öğrenilir. Denetimli öğrenme, bir modele girdi verilerini ve doğru çıktı verilerini besleyerek öğretme sürecidir. Bu giriş/çıkış çiftine genellikle "etiketli veri" olarak ifade edilmektedir. Doğru cevabı bilen, bir soruya verdiği yanıtın doğruluğuna bağlı olarak bir öğrenciye ya puan verecek ya da öğrenciden puan alacak bir öğretmen düşünülebilir. Denetimli öğrenme, genellikle iki tür sorun için makine öğrenimi modelleri oluşturmak için kullanılır. [12], [14].

2.3.1.1. Naive Bayes

Naive Bayes sınıflandırıcı, bağımsızlık varsayımıyla Bayes Teoremine dayalı bir sınıflandırma tekniğidir. Bir sınıftaki belirli bir özelliğin varlığının başka herhangi bir özelliğin varlığıyla ilgisi olmadığını varsayar. Örneğin, kırmızı, yuvarlak ve çapı yaklaşık 3 inç olan bir meyve elma olarak kabul edilebilir. Bu özellikler birbirine veya diğer özelliklerin varlığına bağlı olsa bile, tüm bu özellikler bağımsız olarak bu meyvenin bir elma olma olasılığına katkıda bulunur ve bu nedenle ‘Naive’ olarak bilinir. Naive Bayes modelinin oluşturulması kolaydır ve özellikle çok büyük veri kümeleri için kullanışlıdır. Sadeliğin yanı sıra, Naive Bayes'in son derece karmaşık sınıflandırma yöntemlerinden bile daha iyi performans gösterdiği bilinmektedir. Bayes teoremi, $P(c)$, $P(x)$ ve $P(x|c)$'den $P(c|x)$ sonsal olasılığını hesaplamanın bir yolunu sağlar.

2.3.1.2. Destek Vektör Makinesi

Destek Vektör Makinesi (SVM), sınıflandırma için denetimli bir öğrenme yöntemidir. SVM' de, yüksek boyutlu uzayda çeşitli örnek sınıflarını ayırmak için bir hiper düzlem oluşturulur. Hiper düzlemi bulmak, hiper düzlem ile farklı sınıflardaki

en yakın veri noktası arasındaki mesafeyi maksimize eden bir optimizasyon problemidir. Destek vektör makinesi algoritmaları, kategorize etme ve tepki incelemesi için kullanılan bilgileri inceleyen denetimi sağlanan öğrenme türleridir. Arızaları tespit etmek için SVM sıklıkla kullanılmaktadır. Lojistik Regresyon ve Karar Ağacına benzer şekilde, eğitim örnekleri etiketleriyle birlikte olay kaydı anahtarı vektörleridir. SVM aracılığıyla anormallik tespitinde, hiper düzlemin üzerinde yeni bir örnek bulunursa, aksi takdirde normal olarak işaretlenirken bu bir anormallik olarak rapor edilir. Doğrusal SVM ve doğrusal olmayan SVM olmak üzere iki tür SVM vardır. Yalnızca doğrusal SVM' yi tartışılmıştır, çünkü doğrusal SVM, yapılan deneylerin çoğunda doğrusal olmayan SVM' den daha iyi performans göstermektedir [14].

2.3.1.3. Doğrusal Regresyon

Doğrusal regresyon, gözlemlenen verilere doğrusal bir denklem uydurarak iki değişken arasındaki ilişkiyi modellemeye çalışır. Bir değişken açıklayıcı değişken, diğeri bağımlı değişken olarak kabul edilir. Örneğin, bir modelci, doğrusal bir regresyon modeli kullanarak bireylerin ağırlıklarını boylarıyla ilişkilendirmek isteyebilir. Gözlemlenen verilere doğrusal bir model uydurmaya çalışmadan önce, bir modelci öncelikle ilgilenilen değişkenler arasında bir ilişki olup olmadığını belirlemelidir. Bu mutlaka bir değişkenin diğerine neden olduğu anlamına gelmez ancak iki değişken arasında önemli bir ilişki vardır. Bir dağılım grafiği, iki değişken arasındaki ilişkinin gücünü belirlemede yardımcı bir araç olabilir. İki değişken arasındaki ilişkinin değerli bir sayısal ölçüsü, iki değişken için gözlemlenen verilerin ilişkisinin gücünü gösteren -1 ile 1 arasında bir değer olan korelasyon katsayısıdır.

2.3.1.4. Lojistik Regresyon

Lojistik regresyon, sınıflandırma için yaygın olarak kullanılan istatistiksel bir modeldir. Bir eş görünüm durumuna karar vermek için lojistik regresyon, tüm olası durumların (normal veya anormal) olasılığını p tahmin eder. Olasılık p, etiketlenmiş eğitim verileri üzerine kurulmuş bir lojistik fonksiyon tarafından hesaplanır. Yeni bir örnek göründüğünde, lojistik fonksiyon tüm olası durumların olasılığını p ($0 < p < 1$) hesaplayabilir. Olasılıklar elde edildikten sonra en büyük olasılığa sahip durumlar

sınıflandırma çıktısıdır. Lojistik regresyon, sağlanan önceki verilere dayanarak meydana gelen bir olayın olasılığını tahmin etmeye odaklanır. İkili bağımlı bir değişkeni kapatmak için kullanılır, yani sadece iki ve 0, 1 sonuçları gösterilmektedir. Anormallikleri saptamak için, her olay kaydı dizisinden bir olay kaydı anahtar vektörü oluşturulur ve her olay kaydı anahtar vektörü, etiketiyle birlikte bir örnek olarak adlandırılır. Örnek olarak, aslında bir lojistik fonksiyon olan lojistik regresyon modelini kurmak için eğitim örnekleri kullanıldığı düşünülürse model elde edildikten sonra, olasılık p anomalisini hesaplamak için lojistik fonksiyona bir test örneği X beslenir, X etiketi $p \geq 0,5$ olduğunda anormal ve aksi halde normal olarak işaretlenmektedir.

2.3.1.5. Rastgele Orman

Rastgele Orman veya Rastgele Karar Orman; sınıflandırma, regresyon ve diğer görevler için daha iyi sonuçlar üretmek üzere birden fazla algoritmayı birleştiren bir topluluk öğrenme yöntemidir. Her bir sınıflandırıcı zayıf, ancak başkalarıyla birleştiğinde, mükemmel sonuçlar üretebilmektedir. Algoritma bir "Rastgele Ormanlar" ile başlar (ağaç benzeri bir grafik veya karar modeli) ve en üste bir girdi girilir. Daha sonra ağaçtan aşağı doğru hareket eder, veriler belirli değişkenlere bağlı olarak daha küçük kümelere bölünerek yöntem uygulanmaktadır.

2.3.1.6. Karar Ağacı

Karar Ağacı, her bir örnek için tahmin edilen durumu göstermek için dalları kullanan bir ağaç yapısı diyagramıdır. Karar ağacı, eğitim verileri kullanılarak yukarıdan aşağıya bir şekilde oluşturulur. Her ağaç düğümü, özniteliğin bilgi kazancı tarafından seçilen mevcut "en iyi" özniteliği kullanılarak oluşturulur.

2.3.2. Denetimsiz Öğrenme

Denetimli yöntemlerden farklı olarak denetimsiz öğrenme, diğer bir yaygın makine öğrenimi görevidir ancak eğitim verileri etiketlenmemiştir. Denetimsiz yöntemler, etiketlerin olmaması nedeniyle gerçek dünya üretim ortamında daha uygulanabilir olmaktadır. Yaygın denetimsiz yaklaşımlar arasında çeşitli kümeleme

yöntemleri, birliktelik kuralı madenciliği, temel bileşen analizi vb. bulunur. Denetimsiz öğrenme, sınıflandırılmamış veya etiketlenmemiş veri noktalarını içeren veri kümelerindeki kalıpları tanımlamak için yapay zekâ algoritmalarının kullanımını ifade eder. Algoritmaların, bu görevi yerine getirirken herhangi bir dış rehberliğe sahip olmadan veri kümeleri içinde yer alan veri noktalarını sınıflandırmasına, etiketlemesine ve/veya gruplamasına izin verilir. Başka bir deyişle, denetimsiz öğrenme, sistemin veri kümeleri içindeki kalıpları kendi başına tanımlamasına olanak tanır. Denetimsiz öğrenmede, bir yapay zekâ sistemi, hiçbir kategori sağlanmasa bile, sıralanmamış bilgileri benzerliklere ve farklılıklara göre gruplayacaktır. Denetimsiz öğrenme algoritmaları, denetimli öğrenme sistemlerinden daha karmaşık işleme görevleri gerçekleştirebilir.

2.3.2.1. K-Means Algoritması

K-means algoritması, veri kümesini, her bir veri noktasının yalnızca bir gruba ait olduğu, K-tanımlı farklı örtüşmeyen alt gruplara (kümelere) bölmeye çalışan yinelemeli bir algoritmadır. Küme içi veri noktalarını mümkün olduğunca benzer hale getirmeye çalışırken, kümeleri mümkün olduğunca farklı (uzak) tutmaya çalışır. Veri noktaları ile kümenin ağırlık merkezi (o kümeye ait tüm veri noktalarının aritmetik ortalaması) arasındaki kare uzaklığın toplamı minimum olacak şekilde bir kümeye veri noktaları atar. Kümeler içinde ne kadar az varyasyona sahipsek, aynı küme içindeki veri noktaları o kadar homojen (benzer) olur.

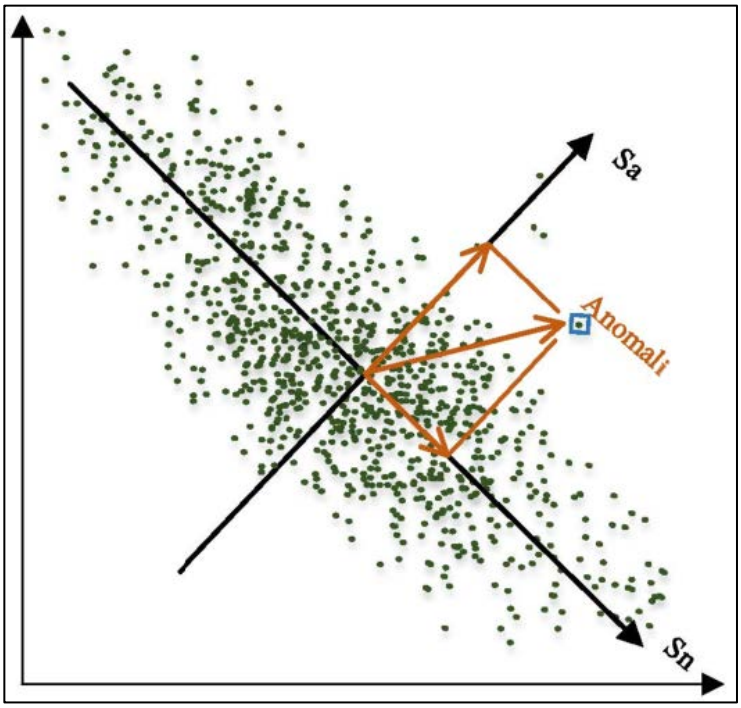
2.3.2.2. Kümeleme

Kümeleme temel olarak bir tür denetimsiz öğrenme yöntemidir. Denetimsiz öğrenme yöntemi, etiketli yanıtlar olmadan girdi verilerinden oluşan veri kümelerinden referanslar alınan bir yöntemdir. Genellikle, bir dizi örnekte bulunan anlamlı yapıyı, açıklayıcı temel süreçleri, üretken özellikleri ve gruplamaları bulmak için bir süreç olarak kullanılır. Kümeleme, popülasyonu veya veri noktalarını, aynı gruplardaki veri noktalarının aynı gruptaki diğer veri noktalarına daha benzer ve diğer gruplardaki veri noktalarından farklı olacak şekilde birkaç gruba ayırma görevidir. Temel olarak, aralarındaki benzerlik ve farklılık temelinde bir nesneler topluluğudur. Kümeleme, en önemli denetimsiz öğrenme sorunu olarak kabul edilebilir; bu nedenle,

bu tür diğer tüm problemler gibi, etiketlenmemiş veri koleksiyonunda bir yapı bulmakla ilgilenir. Kümelemenin gevşek bir tanımı, “nesneleri, üyeleri bir şekilde benzer olan gruplar halinde organize etme süreci” olabilir. Bu nedenle bir küme, aralarında "benzer" olan ve diğer kümelere ait olan nesnelere "benzer olmayan" nesnelerin bir koleksiyonudur.

2.3.2.3. Temel Bileşen Analizi

Temel Bileşen Analizi (PCA), boyut küçültme yapmak için yaygın olarak kullanılan istatistiksel bir yöntemdir. PCA' nın arkasındaki temel fikir, yüksek boyutlu verileri (örneğin, yüksek boyutlu noktalar), k'nin orijinal boyuttan daha az olarak ayarlandığı k ana bileşenden (yani, k boyutlarından) oluşan yeni bir koordinat sistemine yansıtmaktır. PCA, yüksek boyutlu veriler arasında en fazla varyansı yakalayan bileşenleri (yani eksenleri) bularak k temel bileşeni hesaplar. Böylece, PCA ile dönüştürülmüş düşük boyutlu veriler, orijinal yüksek boyutlu verilerin ana özelliklerini (örneğin, iki nokta arasındaki benzerlik) koruyabilir.



Şekil 2.8: Temel Bileşen Analizi yöntemi gösterimi.

Örneğin, Şekil 2.8’ de PCA, iki boyutlu noktaları tek boyutlu noktalara dönüştürmeye çalışır. Sn ana bileşen olarak seçilmiştir, çünkü noktalar arasındaki mesafe en iyi şekilde onları Sn ile eşleştirerek tanımlanabilir. Anormallik saptama yöntemlerinde, her günlük dizisi bir olay kaydı anahtar vektörü olarak vektörler haline getirilir. Bundan sonra, olay kaydı anahtar vektörlerinin boyutları arasındaki kalıpları bulmak için PCA kullanılır. PCA kullanılarak, normal uzay S_n ve anomali uzayı S_a olmak üzere iki alt uzay oluşturulur. S_n, ilk k ana bileşen tarafından oluşturulur ve S_n, kalan (n-k) tarafından oluşturulur; burada n, orijinal boyuttur. Ardından, bir olay kaydı anahtar vektörünün y'den S_a'ya izdüşümü $y_a=(1 - PP^T)y$ hesaplanır, burada $P = [v_1,v_2, \ldots, v_k,]$ ilk k ana bileşendir. y_a'nın uzunluğu bir eşikten büyükse, karşılık gelen olay sayım vektörü bir anormallik olarak rapor edilmektedir. Örneğin, Şekil 2.8' de seçilen nokta, S_a üzerindeki izdüşümünün uzunluğu çok büyük olduğu için bir anormallik olarak işaretlenir. Spesifik olarak, bir olay kaydı anahtar vektörü, bu durumlarda anomali olarak kabul edilmektedir [14].

2.3.3. Yarı Denetimli Öğrenme

Yarı denetimli öğrenme, bir tür makine öğrenimi yöntemidir. Yarı denetimli öğrenme, az sayıda etiketlenmiş örnek ve çok sayıda etiketlenmemiş örnek içeren bir öğrenme problemidir. Ne denetimli ne de denetimsiz öğrenme algoritmaları etiketli ve anlaşılmaz verilerin karışımlarını etkili bir şekilde kullanamadığından bu durumu kontrol altında tutamazlar. Bu nedenle, özel yarı denetimli öğrenme algoritmaları gereklidir. Bir modelin öğrenmesi ve yeni örnekler üzerinde tahminlerde bulunması gereken etiketli örneklerin küçük bir bölümünü ve çok sayıda etiketlenmemiş örneği içeren bir öğrenme problemini (ve öğrenme problemi için tasarlanmış algoritmaları) ifade etmektedir.

2.3.4. Takviyeli Öğrenme

Takviyeli öğrenme, bir tür makine öğrenimi yöntemidir. Takviyeli öğrenme, bir dizi karar vermek için makine öğrenimi modellerinin eğitimidir. Temsilci, belirsiz, potansiyel olarak karmaşık bir ortamda bir hedefe ulaşmayı öğrenir. Takviyeli öğrenmede, yapay zekâ oyun benzeri bir durumla karşı karşıyadır. Bilgisayar, soruna bir çözüm bulmak için deneme yanılma yöntemini kullanır. Makinenin programcının

istediğini yapmasını sağlamak için, yapay zekâ gerçekleştirdiği eylemler için ya ödül ya da ceza alır. Amacı, toplam ödülü en üst düzeye çıkarmaktır. Tasarımcı ödül politikasını, yani oyunun kurallarını belirlemesine rağmen, modele oyunun nasıl çözüleceğine dair hiçbir ipucu veya öneri vermez. Tamamen rastgele denemelerden başlayıp karmaşık taktikler ve insanüstü becerilerle bitirerek, ödülü en üst düzeye çıkarmak için görevin nasıl gerçekleştirileceğini bulmak modele bağlıdır. Aramanın gücünden ve birçok denemeden yararlanarak, pekiştirmeli öğrenme şu anda makinenin yaratıcılığını göstermenin en etkili yoludur. İnsanların aksine, yapay zekâ, yeterince güçlü bir bilgisayar altyapısı üzerinde bir takviye öğrenme algoritması çalıştırılırsa, binlerce paralel oyundan deneyim toplayabilir.

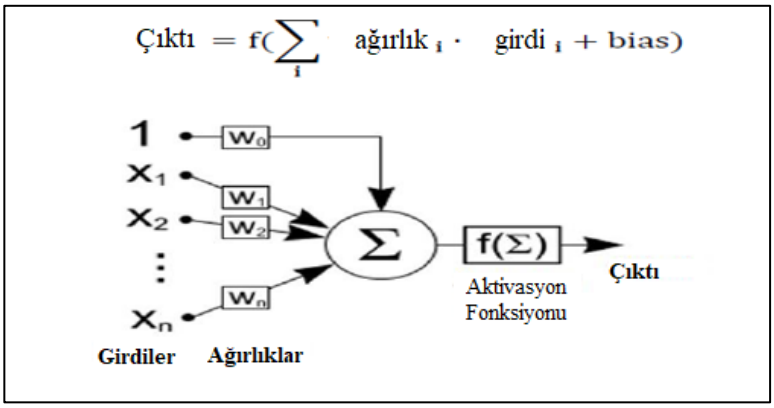
2.4. Derin Öğrenme

Derin öğrenme, çok katmanlı yapay sinir ağlarının geri yayılım isimli bir algoritma ile eğitilmesidir. Bu çok katmanlı yapılar farklılıklar göstermektedir. Görüntü işleme, doğal dil işleme gibi birçok alanda kullanılan derin öğrenme yöntemleri geleneksel makine öğrenmesi yöntemlerine göre daha karmaşık problemler üzerinde ciddi sonuçlar alabilmektedir.

2.4.1. Yapay Sinir Ağları

Yapay sinir ağları, insan beyninin çalışma yapısını taklit eden ve mevcut verileri analiz eden ve bu verilerden farklı öğrenme algoritmaları ile yeni bilgiler oluşturan bilgi işleme teknolojisidir. Yapay sinir ağlarının ilk temelleri, 1940'ların başında araştırmaya başlayan Mc Cullogh ve Pitts tarafından 1943'te yayınlanan bir makaleyle atılmıştır. Ancak XOR problemi olarak bilinen yapay sinir ağları literatürünün başarısız olması nedeniyle yapay sinir ağlarına olan ilgi belirli bir süreliğine azalmıştır. Hopfield'in 1980'lerde yaptığı çalışmalar, yapay sinir ağlarının genellenebileceğini ve geleneksel bilgisayar programlama ile çözülmesi zor olan problemler için kullanılabileceğini göstermektedir. 1988 yılında Rumelhart, “Paralel Dağıtılmış İşleme” de Geri Yayılım Algoritması adlı yeni bir öğrenme modeli geliştirmiş ve bu, daha önce bahsedilen problemlerin (XOR problemi gibi) geri yayılım algoritması adı verilen yeni bir öğrenme modeli geliştirilerek aşılabileceğini göstermektedir. Yapay

sinir ağları, biyolojik nöronlardan (sinir hücreleri) esinlenerek, beynin çalışma sisteminin yapay simülasyonu ile sonuçlanır.



Şekil 2.9: Yapay Sinir Ağının görselleştirilmesi.

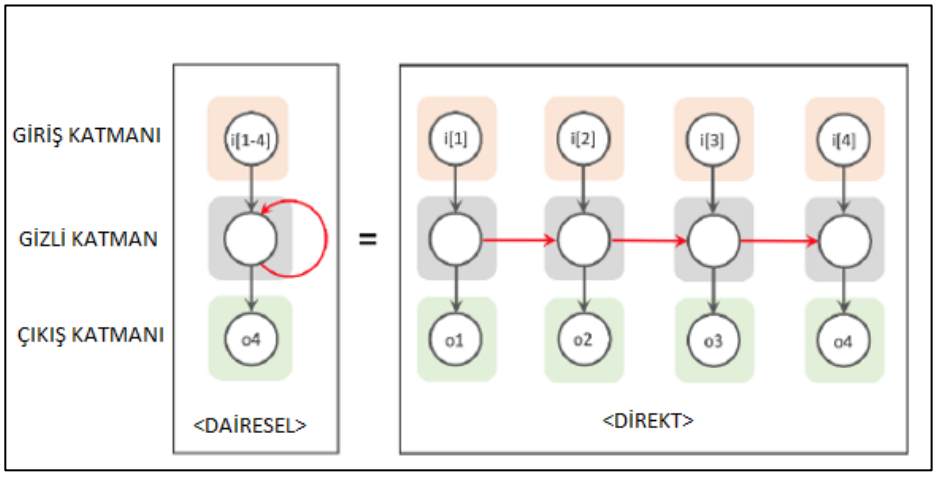
İnternetin yaygınlaşması ve sosyal medya platformlarının popülerleşmesi ile kullanılabilecek devasa bir veri topluluğu oluşturmuştur. Günümüzde bu kadar fazla verinin bulunması veri kümesi boyutuna göre başarımları artan yapay sinir ağı modellerinin potansiyellerini tam olarak göstermelerine imkân tanımaktadır. Bilgisayarların hesaplama gücünün artması artık eskiye kıyasla çok daha büyük, diğer bir deyişle "derin" yapay sinir ağlarının tasarlanabilmesini ve eğitilebilmesini sağlamaktadır. Bunun üzerine birçok büyük özel ve kamu kurumu araştırmacıları artık derin öğrenme olarak anılacak bu alan üstüne çok fazla çalışma yapmaktadır.

2.4.2. Evrişimsel Sinir Ağları

Evrişimsel Sinir Ağı (ConvNet/CNN), bir girdi görüntüsünü alabilen, görüntüdeki çeşitli yönlere/nesnelere önem (öğrenilebilir ağırlıklar ve önyargılar) atayan ve birini diğerinden ayırt edebilen bir derin öğrenme algoritmasıdır. Bir evrişimsel sinir ağında gereken ön işleme, diğer sınıflandırma algoritmalarına kıyasla çok daha düşüktür. İlkel yöntemlerde filtreler elle tasarlanırken, yeterli eğitimle evrişimsel sinir ağları bu filtreleri/özellikleri öğrenme yeteneğine sahiptir. Evrişimsel sinir ağın mimarisi, insan beynindeki nöronların bağlantı modeline benzer ve görsel korteksin organizasyonundan ilham almaktadır. Bireysel nöronlar, uyaranlara yalnızca alıcı alan olarak bilinen görme alanının sınırlı bir bölgesinde yanıt verir. Bu tür alanların bir koleksiyonu, tüm görsel alanı kapsayacak şekilde örtüşür. [15].

2.4.3. Tekrarlayan Sinir Ağları

Tekrarlayan Sinir Ağı, bilgilerin ağ içinde depolanmasına izin veren döngüler içeren bir tür sinir ağıdır. Kısacası, Tekrarlayan Sinir Ağları, yaklaşan olayları bilgilendirmek için önceki deneyimlerinden akıl yürütmelerini kullanır. Tekrarlayan modeller, daha karmaşık görevleri gerçekleştirmeye açan vektörleri sıralama yetenekleri açısından değerlidir. Tekrarlayan Sinir Ağları, birbirine bağlı bir dizi ağ olarak düşünülebilir. Genellikle zincir benzeri bir mimariye sahiptirler, bu da onları konuşma tanıma, dil çevirisi vb. görevlere uygulanabilir kılar. Bir RNN, giriş, çıkış veya her ikisindeki vektör dizileri arasında çalışacak şekilde tasarlanabilir. Örneğin, sıralı bir girdi, girdi olarak bir cümle alabilir ve pozitif veya negatif bir duyarlılık değeri verebilir. Alternatif olarak, sıralı bir çıktı girdi olarak bir görüntü alabilir ve çıktı olarak bir cümle üretebilmektedir. RNN mimarisinin yinelenen (recurrent) olarak adlandırılmasının sebebi, bir dizinin her öğesi için (cümledeki kelimeler gibi) aynı görevi önceki çıktılara bağlı olarak yerine getirmesidir. RNN' lerin ek özelliği, gizli katmandaki geri besleme döngüsüdür. Bu sıralı veriler hakkında ek bilgi sağlar. Örneğin, sinir ağındaki girdi bir kelime cümlesiyse ve her bir girdi bir kelimeyse, gizli katmandaki geri besleme döngüsü ile RNN düğümlerinin ağırlıklarını eğitmek ve güncellemek için tüm cümleyi dikkate alabilir [12], [15], [17].



Şekil 2.10: Tekrarlayan sinir ağının yapısı.

Tekrarlayan sinir ağlarında ileri beslemeli ağlara göre önceki gizli durum ve mevcut gizli durum (ve tekrarlayan katman ağırlığı parametreleri) arasındaki bağlantıyı tanıtır. Bu yinelenen katman geçmiş bilgileri depolamak için tasarlanmıştır.

Gizli katmanındaki değer (2.1) ile hesaplanmaktadır. Çıktı katmanlarındaki değerler ise (2.2) ile hesaplanmaktadır. Burada U, W ve V, eğitim süresinde hesaplanacak bağlantı ağırlıklarıdır ve f(z) ve g(z) aşağıdaki gibi (2.3) sigmoid ve (2.4) softmax aktivasyon fonksiyonlarıdır [16].

$$h(t) = f\big(Ux(t) + Wh(t-1)\big) \tag{2.1}$$

$$y(t) = g\big(Vh(t)\big) \tag{2.2}$$

$$f(z) = \frac{1}{1+e^{-z}} \tag{2.3}$$

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{2.4}$$

2.4.3.1.Geçitli Tekrarlayan Ünite

Geçitli Tekrarlayan Üniteler - GRU'lar daha kalıcı belleğe sahip olmak için tasarlanmıştır, böylece tekrarlayan sinir ağlarının uzun vadeli bağımlılıklarını yakalamasını kolaylaştırır. Eşitlik (2.5) güncelleme sinyali bir sonraki duruma ne kadar geçmiş gizli durum (2.8) taşınması gerektiğine karar verir. (2.6) önceki katmandan gelen gizli durumun yeni belleğin özetlenmesinde ne kadar önemli olduğunu belirlemekten sorumludur. Eğer yeni durumla alakasız olduğunu tespit ederse geçmiş gizli durumu tamamen sıfırlama azaltma yeteneğine sahiptir. Eşitlik (2.7) yeni hafıza eski gizli durumun bir (t-1) konsolidasyonudur. Eşitlik (2.8) ile yeni durumu geçmiş bağlam bilgisi ile özetler. Eşitlik (2.8) ise en son gizli durumdur [18].

$$z^{(t)} = \sigma(W^{(z)}x^{(t)} + U^{(z)}h^{(t-1)}) \tag{2.5}$$

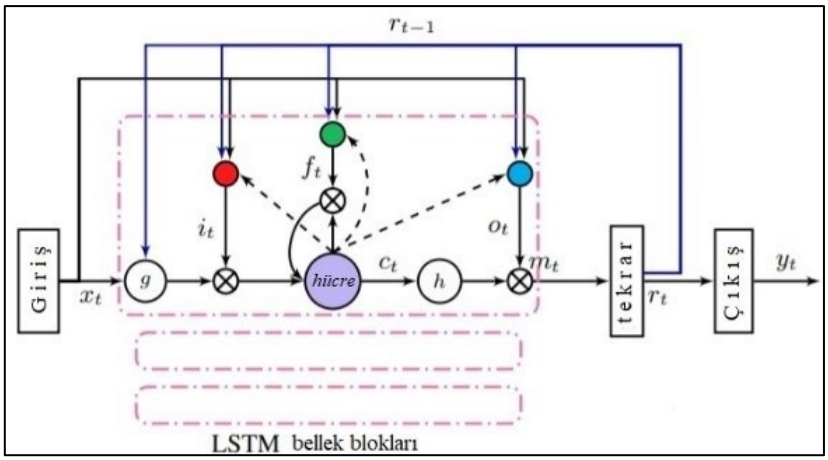
$$r^{(t)} = \sigma(W^{(r)}x^{(t)} + U^{(r)}h^{(t-1)}) \tag{2.6}$$

$$\tilde{h}^{(t)} = tanh(r^{(t)}.Uh^{(t-1)} + Wx^{(t)}) \tag{2.7}$$

$$h^{(t)} = \left(1 - z^{(t)}\right).\ \tilde{h}^{(t)} + z^{(t)}\ .h^{(t-1)} \tag{2.8}$$

2.4.3.2. Uzun Kısa Süreli Bellek

Uzun kısa süreli bellek (LSTM), ezberlenecek veya unutulacak belleği seçerek kaybolan gradyan sorununun üstesinden gelen geleneksel RNN' in geliştirilmiş bir algoritmasıdır. Tüm belleği RNN gibi birimlerde geçirmek yerine, RNN mimarilerinde önceki bilgi kullanımına dayalı bir yaklaşım vardır. Bu uzun vadeli bağımlılıkları öğrenebilen özel bir RNN türü olan algoritmalara Uzun Kısa Vadeli Bellek (Long Short Term Memory - LSTM) ismi verilmiştir [19].



Şekil 2.11: Uzun Kısa Süreli Bellek ağının mimarisi.

LSTM, GRU'lardan biraz farklı olan başka bir karmaşık aktivasyon birimi türüdür. Bunları kullanma motivasyonu GRU'lara benzer, ancak bu tür birimlerin mimarisi farklıdır. LSTM birimlerinin matematiksel formülasyonlarını tanımlayacak olursak, Eşitlik (2.9), yeni hafıza birimi oluşturmadan önce gelen değerin önemli olup olmadığını kontrol ederek giriş kapısı görevi görmektedir. Girişin korunmaya değer olup olmadığını belirlemek için giriş değerini ve geçmiş gizli durumu kullanır. Eşitlik (2.10), giriş değerinin kullanışlılığını belirlemesi dışında giriş kapısına (2.9) benzer. Geçmiş hafıza hücresinin, yeni hafıza hücresinin hesaplanması için yararlı olup olmadığına dair bir değerlendirme yapar. Eşitlik (2.11) GRU'larda açıkça bulunmayan kapıdır. Görevi son belleği gizli durumdan ayırmaktır. Son bellek (2.13), gizli durumda kaydedilmesi zorunlu olmayan birçok bilgiyi içerir. Gizli durumlar bir LSTM'nin her bir kapısında kullanılır ve bu nedenle, bu kapı (2.13) belleğinin hangi gizli durumda (2.14) açığa çıkarılması gerektiğine ilişkin değerlendirme yapar. Bunu belirtmek için ürettiği sinyal (2.11) dir. Bu belleğin noktasal tanh değerini geçmek

için kullanılır. Eşitlik (2.12) GRU'larda görülen yeni bellek oluşturma aşamasına benzer. Yeni çıktı değerlerini içeren bir bellek oluşturmak için girdi değerini ve geçmiş gizli durumu kullanır. Eşitlik (2.13) aşamasında ilk önce ilgili son gizli durumdaki tavsiyeyi saklar, (2.10) kapısını unutur ve buna göre geçmiş hafıza değerini unutur. Daha sonra son belleği üretebilmek için bu iki sonucu toplar. Eşitlik (2.14), GRU'larda olduğu gibi en son gizli durumdur [18]. Bu eşitliklere dayanan tekrarlayan sinir ağları ile modeller eğitildikten sonra her olay kaydı anahtarı için ayrı ayrı tahmin yapılmıştır.

$$i^{(t)} = \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)}) \tag{2.9}$$

$$f^{(t)} = \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)}) \tag{2.10}$$

$$o^{(t)} = \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)}) \tag{2.11}$$

$$\tilde{c}^{(t)} = tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)}) \tag{2.12}$$

$$c^{(t)} = f^{(t)} \,.\, \tilde{c}^{(t-1)} + i^{(t)} \,.\, \tilde{c}^{(t)} \tag{2.13}$$

$$h^{(t)} = o^{(t)} \,.\tanh(c^{(t)}) \tag{2.14}$$

2.4.4. Dikkat Mekanizması

Dikkat, insanlar için vazgeçilmez olan karmaşık bir bilişsel işlevdir. Algının önemli bir özelliği, insanların tüm bilgiyi bir kerede bütünüyle işleme eğiliminde olmamasıdır. Bunun yerine, insanlar ihtiyaç duyulduğu zaman ve yerde bilginin bir kısmına seçici olarak konsantre olma eğilimindedirler, ancak aynı zamanda diğer algılanabilir bilgileri görmezden gelirler. Örneğin, insanlar bir şeyleri görsel olarak algılarken genellikle başından sonuna kadar tüm sahneleri görmezler, bunun yerine gözlemler ve gerektiğinde belirli kısımlara dikkat ederler. İnsanlar bir sahnenin genellikle belirli bir bölümünde gözlemlemek istedikleri bir şeye sahip olduğunu fark ettiklerinde, benzer sahneler tekrar ortaya çıktığında o kısma odaklanmayı öğrenecek ve faydalı kısma daha fazla odaklanmayı öğreneceklerdir. Bu, insanların sınırlı işleme kaynakları kullanarak büyük bilgiden yüksek değerli bilgileri hızla seçmesi için bir araçtır. Dikkat mekanizması, algısal bilgi işlemenin etkinliğini ve doğruluğunu büyük ölçüde artırır. İnsanların dikkat mekanizması, oluşum şekline göre iki kategoriye ayrılabilir. İlk kategori, dış uyaranlar tarafından yönlendirilen, belirginlik temelli

dikkat olarak adlandırılan aşağıdan yukarıya bilinçsiz dikkattir. Örneğin, insanların bir konuşma sırasında yüksek sesler duyma olasılığı daha yüksektir. Daha uygun değerleri (yani daha büyük değerleri) bir sonraki adıma geçiren derin öğrenmedeki maksimum havuzlama mekanizmasına benzer. İkinci kategori, odaklanmış dikkat adı verilen yukarıdan aşağıya bilinçli dikkattir. Odaklanmış dikkat, önceden belirlenmiş bir amacı olan ve belirli görevlere dayanan dikkati ifade eder. İnsanların bilinçli ve aktif olarak belirli bir nesneye dikkat çekmesini sağlar. Derin öğrenmedeki dikkat mekanizmalarının çoğu, belirli görevlere göre tasarlanmıştır, böylece çoğu odaklanmış dikkattir. Dikkat mekanizması, aşırı bilgi yüklemesi sorununu çözmenin ana yolu olan bir kaynak tahsis şeması olarak kullanılabilir. Sınırlı bilgi işlem gücü durumunda, sınırlı bilgi işlem kaynakları ile daha önemli bilgileri işleyebilir. Bahdanau vd. [20], makine çevirisi görevlerinde çeviri ve hizalamayı aynı anda gerçekleştirmek için dikkat mekanizmasını kullandı. Daha sonra, dikkat mekanizması nöral mimarilerin giderek daha yaygın bir bileşeni haline geldi. Dikkat mekanizması, performans iyileştirmeleri sağlamanın yanı sıra, anlaşılmaz sinir mimarisi davranışını açıklamak için bir araç olarak da kullanılabilir [21].

2.5. İlgili Çalışmalar

Son yıllarda bilişim sistemlerinde olay kayıtlarının analizi, ayrıştırılması, toplanması ve saklanması gibi birçok konuda yeni çalışmalar ve araştırmalar yapılmaktadır. Her geçen gün daha iyi sonuçlar elde edilmeye devam etmektedir. Bu bölüm olay kaydı analizi için araştırmacılar tarafından yapılan geçmiş çalışmaların kısa bir sunumunu içermektedir. Olay kayıtları ham veri halinde işlemeye ve analiz yapmaya uygun değildir. Bunun için öncelikle ayrıştırma işlemi yapılması gereklidir. Ayrıştırma işlemi esnasında birçok yöntem bulunmaktadır, kümeleme, graflar, ağaçlar vs. Zhang ve diğerleri [22] olay kayıtlarının ayrıştırma işlemini, örüntü tabanlı TF-IDF özellik çıkarımı ile yapmış ve Uzun Kısa Süreli Bellek (LSTM) tekrarlayan sinir ağı kullanarak geliştirdikleri derin öğrenme modeli ile BT sistemlerinde sistem arıza tahmini yapmışlardır. Wang ve diğerleri [23] sistem olay kayıtlarından özellik çıkarabilmek için doğal dil işleme tekniklerinden olan Word2Vec ve TF-IDF yöntemlerini kullanmışlardır. Olay kayıtlarını bu teknikler ile sayısal sekanslar haline getirdikten sonra LSTM tekrarlayan sinir ağıyla bir model geliştirmişlerdir. Bu yöntemi Gradyan Artırıcı Karar Ağaçları (GBDT) ve Naïve Bayes yöntemleri ile

karşılaştırmışlardır. Hammoud [10], olay kayıtlarını saklamak için geliştirilen uygulamaların bağımsız yapılarının, olay kayıtlarının korelasyonunu zorlaştırdığından bahsetmiştir. Dağıtık yapıda bulunan bu olay kayıtlarının veri tabanı alanında geliştirilen sürekli sorguları uyarlayan ve uygulayan bir yöntem önermiştir. Bu yöntem ile dağıtık ve bağımsız yapıdaki olay kayıtlarını gerçek zamanlı olarak ilişkilendirmeye çalışmıştır. Olay kayıtlarının gerçek zamanlı olarak izlenmesi ve işlenmesi de bir başka problemdir. Sürekli tekrar eden ve aynı mesajı taşıyan olay kayıtlarının gelmesi izlenen olay kayıtları içerisinde önemli olanların analistler tarafından gözden kaçmasına neden olmaktadır. Özellikle bir siber saldırı esnasında diğer olay kayıtlarıyla birlikte tespit edilmesi çok zordur. Dwivedi ve Triphati [25], bu zorluğa karşı Saldırı Tespit Sistemleri (IDS) için olay kaydı korelasyonu hazırlarken dikkat edilmesi gereken noktaları incelemiştir. Tekrar eden değerlerin doğru bir şekilde temizlenmesi ve yanlış alarm olasılığının düşürülmesini hedefleyen bir yöntem önermişlerdir. Giderek artan bir şekilde büyüyen bilişim teknolojileri sistemleri çok büyük boyutlarda olay kaydı ürettiğinden bu olay kaydı verilerini işleyebilecek teknolojiler geliştirilmesi gerekmektedir. Gupta [25], olay kaydı madenciliğinin en temel şeklinin korelasyon bulmak olduğundan bahsetmiştir. Operasyon merkezlerinde toplanan olay kayıtları arasında manuel olarak korelasyon oluşturma ve bu korelasyonların atanma sorununa korelasyon analizi yaparak çözüm önerisinde bulunmuştur. Cheng ve diğerleri [26], filtreleme ve doğrulama ilkesini takip ederek, dağıtılmış platformlar kullanarak daha verimli bir olay korelasyonunu amaçlayan RF-GraP (Kural Filtreleme ve Grafik Bölümleme) adlı yeni bir yaklaşım önermiştir. Bu yöntem de korelasyonları graflar halinde filtreleyip işlemişlerdir. Deneysel sonuçlar, önerilen algoritmanın son derece verimli olduğunu ve son teknoloji ürünü üzerinde ve çok daha az ağ iletişimiyle önemli hız artışları sağlayabileceğini göstermiştir. Du ve diğerleri [27], bir olay kaydını doğal bir dil dizisi olarak modellemek için Uzun Kısa Süreli Bellek (LSTM) kullanan derin bir sinir ağı modeli olan DeepLog'u önermiştir. Model eğitilmeden önce mevcut olay kayıtlarını Spell [28] isimli yöntem ile ayrıştırmışlardır. Spell olay kayıtlarını En Uzun Ortak Sıra (LCS) algoritmasını kullanır. Bu yöntem ile olay kaydı anahtarları oluşturarak olay kayıtlarını sayısal diziler halinde temsil etmişlerdir. DeepLog, bu olay kaydı anahtarlarını kullanarak anahtar dizisinde oluşan anormallikleri algılayan sinir ağı modelidir. Bu modeli eğitmek için ilk olarak eğitim aşamasında normal olarak etiketlenmiş olay kayıtlarının bir dizi temsili kullanılır ve anomali tanılama amacıyla bu sistem akış modelleri

oluşturur. Tespit aşamasında, DeepLog, sistemin anormal olup olmadığını belirlemek için anahtar anomali tespit modelinin tahmin edilen sonuçlarını gerçek olay kaydı anahtar değerleriyle karşılaştırır. Xu ve diğerleri [7], program analizinin bir kombinasyonunu kullanarak madencilik konsol olay kayıtları için otomatik bir metodoloji önerdi. Bu yöntem, konsol günlüklerinden yapı çıkarmak için ilk olarak olay kaydı ayrıştırma ve metin madenciliğini kaynak kod analiziyle birleştirir. Daha sonra, Temel Bileşen Analizini (PCA) kullanarak olay kayıtlarındaki anormal kalıpları tespit etmek için yapılandırılmış bilgilerden özellikler çıkarır. Son olarak, PCA tabanlı anomali tespitinin sonuçlarını kolayca anlaşılabilir bir formata ayırmak için bir karar ağacı kullanırlar. Bu yöntemin dezavantajı, kaynak kodun pratikte her zaman erişilebilir olmaması ve algılama verimliliği ve doğruluğunun boyutlardan büyük ölçüde etkilenmesidir. Terim Frekansı-Ters Döküman Frekansı (TF-IDF) gibi çıkarma yaklaşımı daha sonra kayan bir pencerede bir olay kaydı anahtarları dizisini temsil edecek bir özellik vektörü oluşturmak için kullanılır. Son olarak, çoğu durumda, anormal dizileri saptamak için denetimsiz bir yaklaşım uygulanır [29], [30]. Son zamanlarda, olay kaydı anomali tespiti için birçok derin öğrenme tabanlı olay kaydı anomali tespit yaklaşımı önerilmiştir. Mevcut yaklaşımların çoğu, normal olay kaydı anahtar dizilerini modellemek ve anormal olay kayıt dizilerini tespit etmek için anormal puanlar türetmek için özellikle tekrarlayan sinir ağlarını kullanır. Ambre ve Shekokar [31] çalışmaları belirli bir tehdidin veya saldırının temel nedenini bulmak için olayın meydana gelme sıklığını yüzde olarak gösteren olasılıksal bir yaklaşım önermişlerdir. Bu çalışmada, düşük yanlış alarm oranı elde ederken kötü niyetli içeridekileri belirlemeye odaklanmışlardır.

3. GELİŞTİRİLEN YÖNTEM

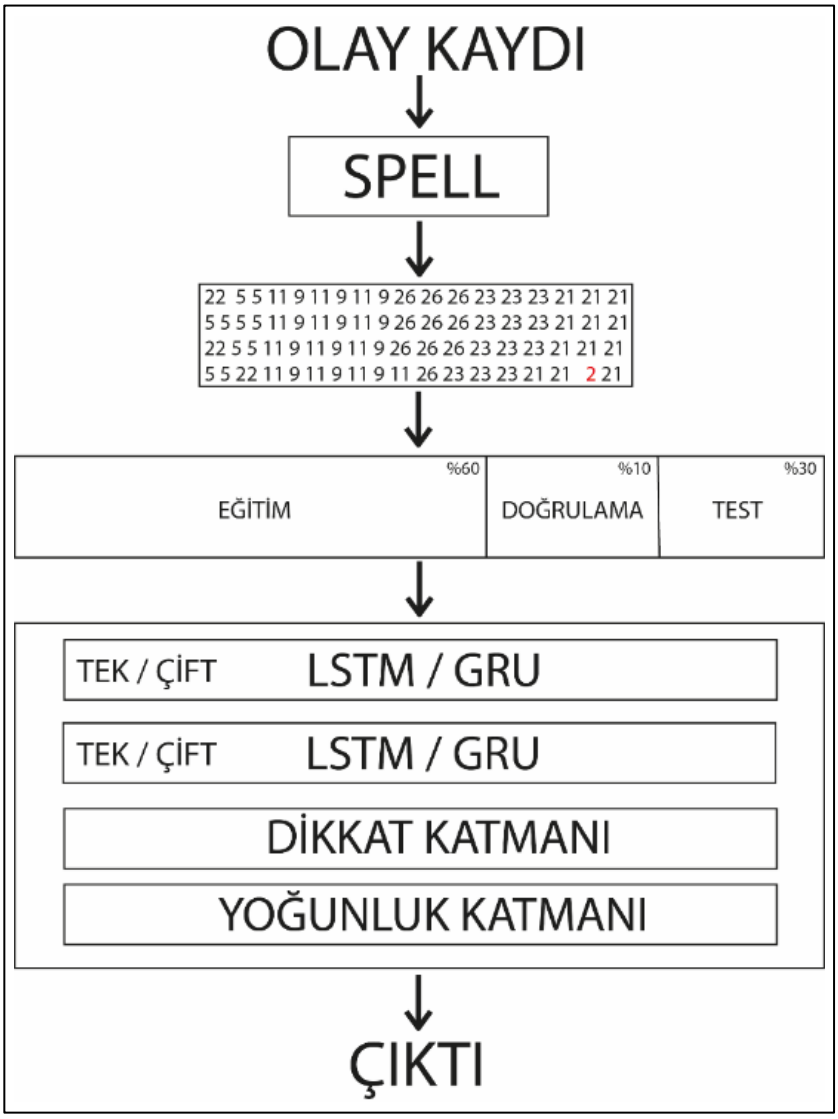
Bilişim sistemlerinden toplanan olay kayıtlarının analizinin yapılması için derin sinir ağı modelleri geliştirilmiştir. Bu bölümde olay kayıtlarının analizi için derin sinir ağı modeli önerilmiştir. Öncelikle olay kayıtları literatürde sıklıkla kullanılan Spell [28] isimli bir olay kaydı ayrıştırma aracı kullanılmıştır. Spell ile olay kaydı anahtar dizileri haline getirilen veri kümesi tekrarlayan sinir ağı yapısı olan LSTM/GRU kullanılarak oluşturulan derin sinir ağı modelinin eğitimi ve testinde kullanılmıştır. Geliştirilen yöntemde Hem LSTM hem de GRU için tek veya çift yönlü olarak iki tür RNN tipi kullanılmıştır. Önerilen yöntemde dikkat mekanizması [33], [34] de kullanılmıştır. Dikkat mekanizması basitçe Kodlayıcı-Çözücü tekrarlayan sinir ağlarının son gizli katmanlarından çıkan değerlerden bir sonraki katmana geçmeden önce bir dizi işlemden geçtikten sonra bir hizalama skoru oluşturur. Bu hizalama skorlarıyla dikkat ağırlıklarını hesapladıktan sonra kodlayıcının çıktısı ile bu ağırlıkları çarparak bir bağlam vektörü oluşturur. Son olarak bir önceki çözücünün çıkış değeri ile oluşturulan bu bağlam vektörünü toplayarak sonraki çözücünün girdi değerlerine gönderir. Bu sayede tekrarlayan sinir ağında ağırlıklar sinir ağının en çok dikkat etmesi gereken ağırlıklara odaklanmasını sağlar. Önerilen yöntemde hem Uzun Kısa Süreli Bellek (LSTM) hem de Geçitli Tekrarlayan Ünite (GRU) tekrarlayan sinir ağları kullanılmıştır. Ayrıca her yöntem için dikkat mekanizması içeren varyasyonlar da geliştirilmiştir. Önerilen yöntem Spell olay kaydı ayrıştırıcı tarafından parçalanmış olay kayıtlarının içerisinden, HDFS'e özgü block_id değerlerine göre oluşturulmuş anahtar dizilerinin sıralarıyla eğitilmiştir. Bu anahtar dizilerinin her biri block_id değerlerine göre toplanmış iş akış sıralarıdır. Yani gerçekleşen olaylar zamansal olarak bir iş akışını temsil etmektedir. Bu iş akışında oluşabilecek farklı akışlar veya beklenmeyen olaylar anomali olarak işaretlenmiştir. Bu yöntemde iş akış sekanslarından belirlenen pencere boyutunca bir aralığı alarak eğitilen modelden, bir sonraki olay kaydı anahtar değerinin tahmin edilmesine dayanılarak geliştirilen bir yöntemdir. Bu olay kaydı anahtar sekansları ile LSTM/GRU tekrarlayan sinir ağı eğitildikten sonra çıkış katmanı olarak 28 hücreden oluşan tam bağlı yoğunluk katmanına gönderilmiştir. Çıkış katmanında 28 hücrenin olmasının sebebi veri kümesi içerisinden 28 farklı olay kaydı anahtarı çıkarılmış olmasıdır. Bu 28 olay kaydı şablonuna göre sinir ağının çıktısında her bir hücreden farklı ağırlıkta bir değer çıkmaktadır. Bu değerler tahmin edilen değerin hangi olay kaydı anahtarı olduğunu

gösterecektir. Yöntemde 8-9-10-11-12 pencere boyutları sırasıyla denenmiş ve en iyi sonuç 10 pencere boyutunda alınmıştır. On pencere boyutunda ilk 9 olay kaydı anahtar değeri öğrenildikten sonra 10. olay kaydı anahtar değeri tahmin edilmeye çalışılmıştır. Yalnızca olay kaydı anahtarının tahmin edilmesi anomali tespiti için yeterli bir durum değildir. Bu yüzden tahmin edilen olay kaydının ardından oluşan iş akış sekansının normal iş akış sekanslarının içerisinde mi yoksa anormal iş akış sekanslarının içerisinde olup olmadığı kontrol edilmiştir. Bu kontrol sonucunda oluşan ve tahmin edilen iş akış sekansının hangi küme içerisinde olduğuna göre oluşan durumun normal veya anormal olarak değerlendirilmesi gerçekleştirilmiştir. Bir sonraki olay kaydı anahtarını tahmin ederken yapılan tahminlerin içerisinden 9 tahmin alınarak en iyi değer seçilmiştir. Ayrıca alınan sonuçlar 10 adımlı çapraz doğrulama ile alınarak ortalama değer üzerinden hesaplanmıştır.

#	Şablonlar		#
1	Adding an already existing block (.*)	Changing block file offset of block (.*) from ([-]?[0-9]+) to ([-]?[0-9]+) meta file offset to ([-]?[0-9]+)	15
2	(.*)Verification succeeded for (.*)	(.*):Transmitted block (.*) to (.*)	16
3	(.*) Served block (.*) to (.*)	(.*):Failed to transfer (.*) to (.*) got (.*)	17
4	(.*):Got exception while serving (.*) to (.*):(.*)	(.*) Starting thread to transfer block (.*) to (.*)	18
5	Receiving block (.*) src: (.*) dest: (.*)	Reopen Block (.*)	19
6	Received block (.*) src: (.*) dest: (.*) of size ([-]?[0-9]+)	Unexpected error trying to delete block (.*)\. BlockInfo not found in volumeMap\.	20
7	writeBlock (.*) received exception (.*)	Deleting block (.*) file (.*)	21
8	PacketResponder ([-]?[0-9]+) for block (.*) Interrupted\.	BLOCK* NameSystem\.allocateBlock: (.*)\. (.*)	22
9	Received block (.*) of size ([-]?[0-9]+) from (.*)	BLOCK* NameSystem\.delete: (.*) is added to invalidSet of (.*)	23
10	PacketResponder (.*) ([-]?[0-9]+) Exception (.*)	BLOCK* Removing block (.*) from neededReplications as it does not belong to any file\.	24
11	PacketResponder ([-]?[0-9]+) for block (.*) terminating	BLOCK* ask (.*) to replicate (.*) to (.*)	25
12	(.*):Exception writing block (.*) to mirror (.*)(.*)	BLOCK* NameSystem\.addStoredBlock: blockMap updated: (.*) is added to (.*) size ([-]?[0-9]+)	26
13	Receiving empty packet for block (.*)	BLOCK* NameSystem\.addStoredBlock: Redundant addStoredBlock request received for (.*) on (.*) size ([-]?[0-9]+)	27
14	Exception in receiveBlock for block (.*) (.*)	BLOCK* NameSystem\.addStoredBlock: addStoredBlock request received for (.*) on (.*) size ([-]?[0-9]+) But it does not belong to any file\.	28

Şekil 3.1: Spell aracılığıyla çıkarılan şablonların gösterimi.

Modellerin eğitilmesinde Adam optimize edici kullanılmıştır. Genellikle 300 adımda eğitilmiştir. Bazı yöntemlerde farklı adım sayıları da kullanılmıştır. Farklı adım sayılarında bazı yöntemlerin nispeten biraz daha iyi veya kötü sonuç verebildiği gözlenmiştir. Sinir ağları 64 veya 128'er birimlerden oluşmaktadır. Öğrenme katsayısı da bazı modellerde sabit 0.001 seçilirken bazılarında öğrenme oranı düştükçe 0.0001'e kadar katlanarak düşürülmüştür. Gerektiği zaman GRU veya LSTM katmanlarının çıktıları ek bir yoğun katman üzerinde seyreltilerek çıkış katmanına gönderilmiştir. Şekil-3.1'de geliştirilen yöntemlerin genel yapısı gösterilmiştir.



Şekil 3.2: Geliştirilen yöntemlerin genel mimarisi.

4. DENEY SONUÇLARI VE DEĞERLENDİRME

Bu bölümde geliştirilen modellerin deney sonuçları ve değerlendirmeleri verilmiştir.

4.1. Olay Kaydı Veri Kümesi

Veri kümesi olarak Hadoop dağıtık dosya sistemi (Hadoop Distributed File System – HDFS) olarak adlandırılan bir sistemin olay kayıtları kullanılmıştır [34]. Bu olay kayıtları Amazon EC2 isimli veri tabanı üzerinde 200 düğümden oluşan dağıtık bir sistem üzerinden toplanmıştır. Yaklaşık 40 saat sürede toplanmış bir olay kaydı veri kümesidir. Veri kümesindeki verilerin %2,9'u uzmanlar tarafından anormal olarak etiketlenmiştir. Literatürde sıklıkla kullanılan bir veri kümesidir. Spell [28] ile ayrıştırıldıktan sonra HDFS'e özel blk_id değerlerine göre gruplanarak olay kaydı anahtar dizilerine dönüştürülmüşlerdir.

Tablo 4.1: Veri Kümesinin Dağılımı.

Veri Kümesi	**Normal**	**Anormal**	**Toplam**
Eğitim	4855	4855	9170
Test	553366	11983	565349
Toplam	**574519**		

Tablo 4.1'de bu çalışmada geliştirilen yöntemlerin eğitiminde ve testinde kullanılan veri kümesinin detaylı dağılımı verilmiştir. Eğitim veri kümesi ile geliştirilen yöntemler eğitilirken doğrulama veri kümesiyle bu modeller üzerinde performans iyileştirmeleri yapılmıştır.

4.2. Deney Ortamı

Geliştirilen yöntemler ile yapılan deneyler Nvidia GTX 1050 ekran kartına sahip i7-7700HQ işlemci ve 8GB RAM bulunan bir dizüstü bilgisayar ile gerçekleştirilmiştir. Geliştirilen yöntemler Anaconda isimli veri bilimi platformunda Python 3.9 dili ve Tensorflow, Keras kütüphaneleri kullanılarak gerçekleştirilmiştir.

4.3. Değerlendirme Ölçütleri

Eğitim veri kümesi ile eğitilen model üzerinde test veri kümesiyle test yapabilmek için öncelikle tahmin edilecek olay kaydının anahtarı hesaplanır. Daha sonra gerçek değeri ile tahmin edilen değer karşılaştırıldıktan sonra gerekli ölçütlerde güncellemeler yapılır. Bu ölçütlerin anlaşılabilir olması için öncelikle tanımlamaları yapılmıştır.

True Positive (TP): Doğru bir şekilde anomali olarak tespit edilen olay kayıtlarının sayısı.

True Negative (TN): Doğru bir şekilde normal olarak tespit edilen olay kayıtlarının sayısı.

False Positive (FP): Normal olmasına rağmen anomali olarak işaretlenen olay kayıtlarının sayısı

False Negative (FN): Anomali olmasına rağmen normal olarak tespit edilen olay kayıtlarının sayısı.

Tablo 4.2: Sınıflandırma karmaşıklık matrisi.

Tahmin / **Gerçek**	**0 (Normal)**	**1 (Anomali)**
0 (Normal)	TN	FP
1 (Anomali)	FN	TP

- Doğruluk (Accuracy)

Doğruluk, doğru tahmin edilen anomali olay kayıtlarının toplam olay kayıtlarının sayısına oranıdır.

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \tag{4.1}$$

- Kesinlik (Precision)

Kesinlik, doğru tahmin edilen pozitif örneklerin (olay kayıtlarının) toplam pozitif tahminlere oranıdır:

$$Precision = \frac{TP}{TP+FP} \tag{4.2}$$

• Duyarlılık (Recall)

Duyarlılık veya algılama olasılığı (Recall), doğru tahmin edilen anomali olay kayıtlarının toplam olay kayıtlarının sayısına oranıdır.

$$Recall = \frac{TP}{TP+FN} \tag{4.3}$$

• F-Skor

F-skoru, hassasiyet ve hatırlamanın harmonik ortalamasıdır. F-skoru 0 ile 1 arasında bir değer aralığına sahiptir, burada 1 mükemmel hassasiyete eşittir ve her ikisi de 1'e eşittir. F1 skoru, hassasiyet ve hatırlama arasında bir ilişki metriği sağlamak için yararlıdır, çünkü bu iki değer bir şekilde ters benzetmeye sahiptir [13].

$$F - measure = 2 * \frac{precision*recall}{precision+recall} \tag{4.4}$$

Eşitlik (4.5), Kayıp fonksiyonu tasarlanan modelin hata oranını aynı zamanda başarımını ölçen fonksiyondur. Derin sinir ağların son katmanı kayıp fonksiyonun tanımlandığı katmandır. Kayıp fonksiyonu temelde modelin yaptığı tahminin, gerçek değerden ne kadar farklı olduğunu hesaplamaktadır. Bu nedenle iyi tahmin eden bir model oluşturmamışsak, gerçek değer ile tahmin edilen değer arasındaki fark yüksek olacak dolayısıyla kayıp değeri yüksek olacak, iyi modele sahipsek kayıp değeri az olacaktır. Tahmin edilen değer ve gerçek değerin birebir aynı olduğu durumda kayıp değeri 0 olacaktır. Bu modelde kategorik çapraz entropi (categorical cross entropy) kayıp fonksiyonu kullanılmıştır.

$$Kayıp\ Fonksiyonu = -\sum_{i=1}^{çıkış\ boyutu}(y_i\ .\log \hat{y}_i) \tag{4.5}$$

4.4. Test Sonuçları ve Değerlendirme

Belirtilen test kümesi üzerinde yapılan deneysel sonuçlarda elde edilen değerler Tablo 4.3'te verilmiştir. Çift yönlü GRU modellerinin tek yönlü GRU modellerine göre daha iyi sonuçlar verdiği gözlemlenmiştir. Fakat çift yönlü LSTM modelleri tek yönlü LSTM modellerine göre daha düşük performans göstermiştir. Dikkat mekanizması seq2seq modellerde [35] katmanlar arasında sonraki girdiyi oluştururken, önceki katmandan gelen çıktı değerinin yalnızca bir bölümüne odaklanma problemini çözmek için geliştirilmiştir. Fakat artık birçok sinir ağı yönteminde gereksiz ağırlıklardan kurtulmak ve modelin iyileştirilmesi için sıklıkla kullanılmaktadır. Dikkat mekanizmasına sahip çift yönlü GRU tek yönlü GRU'dan daha iyi sonuç vermiştir.

Tablo 4.3: Değerlendirme ve karşılaştırma tablosu.

Yöntem	**Doğruluk**	**Kesinlik**	**Duyarlılık**	**F-skor**
DeepLog	-	95	96	96
Bi-LSTM	91,13	84,18	94,1	88,9
LSTM + Attention	94,61	96,91	97,28	97,09
Bi-LSTM + Attention	92,39	91,13	95,42	93,22
GRU	91,35	84,37	93,45	88,68
Bi-GRU	92,51	91,72	97,52	94,53
GRU + Attention	90	84,24	94,78	89,2
Bi-GRU + Attention	92	89,76	94,83	92,22

Daha önce yapılmış bir çalışma olan DeepLog [27] çalışmasını referans alarak geliştirilen yöntemlerin ölçüm sonuçları karşılaştırılmıştır. DeepLog yönteminde yalnızca LSTM kullanılarak bir model geliştirildiği için yalnızca LSTM kullanılarak bir çalışma yapılmamıştır.

5. SONUÇLAR ve GELECEK ÇALIŞMALAR

Bu çalışmada, BT sistemlerinde üretilen olay kayıtlarının analizi yapılarak olası ilişkilerin çıkartılması için derin öğrenme tabanlı bir yöntem önerilmiştir. Yöntemde, dikkat mekanizması eklenmiş LSTM ve çift yönlü LSTM olmak üzere farklı varyasyonları kullanılmıştır. LSTM modeline eklenen dikkat mekanizması DeepLog ile önerilen yönteme göre başarı artışı sağlamıştır. Çift yönlü RNN'ler tek yönlülere göre daha fazla bağlam bilgisi yakalayabildikleri için başarım artışı sağlamışlardır. Dikkat mekanizması RNN'lerde kaybolan önemli bilgilerin bağlam vektörleriyle tekrar sürece dahil edilmesini sağlayarak başarımları artırmıştır. Geliştirilen yöntemler daha fazla çalışma ile iyileştirilebilecektir. RNN'lerin eğitim ve test maliyetlerinin düşürülmesi içinde ek yöntemler geliştirilmelidir.

Geliştirilen yöntemler, tekrarlayan sinir ağlarıyla oluşturulan modellerin olay kaydı analizi konusunda yüksek başarıma sahip olduklarını ispatlamışlardır. Çift yönlü LSTM/GRU kullanılarak daha iyi bağlam bilgisi yakalanması hedeflenmiştir. Aynı zamanda bu çift yönlü katmandan sonra çıkış katmanında bir dikkat mekanizması bulunan bir katman eklenmiştir. Bu dikkat katmanı sayesinde önceki olay kayıt anahtarlarından hangileri sonraki giriş değerini daha çok etkiliyorsa bunların ağırlıklarının daha da yükseltilerek bu bağlamlara daha fazla dikkat etmesi sağlanmıştır.

Gelecekteki araştırmalar, önerilen modeller üzerindeki hiperparametre ayarının etkisini göz önünde bulundurabilir. Bu hiperparametrelerin daha detaylı değerlendirilmesi ve farklı veri kümelerinde bu yöntemlerin nasıl sonuçlar alacağı test edilecektir. Modellerin güçlü cihazlarda en performanslı nihai model bulunana kadar eğitilmesi gerekmektedir. Ayrıca, Evrişimli Sinir Ağı (CNN) gibi algoritmalar, olay kayıtlarından anomali tespiti ve sınıflandırması için kullanılabilir. Tekrarlayan sinir ağlarının (LSTM ve GRU) eğitilmeleri ve tespit için tahmin yapmaları hem zaman hem enerji açısından maliyetlidir. CNN'ler bu durumu ele alabilmek için kullanılabilir. Çevrimiçi olay analizi sistemlerinde olay kayıtlarında anomalileri en hızlı şekilde yakalamak çok önemlidir. CNN'lerin RNN'lere nispeten daha hızlı çalıştıkları için çevrimiçi olay analizlerinde uygulamaların entegrasyonları açısından tercih edilebilirler. Gelecek çalışmalarda CNN'lerin olay kayıtlarının anomali tespitinde kullanımı ve RNN'lerin daha da iyileştirilmesi hedeflenmektedir.

KAYNAKLAR

[1] Xu W., Huang L., Fox A., Patterson D., Jordon M.I., (2009), "Detectinglarge-scale system problems by mining console logs", Proc. of the ACM Symposium on Operating Systems Principles.

[2] Ankerst M., Breunig M. M., Kriegel H.-P., J Sander., (1999), "Optics: Ordering points to identify the clustering structure," in Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, SIGMOD '99. New York, USA: ACM, 1999, 49–60.

[3] Wang M., Xu L., Guo L., (2018), "Anomaly Detection of System Logs Based on Natural Language Processing and Deep Learning," In 2018 4th International Conference on Frontiers of Signal Processing (ICFSP), 140-144, IEEE.

[4] Hanemann, A., Sailer M., (2005), A "framework for service quality assurance using event correlation techniques," In Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05), 428-433, IEEE.

[5] Yuan D., Mai H., Xiong W., Tan L., Zhou Y., Pasupathy S., (2010), "Sherlog: Error diagnosis by connecting clues from run-time logs," SIGARCH Comput. Archit. News, 143–154.

[6] Kim C. H., Rhee J., Zhang H., Arora N., Jiang G., Zhang X., Xu D., (2014), "Introperf: Transparent context-sensitive multilayer performance inference using system stack traces," SIG-METRICS Perform. Eval. Rev., 235–247.

[7] Xu W., Huang L., Fox A., Patterson D., Jordan M. I., (2009), "Detecting large-scale system problems by mining console logs," in Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, 117–132.

[8] Kimura T., Ishibashi K., Mori T., Sawada H., Toyono T., Nishimatsu K., Watanabe A., Shimoda A., Shiomoto K., (2014), "Spatio-temporal factorization of log data for understanding network events," in 2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, 610–618.

[9] Cheng, L., Van Dongen, B. F., Van Der Aalst, W. M., (2017), "Efficient event correlation over distributed systems," In 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 1-10.

[10] Hammoud N., (2009), "Decentralized log event correlation architecture.", In Proceedings of the International Conference on Management of Emergent Digital EcoSystems, 480-482.

[11] Myers J., Grimaila M. R., Mills R. F., (2010), "Adding value to log event correlation using distributed techniques," In Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, 1-4.

[12] Ahi Ş., (2020), “Gelişmiş sosyal mühendislik saldırıları analizi ve tespiti”, Yüksek Lisans Tezi, Gebze Teknik Üniversitesi.

[13] Ahi Ş., Soğukpınar İ., (2020), "Derin Öğrenme Modelleri ile Kimlik Avı E-posta Tespiti", Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 17-31.

[14] He S., Zhu J., He P., Lyu, M. R., (2016), “Experience report: System log analysis for anomaly detection,” In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE),207-218, IEEE.

[15] Şeker A., Diri B., Balık H. H., (2017), “Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme,” Gazi Mühendislik Bilimleri Dergisi (GMBD), 47-64.

[16] Huang Z., Xu W., Yu K., (2015), “Bidirectional LSTM-CRF models for sequence tagging,” arXiv preprint arXiv:1508.01991.

[17] Mandic D., Chambers J., (2001), “Recurrent neural networks for prediction: learning algorithms, architectures and stability”, Wiley.

[18] Socher R., Class Lecture, (2015), Topic: "Part IV-Language Models. RNN-Bi-RNN-GRU-LSTM" CS224D/Deep Learning for Natural Language Processing, Computer Science Department/Linguistics Department, Stanford University.

[19] Hochreiter S., Schmidhuber J., (1997), “LSTM can solve hard long time lag problems,” Advances in neural information processing systems, 473-479.

[20] Bahdanau D., Cho K., Bengio Y., (2014), “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473.

[21] Niu Z., Zhong G., Yu H., (2021), “A review on the attention mechanism of deep learning,” Neurocomputing, 452, 48-62.

[22] Zhang K., Xu J., Min M. R., Jiang G., Pelechrinis K., H. Zhang, “Automated IT system failure prediction: A deep learning approach,” IEEE International Conference on Big Data (Big Data), 2016, 1291-1300.

[23] Wang M., Xu L., Guo L., “Anomaly Detection of System Logs Based on Natural Language Processing and Deep Learning,” (2018) 4th International Conference on Frontiers of Signal Processing (ICFSP), 2018, 140-144.

[24] Dwivedi N., Tripathi A., (2015), “Event Correlation for Intrusion Detection Systems,” IEEE International Conference on Computational Intelligence & Communication Technology, 133-139.

[25] Gupta C., (2012), “Event correlation for operations management of largescale IT systems,” Proceedings of the 9th international conference on Autonomic computing (ICAC '12), Association for Computing Machinery, New York, USA, 91–96.

[26] Cheng L., Van Dongen B. F., Van Der Aalst W. M. P., (2017), “Efficient Event Correlation over Distributed Systems,” 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 1-10.

[27] Du M., Li F., Zheng G., Srikumar V., (2017), “DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning,” Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17), Association for Computing Machinery, NY, USA, 1285–1298.

[28] Du M., Li F., (2016), “Spell: Streaming Parsing of System Event Logs,” 2016 IEEE 16th International Conference on Data Mining (ICDM), 859-864.

[29] Lou J.G., Fu Q., Yang S. Y., Li J., (2010), “Mining invariants from console logs for system problem detection.” Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference.

[30] Xu W., Huang L., Fox A., Patterson D., Jordan M. I., (2009), “Online system problem detection by mining patterns of console logs,” Ninth IEEE International Conference on Data Mining, 588–597.

[31] Ambre A., Shekokar N., (2015), “Insider threat detection using log analysis and event correlation. Procedia Computer Science,”, 436-445.

[32] Chorowski J., Bahdanau D., Serdyuk D., Cho K., Bengio Y., (2015), “Attention-based models for speech recognition,”, arXiv preprint arXiv:1506.07503.

[33] Luong M. T., Pham H., Manning C. D., (2015), “Effective approaches to attention-based neural machine translation,”, arXiv preprint arXiv:1508.04025.

[34] Xu W., Huang L., Fox A., Patterson D., Jordan M., (2009), “Large-scale system problem detection by mining console logs,”, Proc. of the 22nd ACM Symposium on Operating Systems Principles (SOSP’ 09), Big Sky, MT.

[35] Sutskever I., Vinyals O., Le Q. V., (2014), “Sequence to sequence learning with neural networks,” arXiv preprint arXiv:1409.3215.

ÖZGEÇMİŞ

Sivas Cumhuriyet Anadolu Lisesinde ortaöğrenimini tamamlayan Atahan Duman, Sivas Cumhuriyet Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünü 2018 yılında başarıyla tamamladı. 2019'da yüksek lisans eğitimine Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Siber Güvenlik Programında başladı. 2019 yılından bu yana GTÜ Teknopark bünyesinde özel bir firmada Ar-Ge Mühendisi olarak çalışmaktadır.