T.C.
GİRESUN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN SİNİR AĞI YÖNTEMLERİ İLE WIKIPEDIA TOKSİK YORUM SINIFLANDIRMASI

YÜKSEK LİSANS TEZİ

Öğrencinin Adı SOYADI : Mete ÖZDEMİR

Tezin Enstitüye Verildiği Tarih : …/…/……

Enstitü Anabilim Dalı : İstatistik Anabilim Dalı

Tez Danışmanı : Dr. Öğr. Üyesi Ali Zafer DALAR

Temmuz 2021
GİRESUN

T.C.
GİRESUN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN SİNİR AĞI YÖNTEMLERİ İLE WIKIPEDIA TOKSİK YORUM SINIFLANDIRMASI

YÜKSEK LİSANS TEZİ

Mete ÖZDEMİR

Enstitü Anabilim Dalı : İstatistik

Bu tez ../../20.. tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

Prof. Dr.	**Prof. Dr.**	**Dr. Öğr. Üyesi**
Erol EĞRİOĞLU	**Ufuk YOLCU**	**Ali Zafer DALAR**
Jüri Başkanı	**Üye**	**Üye**

Doç. Dr.
Bahadır KOZ
Enstitü Müdürü

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu beyan ederim.

Mete ÖZDEMİR

30/07/2021

TEŞEKKÜRLER

Yüksek lisans tezimde danışmanlık alma şansı bulduğum değerli hocam Dr. Öğr. Üyesi Ali Zafer DALAR'a tez süresince olan desteği, değerli bilgilerini paylaşması ve ilgisinden dolayı teşekkürlerimi sunarım.

Ayrıca yüksek lisans eğitimim süresince her zaman olumlu mesajlar veren, bilgilerini paylaşan ve destek olan hocalarıma, bilgilerinden ve tecrübelerinden faydalandığım değerli dostlarım Fatih Samet Çetin, Salih Cemil Çetin, Selman Baş ve Alper Demirel'e teşekkürlerimi sunarım.

Tez süresince desteklerini esirgemeyen hep yanımda olan sevgili eşim ve aileme de teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

σ	: Sigmoid Fonksiyonu
α	: Öğrenme Hızı
ϵ	: Epsilon
API	: Uygulama Programlama Arayüzü (Application Programming Interface)
Bi-LSTM	: Çift Yönlü Uzun Kısa-Süreli Hafıza (Bidirectional Long Short-Term Memory)
BRNN	: Çift Yönlü Tekrarlayan Sinir Ağı (Bidirectional Recurrent Neural Network)
CNN	: Evrişimli Sinir Ağları (Convolutional Neural Network)
CPU	: Merkezi İşlem Birimi / İşlemci (Central Processing Unit)
GPU	: Grafik İşlemci Birimi (Graphics Processing Unit)
GRU	: Geçitli Tekrarlayan Birim (Gated Recurrent Unit)
IQR	: Çeyreklikler Arası Açıklık (Interquartile Range)
LSTM	: Uzun Kısa-Süreli Hafıza (Long Short-Term Memory)
NLP	: Doğal Dil İşleme (Natural Language Processing)
SS	: Standart Sapma
ReLU	: Doğrultulmuş Doğrusal Birim (Rectified Linear Units)
TSA	: Tekrarlayan Sinir Ağı (Recurrent Neural Network)
YSA	: Yapay Sinir Ağı (Artificial Neural Network)

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

DERİN SİNİR AĞI YÖNTEMLERİ İLE WIKIPEDIA TOKSİK YORUM SINIFLANDIRMASI

ÖZET

Teknoloji'nin ilerlemesi ve internet erişiminin yayılması ile birlikte çevrim içi bilgi paylaşımı da ciddi oranda artış göstermiştir. Özellikle çevrim içi forum ve sosyal medya gibi platformlarda insanlar birbirleri ile bilgi paylaşımı yapıp özgürce yorum yapıp paylaşabilmektedirler. Bu sayede bilgiye ulaşmak çok daha kolay bir hale gelmiştir. Fakat bu platformlarda bilgi paylaşımının yanı sıra içinde bir ya da daha fazla bireyi rencide edecek yorumlar da paylaşılmaktadır. Bunun olumsuz bir sonucu olarak da faydalı paylaşım yapan kişilerin bir kısmının artık çevrim içi paylaşımlardan uzak durdukları görülmektedir. Çalışma kapsamında kullanılan veriler Kaggle internet sitesindeki bir yarışma için paylaşılan Wikipedia toksik yorum verilerinden alınmıştır. Veri setindeki rencide edici, nefret dolu, hakaret veya taciz içerikli yorumlar yarışma için 6 başlık altında sınıflandırılmıştır. Bu başlıklar "toxic", "severe toxic", "obscene", "threat", "insult" ve "identity hate"'dir.

Bu tez kapsamında; kelime gömme işlemi için Keras kütüphanesi, model içindeki ağırlıkları optimize etmek için Adam optimizasyon algoritması, toksik yorumların sınıflandırılması için LSTM, Bi-LSTM ve GRU algoritmaları kullanılmıştır. Doğruluk ve F1 skor metrikleri ile performans ölçümü yapılmıştır. Her bir yöntem 30 kez çalıştırılıp elde edilen doğruluk ve F1 skor metrikleri üzerinden istatistiksel olarak karşılaştırıldığında yöntemler arasında bir fark olmadığı gözlemlenmiştir.

Anahtar kelimeler: Derin öğrenme, doğal dil işleme, kelime gömme, metin sınıflandırma, wikipedia

WIKIPEDIA TOXIC COMMENT CLASSIFICATION VIA DEEP NEURAL NETWORKS

SUMMARY

With the technological developments and global Internet usage, online information sharing has increased considerably. Especially on platforms such as online forums and social media, people can share information with each other and comment freely on topics. Thus, it has become much easier to access information. However, insulting comments may be shared on these platforms apart from information sharing. As a negative result of this condition, it seems that some of who make useful shares, begin avoiding online sharing. The data used within the research has been collected from Wikipedia toxic comment data which was shared for a competition on Kaggle website. Insulting, hateful, offending or harassing comments have been classified under 6 titles. These titles are “toxic”, “severe toxic”, “obscene”, “insult”, “threat”, "identity hate".

Within the concept of this study, Keras library was used for word embeddings, Adam optimizer algorithm was used for optimizing weights in the models; LSTM, Bi-LSTM and GRU algorithms were used for classifying toxic comments. Performance measurements were carried out by accuracy and F1 score metrics. Each method was processed 30 times and when compared statistically on collected accuracy and F1 score metrics, it has been observed that there is no difference between the methods.

Keywords: Deep learning, natural language processing, word embeddings, text classification, wikipedia

BÖLÜM 1. GİRİŞ

Derin öğrenme, özellikle 2005 yılından sonra birçok alanda kullanılmaya başlanan bir makine öğrenimi yaklaşımıdır. Derin öğrenme ifadesi 2000 yılında Aizenberg ve arkadaşları tarafından tanıtılmıştır (Aizenberg ve ark. 2000). Ivankhnenko ve Lapa 1965'te ilk denetimli derin beslemeli öğrenme algoritmasını yayınlamışlardır (Ivakhnenko ve Lapa, 1965). Ivankhnenko'dan sonra ilk derin öğrenme mimarisi olan Neokogntiron 1979 yılında Fukushima tarafından önerilmiştir (Fukushima, 1979). Yann LeCun ve arkadaşları el yazısı rakamlarını sınıflandırmak için kıvrımlı ağlarla geri yayılımı birlikte uygulamıştır (LeCun ve ark., 1989).

Beynin yapısal ve işlemsel özelliklerinden faydalanarak geliştirilen yapay sinir ağları makine öğrenmesi alanında birçok konuda kullanılmıştır. Yapay sinir ağı (YSA) algoritmaları her ne kadar kullanışlı olsa da hesaplama maliyetlerinden dolayı destek vektör makineleri daha fazla tercih edilmiştir. 2000'lerin başında tekrar gözde bir alan olmaya başlayan yapay sinir ağları, teknolojinin gelişmesiyle, özellikle grafik işlemci birimlerinin (GPU) gelişmeleriyle birlikte daha popüler hale gelmiştir. GPU'lar görüntü işlemeden doğal dil işlemeye kadar birçok alanda kullanılmaktadır. Doğal dil işleme konusunda metin özetleme, sınıflandırma gibi problemlerde tekrarlayan sinir ağları (TSA), uzun kısa-süreli hafıza (LSTM) ve evrişimli sinir ağları (CNN) gibi derin öğrenme algoritmalarının başarılı bir şekilde kullanıldığı görülmüştür.

Doğal dil işleme (NLP) alanı, İngilizce ve diğer doğal dillerde yazılmış metinleri, insanlarda olduğu gibi yorumlama ve üretme yetenekleriyle işleyebilen bilgisayar yazılım sistemlerinin inşası ile ilgilidir. Metinleri az çok yüzeysel seviyelerde işleyen metin işleme veya kelime işlemeden farklı olarak, birçok NLP sistemi bir cümlede veya metnin tamamında yer alan anlamları çıkarmaya çalışır. Bu şekilde çıkarılan anlamlar, robot eylemleri gerçekleştirmek, bir veri tabanından bilgi almak veya

makine çeviri sistemleri durumunda, farklı bir doğal dilde eşdeğer bir çeviri üretmek gibi çeşitli görevler için kullanılabilir (Mahesh ve Nirenburg, 1995).

İnternetin gelişmesi ve milyonlarca belgenin internette bulunmasından dolayı NLP çalışmalarının önemi de artmıştır. Sosyal medyanın sık kullanımıyla NLP için duygu analizi gibi alanlar oluşmuştur. Microsoft, Apple ve Google gibi büyük firmalar bu alanda ciddi yatırımlar yapmaktadırlar.

1990'lı yıllara kadar dayanan kelimelerin vektör ile temsil edilmesinde, günümüzde NLP'deki en güçlü yöntemlerden birisi kelime gömmedir. Bu yöntem metin sınıflandırma, belge kümeleme, konuşma etiketlemenin bir parçası, adlandırılmış varlık tanıma, duygu analizi ve benzeri NLP görevleri için kullanılabilmektedir.

You ve ark. (2016), hapishanelerdeki kısa mesajların otomatik güvenlik denetimi için kısa mesajları güvenli ve güvensiz olarak sınıflandıran TSA modeli kullanılmıştır. TSA modeli ile ortalama olarak %92,7 doğruluk elde edilmiştir.

Purwarianti ve ark (2016) yaptıkları çalışmada, Endonezya dili için twitter yorumları şikayet sınıflandırması çalışması yapmışlardır. InaNLP olarak adlandırdıkları araç seti, cümle ayırıcı, simgeleştirme, sözcük normalleştirme, kelime türü etiketleyici, adlandırılmış varlık etiketleyici, tümcecik etiketleyici, sözdizimsel ayrıştırıcı ve anlamsal çözümleyiciden oluşmaktadır. Birçok araştırmacı dilden bağımsız bir NLP modülü oluşturmaya çalışsa da belirli bir dil özelliği eklemenin her zaman doğruluk, hız ve alan açısından daha iyi performans sağladığını belirtmişlerdir. Yalnızca sözcük özellikleri ve istatiksel makine çevirisi algoritmalarını kullanarak, InaNLP ile sınıflandırma çalışmalarında ortalama 0,892'lik bir F1 skoru elde etmişlerdir.

Bir diğer çalışmada, Young ve ark. (2019) olay raporlama ve olumsuz olay analizi alanındaki sınıflandırma görevleri için doğal dil işlemenin sistematik bir incelemesini yapmışlardır. Olay raporları, sağlık hizmetlerinde hatalar ve istenmeyen verileri toplamak için kullanılan raporlardır. Bu sistemin faydası, gelecekteki hastaların riskini azaltmak olarak raporlanmasına dayanmaktadır. Sistematik olmayan raporların daha

iyi sonuçlara dönüştürülmesini zorlaştıran olumsuzluklar vardır. Tüm raporlara aynı önemin verilmemesi, raporların zamanında okunmaması ve dikkatli incelenmemesi gibi durumların çıkan sonucun verimini azalttığını belirtmişlerdir. Sınıflandırma çalışması sayesinde olay türünün, ortaya çıkan zararın türü ve ciddiyeti veya olayın meydana gelmesini sağlayan faktörler daha doğru bir şekilde analiz edilmiştir.

Bir vektör uzayında sözcüklerin dağıtılmış temsilleri, benzer sözcükleri gruplayarak doğal dil işleme görevlerinde daha iyi performans elde etmek için öğrenme algoritmalarına yardımcı olur. Kelime temsillerinin en eski kullanımlarından biri, Rumelhart, Hinton ve Williams'a bağlı olarak 1986 yılına dayanmaktadır. Bu fikir o zamandan beri istatistiksel dil modellemesine başarı ile uygulanmıştır. Takip çalışması, otomatik konuşma tanıma ve makine çevirisi uygulamaları ve çok çeşitli NLP görevleri içerir (Mikolov ve ark., 2013).

Shaunak Varudandi (towardsdatascience.com) 2021'deki çalışmasında, Kaggle'daki Jigsaw ve Conversation AI tarafından düzenlenen Kelime Sınıflandırma Yarışması için paylaşılan Wikipedia yorum verilerini kullanarak sınıflandırma çalışması yapmıştır. Kelime gömme için *fasttext* modelinden faydalanan Shaunak, sınıflandırma için ise LSTM ve LSTM-CNN melez algoritmalarını kullanmıştır. Çevrim içi yorumlar tutarsız olabildiğinden, veriyi ilk olarak metin ön işleme prosedürlerinden (harf yerine rakam kullanılması gibi) geçirip daha anlamlı bir hale getirmiştir. Verideki tutarsız karakterler için normalleştirme kullanılmıştır. Ardından kelimenin kök haline inmek için Lemmatization yöntemi kullanılmıştır. Eğitim veri setinin %80'ini eğitim için, %20'lik kısmı da doğrulama için kullanılmıştır. Yapılan çalışmanın sonucunda LSTM algoritmasının, melez bir model olan LSTM-CNN algoritmasına göre daha performanslı olduğunu görülmüştür.

Siyuan Li 2018'deki toksik yorum sınıflama çalışmasında Wikipedia yorum verilerini kullanılarak kendi oluşturduğu derin öğrenme sınıflandırma modelini test etmek için oluşturduğu bu modeli diğer modellerle karşılaştırmıştır. Çalışmada kelime gömme için *word2vec* modelinden yararlanılmıştır. Sinir ağını eğitirken, minimum kayıp fonksiyonunu bularak parametreleri güncellemek için gradyan iniş yönteminden daha

verimli olduğu düşünülen stokastik gradyan iniş yönteminden bahsedilmiştir. Çalışmasında veri içindeki sınıf dengesizliğine de değinen Li, spamlerin kelime gömme ve modelin eğitimi için olumsuz etki yaptığına da değinmiştir. Metin ön işleme yapıldıktan sonra, Keras kütüphanesi kullanılarak yorum metnindeki en yüksek frekansa sahip sözcükler bulunup kelime dağarcığına eklemiştir. Bu işlem öncesinde spam temizliği yapmak için oluşturulan dağarcık için daha sağlıklı bir yapı oluşturmuştur. Li'nin modelinin ilk katmanı girdi/gömme katmanıdır. İkinci katman Bi-LSTM katmanıdır. Tekrarlayan sinir ağında 2 adet aktivasyon fonksiyonu kullanılmıştır. Eğitim setinin %10'u doğrulama seti olarak kullanılmıştır. Doğrulama kaybında erken durdurma da yapılmış ve böylece eğitim seti ile birebir aynı olmasının önüne geçilmiştir. LSTM katmanına sahip önerilen modelinin GRU algoritmasına göre daha performanslı çalıştığı gözlemlenmiştir. Uygulanan diğer modeller de performansı kısıtlı bir şekilde geliştirmiştir.

Aken ve ark. (2018) çalışmalarında, siber zorbalık, küfürlü dil, çevrimiçi taciz gibi sınıflandırma çalışmalarının önceden yapıldığı belirtirken, kendi çalışmalarında toksik yorum tespitine odaklanmışlardır. Diğer çalışmalardaki benzer yöntemlerin farklı çalışmalarda kullanılabileceğinden de bahsetmişlerdir. Kaggle'da yayınlanan Wikipedia veri seti üzerinde çalışma yapan ekip çevrimiçi yorumları 6 ana başlık altında sınıflandırmaya çalışmışlardır. Buna ek olarak Twitter API kullanılarak ve CrowdFlower çalışanları tarafından "nefret söylemi", "saldırgan ama nefret söylemi" ve "ne saldırgan ne de nefret söylemi" etiketleriyle açıklanan 24.783 tweet veri seti üzerinde de çalışmışlardır. Çalışmalarında kelime gömme için kullandıkları *FastText* ve *Glove* modellerini ayrı ayrı LSTM, Bİ-LSTM, GRU ve CNN algoritmalarına uygulamışlardır. En başarılı sonuç Ensemble öğrenme modeli ile alınmıştır. Ardından çift yönlü GRU en başarılı sonuçları vermiştir.

Bu tez kapsamında, derin öğrenme yaklaşımlarından LSTM, Bi-LSTM ve GRU'nun sınıflandırma performanslarının istatistiksel olarak karşılaştırılması yapılmıştır. Tez çalışmasının ikinci bölümünde genel bilgiler başlığı altında; yapar sinir ağları, doğal dil işleme, derin öğrenme algoritmaları ve optimizasyon algoritmalarından bahsedilmiştir. Üçüncü bölümde, tez kapsamında kullanılan yöntemlerin sınıflandırma

performanslarını göstermek için Kaggle internet sitesindeki bir yarışma için paylaşılan Wikipedia'da yazılan toksik yorumları içeren veriler kullanılmıştır. Dördüncü bölümde, araştırmaya ait bulgular gösterilmiştir. Son bölümde ise sonuçlar yorumlanarak, gelecek çalışmalara yol gösterecek açıklamalar yapılmıştır.

BÖLÜM 2. KAYNAK ARAŞTIRMASI

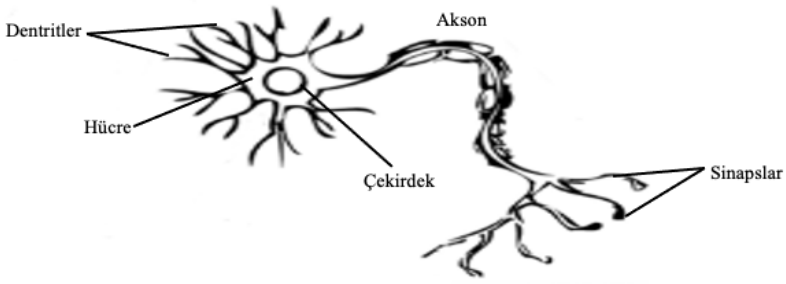
Bu bölümde yapay sinir ağları, kelime gömme, derin öğrenme yöntemlerinden ve ayrıca optimizasyon yöntemlerinden biri olan Adam optimizasyon algoritmasından bahsedilmiştir.

2.1. Yapay Sinir Ağları

Biyolojik sinir ağlarından esinlenen yapay sinir ağları, büyük sayıda ara bağlantıya sahip çok sayıda basit işlemciden oluşan büyük ölçüde paralel bilgi işlem sistemleridir. Yapay sinir ağı (YSA) modelleri, insan beyninde kullanıldığına inanılan bazı ilkeleri kullanmaya çalışır. Teknolojinin gelişmesiyle YSA'lar kullanılarak biyolojik sinir ağları modellenebilmektedir.

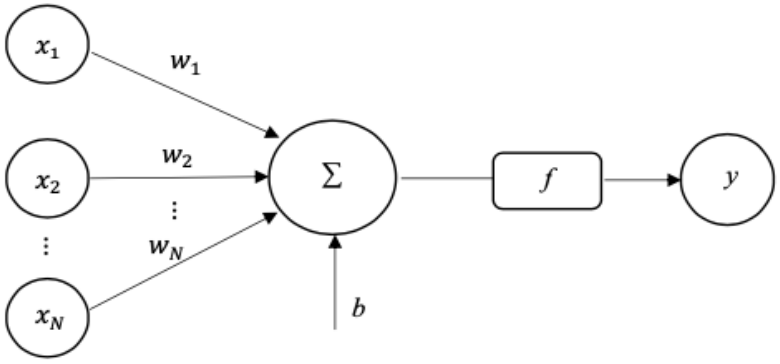
YSA'nın ilk matematiksel modelinin temelleri McCulloch ve Pitts'in 1943 yılında yayınladıkları makaleyle atılmıştır. YSA sınıflandırma, öngörü, ilişkilendirme, öğrenme, genelleme ve optimizasyon gibi farklı konularda kullanılmaktadır.

Nöron sinir ağının temel unsuruna denir. Şekil 2.1'de gösterildiği gibi bir nöron, dentritler, hücre ve akson olarak üç ana bölümden oluşur. Dentritler etrafındaki nöronlardan sinyal alan yapılardır. Sinyali bir nörondan diğerine ileten ince bir silindir yapısına sahip olan aksonun sonunda dentritlerle temas sinapslar aracılığı ile oluşur. Sinapstaki nöronların arasındaki sinyallerin birbirlerine pozitif veya negatif ağırlık etkisi vardır. Bir sinapstan nörona gelen dürtü sinyalinin aksona sinyal gönderebilmesi için sinapsların toplam ağırlık eşiğini aşması gerekmektedir..



Şekil 2.1. Biyolojik sinir hücresi yapısı (Meng ve ark., 2020)

Yapay sinir ağları temel olarak girdi katmanı, gizli katman ve çıktı katmanından oluşmaktadır. Giriş katmanı girdileri $X_1, X_2, \ldots, X_N$ şeklindedir. Gizli katmanda her girdi bağlantı ağırlık değerleriyle çarpılır. Girdi, ağırlık değerlerinin etkisiyle pozitif veya negatif yönde etkilenebilir. Girdiler ağırlık değerleri ile çarpıldıktan sonra ön bilgi (bias) ile toplanır ve elde edilen sonuç bir aktivasyon fonksiyonundan geçer. Aktivasyon fonksiyonundan geçen sonuç, çıktı katmanına aktarılır. Basit bir YSA mimarisi Şekil 2.2'de gösterilmiştir.



Şekil 2.2. Basit bir yapay sinir ağı yapısı

Yapay sinir ağı matematiksel olarak ifade edildiğinde;

- N : Girdi sayısı
- x : Girdi değeri
- w : Ağırlık
- b : Ön bilgi
- f : Aktivasyon fonksiyonu
- y : Çıkış değeri

olmak üzere,

$$y = f(\sum_{i=1}^{N} x_i w_i + b) \quad (2.1)$$

şeklinde ifade edilir (Zayegh ve Bassam, 2018).

2.2. Doğal Dil İşleme

Doğal dil işleme, bilgisayarların doğal dildeki metinleri veya konuşmayı yararlı şeyler yapmak için anlamak ve işlemek için nasıl kullanılabileceğini araştıran bir araştırma ve uygulama alanıdır. NLP araştırmacıları, bilgisayar sistemlerinin istenen görevleri yerine getirmek için doğal dilleri anlamasını ve manipüle etmesini sağlamak için uygun araç ve tekniklerin geliştirilebilmesi için insanların dili nasıl anladığı ve kullandığı hakkında bilgi toplamayı amaçlar. NLP'nin temelleri, bilgisayar ve bilgi bilimleri, dilbilim, matematik, elektrik ve elektronik mühendisliği, yapay zeka ve robotik ve psikoloji gibi bir dizi disiplinde yatmaktadır (Chowdhury, 2003).

2.2.1. Kelime Gömme

Kelime gömme, NLP'de kelimelerin sayısal vektör temsili haline dönüştürülmesini hedefleyen bir tekniktir. Bu sayede metinler veya dokümanlar, vektörler ile ifade edildiği bir uzay olarak düşünülebilir. Aranılan bir metni bu uzay düzleminde bularak, vektöre en yakın dokümanlara erişim sağlanabilir. Metinler ve dokümanlar arasındaki uzaklık kıyaslanabilir ve belgeler belli gruplara ayırılabilir.

2.2.2. Keras Kelime Gömme

Girdi verilerin her biri tamsayı olarak kodlanır ve kelimelerin hepsi bir tamsayı olarak ifade edilir. Bu veri hazırlama adımı Keras içinde bulunan *Tokenizer API* sayesinde sağlanır. Cümleleri kelimelere böler ve her birini tam sayılara kodlar. Rastgele ağırlıklarla başlatılan gömme katmanı, eğitim veri kümesindeki tüm kelimeler için gömmeyi öğrenir.

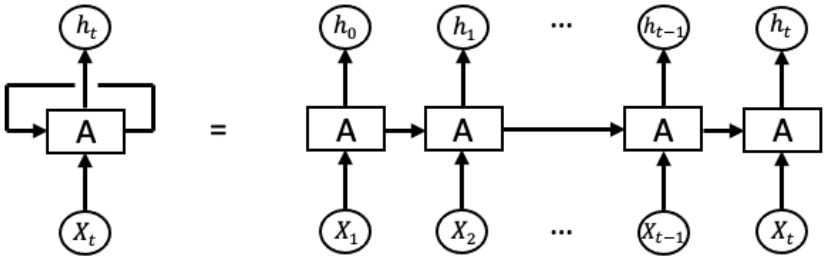
2.3. Derin Öğrenme Algoritmaları

Derin öğrenme, öğrenme sürecini yürütmek için yapay sinir ağlarının hiyerarşik seviyelerini kullanan bir yapay zeka yaklaşımıdır. Makine öğrenimi metotları içerisinde yer alan derin öğrenme, yapılandırılmamış veya etiketlenmemiş verilerden denetimsiz öğrenebilen ağlara sahip makine öğrenme mimarilerini kullanır (Bengio, 2009).

Sınıflandırma, doğal dil işleme, video analizi ve konuşma tanıma gibi birçok alanda kullanılan derin öğrenmede, teknolojinin de gelişmesiyle GPU'lar kullanılmaya başlanmıştır. GPU'lar, CPU'lara göre daha fazla hesaplama gücüne sahiptir.

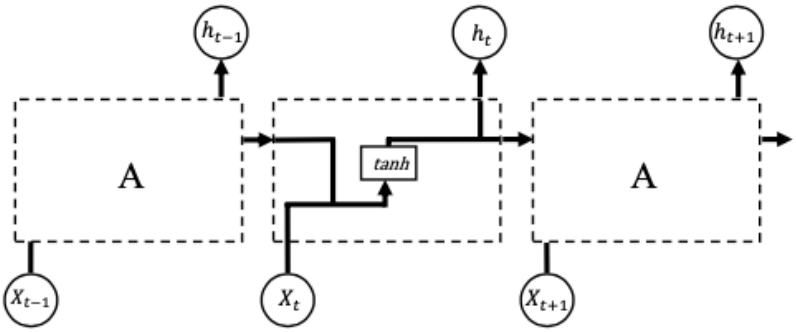
2.3.1. Tekrarlayan Sinir Ağları

1980'li yıllarda temelleri atılan bir yapay sinir ağı modeli olan tekrarlayan sinir ağları özellikle doğal dil işleme alanında dizi işleme özelliği olmasından dolayı sıklıkla tercih edilmektedir. Konuşma, metin, müzik, video gibi sıralı akışa sahip verilerde başarılı sonuçlar üreten TSA yapısında klasik sinir ağı yapısından farklı olarak ilerideki katmanlardan girdilere geri besleme yapılmaktadır. Ağ bu sayede önceki bilgilere bağlı bir çıktı üretmektedir. TSA yapısı Şekil 2.3'de verilmiştir.



Şekil 2.3. Bir TSA yapısı

TSA, Şekil 2.4'deki gibi üzerinde bilgiyi tekrar eden modüllerden oluşan bir zincire sahiptir. Standart tekrarlayan sinir ağında tekrar eden bu modüller tek bir *tanh* fonksiyonu kullanan katmandan oluşan basit bir yapıya sahiptir (Olah, 2015).

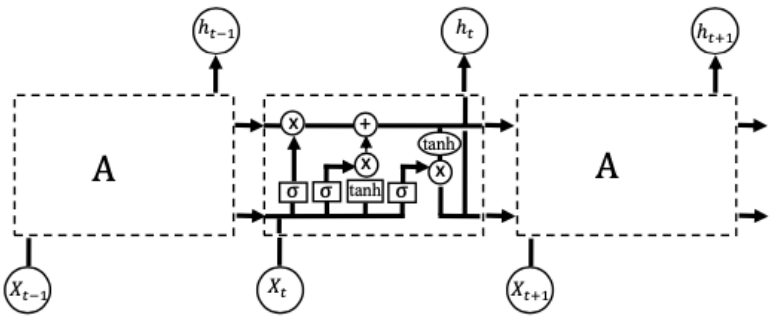


Şekil 2.4. Tek katmanlı standart bir TSA modeli (Olah, 2015)

TSA etkili bir modeldir ancak TSA ile iyi çıktılar alabilmek için uygun mimariyi belirleyebilmek önemlidir. TSA'nın eksi yönlerinden biri, kaybolan ve patlayan gradyan problemidir ve bu yüzden uzun süreli girdilerin öğrenilmesinde güçlük çekmektedir (Hermans ve Schrauwen, 2013).

2.3.2. Uzun Kısa-Süreli Hafıza

Uzun kısa-süreli hafıza (LSTM) ilk olarak Hochreiter ve Schmidhuber tarafından 1997'de tanıtılmıştır. LSTM mimarisi, standart bir TSA'da olduğu gibi her zaman adımı için bir dizi tekrarlanan modüle sahiptir. Uzun süreli bağımlılıkları öğrenebilen bir TSA türüdür. TSA gibi tek bir katmana değil, Şekil 2.5'deki gibi birbiri ile iletişime sahip 4 katmanlı bir yapıya sahiptir.

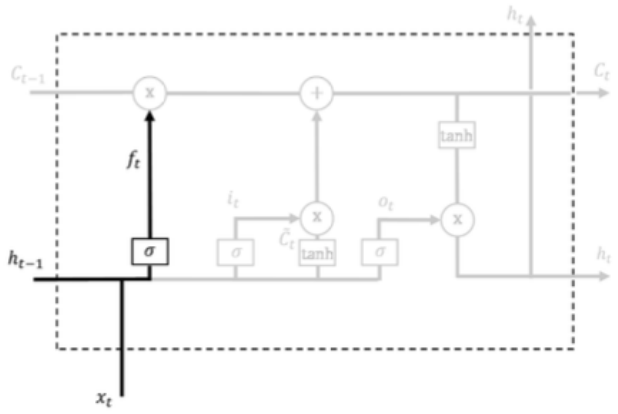


Şekil 2.5. Dört katmanlı LSTM modeli (Olah, 2015)

Tüm vektörler bir düğüm çıktısından diğerinin girdisine kadar her satırda taşınır. Modeldeki dikdörtgen kutular öğrenilmiş sinir ağı katmanlarını, daireler de noktasal işlemleri gösterir, vektör eklenmesi gibi.

LSTM'de ilk olarak, hücre durumundan atılacak bilgiler belirlenir. Unutma kapısı katmanı (forget gate layer) adı verilen bir sigmoid katman tarafından bu karar verilir. Unutma kapısı Şekil 2.6'da gösterilmiştir. h_{t-1} bir önceki blok çıktı değeri, x_t girdi değeri ve b_f ön bilgi değeri olmak üzere; f_t unutma kapısı fonksiyonu denklem 2.2 ile hesaplanır. Unutma kapısı, h_{t-1} ve x_t'ye bakarak C_{t-1} hücre durumundaki her sayı için 0 ve 1 arasında bir sayı verir. 1 değeri "bunu tamamen koru", 0 değeri "bundan tamamen kurtul" demektir. Hesaplamalar boyunca kullanılan $W_{f,x}$, $U_{f,\mathrm{h}}$, $W_{c,x}$, $U_{c,\mathrm{h}}$, $W_{i,x}$, $U_{i,\mathrm{h}}$, $W_{o,x}$, $U_{o,\mathrm{h}}$ değerleri ilgili kapıların ağırlık değerlerini ifade etmektedir.

$$f = \sigma\left(W_{f,x}x_t + U_{f,h}h_{t-1} + b_f\right) \tag{2.2}$$

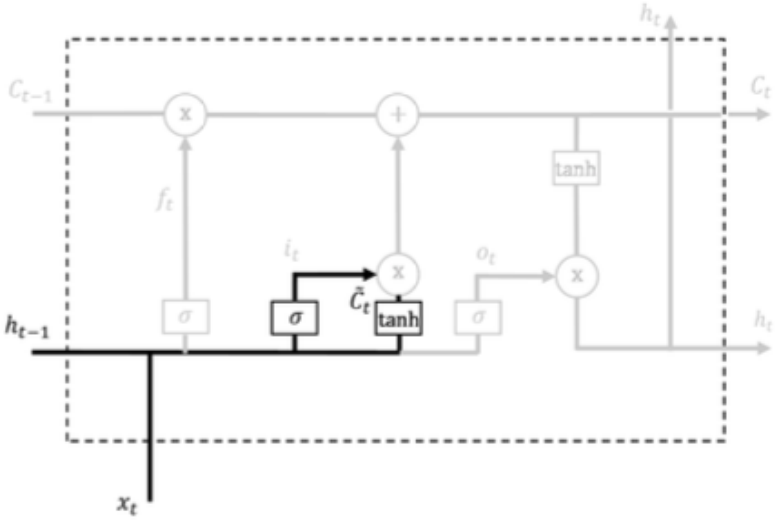


Şekil 2.6. LSTM unutma kapısı katmanı (Olah, 2015)

Daha sonraki adımda hücre durumunda hangi yeni bilgilerin saklanacağı kararı verilmektedir. İki aşamada gerçekleşen bu karar süreci, önce giriş kapısı katmanı (input gate layer) olarak adlandırılan sigmoid katmanı hangi değerlerin güncelleneceğine karar vermesiyle başlar. Daha sonra *tanh* katmanı hücre durumuna eklenebilecek yeni aday değerlerin ($\hat{\mathrm{C}}_t$) vektörünü oluşturur. Sonraki adımda Şekil 2.7'de gösterildiği gibi hücre durumunu güncellemek için bu ikisi birleştirilir. i_t giriş kapısı değeri, $\hat{\mathrm{C}}_t$ aday değer, b_i giriş ön bilgi ve b_c hücre ön bilgi değeri olarak denklem 2.3 ve 2.4'deki gibi hesaplanır.

$$i_t = \sigma\left(W_{i,x}x_t + U_{i,h}h_{t-1} + b_i\right) \tag{2.3}$$

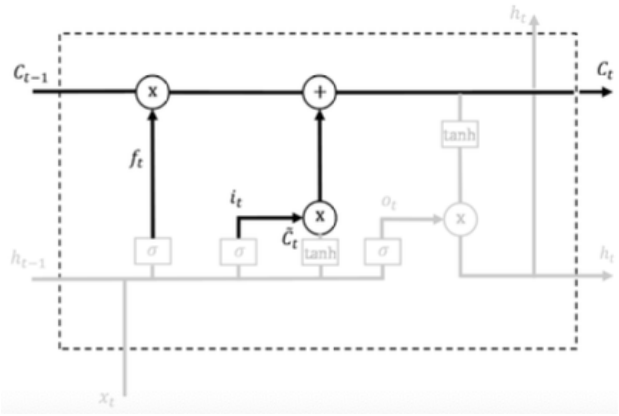
$$\hat{\mathrm{C}}_t = \tanh\left(W_{c,x}x_t + U_{c,h}h_{t-1} + b_c\right) \tag{2.4}$$



Şekil 2.7. LSTM'de yeni bilginin hücre durumunda saklanması (Olah, 2015)

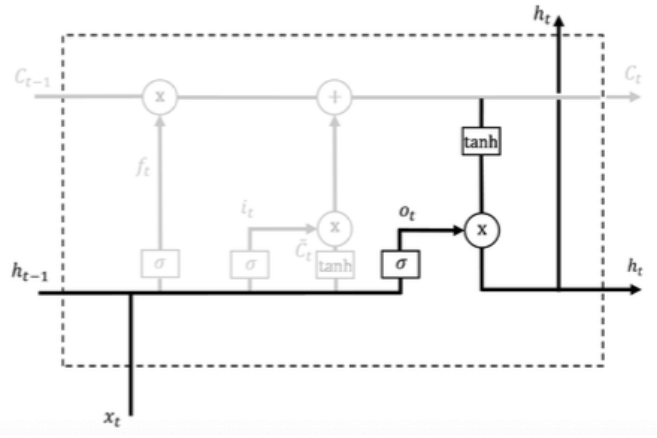
Burada; eski hücre durumu olan C_{t-1}, yeni hücre durumu olan C_t'ye güncellenmiş olur. $\otimes$ işlemi vasıtası ile, eski durum değeri C_{t-1} ile unutma kapısı değeri f_t çarpılır. Önceki adımda unutulmasına karar verdiğimiz bilgiler unutulur. i_t giriş kapısı değeri ile $\hat{C}_t$ yeni hücre aday değeri $\otimes$ işleminden geçirildikten sonra bu iki değer Şekil 2.8'de gösterildiği gibi toplanır. Yeni elde edilen bu değer, her bir durum değerinin ne kadar güncelleceğine karar verilen ve bir sonraki bloğa aktarılmak üzere hafızaya alınan değer olur.

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \hat{C}_t \tag{2.5}$$



Şekil 2.8. LSTM'de bilginin unutulması ve yeni bilginin aktarılması (Olah, 2015)

Son aşamada çıktının ne olacağına karar verilmesi gerekmektedir. Şekil 2.9'da çıktı değerinin oluşturulması gösterilmiştir. Çıktı, kapı durumuna bağlı olarak belirlenir ancak bu değer filtrelenmiş bir değerdir. İlk olarak hücre durumunun hangi kısımlarının çıktısının verileceğine karar verilmesini sağlayan bir sigmoid katman çalıştırılır. Sonrasında hücre durumuna *tanh* fonksiyonu uygulanarak değerlerin -1 ile 1 arasında olması sağlanır. Ardından sigmoid katman ile çarpılarak çıktının karar verilen kısmı için h_t değeri elde edilmiş olur (Olah, 2015).



Şekil 2.9. LSTM'de çıktı değerinin oluşturulması (Olah, 2015)

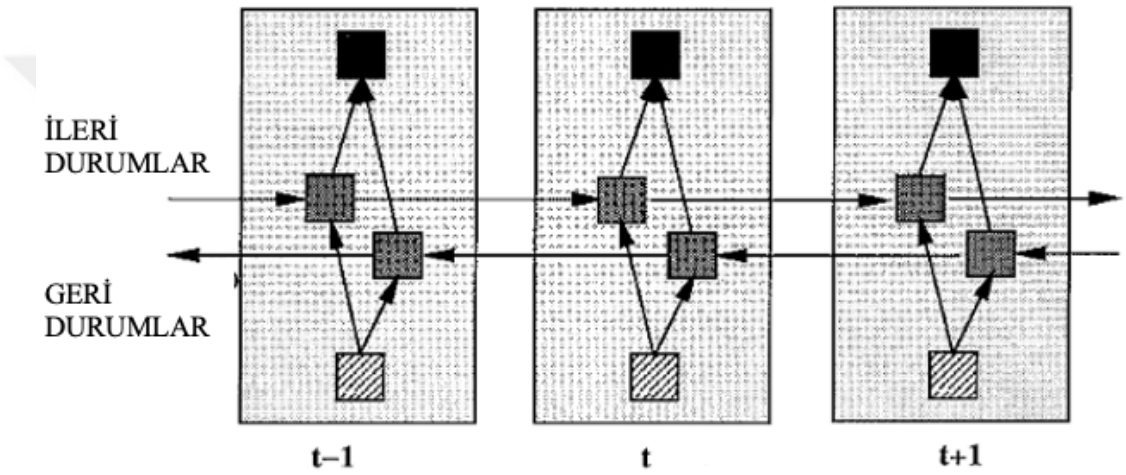
o_t ve h_t çıktı değeri, b_o ön bilgi çıktı değeri olmak üzere sırasıyla denklem 2.6 ve 2.7'deki gibi hesaplanır.

$$o_t = \sigma(W_{o,x}x_t + U_{o,h}h_{t-1} + b_o) \tag{2.6}$$

$$h_t = o_t \otimes \tanh(C_t) \tag{2.7}$$

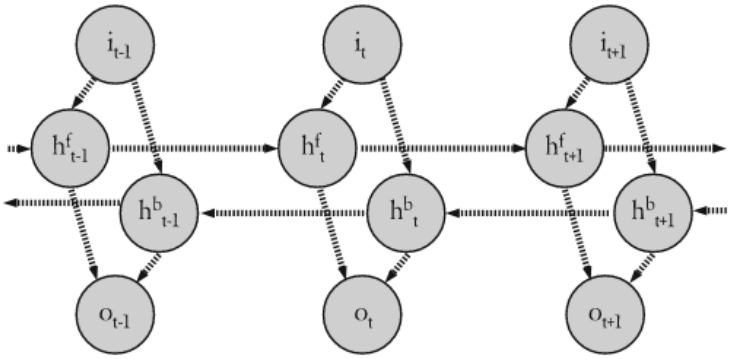
2.3.3. Çift Yönlü Uzun Kısa-Süreli Hafıza

TSA'nın bir başka türü, Schuster ve Paliwal tarafından belirli bir zaman içinde geçmiş ve gelecekteki tüm kullanılabilir girdi bilgilerini kullanarak ağı eğitmek için geliştirilen Çift Yönlü TSA'dır. Bu varyasyon TSA'nın sınırlamalarının üstesinden gelmek için 1997 yılında önerilmiştir. Şekil 2.10'da çift yönlü TSA gösterilmiştir.



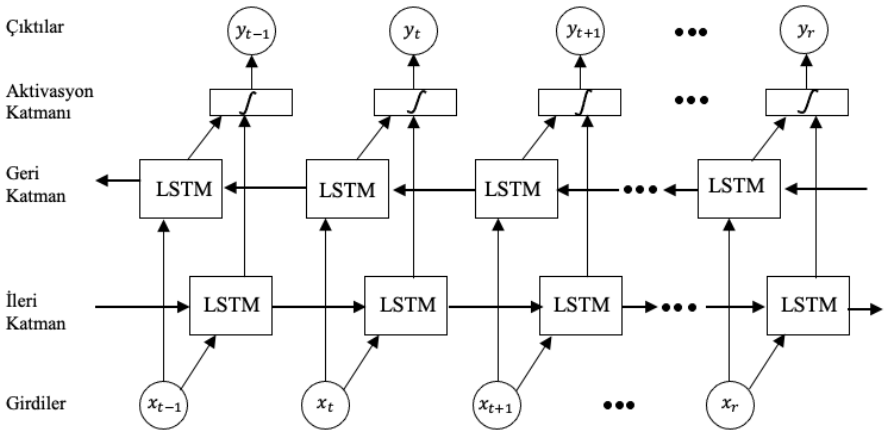
Şekil 2.10. Üç zaman adımı için zaman içinde açılmış gösterilen çift yönlü tekrarlayan sinir ağının (BRNN) genel yapısı (Schuster ve Paliwal, 1997)

Standart TSA'larla ilgili sorunlardan birisi, geçmişe erişimleri olup gelecekteki bilgilere erişimlerinin olmamasıdır. Bu sorun, iki ayrı çift yönlü TSA kullanılarak çözülebilir. İki ayrı tekrarlayan gizli katman girdi dizilerini zıt yönlerde tarar. Aynı çıktı katmanına bağlı iki gizli katman, bağlam bilgisine her iki yönde de erişime sahiptir. Ağın kullandığı bağlam bilgisi miktarı eğitim sırasında öğrenildiği için önceden belirtilmesi gerekmez. Bu sayede ileri ve geri bağlam birbirinden bağımsız bir şekilde öğrenilebilir. Basit bir çift yönlü ağ 2.11'de gösterilmiştir.



Şekil 2.11. Basit bir çift yönlü ağ yapısı: girdi *i*, çıktı *o*, ileri ve geri işleme için iki gizli katman (h^f ve h^b) (Wöllmer ve ark., 2010)

Çift yönlü ağların LSTM ile birleştirilmesi sonucu çift yönlü uzun kısa-süreli hafıza (Bi-LSTM) ortaya çıkar. Ses birimi tanıma, anahtar kelime saptama, el yazısı tanıma, gürültü modelleme ve konuşmadan duygu tanıma konusunda başarılı performanslar gösterebilen Bi-LSTM'de, iki farklı LSTM ağı eğitilir. İlk LSTM'de girdi olarak verilen bilgiler oldukları gibi kullanılır. İkinci LSTM ağında ise ilk girdi bilgilerinin ters çevrilmiş versiyonları kullanılarak eğitim gerçekleştirilir. Çift Yönlü LSTM yapısı Şekil 2.12'de gösterilmiştir.



Şekil 2.12. Çift Yönlü LSTM Yapısı

Bi-LSTM, aynı çıktıya zıt yönlerde iki gizli katmanı bağlar. İlk olarak ileriye dönük gizli diziyi $\vec{h}_t$ hesaplar, daha sonra geriye doğru gizli $\overleftarrow{h}_t$ dizisini hesaplar. Son olarak,

y_t çıktısını üretmek için $\vec{h}_t$ ve $\overleftarrow{h}_t$'yi birleştirir. h gizli durum LSTM blokları olduğunda, Bi-LSTM aşağıdaki fonksiyonlar uygulanarak bulunur.

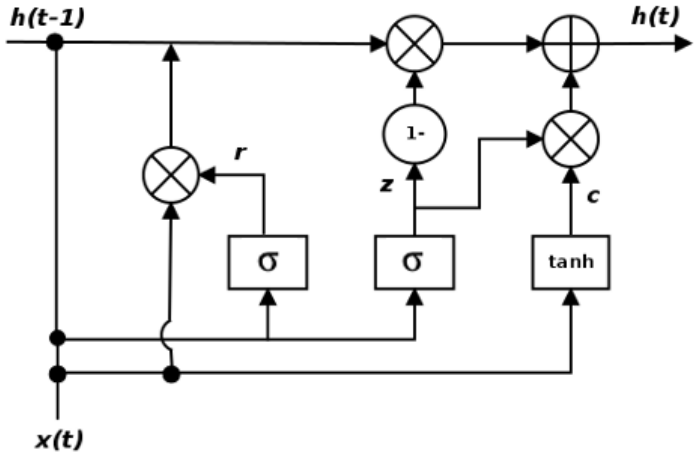
$$\vec{h}_t = H(W_{x\vec{h}}x_t + W_{\vec{h}\vec{h}}\vec{h}_{t-1} + b_{\vec{h}}) \tag{2.8}$$

$$\overleftarrow{h}_t = H(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}) \tag{2.9}$$

$$y_t = W_{\vec{h}y}\vec{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y \tag{2.10}$$

2.3.4. Geçitli Tekrarlayan Birimler

Geçitli tekrarlayan birim (GRU) ilk olarak Cho ve arkadaşları tarafından 2014 yılında, her bir tekrar eden birimin farklı zaman skalalarının bağımlılıklarını uyarlamalı olarak yakalattırmak için önerilmiştir. GRU, LSTM'ye benzer şekilde ünite içindeki bilgi akışını modüle eder. LSTM'den farklı olarak ayrı bir hafıza hücresine sahip olmayan geçit ünitelerine sahiptir. GRU yapısı Şekil 2.13'de gösterilmiştir.



Şekil 2.13. GRU yapısı

GRU'nun t anındaki h_t^j aktivasyonu, bir önceki aktivasyon olan h_{t-1}^j ve aday aktivasyon olan $\tilde{h}_t^j$ arasındaki doğrusal interpolasyon eğrisidir.

$$h_t^j = \left(1 - z_t^j\right)h_{t-1}^j + z_t^j\tilde{h}_t^j \tag{2.11}$$

Güncelleme kapısı z_t^j birimin aktivasyonunu veya içeriğini ne kadar güncelleyeceğine karar verir. Güncelleme kapısı denklem 2.12'deki gibi hesaplanır.

$$z_t^j = \sigma(W_r x_t + U_z h_{t-1})^j \tag{2.12}$$

Mevcut durum ile yeni hesaplanan durum arasında doğrusal bir toplam alma prosedürü LSTM ile benzer şekildedir. Bellek içeriğinin kontrollü olarak açığa çıkması GRU'da eksik olan bir özelliktir. LSTM'de ağdaki diğer birimler tarafından görülen veya kullanılan bellek içeriğinin miktarı çıkış kapısı tarafından kontrol edilirken, GRU'da herhangi bir kontrol olmadığından tüm içerik gösterilir.

Aday aktivasyon $\tilde{h}_t^j$ geleneksel tekrarlayan birime benzer şekilde hesaplanır.

$$\tilde{h}_t^j = \tanh\left(W x_t + U(r_t \odot h_{t-1})\right)^j \tag{2.13}$$

r_t bir dizi sıfırlama kapısıdır ve $\odot$ eleman bazında bir çarpmadır. r_t^j 0'a yakınken sıfırlama kapısı, birimi etkin bir şekilde girdi sırasının ilk sembolünü okuyormuş gibi hareket ettirir. Bunu bir önceki hesaplanmış durumu unutturarak yapar.

Sıfırlama kapısı r_t^j, güncelleme kapısı ile benzer şekilde hesaplanır (Chung ve ark, 2014).

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j \tag{2.14}$$

2.4. Optimizasyon Algoritmaları

Optimizasyon yöntemleri sayesinde, ağın ürettiği çıktı değeri ile gerçek değer arasındaki fark minimuma indirilebilir. Ağın ürettiği çıktı değeri ile gerçek değer arasındaki farkın minimum değeri bulunur. Bu minimum değer ile derin öğrenme uygulamalarında, öğrenme işleminin sağlıklı sonuçlanması sağlanır.

2.4.1. Adam Optimizasyon Algoritması

Kingma ve Ba tarafından 2014 yılında önerilen Adam, AdaGrad ve RMSProp algoritmalarının en iyi özelliklerini birleştirir. AdaGrad'ın seyrek gradyanlarla başa çıkması ve RMSProp'un sabit olmayan hedeflerle başa çıkma yetenekleri birleştirilmiştir.

Adam, uyarlanabilir bir öğrenme oranı yöntemidir, yani farklı parametreler için bireysel öğrenme oranlarını hesaplar. Adam, sinir ağının her ağırlığı için öğrenme oranını uyarlamak için birinci ve ikinci gradyan anlarının tahminlerini kullanır. Rastgele bir değişkenin N'ninci momenti, o değişkenin *n*'nin gücüne beklenen değeri olarak tanımlanır.

- α : Adım boyutu veya öğrenme hızı. Varsayılan değer 0,001 olarak alınmıştır.
- $\beta 1$: Birinci moment kestirimi için üstel azalma oranı. Varsayılan değer 0,9 olarak alınmıştır.
- $\beta 2$: İkinci moment kestirimi için üstel azalma oranı. Varsayılan değer 0,999 olarak alınmıştır.
- $f(\theta)$: θ parametresi ile stokastik hedef fonksiyonu : İlk parametre vektörü
- $\theta 0$: İlk parametre vektörü
- $m0$: Birinci moment vektörü
- $v0$: İkinci moment vektörü
- ϵ : Sıfıra bölünmeyi önlemek için kullanılan değer, epsilon. Varsayılan değer $10{-}8$ olarak alınmıştır.

Ağırlıkların güncellenme adımları;

t zaman adımında stokastik hedefe göre gradyanların elde edilmesi,

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \tag{2.15}$$

yanlı birinci moment kestiriminin güncellenmesi,

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.16}$$

yanlı ikinci ham moment kestiriminin güncellenmesi,

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.17}$$

yanlılığı düzeltilmiş birinci moment kestiriminin hesaplanması,

$$m'_t = \frac{m_t}{1 - \beta_1^t} \tag{2.18}$$

yanlılığı düzeltilmiş ikinci moment kestiriminin hesaplanması,

$$v'_t = \frac{v_t}{1 - \beta_2^t} \tag{2.19}$$

Parametrelerin güncellenmesi,

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{v'_t + \epsilon}} m'_t \tag{2.20}$$

şeklinde gerçekleştirilir (Kingma ve Ba, 2017).

BÖLÜM 3. MATERYAL VE YÖNTEM

Bu tez çalışmasında, Bi-LSTM, LSTM ve GRU modelleri için Python programlama dili kullanılmıştır. Kodu çalıştırmak için GPU desteği sağlayan ve birçok Python kütüphanesini içinde barından Google Colaboratory (Colab) kullanılmıştır. Colab üzerinde model çalıştırılırken Python 3.7 versiyonu kullanılmıştır.

3.1. Kullanılan Veri Seti

Bu çalışmada, Google Jigsaw tarafından Aralık 2017'de Kaggle'da düzenlenen "Toksik Yorum Sınıflandırma" yarışması için herkese açık bir şekilde yayımlanan Wikipedia tartışma sayfalarından toplanan veri seti üzerinde çalışılmıştır. Değerlendiriciler tarafından 'toxic', 'severe toxic, 'insult', 'threat', 'obscene' ve 'identity hate' olarak 6 sınıf açıklanmıştır. Yorumlar birden fazla sınıfa girebilmektedir. Jigsaw, altı sınıf için resmi bir tanım yayınlamamıştır fakat toxic bir yorumu, kaba, saygısız veya mantıksız bir yorum olarak tanımlamışlardır. Çalışmada bu veri seti ile Bİ-LSTM, LSTM ve GRU modellerinin performansları incelenecektir.

3.2. Kullanılan Programlama Dili Kütüphaneleri

TensorFlow, Google tarafından geliştirilmiştir. Makine öğrenimi uygulamaları için kullanılan açık kaynaklı bir kütüphanedir. TensorFlow, görüntü tanıma, doğal dil işleme, el yazısı sınıflandırma, sözcük yerleştirme, TSA ve makine çevirisi için derin sinir ağlarını eğitir ve çalıştırabilir. TensorFlow, geliştiricilerin veri akış grafikleri oluşturmalarını sağlar. Grafikteki her düğüm matematiksel bir işlemi temsil eder ve düğümler arasındaki her bağlantı çok boyutlu bir veri dizisi veya tensör ile ifade edilmektedir. Bu yapılar programcıya Python dili aracılığıyla sağlanmaktadır. TensorFlow'daki düğümler ve tensörler Python nesneleridir ve TensorFlow uygulamaları da Python uygulamalarıdır.

Keras, Python için geliştirilmiş bir derin öğrenme kütüphanesidir ve derin öğrenme modellerinin oluşturulması ve eğitilmesi için uygun ortam sağlar. Aynı kodun değiştirilmeden hem CPU'da hem de GPU'da çalışmasını sağlar. TensorFlow ve Theano kütüphaneleri ile uyumlu ve derin öğrenme modellerinin prototipinin oluşmasını hızlıca sağlayan API'ye (Uygulama Programlama Arayüzü) sahiptir. Evrişimli ağlar, tekrarlayan ağlar ve her ikisinin beraber kullanımı için önceden tanımlı desteğe sahiptir. Birçok farklı ağ yapısını, çoklu girdi ya da çoklu çıktıyı, katman veya model paylaşımını uygulayabilmek mümkündür. Keras, MIT lisansı (Massachusetts Teknoloji Enstitüsü tarafından hazırlanan açık kaynak ve özgür yazılım lisansı) ile dağıtılmaktadır ve bu sayede ticari projeler dahil her yerde serbest olarak kullanılabilmektedir (Chollet, 2018).

NumPy (Numerical Python), çok boyutlu diziler ve matrislerle çalışmayı sağlayan ve matematiksel işlemler yapabilen açık kaynak kodlu bir Python kütüphanesidir.

Pandas, veri yapıları sunan ve veri üzerinde analiz yapmayı sağlayan kullanımı kolay, hızlı ve esnek bir açık kaynak kodlu Python kütüphanesidir.

Matplotlib, statik, animasyonlu ve etkileşimli görseller oluşturmak için kapsamlı bir Python kütüphanesidir. Çoğunlukla 2D grafikler ve NumPy kütüphanesi için bir çizim kütüphanesidir.

Seaborn, matplotlib tabanlı bir Python veri görselleştirme kütüphanesidir. Çekici ve bilgilendirici istatistiksel grafikler çizmek için üst düzey bir arayüz sağlar.

Sklearn veya *Scikit-learn*, makine öğrenmesi modelleri oluşturmada kullanılan, regresyon, kümeleme ve sınıflandırma için kullanılan pek çok öğrenme algoritmasına sahip veri işlemede kullanılan ve NumPy ile uyumlu çalışabilen bir Python kütüphanesidir.

3.3. Kullanılan Yöntemler

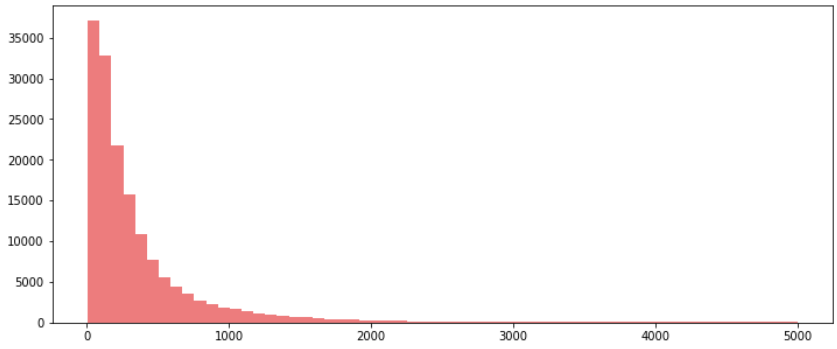
Tez kapsamında kullanılan yöntemlere ait algoritma adımları aşağıdaki gibidir.

Adım 1. *Veri setini yükle ve incele*

Materyaller bölümünde belirtilen Python kütüphaneleri yüklendikten sonra veri seti, *Pandas* kütüphanesi sayesinde csv formatında sisteme yüklenmiştir. Ardından yorumların ilk 5 satırı Şekil 3.1'de gösterildiği gibi yazdırılmıştır ve *comment_text* sütunundaki cümlelerin karakter uzunluğu büyükten küçüğe Şekil 3.2'de histogram grafiği şeklinde gösterilmiştir. Veriyi daha iyi anlamak için bazı istatistiksel veriler çıkarılmış ve incelenmiştir.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

Şekil 3.1. Veri seti içindeki ilk 5 yorum



Şekil 3.2. Cümlelerin uzunluğunun grafiksel gösterimi

Adım 2. *Eksik gözlem sayısı kontrolü*

Veri setinde eksik gözlem olup olmadığı print(train.comment_text.isna().sum()) kodu ile kontrol edilir. Kontrol sonunda eğitim veri setinde eksik veri olmadığı görülmüştür.

Adım 3. *Veri setinin bağımlı ve bağımsız değişkenlerinin birbirinden ayrılması*

Şekil 3.3'de *X* değeri (bağımsız değişken) modele girecek girdi değerini temsil eder. Eğitim veri setinde girdi olarak "comment_text" sütununun alınacağı gösterilmiştir. *Y* değeri (bağımlı değişken, hedef) de tahmin edilecek sütunların gösterimidir. Yani çıktı değerleri.

```
X = train.comment_text
y = train[["toxic", "severe_toxic", "obscene", "threat", "insult", "identity_hate"]].values
```

Şekil 3.3. Girdi ve çıktı değerleri

Adım 4. *Veri setindeki cümlelerin modele uygun bir şekilde dönüştürülmesi*
Elimizdeki string cümlelerin alınıp tokanizer api kullanılarak kelime kelime ayrılması, daha sonra *fit_on_text* ve *text_to_sequence* fonksiyonları kullanılarak her cümleye ait kelimelerin sıklıklarına göre tamsayı değerlerinin atanması sağlanır. Bu tam sayı değerlerinde küçük olan daha sık geçtiği anlamına gelmektedir. Bu bizim modele vereceğimiz cümleleri bir liste içinde Şekil 3.5'deki gibi diziler (array) halinde tutmaktadır.

```
0         Explanation\nWhy the edits made under my usern...
1         D'aww! He matches this background colour I'm s...
2         Hey man, I'm really not trying to edit war. It...
3         "\nMore\nI can't make any real suggestions on ...
4         You, sir, are my hero. Any chance you remember...
                              ...
159566    ":::::And for the second time of asking, when ...
159567    You should be ashamed of yourself \n\nThat is ...
159568    Spitzer \n\nUmm, theres no actual article for ...
159569    And it looks like it was actually you who put ...
159570    "\nAnd ... I really don't think you understand...
Name: comment_text, Length: 159571, dtype: object
```

Şekil 3.4. Veri seti özet gösterim

```
[[688, 75, 1, 126, 130, 177, 29, 672, 4511, 12052,
```

Şekil 3.5. Dizi (array)'ler halinde tutulan tam sayı değeri atanmış kelimeler

Üstteki Şekil 3.4'deki indeks 0'daki cümlenin kelimelerine atanan tam sayılar Şekil 3.5'de görülebilir. "the" kelimesi en fazla geçtiği için 1 tam sayısı atanmıştır.

Daha sonra bu dizi(array)'leri modele verebilmek için *pad_sequence* fonksiyonu kullanarak her cümleyi belirlenilen *max_len* parametresi ile standartlaştırılmıştır. Her cümle 150 karakter boyutuna gelecek şekilde başına 0 koyarak Şekil 3.6'daki hale gelmiştir. Veri setimiz boyutu 159571,150 şeklinde modelimize sokulacak hale getirilmiştir.

```
[[    0     0     0 ...  4583  2273   985]
 [    0     0     0 ...   589  8377   182]
 [    0     0     0 ...     1   737   468]
 ...
 [    0     0     0 ...  3509 13675  4528]
 [    0     0     0 ...   151    34    11]
 [    0     0     0 ...  1627  2056    88]]
```

Şekil 3.6. Standart hale gelen cümleler

Adım 5. *Hiperparametrelerin ayarlanması*

Modelin doğru ve istenilen sonuçlar vermesi için model için önemli olan parametrelerin ayarlanmasını içerir. Ayarlanan bazı hiperparametreler aşağıda listelenmiştir.

num_words = Kelime sıklığına bağlı olarak tutulacak maksimum kelime sayısıdır. Biz modelimizde 20000 değerini atadık.

max_len = Metin verisi içindeki kelime haznesinin boyutudur. Biz modelimizde 150 değerini atadık.

emb_size = Kelimelerin gömüleceği vektör uzayının boyutudur. Her kelime için bu katmandan çıktı vektörlerinin boyutunu ayarlar. Biz modelimizde 128 değerini atadık.

Adım 6. *F1 skor değerinin hesaplanması*

F1 skor hesaplaması düzensiz veri tiplerinde doğruluk metriğine göre daha doğru sonuçlar vermektedir. F1 skor hesaplaması detaylı bir şekilde aşağıda anlatılmıştır. Keras kütüphanesinde *backend* fonksiyonu kullanılarak hesapla işlemi Şekil 3.7'de gösterilmiştir.

Doğruya doğru demek (True Positive – TP, Doğru Pozitif)

Yanlışa yanlış demek (True Negative – TN, Doğru Negatif)
Doğruya yanlış demek (False Positive – FP, Yanlış Pozitif)
Yanlışa doğru demek (False Negative – FN, Yanlış Negatif)

Kesinlik (Precision): Kesinlik, TP sayısının TP ve FP sayısına bölümüdür. Başka bir deyişle, tahmin edilen pozitif sınıf değerlerinin, tahmin edilen toplam pozitif sayısına bölünerek hesaplanan orandır. Kesinlik, bir sınıflandırıcının tutarlılığının bir ölçüsü olarak düşünülebilir. Düşük bir tutarlılık aynı zamanda çok sayıda FP'yi de gösterebilir.

$$Kesinlik = \frac{TP}{TP+FP} \tag{2.21}$$

Duyarlılık (Recall): Duyarlılık, TP sayısının TP ve FN sayısına bölünmesidir. Başka bir deyişle, test verilerindeki pozitif sınıf değerlerinin sayısına bölünen pozitif tahmin sayısıdır. Duyarlılık veya doğru pozitif oran olarak da adlandırılır.

Duyarlılık, sınıflandırıcıların eksiksizliğinin bir ölçüsü olarak da düşünülebilir. Düşük bir duyarlılık, birçok FN olduğunu gösterir.

$$Duyarlılık = \frac{TP}{TP + FN} \tag{2.22}$$

F1 Skor: F skor veya F ölçüsü olarak da adlandırılan F1 skor, kesinlik ve duyarlılık arasındaki dengeyi aktarır.

$$F1\ Skor = 2 \times \frac{Kesinlik * Duyarlılık}{Kesinlik + Duyarlılık} \tag{2.23}$$

```
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

Şekil 3.7. F1 Skor hesaplamak için kullanılan kod

Adım 7. *Model oluşturulması*

Adım 4'te ayarlanan hiperparametreler ile önce sinir ağının girdi katmanı oluşturulur. Daha sonra tamsayılara dönüştürülen değerleri alıp Şekil 3.8'de gösterilen gömme (embedding) katmanında sabit boyutlu yoğun vektörlere dönüştürülür.

```
def create_model(type='LSTM', bidirectional=True):
    inp = Input(shape = (max_len, ))
    layer = Embedding(num_words, emb_size)(inp)
```

Şekil 3.8. Embedding katmanı kod bloğu

Daha sonra modelde kullanılacak olan tüm algoritmalar belirlenir. Bu tez kapsamında LSTM, GRU ve Bi-LSTM modellerinin kullanılabilmesi için Şekil 3.9'da gösterilen kod bloğu oluşturulmuştur. Birden fazla katman olduğu için *return_sequences*, "True" olarak tanımlanmıştır.

```
if type=='GRU' and bidirectional==False:
  layer = GRU(50, return_sequences = True, recurrent_dropout = 0.15)(layer)
elif type=='LSTM' and bidirectional==True:
  layer = Bidirectional(LSTM(50, return_sequences = True, recurrent_dropout = 0.15))(layer)
else:
  layer = LSTM(50, return_sequences = True, recurrent_dropout = 0.15)(layer)
```

Şekil 3.9. Farklı modellerin seçilmesi

Modele girecek olan veri setindeki düşük seviye özelliklerin çıkarılmasına yardımcı olmak için *GlobalMaxPooling* katmanı eklenir. Böylece verideki maksimum değerler alınmış olur. Bir çeşit filtreleme işlemi gerçekleştirilir. Sonrasında modeli daha sağlam

bir hale getirmek ve aşırı uyumu engellemek için modele *Dropout* katmanı eklenir ve her seferinde rastgele bir şekilde nöronların eğitilmesi sağlanır. Bu tez çalışmasında nöronların rastgele %20'sinin çıkarılması sağlanmıştır.

Daha sonra oluşturulan parametrelerin doğrusal bir şekilden doğrusal olmayan bir şekle dönüştürülmesi için gizli katman eklenir ve bir aktivasyon fonksiyonundan geçirilir.

Aktivasyon fonksiyonu olarak doğrultulmuş doğrusal birim (ReLU) fonksiyonu kullanılmıştır. ReLU aktivasyon fonksiyonu girdi pozitif ise doğrudan ileten, negatif olduğu durumlarda da sıfır çıktısı veren parçalı bir doğrusal fonksiyondur. Kaybolan gradyan probleminin üstesinden gelmesine, modellerin hızlı öğrenmesine ve iyi performans göstermesine olanak tanıdığı için sıklıkla kullanılır (Brownlee, 2020).

ReLU aktivasyon fonksiyonundan geçen ağırlıklar bir *Dropout* katmanından daha geçerek tekrardan rastgele nöronların %20'si çıkarılmıştır ve aşırı uyum olması engellenmiştir.

Son olarak oluşturulmuş olan parametreleri bir çıktı katmanı oluşturarak sigmoid fonksiyonundan geçirilerek nihai sonuçlar elde edilir. Şekil 3.10'da oluşturulan modelin özeti gösterilmektedir.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150)]             0
_________________________________________________________________
embedding (Embedding)        (None, 150, 128)          2560000
_________________________________________________________________
gru (GRU)                    (None, 150, 50)           27000
_________________________________________________________________
global_max_pooling1d (Global (None, 50)                0
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 50)                2550
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 306
=================================================================
Total params: 2,589,856
Trainable params: 2,589,856
Non-trainable params: 0
_________________________________________________________________
```

Şekil 3.10. Oluşturulan modelin özeti

Adım 8. *Optimizasyon yöntemi ve erken durdurma kriteri belirleme*
Optimizasyon yöntemi olarak Adam optimizasyon algoritması kullanılmıştır. Model eğitilirken örnek sayısını belirten *batch_size*=32, döngü sayısını belirten *epoch*=3 olarak tanımlanmıştır. Eğitim veri setinin test için ayrılan kısmı yani *test_size*=0,3 olarak tanımlanmıştır.

Erken durdurma kriterleri için parametreler aşağıdaki gibidir:

Monitör = 'val_loss', her döngüdeki kayıp değerleri model adımlarında gösterir.
patience, eğitimin durdurulmasına karar veren iyileştirme olmayan döngü sayısını gösterir.
verbose, erken durdurmanın hangi döngüde gerçekleştiğini gösterir.
min_delta, iyileştirme olarak nitelendirilecek olan minimum değeri belirtir.

BÖLÜM 4. ARAŞTIRMA BULGULARI

Bi-LSTM, LSTM ve GRU algoritmalarının 30 çalıştırma sonunda elde edilen sonuçları Tablo 4.1'de gösterilmiştir. F1 skor ve doğruluk metriklerine göre çeyreklikler arası açıklık, standart sapma, minimum ve maksimum değerler de hesaplanmıştır.

Tablo 4.1. Sonuçlar ve karşılaştırmalar

	Bi-LSTM		**LSTM**		**GRU**	
	Doğruluk	**F1 Skor**	**Doğruluk**	**F1 Skor**	**Doğruluk**	**F1 Skor**
1	0,9941	0,6617	0,9938	0,6552	0,9942	0,6606
2	0,9943	0,6688	0,9943	0,6568	0,9943	0,6571
3	0,9939	0,6647	0,9941	0,6578	0,9937	0,6546
4	0,9936	0,6849	0,9940	0,6419	0,9932	0,6678
5	0,9942	0,6631	0,9942	0,6415	0,9942	0,6594
6	0,9931	0,6672	0,9936	0,6604	0,9936	0,6626
7	0,9942	0,6552	0,9943	0,6553	0,9943	0,6386
8	0,9941	0,6385	0,9941	0,6430	0,9941	0,6341
9	0,9945	0,6648	0,9940	0,6624	0,9941	0,6594
10	0,9947	0,6557	0,9947	0,6578	0,9947	0,6582
11	0,9939	0,6629	0,9947	0,6516	0,9947	0,6507
12	0,9944	0,6621	0,9942	0,6662	0,9939	0,6592
13	0,9938	0,6464	0,9944	0,6474	0,9938	0,6503
14	0,9943	0,6656	0,9939	0,6543	0,9943	0,6546
15	0,9926	0,6291	0,9931	0,6214	0,9909	0,6206
16	0,9935	0,6719	0,9935	0,6433	0,9935	0,6363
17	0,9931	0,6720	0,9922	0,6574	0,9895	0,6629
18	0,9942	0,6529	0,9941	0,6636	0,9942	0,6633
19	0,9934	0,6752	0,9936	0,6613	0,9936	0,6588
20	0,9940	0,6704	0,9941	0,6661	0,9941	0,6726
21	0,9912	0,6672	0,9940	0,6624	0,9937	0,6534
22	0,9939	0,6486	0,9940	0,6497	0,9940	0,6424
23	0,9947	0,6678	0,9947	0,6605	0,9947	0,6590
24	0,9938	0,6628	0,9938	0,6585	0,9936	0,6587
25	0,9942	0,6449	0,9942	0,6064	0,9942	0,6267
26	0,9939	0,6752	0,9939	0,6763	0,9939	0,6794
27	0,9927	0,6396	0,9936	0,6408	0,9933	0,6420
28	0,9939	0,6686	0,9944	0,6542	0,9945	0,6375
29	0,9945	0,6715	0,9944	0,6682	0,9945	0,6740
30	0,9935	0,6412	0,9948	0,6570	0,9948	0,6504
Ort	0,9938	**0,6606***	**0,9940***	0,6532	0,9938	0,6535
SS	0,00072	0,01292	0,00051	0,01381	0,00108	0,01369
IQR	0,00067	0,01527	0,00047	0,01312	0,00067	0,01592
Min	0,9912	0,6291	0,9922	0,6064	0,9895	0,6206
Max	0,9947	**0,6849***	**0,9948***	0,6763	**0,9948***	0,6794

* En iyi değerleri gösterir.

F1 skor ve doğruluk metrikleri için en iyi değerler ayrı ayrı değerlendirilmiştir.

Her modelin 30'ar defa çalıştırılmasının ardından, Bi-LSTM öğrenme adımında 3 defa erken durdurma parametrelerine takılarak 3. eğitim adımına geçemezken, LSTM ve GRU'da 1'er defa erken durdurma olmuş ve 3. adıma geçememişlerdir.

30'ar çalıştırma sonucunda, en düşük doğruluk değeri GRU (0,9895)'da, en düşük F1 skor değeri ise LSTM (0,6064)'de elde edilmiştir. En yüksek F1 skor Bi-LSTM (0,6849) ile elde edilirken, en yüksek doğruluk değerleri GRU (0,9948) ve LSTM (0,9948) algoritmalarında elde edilmiştir. Ortalama doğruluk ve F1 skor değerleri ele alındığında en başarılı doğruluk LSTM (0,9940) ile elde edilirken, en başarılı F1 skor değeri ise Bi-LSTM (0,6606) ile elde edilmiştir.

Doğruluk ve F1 skor değerlerinin standart sapmalarına bakıldığında ise doğruluk metriğine göre en düşük sapma LSTM (0,00051), F1 skora bakıldığında ise en düşük sapma Bi-LSTM (0,01292)'de gerçekleşmiştir. Çeyreklikler arası açıklık değerlerine bakıldığında doğruluk ve F1 skor metriklerinde LSTM en düşük açıklık göstermiştir (0,00047 ve 0,01312).

4.1. İstatistiksel Karşılaştırmalar

30'ar çalıştırma sonrasında 3 yöntem için elde edilen toplamda 90'ar gözlemden oluşan doğruluk ve F1 skor grupları bakımından yöntemler arasında farklılık olup olmadığı test edilmiştir. Öncelikle hem doğruluk hem de F1 skor gruplarının normal dağılım gösterip göstermediği kontrol edilmiştir. Normal dağılım testi sonucu tablo 4.2'de gösterildiği gibidir.

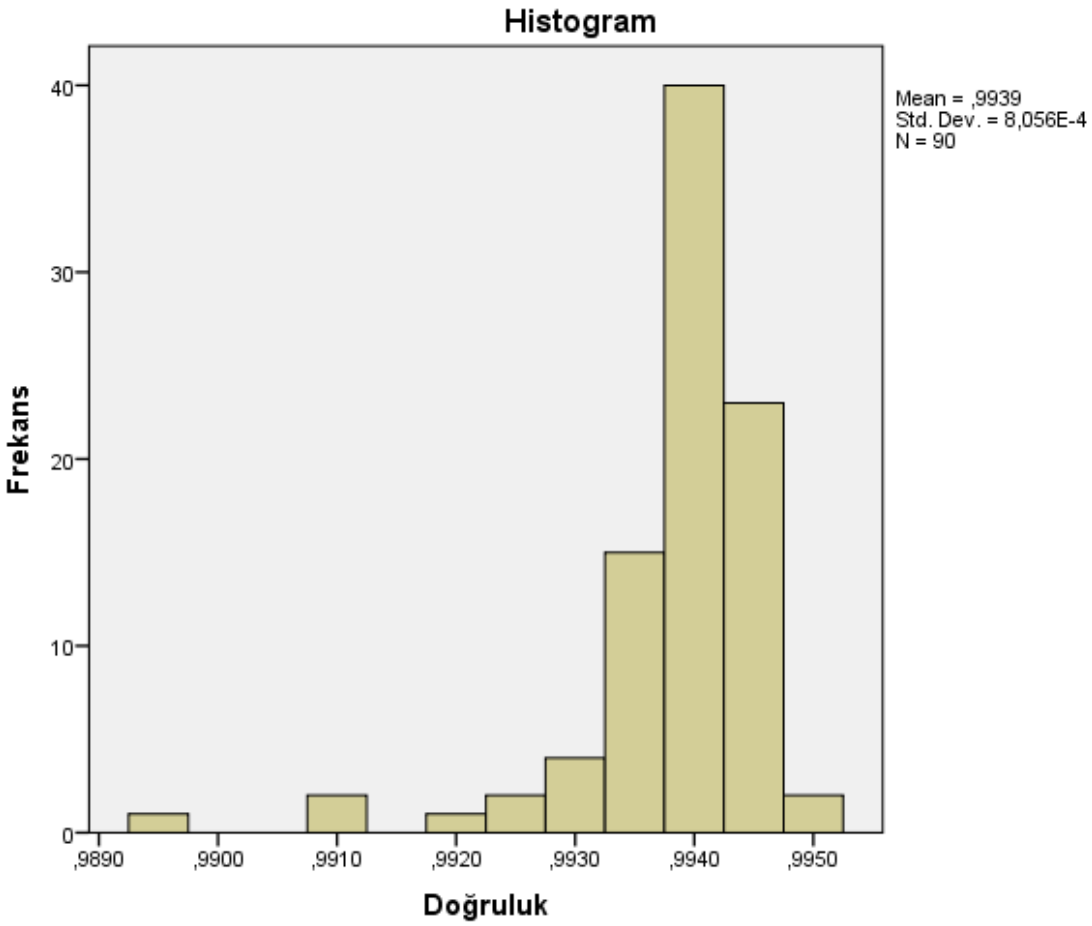
Grupların normal dağılım gösterip göstermediğinin testi Kolmogorov-Smirnov testi ile yapılmış (α=0,05) ve kurulan hipotezler aşağıda verilmiştir.

H_0 = Gruplar normal dağılımlıdır.
H_1 = Gruplar normal dağılım değildir.

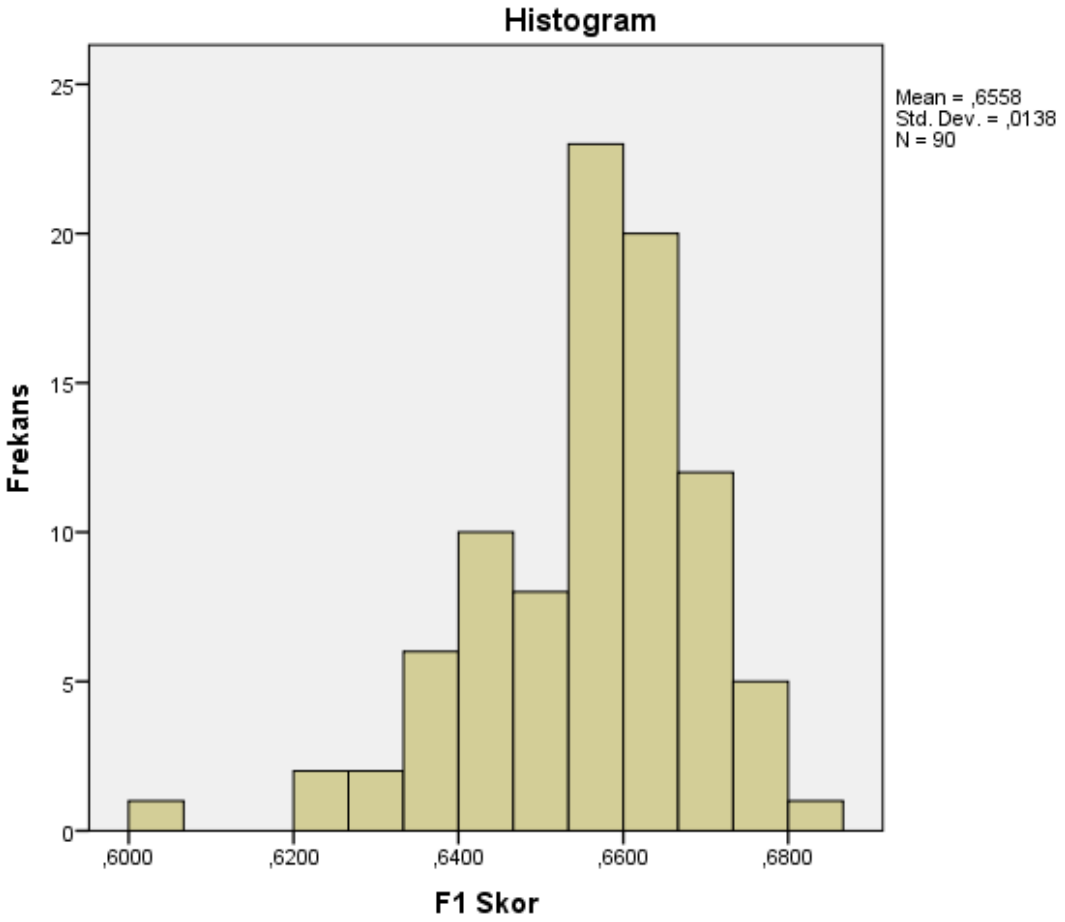
Tablo 4.2. Normal dağılım testi

	Kolmogorov-Smirnov		
	İstatistikler	Serbestlik Derecesi	Önemlilik Düzeyi (*p*)
Doğruluk	0,187	90	0,000
F1 Skor	0,120	90	0,003

Hem doğruluk hem de F1 Skor grupları için Kolmogorov-Smirnov testi önemlilik düzeyi (p=0,000 ve p=0,003) değerleri α=0,05'ten küçük oldukları için sıfır hipotezi kabul edilir ve iki grup da normal dağılımlı değildir. Şekil 4.1 ve Şekil 4.2'deki histogram grafiklerine bakıldığında iki grup için de sola çarpık bir yapının olduğu görülmektedir.



Şekil 4.1. Doğruluk grubuna ait histogram grafiği



Şekil 4.2. F1 skor grubuna ait histogram grafiği

İki grup için de normal dağılım varsayımlarının sağlanabilmesi için dönüşüm yapmadan önce çarpıklık ve basıklık değerlerinin normal dağılım için kabul edilebilir aralıklarının ±2 olup olmadığını (Trochim ve Donnelly, 2006; Field, 2000 & 2009; Gravetter ve Wallnau, 2014) kontrol edilmiştir.

Tablo 4.3. Çarpıklık ve basıklık değerleri

	Çarpıklık	**Basıklık**
Doğruluk	-2,897	11,676
F1 Skor	-0,899	1,354

Tablo 4.2'de F1 skor grubunun çarpıklık ve basıklık değerlerinin ±2 aralığında olduğu ve bu nedenle bu grubun normal dağıldığı varsayılabileceği görülmektedir. Doğruluk grubuna ait çarpıklık ve basıklık değerleri belirtilen aralıkta olmadığı için grubun normal dağıldığı varsayılamamaktadır. Bu yüzden, Şekil 4.1'deki histogram

grafiğinde de görüldüğü üzere, sola çarpık bir dağılım gösteren doğruluk grubu için dönüşümler uygulanmıştır. Ancak uygulanan hiçbir dönüşüm grubun normal dağıldığına dair varsayımı sağlayamamıştır.

F1 skor grubu için tek yönlü varyans analizi testi yapılmış (α=0,05) ve test sonucu Tablo 4.3'te verilmiştir. Kurulan hipotezler aşağıdaki gibidir.

H_0 = Grup ortalamaları eşittir.
H_1 = En az bir grup diğerlerinden farklıdır.

Tablo 4.4. Tek yönlü varyans analizi test sonuçları

	Kareler Toplamı	Serbestlik Derecesi	Kareler Toplamı	F	Önemlilik Düzeyi
Gruplar arası	0,001	2	0,001	2,921	0,059
Gruplar içi	0,016	87	0,000		
Toplam	0,017	89			

Tablo 4.3'tehi önemlilik düzeyi (p=0,059), α=0,05'ten büyük olduğu için sıfır hipotezi reddedilemez ve gruplar arasında fark olmadığı sonucuna varılır.

Doğruluk grubu için ise normal dağılım varsayımını sağlamadığı için parametrik olmayan testlerden Kruskal-Wallis testi kullanılmıştır (α=0,05), kurulan hipotezler aşağıdaki verilmiş ve test sonuçları Tablo 4.4'te gösterilmiştir.

Tablo 4.5. Kruskal-Wallis testi sonuçları

	Doğruluk
Ki-Kare	1,262
Serbestlik derecesi	2
Önemlilik Düzeyi	0,532

Önemlilik düzeyi (p) değeri α=0,05'ten büyük olduğu için doğruluk grubu için de gruplar arasında fark bulunmadığı sonucuna varılmıştır.

BÖLÜM 5. TARTIŞMA VE SONUÇ

Bu tez kapsamında, kelime gömme yöntemi olarak Keras kütüphanesinin kelime gömme tekniği, optimizasyon algoritması olarak Adam, sınıflandırma ve öğrenme için Bi-LSTM, LSTM ve GRU algoritmaları ile çok katmanlı bir yapı kullanılmıştır. Sınıflandırma performanslarını ölçmek için Wikipedia toksik yorum veri seti kullanılmıştır. Eğitim veri setinin %70'lik kısmı eğitim ve kalan %30'luk kısmı test için kullanılmıştır. Epoch sayısı 3 olarak seçilmiş ve sonuçlar doğruluk ve F1 skor metrikleri ile ölçülmüştür. Her bir model 30'ar kez çalıştırılarak, çıkan sonuçların minimum, maksimum, ortalama, çeyreklikler arası açıklık ve standart sapma değerleri hesaplanmıştır. Ortalama değerleri ele alındığında LSTM algoritması doğruluk metriğinde daha başarılı sonuç vermiştir ancak doğruluk metriğine göre daha güvenilir bir sonuç olan F1 skora bakıldığında Bi-LSTM algoritması, LSTM ve GRU'ya göre daha başarılı sonuç vermiştir. Minimum ve maksimum değerlerinin birbirine yakın olması standart sapma değerinin de küçük çıkmasını sağlamış ve yöntemin dar bir aralıkta yayıldığını göstermiştir. Çeyreklikler arası açıklığın düşük olduğu gözlemlenmiştir. İstatistiksel karşılaştırma sonuçlarında hem doğruluk hem de F1 skor metrik değerlerine bakılmış ve 3 yöntem için elde edilen değerler açısından yöntemler arasında bir fark olmadığı sonucuna varılmıştır.

Gelecek çalışmalarda farklı kelime gömme teknikleri de kullanılarak yöntemlerin haricinde kelime gömme tekniklerinin de sonuçlar üzerindeki etkisi araştırılabilir.

KAYNAKLAR

Aizenberg, I.N, Aizenberg, N.N., Vandewalle, J. 2000. Multiple-Valued Threshold Logic and Multi-Valued Neurons, İçinde: Multi-Valued and Universal Binary Neurons. Springer, Boston, 25–80.

Aken, B.V., Risch, J., Krestel, R., Löser, A. 2018. 2nd Workshop on Abusive Language Online to be held at EMNLP 2018, Challenges for Toxic Comment Classification: An In-Depth Error Analysis, Brussels, Belgium, 33-42.

Bengio, Y. 2009. Learning deep architectures for AI. Foundations and trends in Machine Learning, 2(1): 1-127.

Cho, K., Merrienboer, B.V., Bahdanau, D., Bengio, Y. 2014. On the properties of neural machinetranslation: Encoder-decoder approaches. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 103-111.

Chollet, F. 2018. Introduction to Keras, İçinde: Deep Learning with Python. Manning Publications Co., Shelter Island, NY, 61-62.

Chowdury, G.G. 2003. Annual Review of Information Science and Technology. Neural Network Processing, 37(1): 51-89.

Chung, J., Gulcehre, C., Cho, K., Bengio, Y. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. NIPS 2014 Deep Learning Workshop. 1-9.

Field, A. 2002. Discovering statistics using spss for windows, Turkish Online Journal of Distance Education-TOJDE. Sage Publications, London, 3(3): 1-3.

Field, A. 2009. Discovering statistics using SPSS. Sage Publications, London, 1-856.

Fukushima, K.N. 1980. A Self-Organizing Neural Network Model For A Mechanism Of Pattern Recognition Unaffected by Shift In Position, Biological Cybern, 36: 193–202.

Gravetter, F., Wallnau, L. 2014. Essentials Of Statistics For The Behavioral Sciences, 8. Baskı. Wadsworth, Belmont, 1-617.

Hermans, M., Schrauwen, B. 2013. Training and Analysing Deep Recurrent Neural Networks. Advances in Neural Information Processing Systems, 26: 190-198.

Hochreiter, S., Schmidhuber, J. 1997. Long Short-Term Memory. Neural Computation, 9(8): 1735-1780.

Ivakhnenko, A.G, Lapa, V.G. 1966. Cybernetic Predicting Devices, 37. Cilt. Joint Publications Research Service, Washington, 1-256.

Kingma, D. P., Ba, J. 2015. Adam: A Method for Stochastic Optimization. 3rd International Conference, San Diego, 1-15.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hurbard, W., Jackel, L.D. 1989. Handwritten Digit Recognition with a Back-Propagation Network. Morgan Kaufmann Publishers, 396–404.

Li, S. 2018. Application of Recurrent Neural NetworksIn Toxic Comment Classification, University of California, Applied Statistics, Master's thesis.

Mahesh, K., Nirenburg, S. 1995. Semantic Classification for Practical Natural Language Processing. 6th ASIS SIG/CR Classification Research Workshop: An Interdisciplinary Meeting, Chicago, 79-94.

Meng, Z., Hu, Y., Ancey, C. 2020. Using a Data Driven Approach to Predict Waves Generated by Gravity Driven Mass Flows. Water, 12(2): 1-18.

McCulloch, W. S., Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5: 115-133.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems 26 (NIPS 2013), Nevada, 1-9.

Purwarianti, A., Andhika, A., Wicaksono, A.F., Afif, I., Ferdian, F. 2016. InaNLP: Indonesia natural language processing toolkit, case study: Complaint tweet classification. 2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA), Penang, Malaysia, 1-5.

Schuster, M., Paliwal, K.K. 1997. IEEE Transactions On Signal Processing. Bidirectional Recurrent Neural Networks. 45(11): 2673-2681.

Trochim, W. M., Donnelly, J. P. 2006. The research methods knowledge base (3rd ed.). Cincinnati, OH:Atomic Dog, 10-361.

Wöllmer, M., Eyben, F., Graves, A., Schuller, B., Rigoll, G. 2010. Bidirectional LSTM Networks for Context-Sensitive Keyword Detection in a Cognitive Virtual Agent Framework. İçinde: Cognitive Computation 2. 180-190.

You, L., Y. Li, Y. Wang, J. Zhang, Y. Yang 2016. A deep learning-based RNNs model for automatic security audit of short messages. 16th International Symposium on Communications and Information Technologies (ISCIT), Qingdao, China, 225-229.

Young, I.J.B., Luz, S., Lone, N. 2019. A systematic review of natural language processing for classification tasks in the field of incident reporting and adverse event analysis. International Journal of Medical Informatics. 132: 1-7.

Zayegh, A., Bassam, N. A. 2018. Digital Systems. Neural Network Principles and Applications. IntechOpen, 115-130.

www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data, Erişim Tarihi: 22.09.2020.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Erişim Tarihi: 13.01.2021.

https://towardsdatascience.com/toxic-comment-classification-using-lstm-and-lstm-cnn-db945d6b7986, Erişim Tarihi: 26.04.2021.

ÖZGEÇMİŞ

Mete ÖZDEMİR ilk, orta ve lise eğitimini Giresun'da tamamladı. 2003 yılında Giresun Süper Lisesi'nden mezun oldu. 2005 yılında Uluslararası Kıbrıs Üniversitesi Bilgisayar Mühendisli Bölümü'nü 2011 de bitirdi. 2018 yılında Giresun Üniversitesi İstatistik Anabilim Dalında yüksek lisans eğitimine başladı. 2013 yılında Technix Technology şirketinde yazılım uzmanı olarak işe başladı. Daha sonra Turkcell Global Bilgi ve Çiçek Sepeti şirketlerinde test uzmanı, sistem analisti ve ürün yöneticisi pozisyonlarında çalıştı. 2017 yılında Giresun Üniversitesi'nde öğretim görevlisi olarak çalışmaya başladı. Halen öğretim görevlisi olarak görev yapmaktadır.