T.C
DOKUZ EYLÜL ÜNİVERSİTESİ
FEN FAKÜLTESİ
BİLGİSAYAR BİLİMLERİ BÖLÜMÜ

METİNLERİN 6 TEMEL DUYGUYA GÖRE SINIFLANDIRILMASI İÇİN LSTM TABANLI BİR ÖĞRENME MODELİ

Burak Can KUŞ
Giray BUDAN

Danışman: Mete EMİNAĞAOĞLU

Mayıs 2019
İZMİR

Burak Can Kuş ve Giray Budan tarafından **Dr. Öğr. Ü. Mete Eminağaoğlu** yönetiminde hazırlanan **"METİNLERİN 6 TEMEL DUYGUYA GÖRE SINIFLANDIRILMASI İÇİN LSTM TABANLI BİR ÖĞRENME MODELİ"** başlıklı rapor tarafımızca okunmuş, kapsamı ve niteliği açısından bir Bitirme Projesi olarak kabul edilmiştir.

Dr. Öğr. Üyesi
Mete Eminağaoğlu

ÖZET

Günlük iletişimlerde, insanlar çeşitli duygularını aktarır ya da kendilerini farklı duygularla ifade eder. Bu duygular; yazı, söz, jest veya mimiklerle aktarılabilir. Bu proje, yazılı metinlerle aktarılmak istenen duygunun tespit edilmesini amaçlamaktadır. Bu amaç doğrultusunda, psikolojideki duygu kategorileri, çeşitli doğal dil işleme yöntemleri ve son zamanlarda kullanımı yaygınlaşan derin öğrenme yöntemleri kullanılmıştır. Çalışmamızda altı farklı duygu çeşidi (mutluluk, üzüntü, öfke, tiksinti, şaşkınlık, korku) kullanılarak, GloVe sözcük vektörleri LSTM (Long short-term memory) temelli derin öğrenme modeline girdi olarak verilmiş ve farklı duygu içeren cümleler sınıflandırılmıştır. Projede kullanılan veriler, İnternet'teki iki farklı kaynakta bulunan veri kümelerinden elde edilmiştir. Proje yazılımında, Python programlama dili ve çeşitli kütüphaneler kullanılmıştır ve bu yaklaşım duygu sınıflandırmasında literatürdeki benzer çalışmalara göre başarılı sayılabilecek düzeyde sonuçlar elde edilmiştir.

Anahtar kelimeler: Derin öğrenme, duygu tespiti, LSTM, Word2Vec, çoklu sınıflandırma

ABSTRACT

People convey different kinds of emotions in daily communication. These emotions can be transferred by writing, speech, and / or gestures. This project's primary objective is to detect and classify the emotions in written messages correctly. Several natural language processing and deep learning methods have been used for this purpose. Six different emotions (happiness, sadness, anger, disgust, surprise, fear) have been chosen for the categorization or classification process of textual messages. GloVe word vectors are given as inputs to the LSTM (Long short-term memory) based deep learning model to classify the sentences. The benchmark datasets used in this project were obtained from two different resources via Internet. We used Python programming language and various libraries for the implementation of our project and some promising results have been obtained, which can be deduced slightly more successful than the relevant studies in the literature.

Keywords: Deep learning, emotion detection, LSTM, Word2Vec, multi-class classification

İÇİNDEKİLER

ÇİZELGELER DİZİNİ

ŞEKİLLER DİZİNİ

1. GİRİŞ

Dil, iletişim ve bilgi aktarımında önemli bir araçtır. İnsanlar dil aracılığıyla duygularını da ifade ederler. Doğal dil işleme yöntemleri uzun süredir yazılardaki bilgileri otomatik olarak algılamak için kullanılmaktadır. Konu temelli metin kategorize etme, özet çıkarma, soru-cevap sistemleri, arama motorları genel olarak metnin içindeki bilgilere odaklanır.

Son zamanlarda popülerliği artan yapay zeka yöntemleri, sağlık sektörü, otomotiv, finans, ekonomi, psikoloji, devlet işleri, video oyunları, askeri alanlar, reklam ve sanat gibi birçok alandaki problemlerin çözümü için kullanılmıştır.

Sentiment analysis, duygusal durumları ve öznel bilgileri sitematik olarak tanımlamak, çıkarmak, ölçmek ve üzerinde çalışmak için doğal dil işleme, metin analizi, hesaplamalı dilbilim ve biyometri kullanımı anlamına gelir. Sentiment analysis bizim araştırma konumuza yakın bir konudur. Bu araştırmanın farkı, cümlelerin olumlu ve olumsuz olarak ikili sınıflandırmasının yapılması değil, altı farklı duygu kategorisine sınıflandırılmasıdır.

Derin öğrenme, yapay sinir ağlarında kullanılan katmanları temel alan daha geniş bir makine öğrenme ailesinin bir parçasıdır. Öğrenme, supervised, semi-supervised ya da unsupervised olabililir. Bu çalışmada, derin öğrenme alanında kullanılan yapay sinir ağı mimarilerinden biri olan LSTM modeli kullanılmıştır.

Projemizde yazı dili ile aktarılan duyguların kategorize edilmesi üzerine çalışmalar yapılmıştır. Bu çalışmada, doğal dil işleme yöntemleri metinlerin içerdiği duyguları tespit etmek için kullanılmıştır. Yazılım için Python ve Python'un deep learning kütüphanesi olan PyTorch kullanılmıştır.

Eğitilmiş model, sosyal medya analizinde, anket sonuçları analizinde ve buna benzer diğer alanlarda kullanılabilir.

2. YÖNTEM VE ALGORİTMALAR

2.1. Kullanılan Araçlar

Projemiz Python dili 3.6.7 versiyonu kullanılarak Google Colaboratory platformunda yazılmıştır. Projede; Stanford CoreNLP, GloVe: Global Vectors for Word Representation araçlarından ve Python'daki scikit-learn, PyTorch ve çeşitli başka kütüphanelerden yararlanılmıştır.

2.1.1. Stanford Tokenizer

İhtiyacımız olan kelime vektörlerini alabilmek için öncelikle her bir cümledenin kelimelerine ayrılması gerekir. Bunun için de tokenizer denilen araçlar kullanılmaktadır. Bir tokenizer, kabaca metni kelimelere karşılık gelen token'lara böler. Yani cümlenin kelimelerini ayırır.

Yapılan araştırmalar sonucu, veri setimizdeki cümlelerin bir tokenizer yardımıyla token'lara ayrılması gerektiği öngörülmüş, bu ihtiyaç doğrultusunda Christopher Manning, Tim Grow, Teg Grenager, Jenny Finkel ve John Bauer tarafından geliştirilen Stanford Tokenizer'ın kullanılmasına karar verilmiştir. Stanford Tokenizer, verimli, hızlı ve deterministik bir tokenizer'dır. Daha teknik bir tanım yapacak olursak; JFlex tarafından üretilen sonlu bir otomata ile implement edilmiştir. Deterministik olsa da, oldukça iyi sezgisel algoritmalar kullanır. Bu nedenle genellikle tek tırnakların ne zaman sözcüklerin bir parçası olduğuna, noktaların ne zaman cümlelerin sınırı olduğuna karar verebilir.

Cümle bölme tokenizer'ın deterministik bir sonucudur. Cümle, cümle bitirici karakterler (., ! veya ?) diğer tokenların içine dahil olmadığı zamanlarda (kısaltma veya sayılarda) sonlandırılır. Yine de cümle hala cümle bitirici karakterlerden sonra gelen tokenlar içerebilir (tırnak işaretleri ve parantezler gibi). (WEB_1)

Stanford Tokenizer, Stanford CoreNLP araçları ile birlikte gelir. Bu araçlar, eylemlerin çekimlerini, cümlenin öğelerini, özel isimleri (insan, şirket isimleri), tarih, zaman ve sayısal miktar normalizasyonunu, cümle parçalarının birbirine bağlılıklarını, isimler ve varlıklar arasındaki ilişkileri ve alıntıları bulmak gibi çeşitli amaçlarla kullanılabilir.

CoreNLP, Python'da NLTK kütüphanesi aracılığıyla kullanılabilir. Öncelikle kullanılacak bilgisayarda CoreNLP server şu şekilde çalıştırılır:

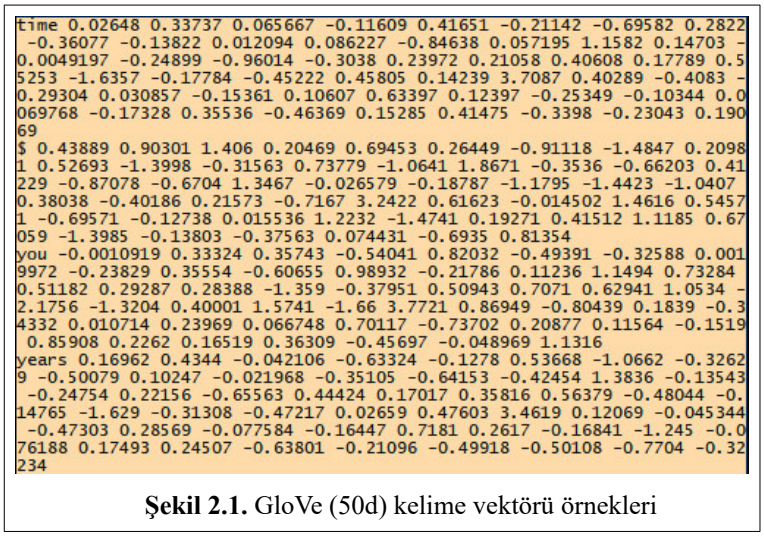
```
java -mx4g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer \
-preload tokenize,ssplit -status_port 9000 -port 9000 -timeout 15000
```

Daha sonra NLTK API yardımıyla Python dilinde bu serverdan cümlelerin tokenize edilmiş halleri alınır.

2.1.2. GloVe: Global Vectors for Word Representation

GloVe, Jeffrey Pennington, Richard Socher ve Christopher D. Manning tarafından, kelimeler için vektör gösterimleri elde etmek amacıyla geliştirilen unsupervised bir öğrenme algoritmasıdır. Eğitim, bir korpustan toplanmış, hedef kelimenin sağında ve solundaki kelimelerle birlikte olma istatistikleri üzerinde gerçekleştirilir ve sonuçta oluşan gösterimler, kelime vektörü uzayında ilginç (anlamsal olarak yakın kelimelerin, birbiri ile bağlantılı vektörleri olması gibi) özellikleri gösterir. (WEB_2)

Projede, hazır Common Crawl (840 milyar token) verisiyle eğitilmiş ve sonuçta oluşan 2.2 milyon kelime (her biri 300 boyutlu vektör) büyüklüğündeki sözlük kullanılmıştır. Veri setimizdeki cümleler tokenize edilerek her bir token'ın GloVe vektörü bulunur ve eğitimler bu vektörler üzerinde yapılır.



Şekil 2.1. GloVe (50d) kelime vektörü örnekleri

Şekil 2.1'de görüldüğü üzere her kelime 300 boyutlu bir vektöre denk gelir (Şekilde kolay anlaşılır olması için 50 boyutlu vektörler gösterilmiştir.). Bu vektörlerin, kelimeleri ifade ettiği varsayılır ve bu sayısal veriler duygu tespit modelinde girdi olarak kullanılır.

2.1.3. FastText

FastText kullanıcıların yazı gösterimlerini öğretebilecekleri ve yazı sınıflandırması yapabilecekleri açık kaynaklı, ücretsiz ve hafif bir kütüphanedir. FastText çalışacağı donanımın çok da güçlü olmasını gerektirmez, standart donanımlarda çalışabilir. Modeller daha sonradan boyutları düşürülerek mobil cihazlara bile sığacak boyuta getirilebilir.(WEB_3)

Projemizde FastText, GloVe ile karşılaştırmalı olarak kullanılmıştır. FastText'in bir diğer özelliği ise, kelime altı parçacıkları kullanarak, corpus'da olmayan kelimeler için kelime vektörleri üretebilmesidir.

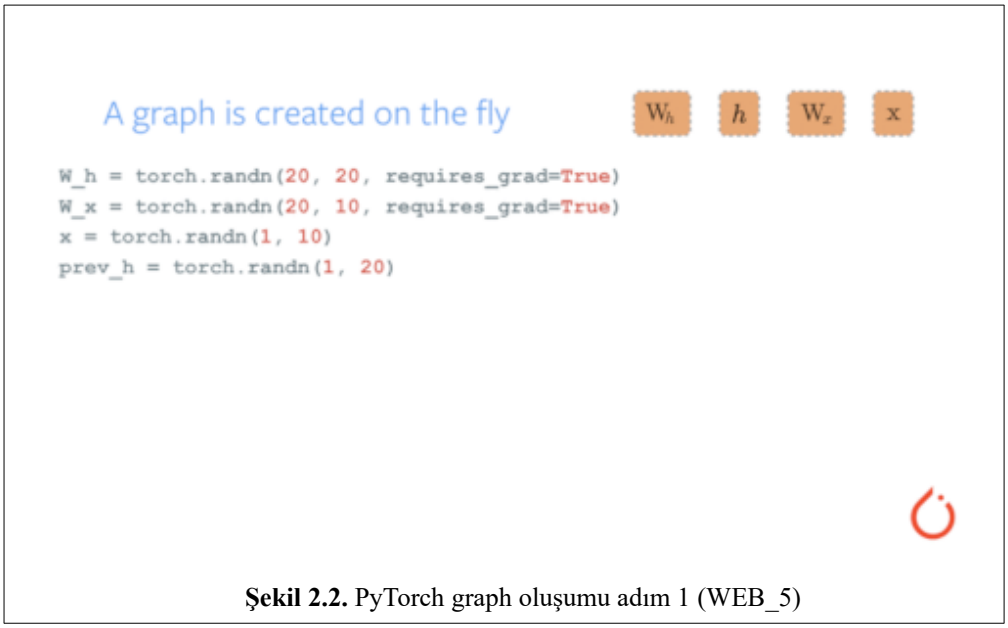
2.1.4. PyTorch

PyTorch, Python için geliştirilmiş Torch temelli açık kaynaklı bir makine öğrenimi kütüphanesidir. Doğal dil işleme gibi alanlarda kullanımı yaygındır. Öncelikle Facebook'un yapay zeka araştırma grubu tarafından geliştirilmiştir. (WEB_4)

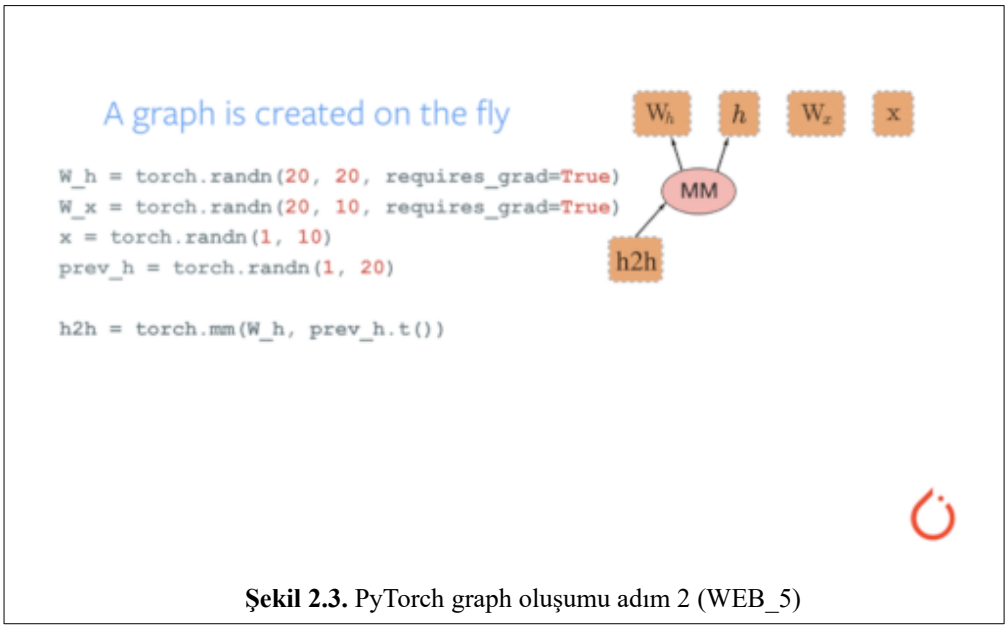
PyTorch, iki adet high-level özellik sunar:

- Tensor işlemlerinin GPU üzerinde yapılabilmesi
- Derin sinir ağlarının otomatik olarak diferansiyel ağaçlarının oluşturulması ve backpropagation (geri besleme) yapılması

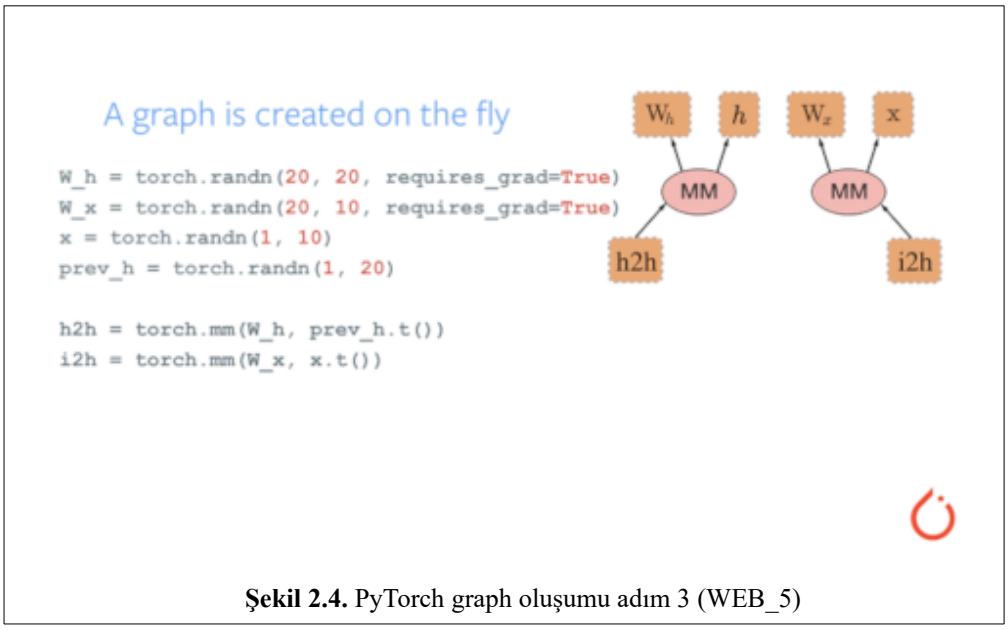
PyTorch birden fazla modül içerir. Autograd, Optim ve nn modülleri bunlara örnek olara gösterilebilir.



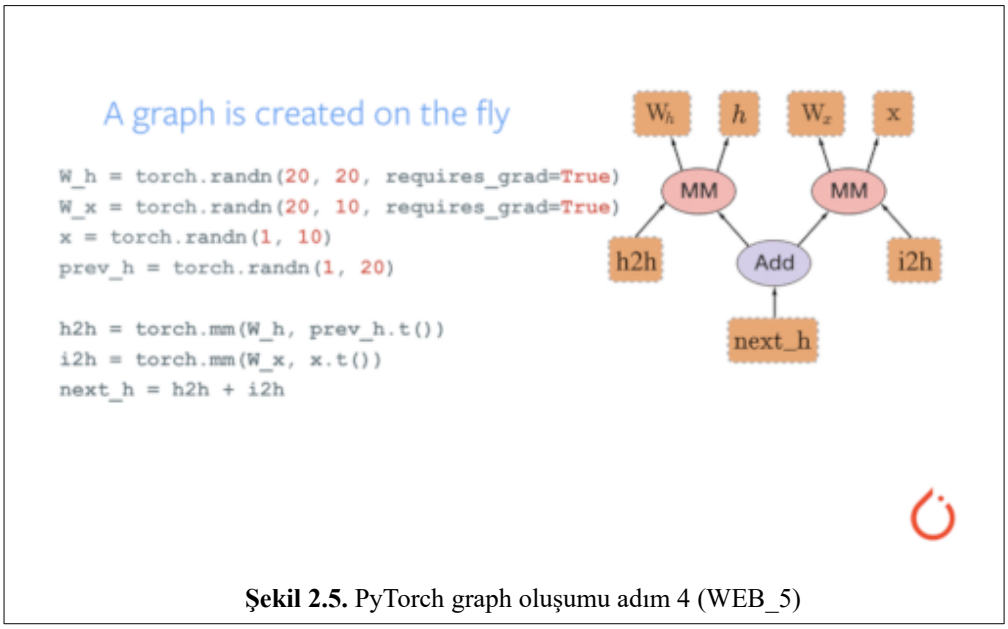
Şekil 2.2. PyTorch graph oluşumu adım 1 (WEB_5)



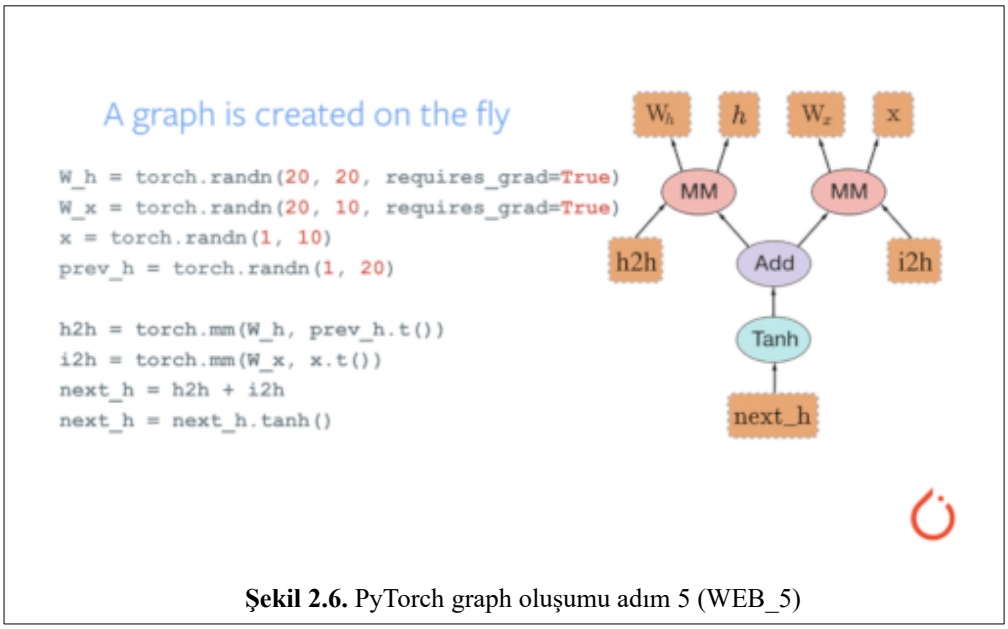
Şekil 2.3. PyTorch graph oluşumu adım 2 (WEB_5)



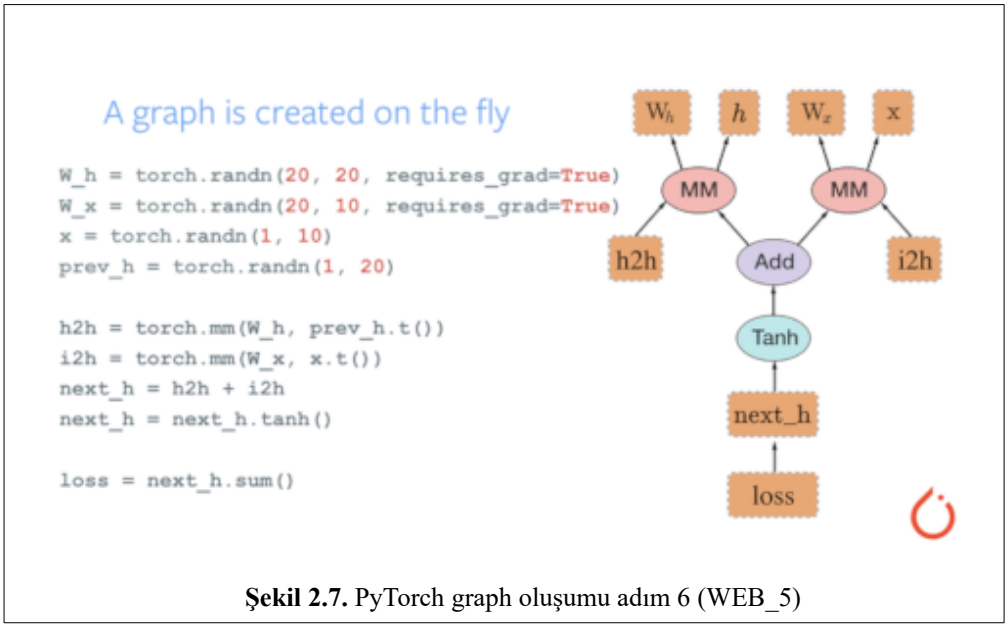
Şekil 2.4. PyTorch graph oluşumu adım 3 (WEB_5)



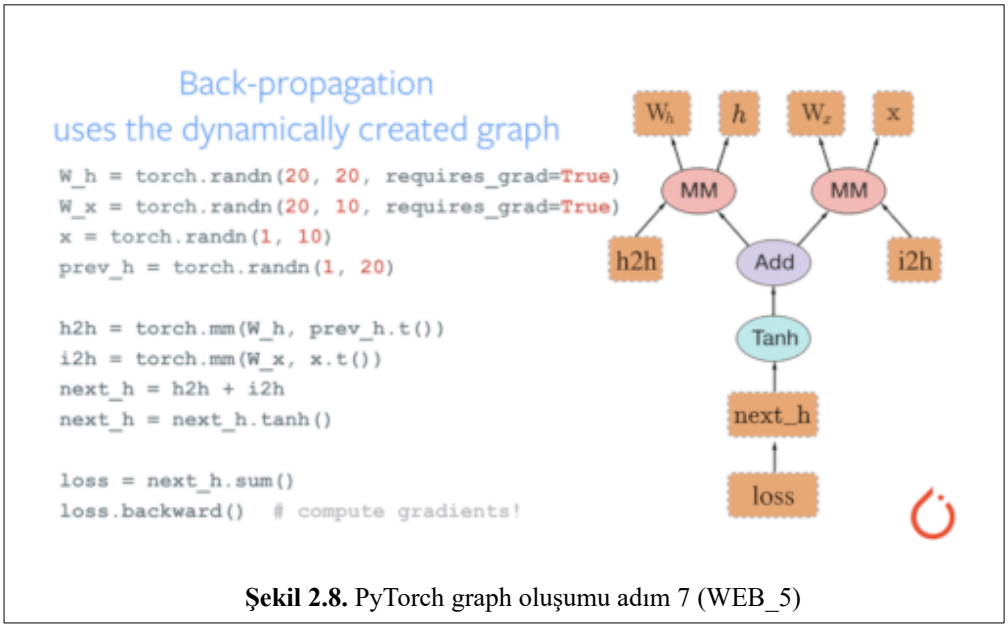
Şekil 2.5. PyTorch graph oluşumu adım 4 (WEB_5)



Şekil 2.6. PyTorch graph oluşumu adım 5 (WEB_5)



Şekil 2.7. PyTorch graph oluşumu adım 6 (WEB_5)



Şekil 2.8. PyTorch graph oluşumu adım 7 (WEB_5)

Projemizde, neural network modeli PyTorch ile implement edilip CUDA üzerinde çalıştırılmıştır. CUDA, NVIDIA'nın GPU (grafik işlem birimi) gücünü kullanarak hesaplama performansında büyük ölçüde artışlara olanak veren paralel hesaplama mimarisidir. PyTorch'da ayarların yapılması kolay olup yüksek düzeyde kolaylıkla programlama yapılabilir. Modelimizde iki yönlü/iki yönlü olmayan LSTM katmanı, batch normalization katmanı, dropout katmanı ve son olarak fully connected bir katman vardır.

2.1.5. scikit-learn

Scikit-learn (önceki adıyla scikits.learn), Python programlama dili için ücretsiz makine öğrenimi kütüphanesidir. Destek vektör makinesi (SVM), random forest, gradient boosting, k-means ve

DBSCAN gibi algoritmaları kullanarak; sınıflandırma, regresyon ve kümeleme gibi çeşitli işlemleri gerçekleştirir. NumPy ve SciPy gibi nümerik ve bilimsel kütüphanelerle ortak çalışılmasına olanak sağlaması için tasarlanmıştır. (WEB_6)

Scikit-learn'ün, bazı önemli algoritmaları Cython'da fakat geneli çoğunlukla Python'da yazılmıştır. Projemizde cross validation ve sınıflandırma raporlarını hazırlamak için PIL (Pillow) kütüphanesi ile ortaklaşa kullanılmıştır.

2.2. Derin Öğrenme Yöntemleri

2.2.1. Derin Öğrenme Nedir?

Derin öğrenme (derin yapılandırılmış öğrenme veya hiyerarşik öğrenme olarak da bilinir) yapay sinir ağlarında kullanılan katmanları temel alan daha geniş bir makine öğrenme yöntemleri ailesinin bir parçasıdır. Öğrenme supervised, semi-supervised ya da unsupervised olabilir.

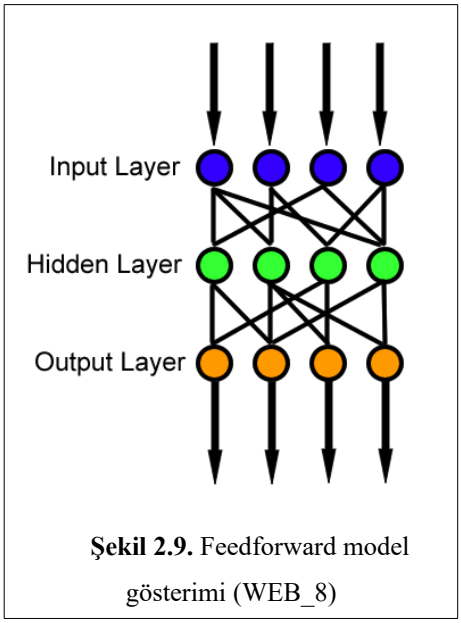
Computer vision, doğal dil işleme, ses tanıma, sosyal ağ filtreleme, metin çevirisi, biyoinformatik, ilaç tasarımı gibi alanlara deep neural network, recurrent neural network ve convolutional neural network gibi derin öğrenme mimarileri uygulanmıştır. Derin öğrenme ile tıbbi görüntü analizi, malzeme muayenesi ve masa oyunu programları gibi alanlarda bazı durumlarda alanında uzman insanlara göre daha iyi sonuçlar elde eden makineler geliştirilmiştir.

Yapay sinir ağları, başlangıçta biyolojik sistemlerin sinaptik yapılarındaki iletişim düğümlerinden (sinir hücreleri) esinlenilmiş ancak biyolojik beyinlerin yapısal ve fonksiyonel özelliklerinden farklılıklar göstermiş ve bu da onları nörolojik kanıtlarla bağdaşmaz hale getirmiştir. Spesifik olarak, yapay sinir ağları statik ve dijital olma eğilimindeyken canlı organizmaların çoğunun biyolojik beyni dinamik ve analogdur.

Derin Öğrenme terimi 1986'da Rina Dechter tarafından makine öğrenme topluluğuna tanıtılmıştır. Supervised, derin, feedforward, çok katmanlı perceptronlar için çalışan ilk öğrenme algoritması 1965'te Alexey Ivakhnenko ve Lapa tarafından yayınlanmıştır. (WEB_7)

2.2.2. Feedforward Neural Networks

Feedforward neural network (ileri beslemeli sinir ağı) yapay bir sinir ağıdır, burada düğümler arasındaki bağlantılar bir döngü oluşturmaz. Bu nedenle, tekrarlayan sinir ağlarından (RNN) farklıdır. Feedforward neural network, tasarlanan ilk ve en basit yapay sinir ağıdır. Bu ağda bilgi, giriş düğümlerinden gizli düğümlere (eğer varsa) ve çıkış düğümlerine doğru sadece bir yönde ilerler. İleri beslemeli bir ağda bilgi her zaman bir yönde hareket eder; asla geriye doğru gitmez.



Şekil 2.9. Feedforward model gösterimi (WEB_8)

Single-layer perceptron (tek katmanlı sinir ağı), tek bir çıktı düğümü katmanından oluşan en basit neural network türüdür. Girdiler doğrudan bir dizi ağırlık üzerinden çıkışa doğru beslenir. Her bir ağırlık ve girdinin çarpımlarının toplamı her düğümde hesaplanır ve değer bir eşiğin üstünde ise (genelde 0) nöron ateşlenir ve aktif değeri alır (genelde 1). Bu tür aktivasyon işlevine sahip nöronlara yapay nöronlar veya lineer eşik birimleri de denir. Literatürde perceptron terimi genellikle bu birimlerden sadece birini ifade eder. 1940'larda Warren McCulloch ve Walter Pitts tarafından da benzer bir nöron tanımlanmıştır.

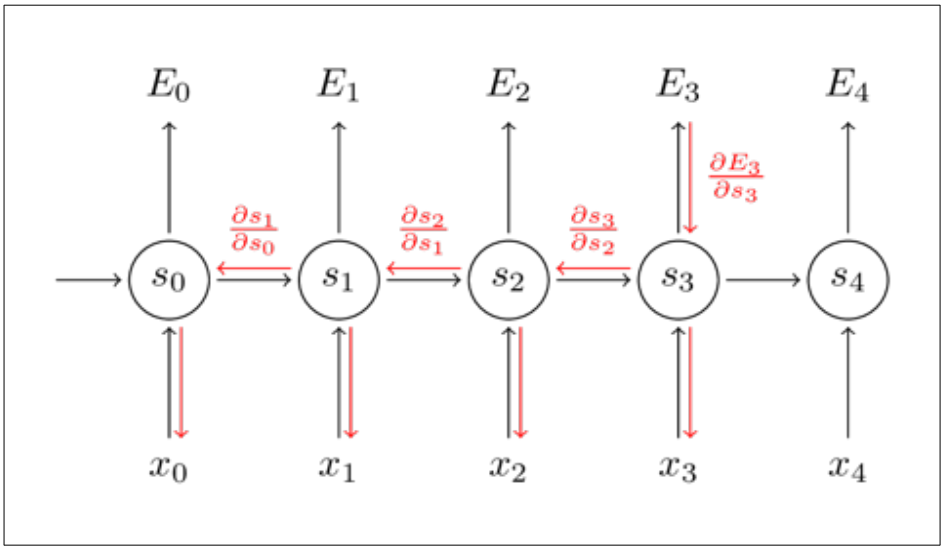
Perceptronlar genellikle delta kuralı denilen basit bir öğrenme algoritması ile eğitilebilirler. Hesaplanan çıktı ve bilinen çıktılar arasındaki hataları hesaplar ve bunu ağırlıklara bir düzeltici oluşturmak için kullanır, böylece gradient descent (bir fonksiyonun minimumunu bulmak için birinci dereceden bir yinelemeli optimizasyon algoritması) uygulanmış olur.

Multi-layer perceptron (çok katmanlı sinir ağı), genellikle ileri beslemeli bir şekilde birbirine bağlanmış çok sayıda katmandan oluşur. Bir katmandaki her bir nöron, bir sonraki katmanın nöronlarına bağlanmıştır. Birçok uygulamada, bu ağların aktivasyon fonksiyonu olarak sigmoid fonksiyonu uygulanır. Çok katmanlı ağlar, en popülerleri backpropagation olan çeşitli öğrenme teknikleri kullanır. Burada, çıkış değerleri önceden tanımlanmış bazı hata fonksiyonlarının değerini hesaplamak için doğru cevap ile karşılaştırılır. Çeşitli tekniklerle hata, ağ üzerinde geriye doğru beslenir. Bu bilgiyi kullanarak, algoritma, hata fonksiyonunun değerini küçük bir miktar azaltmak için her bağlantının ağırlığını ayarlar. Bu işlemi yeterince fazla sayıda eğitim için tekrarladıktan sonra, ağ genellikle hesaplama hatasının küçük olduğu bir duruma dönüşecektir. Bu durumda, ağın belirli bir hedef fonksiyonunu öğrendiği söylenebilir. Ağırlıkları doğru ayarlamak için, gradient descent adı verilen doğrusal olmayan optimizasyon için genel bir yöntem uygulanır. Bunun için, ağ,

ağ fonksiyonuna göre hata fonksiyonunun türevini hesaplar ve ağırlıkları, hatanın azalacağı şekilde değiştirir. Bu nedenle, backpropagation yalnızca türevlenebilir aktivasyon fonksiyonlarına sahip ağlara uygulanabilir. (WEB_8)

Backpropagation (geri besleme) algoritmaları, zincir kuralını kullanan bir gradient descent yaklaşımı izleyerek yapay sinir ağlarını verimli bir şekilde eğitmek için kullanılan bir yöntem ailesidir. Backpropagation'ın ana özelliği, eğitim görevini yerine getirene kadar ağda geliştirmek üzere ağırlık güncellemelerini hesaplamak için yinelemeli, recursive (özyinelemeli) ve verimli bir yöntem olmasıdır. Gauss-Newton algoritması ile yakından ilgilidir ve nöral backpropation konusunda devam eden araştırmaların bir parçasıdır.

Backpropagation, aktivasyon fonksiyonlarının türevlerinin sinir ağının tasarımı yapılırken bilinmesini gerektirir. Automatic differentiation (bir bilgisayar programı tarafından belirtilen bir fonksiyonun türevini sayısal olarak değerlendirmek için kullanılan bir teknikler setidir), türevleri eğitim algoritmasına otomatik ve analitik olarak sağlayabilen bir tekniktir. Eğitim bağlamında, backpropagation genellikle, loss function(tasarlanan modelin hata oranını ve başarımını ölçen fonksiyondur) derecesini hesaplayarak nöronların ağırlığını ayarlamak için gradient descent optimizasyon algoritması tarafından kullanılır.

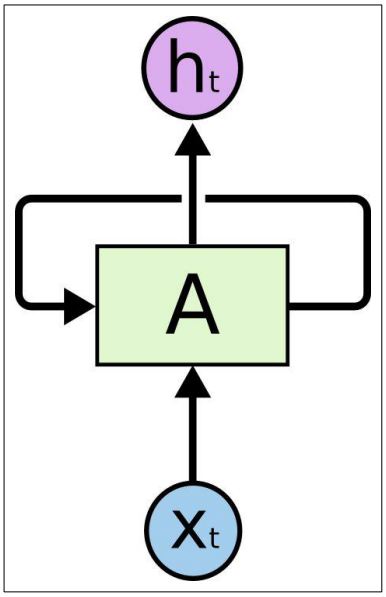


Şekil 2.10. RNN'de backpropagation işlemi (WEB_9)

2.2.3. Recurrent Neural Networks (RNN)

Recurrent Neural Networks (Tekrarlayan sinir ağları), düğümler arasındaki bağlantıların yönlendirilmiş bir döngü oluşturduğu yapay sinir ağı sınıfıdır. İleri beslemeli sinir ağlarından farklı olarak, RNN'ler kendi giriş belleklerini, girdileri işlemek için kullanabilirler. Bu öznitelik RNN'leri, el yazısı tanıma ve konuşma tanımada kullanılabilir bir yöntem yapmaktadır.

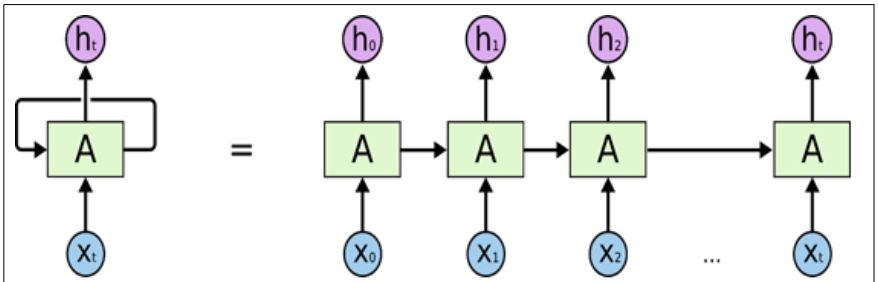
RNN'lerin asıl amacı ardışık bilgileri kullanmaktır. Geleneksel bir sinir ağında tüm girdilerin (ve çıktıların) birbirinden bağımsız olduğunu varsayarız. RNN'ler tekrarlayan olarak adlandırılır çünkü bir dizinin her öğesi için aynı görevi yerine getirirler ve çıktı önceki hesaplamalara bağlıdır. RNN'leri düşünmenin bir başka yolu, şu ana kadar hesaplananlarla ilgili bilgi toplayan "bellek" taşıdıklarıdır. Teorik olarak, RNN'ler uzun dizilerdeki bilgileri kullanabilir, ancak pratikte yalnızca birkaç adım geriye dönmekle sınırlıdır.



Şekil 2.11. RNN gösterimi
(WEB_10)

Üstteki şekilde basit bir tekrarlayan sinir ağ görüntülenmektedir. 'A' ismi verilen dikdörtgen, bir yapay sinir ağındaki hücredir. Ağın girdi değeri X'dir. Yapay sinir ağının çıktı değeri h'dir. Hücreden çıkan bir değer yine kendisine gelerek, bir döngü oluşturmaktadır. Bu döngü ile önceki zamanın bilgileri de kullanılabildiğinden yeni bilgi, eski bilgi kullanılarak anlamlandırılabilmekte ve böylelikle sınıflandırma yapılabilmektedir.

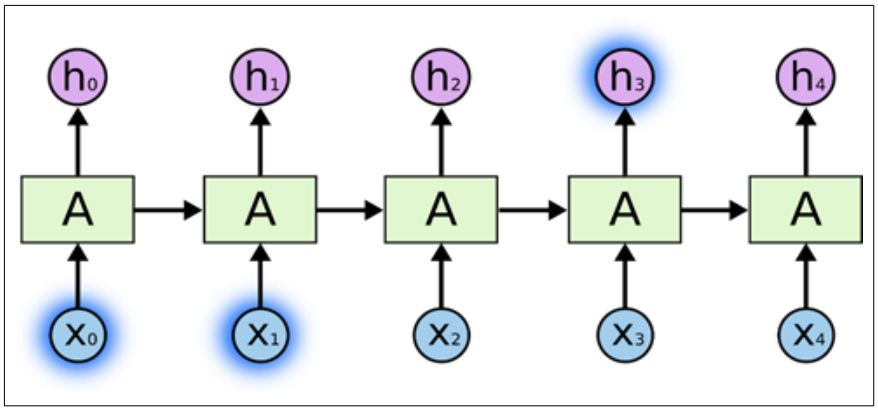
Geleneksel yapay sinir ağlarında, hücrelerden çıkan sonuçlar tekrardan kendilerine girdi olarak gelmemektedir. RNN'de ise hücreden çıkan sonuç, tekrardan kendisine girdi olarak gelmektedir. RNN açılırsa aşağıdaki şekildeki gibi bir mimari ortaya çıkmaktadır. Zaman diliminde, aynı hücre kendini birden fazla tekrar etmektedir. Böylelikle de kareler arasında anlamlandırma kurulabilmektedir.



Şekil 2.12. Açık haliyle gösterilmiş bir RNN modeli (WEB_10)

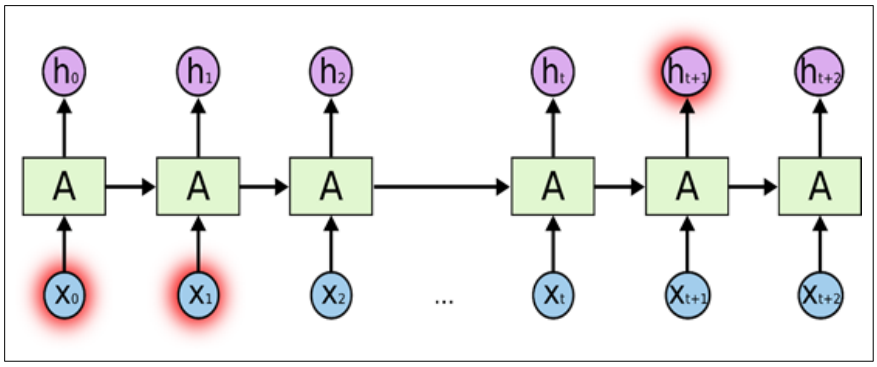
RNN'ler, bir döngü oluşturabildiklerinden, sıralı gelişen aktiviteleri birbirleriyle anlamlandırabilmektedir. Akış içerisindeki aktivitelerin anlamlandırılarak sınıflandırabilmesinden dolayı son yıllarda yaygın olarak kullanılmaktadır. RNN'lerin, birçok kullanım alanı bulunmaktadır; konuşma tanımlama, dil modelleme, metin çevirisi, resim başlığı oluşturma vb.

RNN'ler, geçmiş ile bağlantı kurup anlamlandırma özniteliğinden dolayı, zaman bazlı problemlerde başarılı sonuçlar vermektedir. Ancak RNN'lerde, hangi aktiviteler hatırlanacak, ne kadar süre hatırlanacak bilinmemektedir. Bütün bilgiler, modelin içerisinde tutulmaktadır. Aktiviteler için bazı bilgiler önemliyken, bazı bilgiler gereksizdir. Bu yüzden bazı aktivitelerin sınıflandırılmasında, tüm geçmişin saklanmasına gerek yoktur.



Şekil 2.13. Yakın zamanlı bilginin hatırlanabilmesinin gösterimi (WEB_10)

Örneğin, aktivite sınıflandırılması yapılacak olan videoda, sırayla bazı kişiler yemek masasına oturuyor olsun; böyle bir videoda, önceden masaya oturmuş kişilerden, bir sonraki gelen kişinin de masaya oturacağını tahmin etmek RNN için zor değildir. Çünkü bir sonraki karenin tahmin edilebilmesi için gerekli olan masaya oturma aktivitesi, çok yakın zamanda gerçekleşmiştir. Yukarıdaki şekilde, bu tarz bir RNN örneği gösterilmektedir. Bazı durumlarda ise aktivite sınıflandırılmasında, gerekli bilgi çok önceden oluşmuş ise, bu bilgiye ulaşılamayabilir. Aşağıdaki şekilde bu tarz bir RNN örneği gösterilmektedir. (WEB_10)

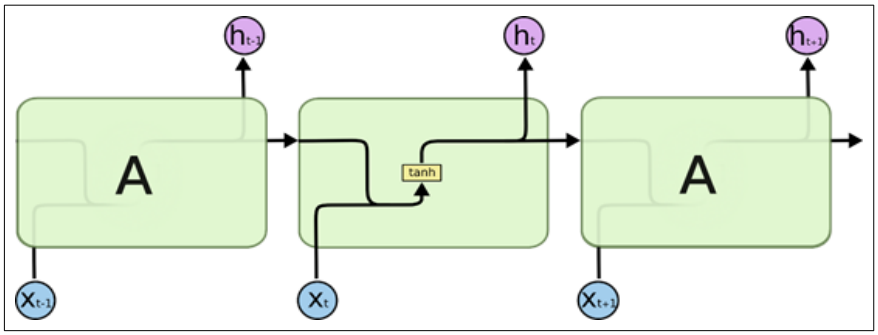


Şekil 2.14. Uzaktaki bilginin hatırlanamamasının gösterimi (WEB_10)

2.2.4. Long-Short Term Memory (LSTM)

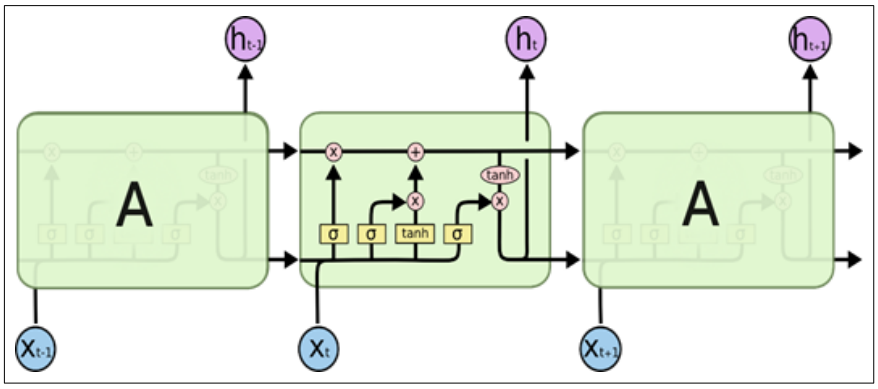
RNN'lerin çok önceden olan olayları tahmin edebilmesi için farklı bir mimari yapıya ihtiyaçları vardır. Bu tür problemlerde, RNN'lerin bir çeşidi olan LSTM (Long-Short Term Memory) yani Uzun Kısa-Süreli Bellek ağları kullanılmaktadır. Teoride, RNN'lerin uzun geçmişteki aktiviteleri, iç mimarisinde kendini tekrarlamakta olduğu için, hatırlayabilme kapasitesine sahiptir. Ancak hatırlanabilmesi için parametrelerin titizlikle seçilmesi gerekmektedir. Böyle bir parametre seçimi, pratikte mümkün olmadığından, RNN'ler uzak geçmişi öğrenemezler (hatırlayamazlar). (S. Hochreiter, J. Schmidhuber. (1997). Long short-term memory. Neural computation, 9(8):1735–1780.)

LSTM ağlarında böyle bir problem bulunmamaktadır. Bu yüzden LSTM'ler birçok aktivite sınıflandırılmasında çoğunlukla tercih edilmektedir. Son zamanlardaki RNN'ler ile yapılmış başarılı çalışmaların, büyük bir çoğunluğunda, LSTM'ler kullanılmıştır.

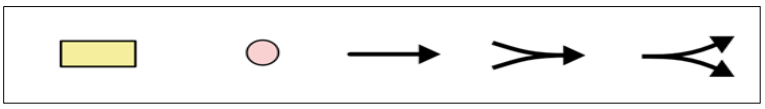


Şekil 2.15. Geleneksel RNN'de tekrarlayan tek katman (WEB_10)

Geleneksel RNN'lerde, tekrarlayan kısımda tek bir tanjant fonksiyonu (katmanı) bulunmaktadır. Şekilde bu tanjant fonksiyonu gösterilmektedir. LSTM'ler tek bir sinir katmanı yerine tekrarlayan zincir şeklinde dört farklı katman bulundurmaktadır. Aşağıdaki şekilde, LSTM'de bulunan 4 farklı katman gösterilmektedir. LSTM mimarisinin görselleştirildiği resimlerde, birçok sembol bulunmaktadır. Bu semboller aşağıda gösterilmiştir.



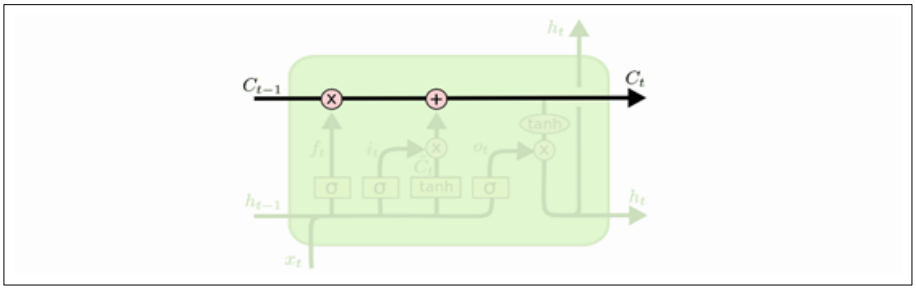
Şekil 2.16. LSTM'deki 4 etkileşimli katman (WEB_10)



Şekil 2.17. LSTM mimarisinde kullanılan işaretler (WEB_10)

Şekilde, okların çizgi kısmı düğümdeki çıkışları, baş kısımları ise diğer düğümlere girdi olarak veriyi taşımaktadır. Okun yönü, verinin taşındığı düğümü göstermektedir. Pembe daire, vektör eklemesi gibi noktasal operasyonları simgelemektedir. Sarı dikdörtgen, yapay sinir katmanını göstermektedir. İki okun birleşmesi, iki farklı vektörün birleşip, başka bir düğüme girdi olarak verildiğini, çatal şeklindeki ok ise düğümden çıkan bir vektörün, birden fazla düğüme girdi olarak kopyalandığını gösterir.

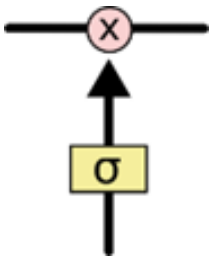
LSTM'lerin kilit noktası, hücrenin durumudur (cell state). Bu hücre durumu şekildeki hücrenin üst tarafında gösterilen yatay çizgidir. Hücrenin durumu, bir çeşit taşıma bandı gibidir. Bilgi, genellikle soldan sağa ilerlemektedir ancak bazı küçük doğrusal etkileşimlerde, zincirin alt tarafına doğru da ilerleyebilmektedir.



Şekil 2.18. LSTM'in kilit noktası olan hücrenin durumu (WEB_10)

LTSM'ler, kapılar (gate) yardımıyla, hücre durumundan, bilgi silme ve hücre durumuna bilgi ekleme yeteneğine sahiptir. Bu kapılar, isteğe bağlı olarak bilginin geçmesine olanak sağlamaktadır.

Bu kapılar, sigmoid sinir katmanı ve noktasal çarpma işleminden oluşmaktadır. Aşağıdaki şekilde bu kapı örneği gösterilmektedir.



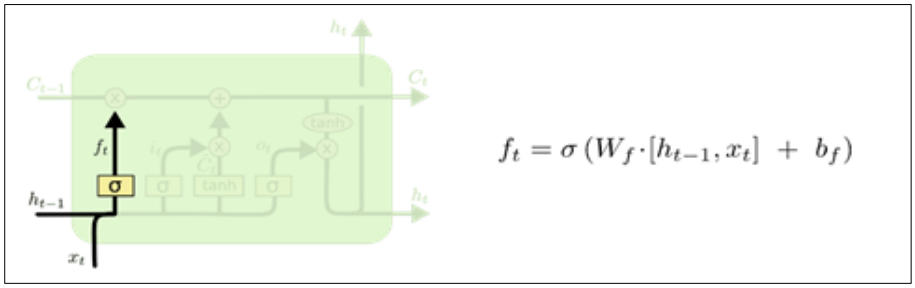
Şekil 2.19. Noktasal çarpma ve sigmoid sinir katmanı (gate) (WEB_10)

Sigmoid sinir katmanı, bilginin, ne kadar geçeceğini belirlemektedir. Sigmoid fonksiyonu, gelen veriye göre, çıktıda **0** ile **1** arasında bir sonuç vermektedir. Bir sigmoid fonksiyonundan, **0** çıkması, "hiçbir şeyin geçmesine izin verme", **1** sonucu çıkması da "hepsinin geçmesine izin ver" anlamına gelir.

LSTM ağları, bu kapılardan üç tanesine sahiptir. Bu kapıların görevi, geçen bilgiyi korumak ve kontrol etmektir. Bu kapılar şu şekilde listelenebilir:

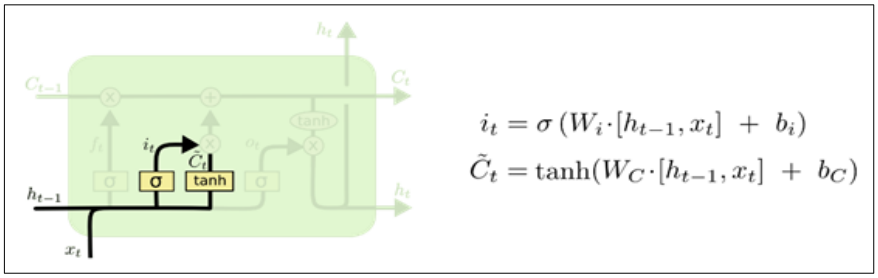
- Input Gate (Girdi Kapısı)
- Forget Gate (Unutma Kapısı)
- Output Gate (Çıktı Kapısı)

LSTM ağının ilk aşaması, unutma kapısı ile yapılmakta olan hücre durumundaki bilginin atılıp atılmayacağının (saklanıp saklanmayacağının) karar verilmesi aşamasıdır. Bu karar forget gate adı verilen bir sigmoid katmanı tarafından verilir. h_{t-1} ve x_t değerlerine bakar ve C_{t-1} hücre durumundaki her sayı için **0** ile **1** arasında bir sonuç çıkarır. Sonucun **1** çıkması "bunu tamamen tut", **0** çıkması ise "bunu tutmana gerek yok" anlamına gelir.



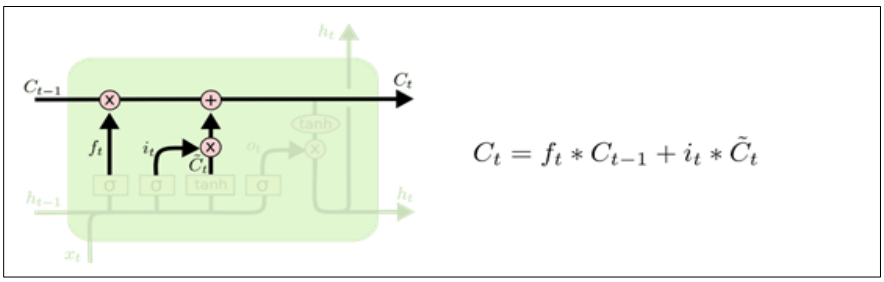
Şekil 2.20. Forget gate'in yapısı ve formülü (WEB_10)

LSTM'lerin bir sonraki aşaması, input gate denilen bir sigmoid sinir ağı ile hangi bilgilerin hücrede güncelleneceği bilgisinin belirlenmesidir. Sonra tanh katmanı, duruma (state) eklenebilecek aday yeni vektör değerlerini, $\widetilde{C}_t$, oluşturur. Bir sonraki adımda bu iki değer birleştirilerek, duruma bir güncelleme oluşturulur.



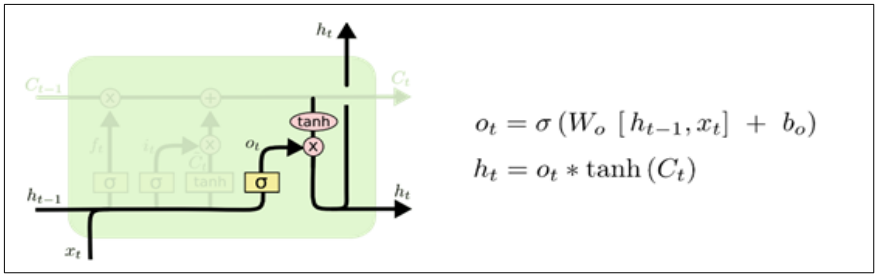
Şekil 2.21. Input gate katmanı ve tanjant kapılarının formülü (WEB_10)

Sırada, eski hücre durumu C_{t-1} 'yi yeni hücre durumu C_t 'ye güncellemek var. Eski hücre durumu f_t ile çarpılıp, unutmaya karar verilen bilgiler unutulur. Sonra $i_t * \widetilde{C}_t$ eklenir. Sonuçta elde edilen değerler, her durumu ne kadar güncellemek istediğimizle ölçeklenmiş yeni aday değerleridir.



Şekil 2.22. Hücre durumundaki eski ve yeni verinin güncellenmesi (WEB_10)

Son olarak çıktı olarak ne verileceğine karar vermek var. Bu çıktı, hücre durumunun filtrelenmiş bir versiyonu olarak olur. Öncelikle bir sigmoid katmanı hücre durumunun hangi parçalarının çıktı olacağına karar verir. Sonra hücre durumu tanh'den geçirilir (çıktılar **-1** ve **1** arasına getirilir) ve sadece istediğimiz parçaları çıktı olarak vermek için sigmoid kapısının çıktısı ile çarpılır. Aşağıdaki şekilde, LSTM hücresinde bulunan çıkış katmanı ve formülü gösterilmiştir. (WEB_10)



Şekil 2.23. Output gate ve formülleri (WEB_10)

2.2.5. Batch Normalization

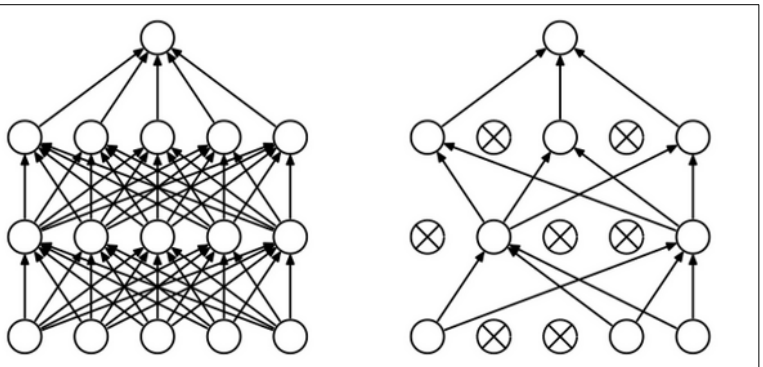
Batch normalization, yapay sinir ağlarının hızını, performansını, dengesini arttırmak ve aktivasyonları ayarlayarak ve ölçekleyerek giriş katmanını normalleştirmek için kullanılır.

Batch normalization'ın etkisi açık olsa da, etkinliğinin arkasındaki nedenler tartışmada kalmaya devam etmektedir. Parametre oluşturma ve her bir katmanın girdi dağılımındaki değişikliklerin ağın öğrenme hızını etkilediği iç değişkenin, internal covariate shift (Eğitim sırasında ağ parametrelerindeki değişiklik nedeniyle ağ aktivasyonlarının dağılımındaki değişiklik) sorununu hafifletebileceğine inanılmaktaydı. Son zamanlarda, bazı araştırmacılar batch normalization'ın internal covariate shift'i azaltmadığını, performansı iyileştirmek için amaç fonksiyonunu pürüzsüzleştirdiğini göstermiştir. (WEB_11)

2.2.6. Dropout

Dropout, overfitting (aşırı öğrenme) olduğu durumlarda verileri dengeleyip bu sorunu çözmek amacıyla kullanılan bir tekniktir.

Dropout, rastgele seçilen nöronların eğitim sırasında ihmal edildiği bir tekniktir. Bazı nöronlar rastgele bir şekilde çıkarılır. Bu, bu nöronların forward pass'de aktivasyona bir etki etmemesi (kaldırılması) ve backward pass'de de yine bu nöronlar için hiçbir ağırlık güncellemesi olunmaması anlamına gelir.



Şekil 2.24. Dropout öncesi ve sonrası bir neural network (WEB_12)

2.2.7. Early Stopping

Early stopping (erken durdurma), makine öğreniminde bir sinir ağını eğitirken overfitting'i (aşırı öğrenme) engellemek için kullanılan bir yöntemdir. Bu yöntem, ağı her tekrarda eğitim verisine daha uygun hale getirecek şekilde günceller. Bir noktaya kadar, bu, sinir ağının eğitim seti dışındaki verilerdeki performansını artırır. Ancak bir noktadan sonra, sinir ağının eğitim verilerine uygunluğunu arttırmak, artan genelleme hatasına (bir algoritmanın daha önce görülmemiş veriler

için sonuç değerlerini ne kadar doğru tahmin edebileceğinin bir ölçüsü) neden olur. Yani, test verilerinde başarı düşer.

Early stopping, ağın overfitting'e başlamadan önce kaç kez çalıştırılabileceği konusunda rehberlik sağlar. (WEB_13)

2.3. Yazılım Dilleri ve Geliştirme Ortamı

Python 3.6.7 projenin tamanında yazılım dili olarak kullanılmıştır. Proje, ortak çalışma yürütülebilmesi açısından Google Colaboratory üzerinde yazılmış olup, Anaconda platformu ile de bilgisayarlarımızda Python kullanımının daha pratik hale getirilmesi sağlanmıştır.

2.3.1. Python 3.6.7

Projede yazılım dili olarak, birçok veri düzenleme ve derin öğrenme kütüphaneleri içermesi nedeniyle Python tercih edilmiştir. Versiyonu 3.6.7'dir.

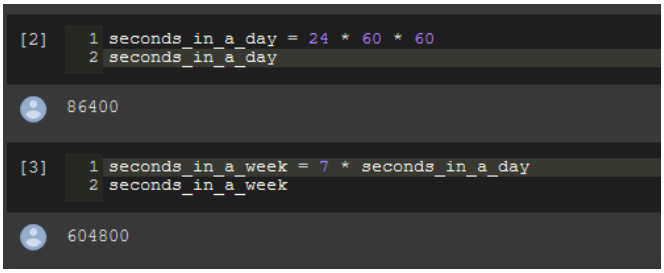
2.3.2. Anaconda

Anaconda, paket yönetimini basitleştirmeyi amaçlayan, bilimsel hesaplama için Python ve R programlama dillerinin (veri bilimi, makine öğrenimi uygulamaları, büyük ölçekli veri işleme, tahmin analizi, vb.) bulunduğu ücretsiz ve açık kaynaklı bir platformdur. Colab'da çalıştırmadığımız kodları Anaconda yardımı ile kendi bilgisayarımızda çalıştırdık.

2.3.3. Google Colaboratory

Colaboratory, kurulum gerektirmeyen ve tamamen bulutta çalışan ücretsiz bir Jupyter notebook ortamıdır. Colaboratory ile kod yazılabilir, uygulanabilir, analizler kaydedilebilir, paylaşılabilir ve güçlü bilgisayar kaynaklarına tarayıcıdan ücretsiz olarak erişim sağlanabilmektedir.

Projemizde bulunan kodların neredeyse tamamı bu platform üzerinde çalıştırılmıştır. Colaboratory, bir Google ürünü olduğu için Google Drive ile entegre bir şekilde çalışabilmektedir. Bu da bize bu projede büyük bir kolaylık sağlamıştır. Kaynak kodlar ve projeyle ilgili hemen hemen tüm veriler dosya halinde burada tutulmuştur.



Şekil 2.25. Google Colab örneği

2.4. Veri Setleri

Bu başlık altında kullanılan veri setlerinden bahsedilmiştir.

2.4.1. Emotion-Annotated Dataset

Kullandığımız **Emotion-Annotated Dataset** sadece akademik araştırmalar için erişilebilir bir veri setidir[1]. Bu veri seti çeşitli bloglardan alınan yazılar üzerinden oluşturulmuştur. Bu yazılar Ekman'ın evrensel duygu kategorilerine göre (*happiness, sadness, anger, disgust, surprise, fear, mixed emotion, no emotion*) cümle cümle sınıflandırılmıştır. Blog cümlelerinin hangi duygu sınıfına ait olduğu, dört kişilik bir kontrol ekibi tarafından belirlenmiştir. A kişisi tüm verileri, B, C ve D kişilerinden her biri, verinin farklı üçte birlik kısmını kategorilendirmiştir. B, C ve D kişilerinin belirlediği kategoriler, A kişisinin kategorileriyle karşılatırılıp, kategorilerde ortak karara varılmışsa verilen kategori veri setine eklenmiştir. (S. Aman & S. Szpakowicz. (2007). Identifying Expressions of Emotion in Text. V. Matousek, P. Mautner (eds.): Proc 10th International Conf. on Text, Speech and Dialogue TSD 2007, Plzeň, Czech Republic, Lecture Notes in Computer Science 4629, Springer, 196-205.) (S. Aman. (2007). Recognizing Emotions in Text. Master of Computer Science, University of Ottawa.)

Çizelge 2.1. Emotion annotated dataset örnekleri

Sentence	Emotion
I have to look at life in her perspective, and it would break anyone's heart.	sadness
We stayed in a tiny mountain village called Droushia, and these people brought hospitality to incredible new heights.	surprise
But the rest of it came across as a really angry, drunken rant.	anger
And I reallllllly want to go to Germany – dang terrorists are making flying overseas all scary and annoying and expensive though!!	mixed emotion
I hate it when certain people always seem to be better at me in everything they do.	disgust
Which, to be honest, was making Brad slightly nervous.	fear

2.4.2. Twitter Dataset

1 Haziran 2009'dan 31 Aralık 2009'a kadar olan süre diliminde 20 milyon Twitter kullanıcısı tarafından atılmış 467 milyon tweet içerir. Bu veri setinin, belirtilen süre içerisinde Twitter sosyal ağında, herkese açık bir şekilde atılan tüm tweet'lerin %20-30'unu kapsadığı düşünülmektedir. (WEB_14) (J. Yang, J. Leskovec. (2011). Patterns of Temporal Variation in Online Media. ACM International Conference on Web Search and Data Mining (WSDM '11).)

Her tweet için şu bilgiler mevcuttur:

- Yazar

1 https://saimacs.github.io/index.html

- Zaman
- İçerik

Projemizde farklı yöntemler kullanılarak filtrelenen Twitter veri seti, ana veri setimiz olan Emotion-Annotated veri setine ek olarak kullanılmıştır.

Çizelge 2.2. Üretilen ek veri seti örnekleri

Sentence	Emotion
I really really hate confusing weird essays about which you can find no good information!	anger
how do you live in this room, you have to call the cleaner.	disgust
When I saw the demon's face, I fell out of the chair because of panic.	fear
they asked: how much does it cost to go to high school in america? i choked on my tears looking down at the dirt on their feet...	sadness
omg! that's crazy!!! i saw like 4 firetrucks.	surprise

2.4.3. Alm Fairy Tale Dataset

Ürettiğimiz modelleri test edip karşılaştırmak için Ebba Cecilia Ovesdotter Alm tarafından "Affect in text and speech" tezi için çeşitli masallardaki cümleleri kategorilendirerek üretilen "Fairy Tale" veri seti kullanılmıştır. (E. C. O. Alm. (2008). Affect in text and speech. PhD Dissertation, Urbana, IL: University of Illinois at Urbana-Champaign.) Sonuçlar Ameeta Agrawal'ın "Unsupervised emotion detection from text using semantic and syntactic relations" tezindeki sonuçlarla karşılaştırılmıştır. (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using semantic and syntactic relations. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.)

Çizelge 2.3. Alm Fairy Tale Dataset örnekleri

Sentence	Emotion
She sat down again, and stared mournfully at the grate.	sadness
She seemed to be in a terrible fright.	fear
The rabbits could not bear him; they could smell him half a mile off.	anger-disgust
He was sitting all over a small rocking chair, twiddling his thumbs and smiling, with his feet on the fender.	happiness
At last the rope gave way with such a sudden jerk that it nearly pulled his teeth out, and quite knocked him over backwards.	surprise

2.4.4. ISEAR Emotion Dataset

Son olarak kullandığımız veri seti 7466 cümle içeren ISEAR (International Survey on Emotion Antecedents and Reactions) veri setidir[2]. Bu veri seti de sonuçlarımızı karşılaştırmak için

2 The ISEAR Dataset is available from : http://emotion-research.net/toolbox/toolboxdatabase.2006-10-13.2581092615

kullanılmıştır. Bu veri seti üretilirken farklı çevrelerden gelen 1096 katılımcıya *anger, disgust, fear, joy, sadness, shame* ve *guilt* duyguları ile ilgili sorular sorularak üretilmiştir. Bizim modelimizde *shame* ve *guilt* duyguları bulunmadığı için, sonuçlar üretilirken bu iki duygu yoksayılmıştır.

Çizelge 2.4. ISEAR Emotion Dataset örnekleri

Sentence	Emotion
When I was robbed in a bus.	anger
The smell of garlic in rush-hour bus.	disgust
When a thief broke into my house at night.	fear
I ignored and offended my parents on the eve of the New Year.	guilt
I felt joy when my two twin sisters were born.	joy
My grandfather’s funeral.	sadness
I am ashamed of the horrible way I used to treat my sister.	shame

3. UYGULAMA

Bu başlık altında, veri seti üzerindeki düzenlemelerden, ara adım kodlarından ve üretilen son modelden bahsedilmiştir.

3.1. Veri Seti Düzenlemeleri

Öncelikle veri setlerinin okunması yapılmalıdır. Emotion-Annotated Dataset için, regex kullanımı ile cümleler dosyası ve karşılık gelen duygular dosyası sadeleştirilmiştir (numaralandırmaların kaldırılması vb.). Kalan dosyalardan, **no emotion** ve **mixed emotion** kategorileri çıkartılmıştır. Bu veri seti, 536 **happiness**, 173 **sadness**, 179 **anger**, 172 **disgust**, 115 **surprise** ve 115 **fear** cümlesi içermektedir. Bu veri setinde toplam 1290 adet cümle vardır. İleriki adımlarda göreceğimiz gibi, modelin eğitimini zorlaştırmak ve sonuçlarda beklenmedik bir değişiklik görüp göremeyeceğimizi test etmek için bu veri seti, grup üyeleri tarafından elle filtrelenip, yeni bir filtrelenmiş subset veri seti oluşturulmuştur. Bu filtrelenmiş veri seti, 372 **happiness**, 149 **sadness**, 154 **anger**, 123 **disgust**, 110 **surprise** ve 103 **fear** cümlesi içermektedir. Bu veri setinde toplam 1011 adet cümle vardır.

Twitter Dataset için ise filtreleme kısmında duygular için belli bir seed word listesi kullanılarak, bu seed wordleri içeren tweetler filtrelenip, içlerinden bu seed wordler çıkarılsa bile duygu bilgisini barındıran cümleler, grup üyeleri tarafından kontrol edilip ortak karara varılan cümleler veri setine eklenmiştir.

Çizelge 3.1. Lists of emotion-related seed words used to build blog corpus (S. Aman. (2007). Recognizing Emotions in Text, sayfa 79. Master of Computer Science, University of Ottawa.)

Emotion Category	Seed Words
Happiness	awesome, happy, amused, fantastic, excited, pleased, cheerful, love, great, amazing
Sadness	sad, lonely, gloomy, depressed, unhappy, down, disheartened, sorrowful, painful, guilty
Anger	angry, annoyed, boiling, enraged, indignant, irate, furious, inflamed, livid, mad
Disgust	stupid, sucks, irritated, humiliated, disgusted, nauseating, sickening, contempt, repelling, unpleasant
Surprise	astonished, bewildered, surprised, confused, sudden, unaware, shocked, perplexed, what, unexpected
Fear	afraid, frightened, fearful, horrified, nervous, panicked, alarmed, phobia, scared, insecure

Sonuçta, 93 **anger**, 13 **disgust**, 44 **fear**, 174 **sadness** ve 49 **surprise** cümlesi bizim tarafımızdan üretilip, küçük bir ek veri seti oluşturulmuştur. Bu ek veri seti 373 adet cümleden oluşmaktadır.

3.2. Veri Setinin Python'da Kullanımı

Emotion-Annotated Dataset satır başına bir girdi olmak üzere, cümleler ve karşılık gelen duygular isimli iki farklı dosyada bulunmaktadır. Bu cümleler ve kategorileri verilen aşağıda kod parçasıyla okunmuştur.

```
from nltk.parse.corenlp import CoreNLPParser
parser = CoreNLPParser(url='http://localhost:9000')
dirname = "D:/Emotion Research/"

our_file_1 = dirname + "Emotion-Data/Annotated Data/annotset.txt"
our_file_2 = dirname + "Emotion-Data/Annotated Data/basefile.txt"
dataset = []

with open(our_file_1, "r") as file1, open(our_file_2, "r") as file2:
    for line_from_file_1, line_from_file_2 in zip(file1, file2):
        output = None
        line1 = line_from_file_1.split()
        line2 = line_from_file_2
        if line1[0] == "hp":
            output = 0
        elif line1[0] == "sd":
            output = 1
        elif line1[0] == "ag":
            output = 2
        elif line1[0] == "dg":
            output = 3
        elif line1[0] == "sp":
            output = 4
        elif line1[0] == "fr":
            output = 5
        dataset.append((output, list(parser.tokenize(line2))))
```

CoreNLPParser ve önemi, bir sonraki başlıkta anlatılmaktadır. Bizim tarafımızdan üretilen ek veri seti ise duygu isimlerine göre ayrılmış dosyalarda bulunmaktadır. *"sadness.txt"* dosyasında sadness duygu cümleleri, *"anger.txt"* dosyasında anger cümleleri gibi. Bu cümleler ise aşağıdaki kod parçasıyla okunmuştur.

```
for emotion in [(1, 'sadness.txt'), (2, 'anger.txt'), (3,
'disgust.txt'), (4, 'surprise.txt'), (5, 'fear.txt')]:
    emotion_our_filename = dirname + "Emotion-Data Generated by us/" +
emotion[1]
    with open(emotion_our_filename, "r") as file:
        for line in file:
            dataset.append((emotion[0], list(parser.tokenize(line))))
```

Sonuçta üretilen dataset isimli değişken, pickle adı verilen serialization-deserialization binary protokollerini implement eden Python modülü ile her seferinde yapılması masraflı olan tokenizer işleminden kaçınmak için bir dosyaya yazılmıştır.

```
import _pickle as cPickle
with open(dirname + "Pickle/dataset", 'wb') as outfile:
    cPickle.dump(dataset, outfile)
```

Bu dosya, bundan sonra veri setini okumak için unpicklelanacak ve zip ile veriler iki ayrı listeye çıkarılacaktır. Bunun için kullanılan kod parçası aşağıdadır.

```
import _pickle as cPickle
with open(dirname + "Pickle/dataset", 'rb') as infile:
    dataset = cPickle.load(infile)

outputs, filelines = zip(*filtereddata)
outputs = torch.LongTensor(outputs).to(device)
```

3.3. Tokenization İşlemi

Bir önceki başlıkta bahsedildiği gibi, dosyadan okunulan cümleler tokenize edilip projede kullanıma hazır hale getirilir. Tokenization işlemi projemizde 2 farklı uygulama için 2 farklı şekilde yapılmaktadır. Birincisi Stanford Tokenizer kullanılarak, **2.1.1. Stanford Tokenizer** başlığında belirtildiği gibi server çalıştırılarak, bu servera istekler gönderilerek tokenization yapılmasıdır. İkinci yöntem ise, fastText kütüphanesi kullanılarak, fastText'te implement edilen tokenization'ın yapılmasıdır. GloVe vektörlü eğitimler için birinci tokenization (bu vektörlerin oluşumunda bu tokenizer kullanıldığı için (Jeffrey Pennington et al. (2014). GloVe: Global Vectors for Word Representation, sayfa 7. Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014) 12.)), fastText vektörlü eğitimler için ikinci tokenization kullanılmıştır. **3.2. Veri Setinin Python'da Kullanımı** başlığında gösterilmiş olan işlem, Stanford Tokenizer tokenization işlemidir. FastText tokenization ise kısaca aşağıdaki satır ile yapılabilir.

```
fastText.tokenize(line)
```

3.4. Vektörlerin Kullanımı

Tokenize edilmiş cümleler, şu haldeki Python listelerine dönüşmüş şekilde filelines isimli listede tutulmaktadır.

Tokenization öncesi:

I have to look at life in her perspective, and it would break anyones heart.

Tokenization sonrası:

['I', 'have', 'to', 'look', 'at', 'life', 'in', 'her', 'perspective', ',', 'and', 'it', 'would', 'break', 'anyones', 'heart', '.']

Bu listeler, vektörler listesine dönüştürülüp, LSTM modeline girmek için uygun hale getirilmek amacıyla `ready_data(outputs, filelines)` fonksiyonuna girdi olarak verilir. Bu fonksiyon çıktı olarak ise "`padded_vectors, targets, sent_lengths`" değişkenlerini verir.

“padded_vectors” girdi olarak verilmiş cümleler içerisindeki en uzun cümle uzunluğuna tamamlanmış vektörlerdir. Toplam cümle sayısı kadar eleman içerir. LSTM eğitimi sırasında padlenmiş bölgeler yok sayılır ve cümleler sadece kendi uzunlukları kadar varsayılır. “targets” cümlelerin çıktıları, “sent_lengths” ise her cümlenin kelime uzunluğudur. LSTM modelinde bu uzunluklar kullanılarak cümle için kaç kelimede eğitim yapılacağı bulunur. Örnek olarak:

```
filelines = (['test', 'sentence'], ['second', 'test', 'sentence'])
outputs = (5, 3)
padded_vectors, targets, sent_lengths = ready_data(outputs, filelines)
```

verilirse, çıktılar bu şekilde olmuştur:

```
>>>print(padded_vectors.shape)
torch.Size([3, 2, 300])
>>>print(targets)
(5, 3)
>>>print(sent_lengths)
tensor([2, 3], device='cuda:0')
```

Bizim tarafımızdan yazılmış bir diğer fonksiyon get_batch(batch, batch_size, x, y, sent_len) fonksiyonudur. Çıktı olarak “x, y, sent_len” verir. Burada x girdiler, y gerçek kategoriler ve sent_len ise cümle uzunluklarıdır. Çıktılar “batch_size” boyutlarında batchlerdir. İlk argüman ise kaçıncı batch'i döndürmesi gerektiğini söyler.

3.5. Early Stopping

Early stopping, overfitting’i engellemek için kullanılan bir regularization tekniğidir. Early stopping, test loss’unu takip eder ve test loss’u eğitimde azalmayı birkaç epoch boyunca bırakırsa, eğitimi durdurur. Kullandığımız EarlyStopping class’ı, PyTorch modelini eğitirken test loss’unu ve modeli aklında tutan bir obje oluşturmak için kullanılır. Test loss’u her azaldığında modelin bir checkpoint’ini diske kaydeder ve patience değeri kadar epoch boyunca test loss’u azalmazsa, kaydedilmiş model en iyi kabul edilir. (WEB_15) Modelimizde patience değeri olarak 20 tercih edilmiştir. Örnek olarak aşağıdaki kod gösterilebilir:

```
early_stopping(testlosses[-1], model)
if early_stopping.early_stop:
    print("Early stopping")
    break
```

3.6. LSTM Modeli

Projenin tasarım aşamasında danışmanımızla beraber en uygun modelin LSTM olduğuna karar verildi. Modelimiz opsiyonel olarak, vektör büyüklüğünü, model büyüklüğünü, katman sayısı, iki yönlü olup olmadığı ve batchnorm yapılıp yapılmayacağı bilgisini, LSTM katmanları arasındaki

dropout değerini ve çıktı katmanındaki dropout değerini alır. Model initialization method tanımı aşağıdaki gibidir:

```
def init(self, input_size, hidden_size, num_layers, bidirectional,
batchnormactive, dropout_hidden, dropout_output):
```

`input_size`, `hidden_size` ve `num_layers` integer, `bidirectional` ve `batchnormactive` boolean, `dropout_hidden` ve `dropout_output` [0, 1] aralığında float değerleri alır.

3.7. Eğitim

Eğitim kısmı, projemizin en geniş kısmıdır. Öncelikle test edilmek ve kaydedilmek istenen parametreler, parametreler listesine aşağıdaki şekilde eklenir:

```
lrlist = [0.001, 0.003]
batchsizelist = [16, 32, 64, 128]
hdlist = [128, 256]
lylist = [1, 2]
bdlist = [True, False]
bnlist = [True, False]
dphlist = [0.0, 0.2, 0.4]
dpolist = [0, 0.4, 0.6, 0.8]
number_of_epochs = 1000
n_splits = 5
early_stopping_patience = 15
```

Bu değerler sırasıyla, learning rate, eğitimde batchlerin boyutu, LSTM büyüklüğü, LSTM katman adedi, LSTM modelinin bidirectional olup olmadığı, batchnorm'un aktif olup olmadığı, LSTM katmanları arası dropout değeri, LSTM katmanı ile çıktı katmanı arasındaki dropout değeri, çalıştırılacak epoch sayısı (early stopping ile daha erken bitebilir), stratified KFold için split sayısı ve early stopping için patience değeridir. Loss fonksiyonu olarak CrossEntropyLoss, optimizer olarak ise Adam optimizer (weight_decay=1e-5 (L2 regularization)) kullanılmıştır. Her fold'da (eğitim epoch sınırına ulaştığında veya early stopping tarafından durdurulduğunda), train/test classification reports ve confidence matrices bir listeye eklenir, her fold'un en iyi modelleri "models/" klasörü içerisine yazılır. Model isimleri:

```
"models/es_n" + str(i) + "+b" + str(batch_size) + "+e" +
str(number_of_epochs) + "+lr" + str(learning_rate) + "+hidden" +
str(hidden_size) + "+ly" + str(num_layers) + ("+bd" if bidirectional
else "") + ("+bn" if batchnorm else "") + "+dp_h" +
str(dropout_hidden) + "+dp_o" + str(dropout_output) + ".pth"
```

şeklindedir. Örnek olarak "*es_n1+b32+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.2+dp_o0.4.pth*" verilirse, buradan, bu modelin early stopping kullanılarak üretildiğini, ilk fold'un sonuç modeli olduğunu, eğitimde 32'lik batchler kullanıldığını, 100 epoch boyunca eğitilmiş olduğunu, learning rate'in 0.001 olduğunu, LSTM boyutunun 256 olduğunu, 2 layerlı bir LSTM

modeli kullanıldığını, LSTM modelinin bidirectional olduğunu, batchnorm kullanıldığını, LSTM katmanları arasında 0.2 dropout olduğunu ve LSTM ile çıktı katmanı arasında 0.4 dropout olduğunu anlayabiliriz.

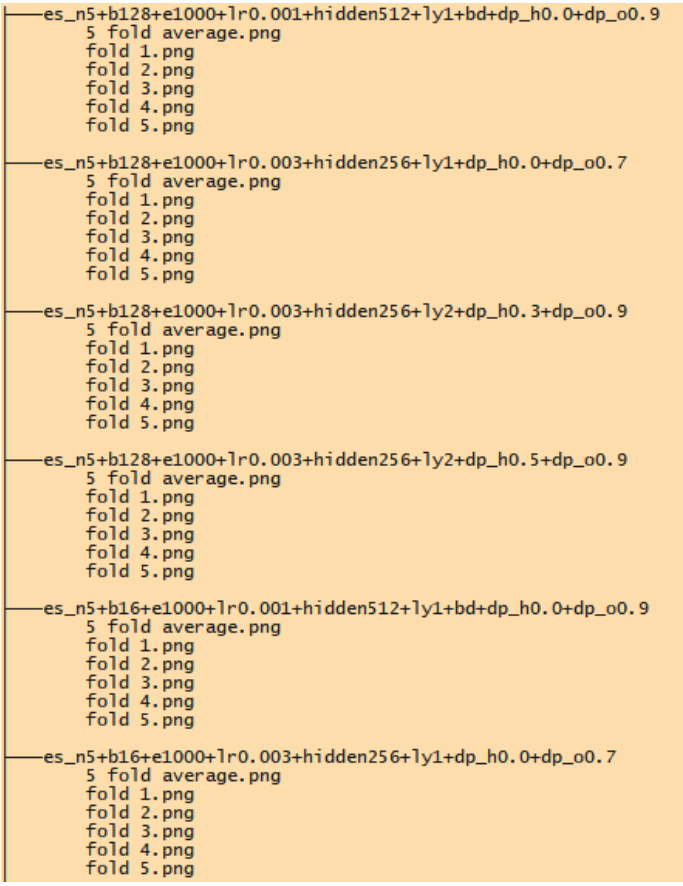
Bu listeler, daha sonradan modelin değerlendirilmiş halini çıktı olarak vermek için kullanılır.

3.8. Plot Fonksiyonları

Eğer eğitim, early stopping veya epoch sayısına ulaşması sebebiyle bitiriyorsa, bu fold'daki en iyi modelin train ve test verilerindeki sonuçları "results/" klasörü içinde kendisine ait bir klasöre yazılır. Klasör adı:

```
"results/" + "es_" + "n" + str(n_splits) + "+b" + str(batch_size) +
"+e" + str(number_of_epochs) + "+lr" + str(learning_rate) + "+hidden"
+ str(hidden_size) + "+ly" + str(num_layers) + ("+bd" if bidirectional
else "") + ("+bn" if batchnorm else "") + "+dp_h" +
str(dropout_hidden) + "+dp_o" + str(dropout_output) + "/"
```

şeklinde, model değerlendirme dosyaları ise "*fold 1.png*", "*fold 2.png*" ... "*fold n.png*" şeklindedir. Fold ortalamaları da "n fold average.png" dosyasında yazılmıştır. Örnek bir klasör ağacı **Şekil 3.1.**'de verilmiştir.



Şekil 3.1. Örnek bir result klasör ağacı

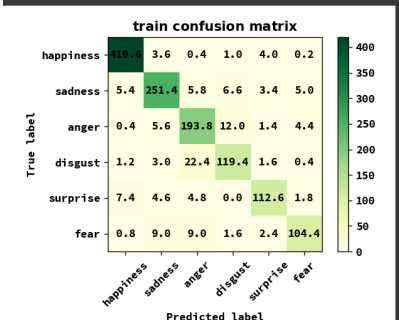
```
LEARNING RATE = 0.001
BATCH SIZE = 16
HIDDEN_SIZE = 512
1 LAYERS
BIDIRECTIONAL YES
BATCHNORM NO
DROPOUT HIDDEN = 0.0
DROPOUT OUTPUT = 0.9

5 FOLD AVERAGE
EPOCH 24
```

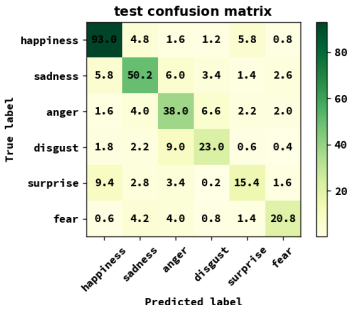
TRAIN DATA

	precision	recall	f1-score	support
happiness	0.968	0.978	0.972	428.800
sadness	0.912	0.906	0.908	277.600
anger	0.822	0.890	0.852	217.600
disgust	0.854	0.806	0.828	148.000
surprise	0.900	0.858	0.876	131.200
fear	0.902	0.822	0.856	127.200
micro avg	0.902	0.902	0.902	1330.400
macro avg	0.892	0.876	0.882	1330.400
weighted avg	0.906	0.902	0.904	1330.400

TEST DATA

	precision	recall	f1-score	support
happiness	0.832	0.868	0.850	107.200
sadness	0.748	0.726	0.732	69.400
anger	0.618	0.700	0.652	54.400
disgust	0.656	0.620	0.636	37.000
surprise	0.576	0.468	0.514	32.800
fear	0.736	0.654	0.690	31.800
micro avg	0.722	0.722	0.722	332.600
macro avg	0.694	0.672	0.678	332.600
weighted avg	0.726	0.722	0.718	332.600





Şekil 3.2. fold average dosyası örneği

4. SONUÇLAR VE DEĞERLENDİRME

Bu proje kapsamında **3. UYGULAMA** başlığında belirtilen şekilde yüzlerce model eğitilmiştir. Bu başlık altında farklı parametrelerle üretilen modellerin karşılaştırması, parametrelerin başarı üzerinde etkisi ve son olarak en iyi modelin, diğer araştırmalarda yapılan classification sonuçları ile karşılaştırması yapılmıştır. Aşağıda görülen modellerin hepsinde 5-fold stratified cross validation kullanılıp, micro ve macro averagelar verilmiştir.

4.1. Veri Seti Karşılaştırması

Tam olarak net yorum yapılamamasına karşın, emotion annotated dataset ile genellikle daha iyi sonuçlar elde edilmiştir.

Çizelge 4.1. Kullanılan veri setinin performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
emotion annotated dataset es_n5+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.656/0.572	0.656/0.512	0.656/0.514
filtrelenmiş emotion annotated dataset es_n5+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.642/0.588	0.642/0.550	0.642/0.556
filtrelenmiş emotion annotated dataset ve kendi verimiz es_n5+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.640/**0.612**	0.640/0.590	0.640/0.594
emotion annotated dataset ve kendi verimiz es_n5+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	**0.666**/**0.612**	**0.666**/**0.594**	**0.666**/**0.596**
emotion annotated dataset es_n5+e1000+lr0.003+hidden512+ly2+dp_h0.5+dp_o0.7	**0.706**/0.640	**0.706**/0.608	**0.706**/0.614
filtrelenmiş emotion annotated dataset es_n5+e1000+lr0.003+hidden512+ly2+dp_h0.5+dp_o0.7	0.640/0.588	0.640/0.546	0.640/0.550
filtrelenmiş emotion annotated dataset ve kendi verimiz es_n5+e1000+lr0.003+hidden512+ly2+dp_h0.5+dp_o0.7	0.658/0.626	0.658/0.616	0.658/0.616
emotion annotated dataset ve kendi verimiz es_n5+e1000+lr0.003+hidden512+ly2+dp_h0.5+dp_o0.7	0.678/**0.642**	0.678/**0.626**	0.678/**0.622**
emotion annotated dataset es_n5+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	**0.672**/0.580	**0.672**/0.540	**0.672**/0.544
filtrelenmiş emotion annotated dataset es_n5+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.648/0.600	0.648/0.570	0.648/0.576
filtrelenmiş emotion annotated dataset ve kendi verimiz es_n5+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.664/**0.632**	0.664/**0.622**	0.664/**0.620**

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
emotion annotated dataset ve kendi verimiz es_n5+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.646/0.578	0.646/0.574	0.646/0.568
emotion annotated dataset es_n5+e1000+lr0.003+hidden300+ly2+dp_h0.3+dp_o0.8	**0.694**/**0.626**	**0.694**/0.612	**0.694**/**0.612**
filtrelenmiş emotion annotated dataset es_n5+e1000+lr0.003+hidden300+ly2+dp_h0.3+dp_o0.8	0.654/0.584	0.654/0.560	0.654/0.560
filtrelenmiş emotion annotated dataset ve kendi verimiz es_n5+e1000+lr0.003+hidden300+ly2+dp_h0.3+dp_o0.8	0.650/0.624	0.650/0.604	0.650/0.604
emotion annotated dataset ve kendi verimiz es_n5+e1000+lr0.003+hidden300+ly2+dp_h0.3+dp_o0.8	0.668/0.620	0.668/**0.614**	0.668/**0.612**

4.2. Veri Setinde Büyük-Küçük Harf Olup Olmama Durumu

Büyük-küçük harf içeren ve sadece küçük harf içeren modeller karşılaştırıldığında, performansın yorumlanabilir bir şekilde değişmediği görülmüştür.

Çizelge 4.2. Veri setinde büyük-küçük harf olup olmama durumlarının performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
emotion annotated dataset, upper case es_n5+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.672/0.580	0.672/0.540	0.672/0.544
emotion annotated dataset, lower case es_n5+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	**0.678**/**0.600**	**0.678**/**0.586**	**0.678**/**0.584**
filtrelenmiş emotion annotated dataset, upper case es_n5+e1000+lr0.003+hidden512+ly1+dp_h0.0+dp_o0.9	**0.670**/**0.634**	**0.670**/**0.600**	**0.670**/**0.608**
filtrelenmiş emotion annotated dataset, lower case es_n5+e1000+lr0.003+hidden512+ly1+dp_h0.0+dp_o0.9	**0.670**/0.630	**0.670**/0.594	**0.670**/0.602
filtrelenmiş emotion annotated dataset ve kendi verimiz, upper case es_n5+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	**0.640**/0.612	**0.640**/**0.590**	**0.640**/**0.594**
filtrelenmiş emotion annotated dataset ve kendi verimiz, lower case es_n5+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.634/**0.616**	0.634/0.574	0.634/0.576
emotion annotated dataset ve kendi verimiz, upper case es_n5+e1000+lr0.003+hidden300+ly2+dp_h0.3+dp_o0.8	**0.668**/0.620	0.668/**0.614**	0.668/**0.612**
emotion annotated dataset ve kendi verimiz, lower case es_n5+e1000+lr0.003+hidden300+ly2+dp_h0.3+dp_o0.8	0.674/**0.644**	**0.674**/0.612	**0.674**/**0.612**

4.3. Batch Size

Batch size'ın değiştirilip, diğer parametrelerin sabit tutulduğu örneklerde, batch_size 16 verildiğinde en iyi sonuç alındığı gözlemlenmiştir. Fakat layer sayısı 2 iken, batch_size 64 ve 128 verildiğinde de sonuçlar iyileşmiştir.

Çizelge 4.3. Batch size'ın performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+b8+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.696/0.672	0.696/0.624	0.696/0.628
es_n5+b16+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	**0.722**/**0.694**	**0.722**/**0.672**	**0.722**/**0.678**
es_n5+b32+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.716/0.692	0.716/0.656	0.716/0.664
es_n5+b64+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.694/0.668	0.694/0.628	0.694/0.632
es_n5+b128+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.696/0.662	0.696/0.638	0.696/0.642
es_n5+b256+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.686/0.650	0.686/0.620	0.686/0.624
es_n5+b1500+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.656/0.584	0.656/0.560	0.656/0.564
es_n5+b8+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	0.682/0.650	0.682/**0.636**	0.682/0.630
es_n5+b16+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	0.680/**0.668**	0.680/0.614	0.680/0.618
es_n5+b32+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	0.678/0.630	0.678/0.622	0.678/0.618
es_n5+b64+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	0.686/0.654	0.686/0.612	0.686/0.616
es_n5+b128+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	**0.694**/0.652	**0.694**/0.632	**0.694**/**0.636**
es_n5+b256+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	0.678/0.634	0.678/0.618	0.678/0.616
es_n5+b1500+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	0.672/0.624	0.672/0.588	0.672/0.598
es_n5+b8+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	0.708/0.678	0.708/0.648	0.708/0.654
es_n5+b16+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	**0.710**/0.676	**0.710**/**0.658**	**0.710**/**0.662**
es_n5+b32+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	0.708/**0.696**	0.708/0.640	0.708/0.654
es_n5+b64+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	0.702/0.682	0.702/0.652	0.702/0.652
es_n5+b128+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	0.690/0.666	0.690/0.628	0.690/0.636
es_n5+b256+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	0.690/0.644	0.690/0.638	0.690/0.638
es_n5+b1500+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	0.670/0.626	0.670/0.606	0.670/0.610
es_n5+b8+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.674/0.632	0.674/0.596	0.674/0.588
es_n5+b16+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.678/0.644	0.678/0.616	0.678/0.620
es_n5+b32+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.694/0.658	0.694/0.634	0.694/0.628
es_n5+b64+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	**0.696**/**0.670**	**0.696**/**0.636**	**0.696**/0.640
es_n5+b128+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.694/0.660	0.694/**0.636**	0.694/**0.642**
es_n5+b256+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.670/0.630	0.670/0.610	0.670/0.608
es_n5+b1500+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.672/0.624	0.672/0.600	0.672/0.604

4.4. FastText vs. GloVe (Kelime Vektörleri)

Kelime vektörleri için yapılan karşılaştırmalarda, her durumda GloVe vektörlerinin daha hızlı ve daha iyi öğrendikleri gözlemlenmiştir.

Çizelge 4.4. FastText ve GloVe performans karşılatırması

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
FastText es_n5+b16+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.540/0.314	0.540/0.352	0.540/0.302
GloVe es_n5+b16+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	**0.744/0.684**	**0.744/0.676**	**0.744/0.672**
FastText es_n5+b16+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	0.582/0.402	0.582/0.416	0.582/0.400
GloVe es_n5+b16+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	**0.758/0.714**	**0.758/0.698**	**0.758/0.700**
FastText es_n5+b16+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	0.540/0.332	0.540/0.368	0.540/0.330
GloVe es_n5+b16+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	**0.734/0.702**	**0.734/0.636**	**0.734/0.640**
FastText es_n5+b16+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.558/0.388	0.558/0.396	0.558/0.360
GloVe es_n5+b16+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	**0.720/0.668**	**0.720/0.636**	**0.720/0.636**

4.5. LSTM Bidirectional

LSTM modelinin bidirectional olup olmamasının performansa yorumlanabilir bir etkisi olmamıştır.

Çizelge 4.5. Bidirectional LSTM modelinin performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.001+hidden256+ly1+bd+dp_h0.0+dp_o0.0	**0.700/0.636**	**0.700/0.608**	**0.700/0.610**
es_n5+e100+lr0.001+hidden256+ly1+dp_h0.0+dp_o0.0	0.688/0.616	0.688/0.580	0.688/0.586
es_n5+e100+lr0.001+hidden256+ly2+bd+dp_h0.4+dp_o0.8	0.674/0.594	0.674/0.570	0.674/0.572
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.4+dp_o0.8	**0.716/0.658**	**0.716/0.626**	**0.716/0.632**

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.003+hidden512+ly2+bd+dp_h0.5+dp_o0.9	**0.716**/**0.676**	**0.716**/**0.650**	**0.716**/**0.642**
es_n5+e100+lr0.003+hidden512+ly2+dp_h0.5+dp_o0.9	**0.716**/0.666	**0.716**/0.638	**0.716**/0.628
es_n5+e100+lr0.003+hidden512+ly1+bd+dp_h0.0+dp_o0.7	0.710/0.648	0.710/**0.638**	0.710/**0.636**
es_n5+e100+lr0.003+hidden512+ly1+dp_h0.0+dp_o0.7	**0.714**/**0.652**	**0.714**/0.628	**0.714**/0.632
es_n5+e100+lr0.001+hidden768+ly4+bd+dp_h0.2+dp_o0.8	**0.680**/**0.606**	**0.680**/**0.590**	**0.680**/**0.586**
es_n5+e100+lr0.001+hidden768+ly4+dp_h0.2+dp_o0.8	0.614/0.488	0.614/0.464	0.614/0.448

4.6. LSTM Modelinin Büyüklüğü

Learning_rate 0.003 iken, LSTM modelinin büyüklüğü arttıkça performansın olumsuz etkilendiği görülmüştür ancak 2 layer'lı bir modelde learning_rate 0.001 iken, 512 boyutlu LSTM modelinin iyi performans gösterdiği gözlemlenmiştir.

Çizelge 4.6. LSTM modelinin büyüklüğünün performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.003+hidden256+ly1+bd+dp_h0.0+dp_o0.9	**0.750**/**0.702**	**0.750**/**0.690**	**0.750**/**0.690**
es_n5+e100+lr0.003+hidden300+ly1+bd+dp_h0.0+dp_o0.9	0.738/0.684	0.738/0.658	0.738/0.664
es_n5+e100+lr0.003+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.722/0.664	0.722/0.632	0.722/0.638
es_n5+e100+lr0.003+hidden768+ly1+bd+dp_h0.0+dp_o0.9	0.742/0.698	0.742/0.668	0.742/0.674
es_n5+e100+lr0.003+hidden1024+ly1+bd+dp_h0.0+dp_o0.9	0.718/0.666	0.718/0.634	0.718/0.640
es_n5+e100+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.7	0.726/0.672	0.726/0.656	0.726/0.650
es_n5+e100+lr0.003+hidden300+ly2+dp_h0.5+dp_o0.7	0.726/0.670	0.726/0.660	0.726/0.662
es_n5+e100+lr0.003+hidden512+ly2+dp_h0.5+dp_o0.7	**0.734**/**0.692**	**0.734**/**0.670**	**0.734**/**0.664**
es_n5+e100+lr0.003+hidden768+ly2+dp_h0.5+dp_o0.7	0.664/0.584	0.664/0.560	0.664/0.562
es_n5+e100+lr0.003+hidden1024+ly2+dp_h0.5+dp_o0.7	0.614/0.532	0.641/0.478	0.614/0.476
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.6+dp_o0.8	0.706/0.640	0.706/0.604	0.706/0.614
es_n5+e100+lr0.001+hidden300+ly2+dp_h0.6+dp_o0.8	0.696/0.628	0.696/0.588	0.696/0.590
es_n5+e100+lr0.001+hidden512+ly2+dp_h0.6+dp_o0.8	**0.732**/0.676	**0.732**/0.640	**0.732**/0.648
es_n5+e100+lr0.001+hidden768+ly2+dp_h0.6+dp_o0.8	0.728/**0.690**	0.728/**0.652**	0.728/**0.656**
es_n5+e100+lr0.001+hidden1024+ly2+dp_h0.6+dp_o0.8	0.722/0.668	0.722/0.650	0.722/0.652

4.7. LSTM Layer Sayısı

Layer sayısı arttıkça performansın olumsuz etkilendiği gözlemlenmiştir.

Çizelge 4.7. LSTM layer sayısının performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	**0.734**/**0.684**	**0.734**/**0.670**	**0.734**/**0.672**
es_n5+e100+lr0.001+hidden512+ly2+bd+dp_h0.0+dp_o0.9	0.678/0.612	0.678/0.578	0.678/0.572
es_n5+e100+lr0.001+hidden512+ly4+bd+dp_h0.0+dp_o0.9	0.680/0.618	0.680/0.598	0.680/0.592
es_n5+e100+lr0.003+hidden256+ly1+bd+dp_h0.0+dp_o0.9	**0.750**/**0.702**	**0.750**/**0.690**	**0.750**/**0.690**
es_n5+e100+lr0.003+hidden256+ly2+bd+dp_h0.0+dp_o0.9	0.710/0.648	0.710/0.626	0.710/0.628
es_n5+e100+lr0.003+hidden256+ly4+bd+dp_h0.0+dp_o0.9	0.724/0.672	0.724/0.644	0.724/0.648
es_n5+e100+lr0.003+hidden300+ly2+dp_h0.3+dp_o0.8	**0.734**/**0.680**	**0.734**/**0.672**	**0.734**/**0.666**
es_n5+e100+lr0.003+hidden300+ly4+dp_h0.3+dp_o0.8	0.624/0.502	0.624/0.524	0.624/0.500
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.5+dp_o0.9	**0.714**/**0.642**	**0.714**/**0.604**	**0.714**/**0.608**
es_n5+e100+lr0.001+hidden256+ly3+dp_h0.5+dp_o0.9	0.668/0.576	0.668/0.552	0.668/0.550
es_n5+e100+lr0.001+hidden256+ly4+dp_h0.5+dp_o0.9	0.586/0.438	0.586/0.432	0.586/0.400

4.8. Learning Rate

Learning rate'in 0.003 olduğu modellerde sonucun, 0.001 ve 0.005 değerlerine göre performansa daha iyi etkisi olduğu gözlemlenmiştir. Ancak modele göre değişen durumlar söz konusu olabildilği için bu etkinin kesinliği hakkında net bir yorum yapılamamıştır.

Çizelge 4.8. Learning rate'in performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.001+hidden256+ly1+bd+dp_h0.0+dp_o0.9	0.714/0.644	0.714/0.624	0.714/0.628
es_n5+e100+lr0.003+hidden256+ly1+bd+dp_h0.0+dp_o0.9	**0.750**/**0.702**	**0.750**/**0.690**	**0.750**/**0.690**
es_n5+e100+lr0.005+hidden256+ly1+bd+dp_h0.0+dp_o0.9	0.738/0.690	0.738/0.650	0.738/0.654
es_n5+e100+lr0.001+hidden512+ly2+bd+dp_h0.3+dp_o0.9	**0.704**/0.632	**0.704**/**0.626**	**0.704**/**0.622**
es_n5+e100+lr0.003+hidden512+ly2+bd+dp_h0.3+dp_o0.9	0.702/**0.642**	0.702/0.616	0.702/0.616
es_n5+e100+lr0.005+hidden512+ly2+bd+dp_h0.3+dp_o0.9	0.690/0.628	0.690/0.592	0.690/0.594
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.3+dp_o0.9	0.722/0.678	0.722/0.628	0.722/0.636
es_n5+e100+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	**0.728**/**0.680**	**0.728**/**0.668**	**0.728**/**0.664**

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.005+hidden256+ly2+dp_h0.3+dp_o0.9	**0.728**/**0.680**	**0.728**/0.666	**0.728**/**0.664**

es_n5+e100+lr0.001+hidden256+ly1+dp_h0.0+dp_o0.9	**0.724**/**0.658**	**0.724**/**0.634**	**0.724**/**0.638**
es_n5+e100+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.9	0.710/0.640	0.710/0.606	0.710/0.610
es_n5+e100+lr0.005+hidden256+ly1+dp_h0.0+dp_o0.9	0.716/0.654	0.716/0.620	0.716/0.624

es_n5+e100+lr0.001+hidden256+ly2+bd+dp_h0.5+dp_o0.9	0.700/0.640	0.700/0.586	0.700/0.584
es_n5+e100+lr0.003+hidden256+ly2+bd+dp_h0.5+dp_o0.9	**0.728**/**0.680**	**0.728**/**0.644**	**0.728**/**0.648**
es_n5+e100+lr0.005+hidden256+ly2+bd+dp_h0.5+dp_o0.9	0.682/0.604	0.682/0.592	0.682/0.590

4.9. LSTM Katmanlar Arasındaki Dropout

LSTM dropout katsayısı, 2 layer'lı bir modelde 0.4 ve 0.5 iken performansa olumlu bir etkisi olmuştur. Dropout'un bu değerlerin altında ve üstünde olduğu modellerde performansta azalma gözlemlenmiştir.

Çizelge 4.9. LSTM Katmanları Arasındaki Dropout'un performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.0+dp_o0.0	0.594/0.536	0.564/0.454	0.594/0.460
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.2+dp_o0.0	0.594/**0.578**	0.594/0.458	0.594/0.452
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.4+dp_o0.0	**0.624**/0.544	**0.624**/**0.494**	**0.624**/**0.494**

es_n5+e100+lr0.001+hidden256+ly2+dp_h0.0+dp_o0.9	0.698/0.634	0.698/0.610	0.698/0.608
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.3+dp_o0.9	0.722/0.678	0.722/0.628	0.722/0.636
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.5+dp_o0.9	**0.740**/**0.692**	**0.740**/**0.646**	**0.740**/**0.654**
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.6+dp_o0.9	0.694/0.630	0.694/0.582	0.694/0.578
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.7+dp_o0.9	0.688/0.614	0.688/0.600	0.688/0.592
es_n5+e100+lr0.001+hidden256+ly2+dp_h0.8+dp_o0.9	0.698/0.632	0.698/0.602	0.698/0.604

es_n5+e100+lr0.001+hidden256+ly4+bd+bn+dp_h0.0+dp_o0.8	0.616/**0.576**	0.616/0.476	0.616/0.474
es_n5+e100+lr0.001+hidden256+ly4+bd+bn+dp_h0.2+dp_o0.8	0.592/0.570	0.592/0.444	0.592/0.444
es_n5+e100+lr0.001+hidden256+ly4+bd+bn+dp_h0.4+dp_o0.8	**0.642**/0.574	**0.642**/**0.534**	**0.642**/**0.536**
es_n5+e100+lr0.001+hidden256+ly4+bd+bn+dp_h0.5+dp_o0.8	0.610/0.514	0.610/0.472	0.610/0.476
es_n5+e100+lr0.001+hidden256+ly4+bd+bn+dp_h0.6+dp_o0.8	0.638/0.562	0.638/0.516	0.638/0.514
es_n5+e100+lr0.001+hidden256+ly4+bd+bn+dp_h0.7+dp_o0.8	0.616/0.482	0.616/0.490	0.616/0.470

es_n5+e100+lr0.001+hidden512+ly2+bd+dp_h0.0+dp_o0.7	**0.684**/**0.606**	**0.684**/**0.586**	**0.684**/**0.588**

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.001+hidden512+ly2+bd+dp_h0.3+dp_o0.7	0.668/0.570	0.668/0.550	0.668/0.546
es_n5+e100+lr0.001+hidden512+ly2+bd+dp_h0.5+dp_o0.7	0.670/**0.606**	0.670/0.562	0.670/0.564

4.10. LSTM Çıkış Katmanındaki Dropout

Çıkış katmanındaki dropout değeri arttırıldığında 1 ve 2 layer'lı modellerin performansında artış görülmüştür ancak 4 layer'lı modellerde dropout artışının performansa olumsuz bir etkisinin olduğu gözlemlenmiştir.

Çizelge 4.10. LSTM Çıkış Katmanındaki Dropout'un performansa etkisi

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.001+hidden256+ly1+bd+bn+dp_h0.0+dp_o0.0	0.634/0.560	0.634/0.520	0.634/0.518
es_n5+e100+lr0.001+hidden256+ly1+bd+bn+dp_h0.0+dp_o0.4	0.662/0.596	0.662/0.554	0.662/0.556
es_n5+e100+lr0.001+hidden256+ly1+bd+bn+dp_h0.0+dp_o0.8	**0.692**/**0.632**	**0.692**/**0.598**	**0.692**/**0.604**
es_n5+e100+lr0.001+hidden256+ly1+bd+dp_h0.0+dp_o0.0	0.700/0.636	0.700/0.608	0.700/0.610
es_n5+e100+lr0.001+hidden256+ly1+bd+dp_h0.0+dp_o0.4	0.696/0.634	0.696/0.598	0.696/0.602
es_n5+e100+lr0.001+hidden256+ly1+bd+dp_h0.0+dp_o0.7	**0.722**/**0.668**	**0.722**/**0.636**	**0.722**/**0.646**
es_n5+e100+lr0.001+hidden256+ly1+bd+dp_h0.0+dp_o0.8	0.706/0.640	0.706/0.620	0.706/0.620
es_n5+e100+lr0.001+hidden256+ly1+bd+dp_h0.0+dp_o0.9	0.714/0.644	0.714/0.624	0.714/0.628
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.0+dp_o0.0	0.594/0.536	0.594/0.454	0.594/0.460
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.0+dp_o0.4	0.612/0.538	0.612/0.498	0.612/0.504
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.0+dp_o0.8	**0.634**/**0.552**	**0.634**/**0.504**	**0.634**/**0.508**
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.2+dp_o0.0	0.594/**0.578**	0.594/0.458	0.594/0.452
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.2+dp_o0.4	0.616/0.544	0.616/0.482	0.616/0.482
es_n5+e100+lr0.001+hidden256+ly2+bd+bn+dp_h0.2+dp_o0.8	**0.632**/0.542	**0.632**/**0.504**	**0.632**/**0.504**
es_n5+e100+lr0.001+hidden256+ly4+dp_h0.0+dp_o0.0	0.576/0.438	0.576/0.414	0.576/0.386
es_n5+e100+lr0.001+hidden256+ly4+dp_h0.0+dp_o0.4	**0.638**/**0.504**	**0.638**/**0.504**	**0.638**/**0.484**
es_n5+e100+lr0.001+hidden256+ly4+dp_h0.0+dp_o0.8	0.604/0.474	0.604/0.458	0.604/0.430
es_n5+e100+lr0.001+hidden512+ly2+dp_h0.0+dp_o0.7	0.720/0.660	0.720/0.630	0.720/0.636
es_n5+e100+lr0.001+hidden512+ly2+dp_h0.0+dp_o0.9	**0.726**/**0.672**	**0.726**/**0.636**	**0.726**/**0.638**
es_n5+e100+lr0.001+hidden768+ly1+bd+dp_h0.0+dp_o0.0	0.702/0.650	0.702/0.606	0.702/0.606

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+e100+lr0.001+hidden768+ly1+bd+dp_h0.0+dp_o0.4	0.686/0.600	0.686/0.570	0.686/0.572
es_n5+e100+lr0.001+hidden768+ly1+bd+dp_h0.0+dp_o0.8	**0.718/0.660**	**0.718/0.632**	**0.718/0.636**
es_n5+e100+lr0.001+hidden768+ly4+bd+dp_h0.4+dp_o0.0	**0.670/0.596**	**0.670/0.588**	**0.670/0.582**
es_n5+e100+lr0.001+hidden768+ly4+bd+dp_h0.4+dp_o0.4	0.646/0.568	0.646/0.532	0.646/0.524
es_n5+e100+lr0.001+hidden768+ly4+dp_h0.4+dp_o0.0	**0.634/0.528**	**0.634/0.518**	**0.634/0.510**
es_n5+e100+lr0.001+hidden768+ly4+dp_h0.4+dp_o0.4	0.602/0.464	0.602/0.446	0.602/0.416
es_n5+e100+lr0.001+hidden768+ly4+dp_h0.4+dp_o0.8	0.616/0.486	0.616/0.484	0.616/0.470
es_n5+e100+lr0.001+hidden1024+ly1+bd+dp_h0.0+dp_o0.0	0.682/0.606	0.682/0.572	0.682/0.576
es_n5+e100+lr0.001+hidden1024+ly1+bd+dp_h0.0+dp_o0.4	0.680/0.608	0.680/0.582	0.680/0.578
es_n5+e100+lr0.001+hidden1024+ly1+bd+dp_h0.0+dp_o0.8	**0.710/0.658**	**0.710/0.606**	**0.710/0.616**
es_n5+e100+lr0.003+hidden512+ly1+dp_h0.0+dp_o0.7	0.714/0.652	0.714/0.628	0.714/0.632
es_n5+e100+lr0.003+hidden512+ly1+dp_h0.0+dp_o0.9	**0.740/0.694**	**0.740/0.672**	**0.740/0.670**
es_n5+e100+lr0.003+hidden512+ly2+bd+dp_h0.3+dp_o0.7	0.686/0.632	0.686/0.582	0.686/0.586
es_n5+e100+lr0.003+hidden512+ly2+bd+dp_h0.3+dp_o0.9	**0.702/0.642**	**0.702/0.616**	**0.702/0.616**

4.11. En İyi Performans Gösteren Modeller

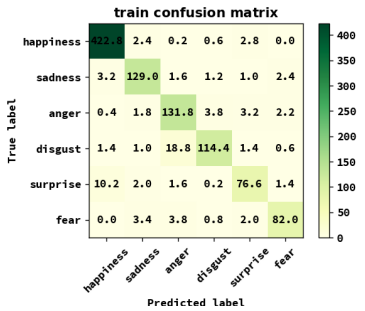
Yukarıdaki karşılaştırmalarımıza göre en iyi performans gösteren bulabildiğimiz 10 model aşağıda gösterilmiştir.

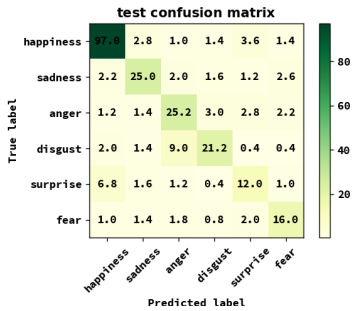
Çizelge 4.11. En iyi performans gösteren modeller

Model	Precision (micro/ macro)	Recall (micro/ macro)	f1-score (micro/ macro)
es_n5+b16+e1000+lr0.003+hidden512+ly1+dp_h0.0+dp_o0.9	0.760/0.708	0.760/0.694	0.760/0.696
es_n5+b16+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7	0.758/0.714	0.758/0.698	0.758/0.700
es_n5+b16+e1000+lr0.003+hidden256+ly1+bd+dp_h0.0+dp_o0.9	0.754/0.700	0.754/0.698	0.754/0.694
es_n5+b16+e1000+lr0.003+hidden300+ly2+dp_h0.3+dp_o0.8	0.746/0.688	0.746/0.664	0.746/0.664
es_n5+b16+e1000+lr0.001+hidden512+ly1+bd+dp_h0.0+dp_o0.9	0.744/0.684	0.744/0.676	0.744/0.672
es_n5+b16+e1000+lr0.001+hidden512+ly2+dp_h0.3+dp_o0.9	0.738/0.700	0.738/0.658	0.738/0.664
es_n5+b16+e1000+lr0.003+hidden512+ly2+dp_h0.5+dp_o0.7	0.736/0.696	0.736/0.672	0.736/0.668
es_n5+b16+e1000+lr0.003+hidden256+ly2+dp_h0.3+dp_o0.9	0.734/0.702	0.734/0.636	0.734/0.640
es_n5+b16+e1000+lr0.003+hidden512+ly2+dp_h0.3+dp_o0.9	0.722/0.682	0.722/0.626	0.722/0.634
es_n5+b16+e1000+lr0.003+hidden256+ly2+dp_h0.5+dp_o0.9	0.720/0.668	0.720/0.636	0.720/0.636

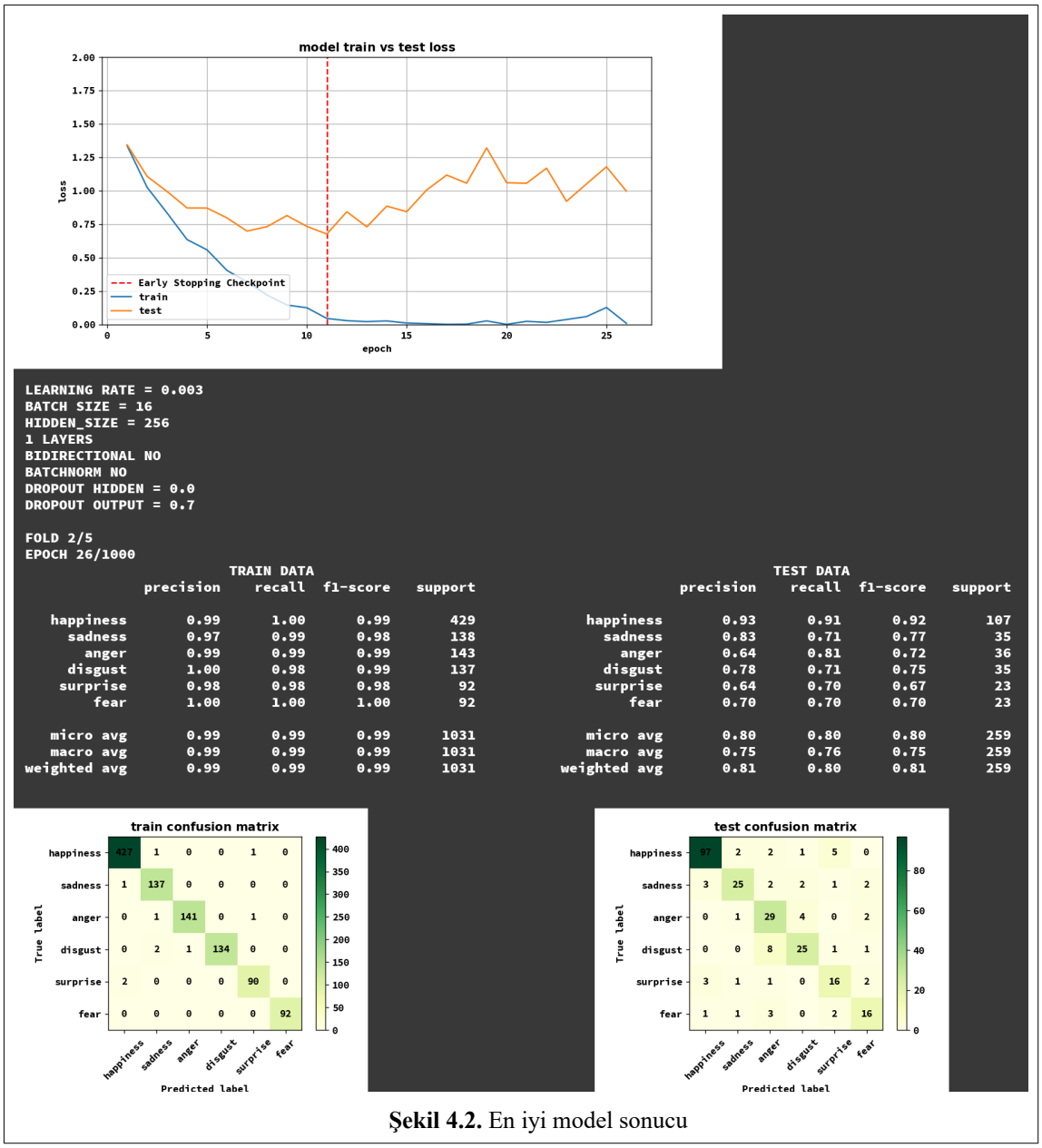
Burada en iyi modelin five fold crossover sonuçları aşağıdaki şekilde gösterilmiştir.

```
LEARNING RATE = 0.003
BATCH SIZE = 16
HIDDEN_SIZE = 512
1 LAYERS
BIDIRECTIONAL NO
BATCHNORM NO
DROPOUT HIDDEN = 0.0
DROPOUT OUTPUT = 0.9

5 FOLD AVERAGE
EPOCH 28
                    TRAIN DATA                                    TEST DATA
              precision recall f1-score  support                precision recall f1-score support
happiness         0.968  0.986    0.976  428.800 happiness         0.884  0.908    0.892 107.200
sadness           0.924  0.930    0.926  138.400 sadness           0.746  0.722    0.734  34.600
anger             0.846  0.922    0.878  143.200 anger             0.632  0.700    0.658  35.800
disgust           0.946  0.832    0.882  137.600 disgust           0.752  0.620    0.676  34.400
surprise          0.876  0.834    0.842   92.000 surprise          0.558  0.522    0.524  23.000
fear              0.932  0.892    0.910   92.000 fear              0.684  0.696    0.686  23.000

micro avg         0.928  0.928    0.928 1032.000 micro avg         0.760  0.760    0.760 258.000
macro avg         0.914  0.900    0.904 1032.000 macro avg         0.708  0.694    0.696 258.000
weighted avg      0.930  0.928    0.928 1032.000 weighted avg      0.766  0.760    0.756 258.000
```





Şekil 4.1. En iyi sonucun kategoriler için detaylı gösterimi



Şekil 4.2. En iyi model sonucu

4.12. Sonuçların Literatürdeki Diğer Çalışmalarla Karşılaştırılması

Bu başlık altında sonuçlarımız, literatürdeki diğer çalışmalarla karşılaştırılmıştır. Alm'ın tüm veri setinde (neutral kategorisi hariç) alınan accuracy, önceki araştırmalara göre daha iyidir, Agrawal'ın en iyi modeline göre %2'lik bir başarı artışı gözlemlenmiştir. (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using semantic and syntactic relations, sayfa 79. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.) Bu başarı artışı, bizim modelimizin bir kategori az tahmin etmesinden dolayı olabilir. Alm'ın 4 kategorili veri setinde en iyi modelimizin ortalama F-score'u 0.590'dır ve Agrawal'ın diğer sonuçları ile kıyaslanabilir bir sonuçtur fakat

herhangi bir başarı artışı sağlanamamıştır. (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using semantic and syntactic relations, sayfa 81-82. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.) ISEAR 7 kategorili veri setinde, *suç* ve *utanç* duyguları dışarıda bırakıldığında ortalama F-score 0.446'dır ve Agrawal'ın methodlarına göre %3'lük bir başarı artışı gözlemlenmiştir. (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using semantic and syntactic relations, sayfa 88. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.) Bu artış da, 2 kategorinin dışarıda tutulması ve bu veri setinin büyük kısmında yüklemi bulunmayan cümleler bulunmasından kaynaklanmış olabilir. Aman veri setinin, train ve test splitlerinde, test başarılarımız, Pragya Arora'nın ISEAR veri setindeki test split başarılarından %12'ye kadar daha başarılı (Pragya Arora et al. (2017). Emotion Analysis using Word Embedding and Neural Network. https://github.com/Harsh24893/EmotionRecognition/blob/master/report/main.pdf), Agrawal'ın Aman veri setindeki başarısından %15'e kadar daha başarılıdır. (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using semantic and syntactic relations, sayfa 94. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.)

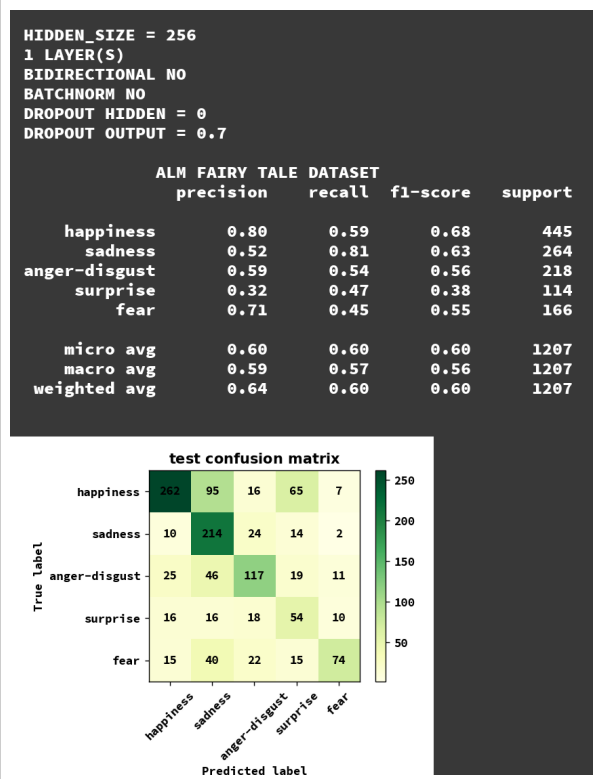
4.12.1. Alm Fairy Tale Dataset

Bu veri seti üzerinde kategorilendirme çalışmaları Alm (E. C. O. Alm, 2008), Ameeta Agrawal (2011) tarafından yapılmıştır ve bu sonuçlar bizim sonuçlarımızla karşılaştırarak aşağıda belirtilmiştir. Sonuçlar Alm'ın, tüm sonuçları en çok var olan kategoriye atama baseline'ını, Awrawal'ın ürettiği basit bir keyword tabanlı WordNet-Affect kullanan bir baseline'ı, Alm'ın gözetimsiz lextag methodunu ve Agrawal'ın Wikipedia corpusundan sonuçları ile karşılaştırılmıştır. Bizim modelimizde *neutral* kategorisi yoktur. (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using semantic and syntactic relations, sayfa 79. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.)

Çizelge 4.12. Alm fairy tale dataset sonuçları

Algorithm	**Happiness**	**Sadness**	**Anger-Disgust**	**Fear**	**Surprise**	**Neutral**
	Accuracy					
Majority class baseline	%31 (Happiness)					
Keyword baseline	%45					
Alm's unsupervised lextag method	%54-55					

Algorithm	Happiness	Sadness	Anger-Disgust	Fear	Surprise	Neutral
Agrawal's method (Without context (Wikipedia))	%56.31					
Agrawal's method (With context (Wikipedia))	57.25%					
Best model (second fold of es_n5+b16+e1000+lr0.003+hidden256+1 y1+dp_h0.0+dp_o0.7)	59.7%					



Şekil 4.3. Alm Fairy Tale Dataset Sonucu

4.12.2. Alm Fairy Tale 4 Kategorili Dataset

Aşağıdaki tabloda sadece 4 kategori üzerinde bazıları (Kim et al., 2010)'dan alınmış sonuçlarla ve Agrawal'ın sonuçlarıyla modelimiz karşılaştırılmıştır. (Kim et al. (2010). Evaluation of unsupervised emotion models to textual affect recognition, sayfa 62-70. Proceedings of the NAACL HLT 2010 Workshopon Computational Approaches to Analysis and Generation of Emotion in Text, Los Angeles, California.) (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using semantic and syntactic relations, sayfa 81-82. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.)

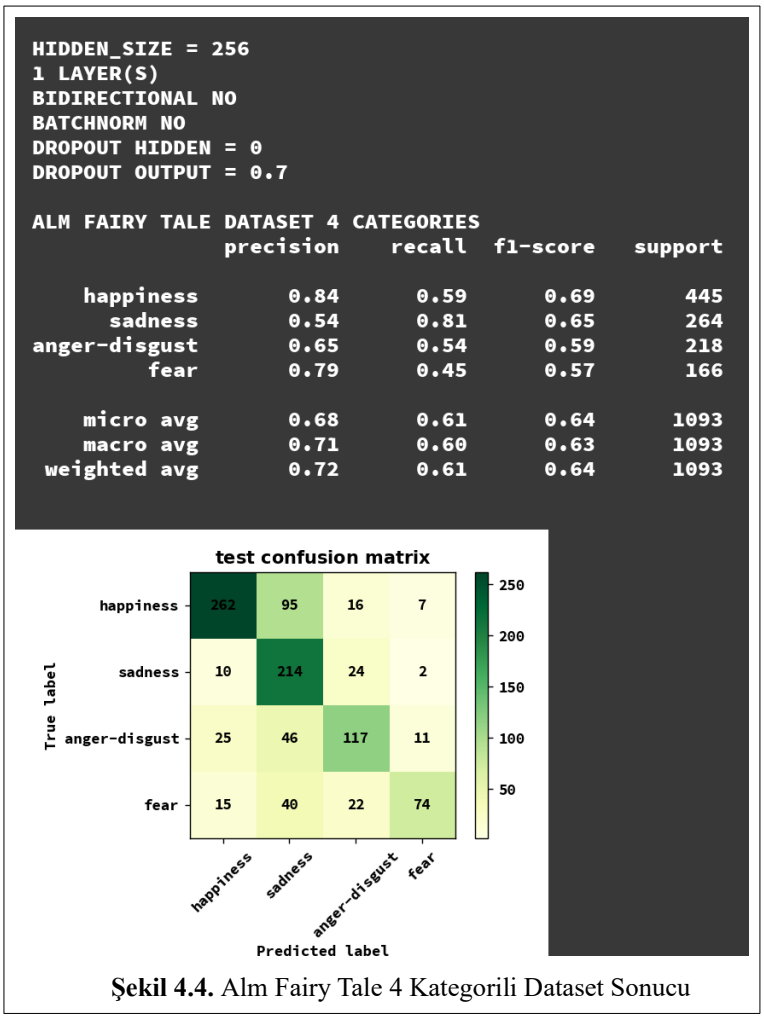
Çizelge 4.13. Alm fairy tale 4 kategorili dataset sonuçları

Algorithm		**Measure**	**Happiness**	**Sadness**	**Anger-Disgust**	**Fear**
Keyword baseline		ρ	0.773	0.667	0.940	0.867
		π	0.481	0.303	0.142	0.157
		F_1	0.593	0.417	0.247	0.265
LSA		ρ	0.847	0.704	0.386	0.710
		π	0.637	0.589	0.749	0.583
		F_1	0.727	0.642	0.510	0.640
PLSA		ρ	0.555	0.333	0.239	0.000
		π	0.358	0.414	0.455	0.000
		F_1	0.436	0.370	0.313	0.000
NMF		ρ	0.802	0.708	0.773	0.704
		π	0.761	0.821	0.560	0.781
		F_1	0.781	0.760	0.650	0.741
DIM		ρ	0.661	0.408	0.604	0.444
		π	0.979	0.169	0.290	0.179
		F_1	0.789	0.240	0.392	0.255
Agrawal's method without context	Wikipedia	ρ	0.758	0.641	0.627	0.465
		π	0.703	0.466	0.486	0.765
		F_1	0.730	0.539	0.548	0.579
	Gutenberg	ρ	0.733	0.722	0.836	0.390
		π	0.703	0.443	0.234	0.807
		F_1	0.718	0.549	0.366	0.525
	Wiki-Guten	ρ	0.785	0.742	0.798	0.433
		π	0.688	0.534	0.417	0.837
		F_1	0.733	0.621	0.548	0.571
Agrawal's method with context	Wikipedia	ρ	0.756	0.629	0.644	0.466
		π	0.710	0.462	0.523	0.777
		F_1	0.732	0.533	0.577	0.582
	Gutenberg	ρ	0.736	0.694	0.828	0.398
		π	0.708	0.447	0.243	0.807
		F_1	0.722	0.544	0.376	0.533
	Wiki-Guten	ρ	0.786	0.760	0.786	0.431
		π	0.694	0.527	0.422	0.849
		F_1	0.737	0.622	0.549	0.572
Best model (second fold of es_n5+b16+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7)		ρ	0.84	0.54	0.65	0.79
		π	0.59	0.81	0.54	0.45
		F_1	0.69	0.65	0.45	0.57

ρ = Precision, π = Recall, F_1 = F-score

Çizelge 4.14. Alm fairy tale 4 kategorili Dataset'de ortalama F-scorelar

Algorithm		Average F-score
LSA		0.629
PLSA		0.279
NMF		0.733
DIM		0.419
Agrawal's method without context	Wikipedia	0.599
	Gutenberg	0.540
	Wiki-Guten	0.618
Agrawal's method with context	Wikipedia	0.606
	Gutenberg	0.544
	Wiki-Guten	0.620
Best model (second fold of es_n5+b16+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7)		0.590



Şekil 4.4. Alm Fairy Tale 4 Kategorili Dataset Sonucu

4.12.3. ISEAR 7 Kategorili Dataset

Tüm 7 kategoride de Agrawal'ın sonuçları alınmış, 5 kategorili ISEAR sonuçlarımızla karşılaştırılmıştır. (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using

semantic and syntactic relations, sayfa 88. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.) ISEAR verisi üzerinde bir çalışma daha (Bincy Thomas et al. (2014). Multiclass Emotion Extraction from Sentences. International Journal of Scientific & Engineering Research, Volume 5, Issue 2.)'da görülebilir. Bizim çalışmamıza çok benzer LSTM ve CNN kullanan bir çalışma (eğitim ve test için ISEAR verisi 80:20 olarak bölünmüş) sonuçları (Pragya Arora et al. (2017). Emotion Analysis using Word Embedding and Neural Network. https://github.com/Harsh24893/EmotionRecognition/blob/master/report/main.pdf)'de görülebilir. Yorum yapılacak olursa, Aman'ın verisinde test sonuçlarında bizim modelimiz, %75 test accuracy elde edebilen duruma gelmişken, (Pragya Arora et al., 2017) çalışmasında LSTM modeli için en iyi test sonucu %63'tür. LSTM modellerinde 128 memory unit (hidden) bulunmaktadır, inputa ve LSTM memory unitleri arasına 0.6 dropout uygulanmış, 128 batch büyüklüğü ile eğitim sağlanmış, crossentropy loss ve ADAM optimizer kullanılmıştır. CNN ve LSTM modellerinde, LSTM modelleri daha iyi sonuçlar vermiştir.

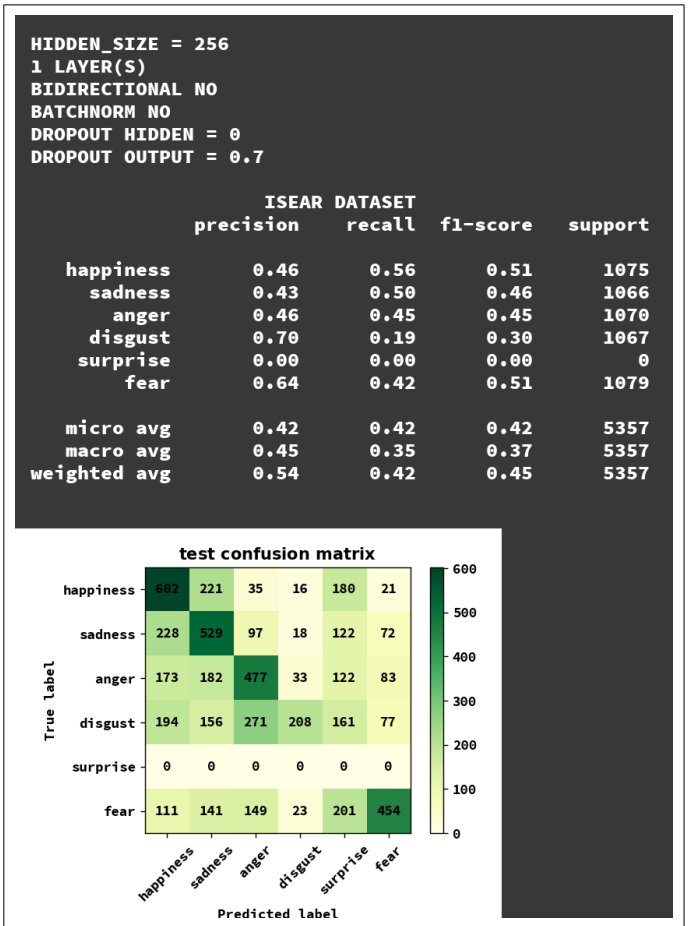
Çizelge 4.15. ISEAR 7 (5) kategorili dataset sonuçları

Algorithm	**Measure**	**Joy**	**Sadness**	**Anger**	**Fear**	**Disgust**	**Shame**	**Guilt**
Agrawal's method Wikipedia	ρ	0.492	0.706	0.347	0.410	0.828	0.378	0.364
	π	0.538	0.276	0.510	0.698	0.290	0.423	0.316
	F_1	0.514	0.396	0.413	0.517	0.430	0.400	0.338
Agrawal's method Gutenberg	ρ	0.428	0.702	0.403	0.309	0.634	0.575	0.416
	π	0.599	0.151	0.429	0.759	0.328	0.303	0.340
	F_1	0.500	0.248	0.415	0.439	0.432	0.397	0.374
Agrawal's method Wiki-Guten	ρ	0.504	0.872	0.323	0.366	0.656	0.542	0.338
	π	0.496	0.174	0.575	0.711	0.367	0.316	0.321
	F_1	0.500	0.290	0.414	0.483	0.470	0.400	0.329
Best model (second fold of es_n5+b16+e1000+lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7)	ρ	0.46	0.43	0.46	0.64	0.70		
	π	0.56	0.50	0.45	0.42	0.19		
	F_1	0.51	0.46	0.45	0.51	0.30		

ρ = Precision, π = Recall, F_1 = F-score

Çizelge 4.16. ISEAR Dataset'inde ortalama F-scorelar

Algorithm	Average F-score
Agrawal's Wikipedia	0.430
Agrawal's Gutenberg	0.401
Agrawal's Wiki-Guten	0.412
Best model (second fold of es_n5+b16+e1000 +lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7)	0.446



Şekil 4.5. ISEAR 7 (5) Kategorili Dataset Sonucu

4.12.4. Emotion Annotated Dataset (Aman'ın Veri Seti)

Bu başlık altında Aman'ın kullandığı methodlar ve Agrawal'ın kullandığı methodlar ile, bizim aynı veri setinin test kısmında aldığımız en iyi sonuç karşılaştırılmıştır. Sonuçlar (Ameeta Agrawal. (2011). Unsupervised emotion detection from text using semantic and syntactic relations, sayfa 94. 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.)'den alınmıştır.

Çizelge 4.17. Aman Dataset'inde ortalama F-scorelar

Algorithm		Average F-score
Keyword baseline		0.331
Aman's supervised ML with unigrams		0.530
Aman's supervised ML with unigrams, Roget's Thesaurus and WordNet-Affect features		0.586
Inkpen's hierarchical approach		0.508
Agrawal's method without context	Wikipedia	0.527
	Gutenberg	0.540
	Wiki-Guten	0.478
Agrawal's method with context	Wikipedia	0.535
	Gutenberg	0.546
	Wiki-Guten	0.516
Best model (second fold of es_n5+b16+e1000 +lr0.003+hidden256+ly1+dp_h0.0+dp_o0.7)		0.696

4.13. Gelecek Çalışmalar

Cümlelerden duygu çıkarımı çalışılmıştır. Gelecek çalışmalarda, tüm veri setleri toplamı üzerinde eğitim yapılması ile daha iyi sonuçlar elde etmek, veri temizliği için otomasyon bir sistem üretmek (stop wordlerin kaldırılması gibi), veri girdileri için, POS tagger gibi ekstra NLP araçları kullanıp, bu taglerin de eğitim için modele verilmesi yapılıp daha robust bir sistem oluşturulabilir.

KAYNAKÇA

S. Hochreiter & J. Schmidhuber, 1997, Long short-term memory, Neural computation, 9(8):1735–1780.

S. Aman & S. Szpakowicz, 2007, Identifying Expressions of Emotion in Text, V. Matousek, P. Mautner (eds.): Proc 10th International Conf. on Text, Speech and Dialogue TSD 2007, Plzeň, Czech Republic, Lecture Notes in Computer Science 4629, Springer, 196-205.

S. Aman, 2007, Recognizing Emotions in Text, Master of Computer Science, University of Ottawa.

J. Yang, J. Leskovec, 2011, Patterns of Temporal Variation in Online Media, ACM International Conference on Web Search and Data Mining (WSDM '11).

Ebba Cecilia Ovesdotter Alm, 2008, Affect in text and speech, PhD Dissertation, Urbana, IL: University of Illinois at Urbana-Champaign.

Ameeta Agrawal, 2011, Unsupervised emotion detection from text using semantic and syntactic relations, 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 346-353. IEEE Computer Society, December.

Jeffrey Pennington et al., 2014, GloVe: Global Vectors for Word Representation, Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014) 12.

Pragya Arora et al., 2017, Emotion Analysis using Word Embedding and Neural Network, https://github.com/Harsh24893/EmotionRecognition/blob/master/report/main.pdf

Kim, S. M., Valitutti, A., & Calvo, R. A., 2010, Evaluation of unsupervised emotion models to textual affect recognition, Proceedings of the NAACL HLT 2010 Workshopon Computational Approaches to Analysis and Generation of Emotion in Text, Los Angeles, California.

Bincy Thomas, Vinod P, Dhanya K. A., 2014, Multiclass Emotion Extraction from Sentences, International Journal of Scientific & Engineering Research, Volume 5, Issue 2.

WEB_1: , Stanford Tokenizer, 2018, https://nlp.stanford.edu/software/tokenizer.shtml

WEB_2: Jeffrey Pennington et al., GloVe, 2014, https://nlp.stanford.edu/projects/glove/

WEB_3: P. Bojanowski et al., fastText, 2019, https://fasttext.cc/

WEB_4: , PyTorch, 2019, https://pytorch.org/

WEB_5: , PyTorch Graph Örneği, 2019, https://github.com/pytorch/pytorch

WEB_6: , scikit-learn, 2019, https://scikit-learn.org/stable/

WEB_7: , Deep Learning, 2019, https://en.wikipedia.org/wiki/Deep_learning

WEB_8: , Feedforward neural network, , https://en.wikipedia.org/wiki/Feedforward_neural_network

WEB_9: Denny Britz, Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients, 2015, http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/
WEB_10: Christopher Olah, Understanding LSTM Networks, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/
WEB_11: , Batch normalization, 2019, https://en.wikipedia.org/wiki/Batch_normalization
WEB_12: Claudia Zhu, The “Less is More” of Machine Learning, 2019, https://towardsdatascience.com/the-less-is-more-of-machine-learning-1f571c0d4481
WEB_13: 2019, Early stopping, , https://en.wikipedia.org/wiki/Early_stopping
WEB_14: , 476 million Twitter tweets, 2011, https://snap.stanford.edu/data/twitter7.html
WEB_15: Bjarte Mehus Sunde, Early Stopping for PyTorch, 2019, https://github.com/Bjarten/early-stopping-pytorch