T.C. İSTANBUL TİCARET ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE SOSYAL MEDYA VERİLERİNDEN DUYGU ANALİZİ

Mesut PEK

Danışman
Dr. Öğr. Üyesi Metin TURAN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
İSTANBUL- 2019

KABUL VE ONAY SAYFASI

Mesut PEK tarafından hazırlanan **" Makine Öğrenmesi Yöntemleri ile Sosyal Medya Verilerinden Duygu Analizi "** adlı tez çalışması 27/06/2019 tarihinde aşağıdaki jüri üyeleri önünde başarı ile savunularak, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Danışman	**Dr. Öğr. Üyesi Metin TURAN** İstanbul Ticaret Üniversitesi
Jüri Üyesi	**Prof. Dr. Selim AKYOKUŞ** İstanbul Medipol Üniversitesi
Jüri Üyesi	**Dr. Öğr. Üyesi Mustafa Alper ÖZPINAR** İstanbul Ticaret Üniversitesi

Onay Tarihi : 09.07.2019

Prof. Dr. Necip ŞİMŞEK
Enstitü Müdürü

AKADEMİK VE ETİK KURALLARA UYGUNLUK BEYANI

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

Beyan ederim.

27/06/2019

Mesut PEK

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE SOSYAL MEDYA VERİLERİNDEN DUYGU ANALİZİ

Mesut PEK

İstanbul Ticaret Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Metin TURAN

2019, 68 sayfa

Günümüzde teknolojinin gelişmesiyle, kişilerin sosyal medyada bulunan yorumları ve paylaşımları artmıştır. Önceki yıllarda internet bu kadar yaygın olmadığı için raporlar daha kolay oluşturulmaktaydı. Şimdi ise internetin ve teknolojin gelişmesiyle bu raporlamalar daha zorlaştı. Teknolojinin gelişmesiyle milyonlarca kişinin üye olduğu ve düşüncelerini paylaştıkları platformlardaki verilerin incelenmesi ve işlenmesi çok daha kolay olmaktadır. Bu alanda yapılan önemli çalışmalar arasında duygu analizi de yer almaktadır. Bu çalışmada makine öğrenmesi kullanılarak duygu analizi yapılmış, diğer yaygın uygulamaların yanı sıra özellik seçimi konusunda iyileştirilmeler yapılarak başarım oranı artırılmaya çalışılmıştır. Naive Bayes sınıflandırıcı eğitim setinin dışında herhangi bir veri setinde sınanarak, %80 oranında başarı elde edilmiştir. Derin öğrenme yöntemleri arasında bulunan konvolüsyonel sinir ağları kullanılarak aynı veri setleri ile çalışmıştır. Konvolüsyonel sinir ağlarında %97 başarı elde edilmiştir.

Anahtar Kelimeler: Duygu Analizi, Konvolüsyon Sinir Ağları, Makine öğrenmesi, Naive Bayes Algoritması.

ABSTRACT

M.Sc. Thesis

EMOTION ANALYSIS OF SOCIAL MEDIA DATA USING MACHINE LEARNING TECHNIQUES

Mesut PEK

İstanbul Commerce University
Graduate School of Applied and Natural Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Metin TURAN
2019, 68 pages

Today, with the improvements in technology, comments and sharing of people increased. In the past, before widespread internet usage, creating reports were easier. But now with advancements in internet and technology reporting got harder. Analyzing and processing data from this sharing platforms where millions of people are registered and expressing their opinions is easier. Important works on this field includes emotion analysis. In this study emotion analysis is done using machine learning and compared to other applications tried to improve success rate with improvements on feature selection. While using any other set than Naïve Bayes classification learning set this study has achieved %80 success rate. A convolutional neural network with deep learning options achieved 97% success in convulsive neural networks.

Keywords: Sentiment Analysis, Convolution Neural Networks, Machine Learning, Naive Bayes Algorithm.

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, değerli bilgilerini benimle paylaşan, kendisine ne zaman danışsam bana kıymetli zamanını ayırıp sabırla ve büyük bir ilgiyle bana faydalı olabilmek için elinden gelenden fazlasını sunan her sorun yaşadığımda yanına çekinmeden gidebildiğim, güler yüzünü ve samimiyetini benden esirgemeyen ve gelecekteki mesleki hayatımda da bana verdiği değerli bilgilerden faydalanacağımı düşündüğüm kıymetli ve danışman hoca statüsünü hakkıyla yerine getiren Dr. Öğr. üyesi Metin TURAN'a teşekkürü bir borç biliyor ve şükranlarımı sunuyorum.

Tezimin her aşamasında beni yalnız bırakmayan aileme sonsuz sevgi ve saygılarımı sunarım.

Mesut PEK
İSTANBUL, 2019

ŞEKİLLER

Sayfa

TABLOLAR

SİMGELER VE KISALTMALAR

ML	Makine öğrenmesi (Machine Learning)
VM	Veri Madenciliği
RF	Rasgele Orman (Random Forest)
UDF	Kullanıcı Tanımlı İşlevler (User defined function)
RNN	Tekrarlanan Yapay Sinir Ağları (Recurrent Neural Network)
ICEWS	Entegre Kriz Erken Uyarı Sistemi (Integrated Crisis Early Warning System)
GDELT	Global Olaylar, Dil ve Ton Veri Tabanı (Global Data on Events, Languages and Tone)
UCI	Makine Öğrenmesi Deposu (Irvine Machine Learning)
TF	Terim Frekansı (Term Frequency)
IDF	Ters Metin Frekansı (Inverse Document Frequency)
KNN	En Yakın Komşu (K nearest neighborhood)
SVM	Destek Vektör Makinesi (Support Vector Machines)
NB	Naive Bayes
CNN	Konvolüsyonel Sinir Ağları (Convolution Neural Network)

1. GİRİŞ

Günümüzde teknolojinin gelişmesiyle, kişilerin sosyal medyada bulunan yorumları ve paylaşımları artmıştır. Önceki yıllarda internet bu kadar gelişmiş olmadığı için raporlamalar daha kolaydı. Yeni iletişim teknolojilerinden biri olan internette Web 2.0'ın gelişimi ile birlikte ortaya çıkan kullanıcı tabanlı "sosyal medya" bireysel kullanıcılara anlık bildirim yapma, yorumda bulunma, şikayet etme gibi pek çok geri bildirim olanağı sunmaktadır. Birçok sosyal medya platformu; Facebook, Twitter, Linkedin, Instagram, Blogger, haber siteleri ve alışveriş siteleri anlık mesajlaşmaları veya yorumları barındırabilmektedir. Verilerin sürekli olarak artması ve büyük miktarlarda sürekli olarak depolanması çok daha kolaylaşmıştır. Bu nedenle internet büyük veri kaynağıdır ve kişiler istedikleri zaman bu verilere erişim sağlayabilmektedirler. Platformlar verileri geliştiriciler için sınırlı olarak açsalarda, istendiğinde verileri parasıyla da satabilmektedirler. Böylece milyonlarca kişinin üye olduğu ve düşüncelerini paylaştıkları platformlardaki verilerin incelenmesi, işlenmesi çok daha kolay olmakta ve duygu analizi yapılabilmektedir.

Duygu analizi ile kişilerin ürünler veya herhangi bir konu üzerinde ne düşündükleri olumlu, olumsuz ve nötr olarak gruplandırılabilir. Kişiler, siyasi partiler veya firmalar bu yönde araştırmalar yaparak, önceden tahminler çıkarmaya çalışmaktadırlar. Örneğin, seçim zamanı atılan tweetlerden siyasi liderler hakkındaki görüşlerin ne kadar olumlu veya olumsuz olduğu yönünde görüş oluşturmak veya hangisinin kazanabileceğini tahmin etmek mümkün olabilir. Firmalar ürünleri hakkında olumlu ve olumsuz görüşlerin istatistiklerini alarak, olumsuz görüşleri ürün geliştirme amacıyla kullanabilmektedirler. Twitter platformu ile kullanıcılar (kişiler) 280 karakterlik (1 Kasım 2017 tarihinden önce 140 karakter) yorumlar paylaşmaktadırlar. Artık bu yorumları anlık olarak, kişilerin düşüncelerini olumlu, olumsuz veya nötr olarak gruplayarak çok hızlı olarak raporlama fimkânımız bulunmaktadır.

İnsan gücü ile gruplama yapmaya çalıştığımızda, milyonlarca satırlık verinin kısa sürede işlenmesi mümkün olmayacaktır ve muhtemelen yapılan

gruplamalarda hatalar oluşacaktır. Medya ajansları personel istihdam etmek isteseler bile edemeyeceklerdir. Finansal olarak personel giderleri çok yüksek çıkacaktır. Halbu ki, makine öğrenmesi yöntemleriyle milyonlarca satır verinin internetten kısa bir süre içinde taranıp, minimum hata ile gruplanması günümüz koşullarında artık mümkündür. Makine öğrenmesi ile kısa sürede raporlanacak bilginin geçerliliği ve doğruluğu uzun süre kalıcı olacaktır.

Bu çalışmada, sosyal medya verileri üzerinde duygu analizi yapmak üzere bir sınıflandırıcı geliştirilmiş, sınıflandırma amacı ile Naive Bayes algoritması ve konvolüsyonel sinir ağları kullanılmıştır.

2. LİTERATÜR ÖZETİ

Yeni iletişim teknolojilerinden biri olan internette Web 2.0'ın gelişimi ile birlikte ortaya çıkan kullanıcı tabanlı "sosyal medya" bireysel kullanıcılara anlık bildirim yapma, yorumda bulunma, şikâyet etme gibi pek çok geri bildirim olanağı sunmaktadır. Facebook, Twitter, Linkedin, Instagram, gibi micro blogların yanı sıra, web tabanlı haber siteleri ve hatta alışveriş siteleri dahil olmak üzere birçok platform, anlık mesaj ve yorum ile kullanıcıyı aktif kılmaktadır. Çok yönlü ileti paylaşımına ve interaktif etkileşime olanak sağlayan kullanıcı tabanlı bu platformlarda, kullanıcının oluşturduğu iletiler ile hızlı bir veri akışı olmakta ve bu veriler birer analiz nesnesine dönüşmektedir. Platformlar depolanan verileri analizini kendi bünyesinde veya farklı firmalar aracılığıyla analiz edebilmektedir. Analizlerde makine öğrenmesi aktif olarak kullanılmaktadır. Makine öğrenmesi son yıllarda gpu ile daha hızlı işlem yapılmasının faydası ile güncel konular arasında yer almaktadır ve birçok alanda kullanılmaktadır. Igor Aizenberg vd. (2000), yapay sinir ağları bağlamında ilk kez derin öğrenme (deep learning) kavramını tanıtmıştır. Hinton vd. (2006) yaptığı bir çalışmada çok katmanlı ileri beslemeli bir sinir ağının her iterasyonda bir katmanı iyi şekilde eğitilebildiğini, sonrasında denetimli bir geri yayılım yönetemiyle ayarlanabileceğini göstermiştir. Krizhevsky vd. (2012) gpu destekli olarak, ezberlemeyi azatmak için "dropout" adında bir normalleştirme yöntemini uygulamıştır. Ciresan vd. (2012) yaptığı çalışmada derin öğrenme yöntemlerinden Konvolüysyonel Sinir Ağları ile MSIT veri kümesi ile %2 hata oranıyla yüksek bir başarı sağlamıştır. Konvolüysyonel Sinir Ağları ile sonucu en hızlı bulduklarını aktarmıştır. Hinton vd. (2014), Dropout yönteminin başarısını, ILSVRC-2012 Imagenet yarışmasında yüksek sonuçları göstererek ispatlamıştır. Akgül vd. (2016), Twitterdan Türkçe tweet mesajları üzerinde çalışma yapmışlardır. Olumlu, olumsuz ve nötr olarak 3 adet sözlük oluşturmuştur. Sözlükte Türkçe kelimeleri elle etiketlemişlerdir. Daha sonrasında bir model oluşturarak sosyal medya verisi olarak tweetlerden duygu analizi yapmıştır ve sonuçlarını paylaşmıştır.

Onan ve Kurukoglu (2016), makine öğrenmesine dayalı yöntemleri; gözetimli öğrenme, yarı gözetimli öğrenme ve gözetimsiz öğrenme olarak üç grupta tanımlamaktadır. Sözlüğe dayalı görüş madenciliği yöntemleri, yüksek ölçeklenebilirliğinden dolayı sıklıkla uygulanmaktadır. Makine öğrenmesi sınıflandırma problemi için içerisinde birçok algoritma barındırmaktadır. Makine öğrenmesi (ML) algoritmalarının basit yapısı ve farklı alanlardaki görüş sınıflandırmalara kısmen kolay uygulanabilmektedir. ML yapısından dolayı görüş madenciliği için önemli bir araştırma alanı durumuna gelmiştir. Çoklu örnek öğrenme yaklaşımı kullanan Wang vd. (2016), cümle düzeyinde etiketler elde etmek zor olsa da belge düzeyinde toplanması nispeten kolay olmasından yararlanmaktadır. Modelde, Latin Amerika ülkesindeki sivil toplumla ilgili olaylar hakkında (İspanyolca metinden) haber makalelerini tespit etme ve bu olaylarla ilgili kilit cümleleri belirleme özelliğini değerlendirmektedir. Açıklamalı cümle etiketleri olmadan eğitilmiş olan model, olay tespit ve cümle tanımlaması için seçilmiş en son modellerle rekabet edebilecek performans sağlamaktadır. Ayrıca, deneysel sonuçlar, çıkarılan özelliklerle ilgili cümlelerin bilgilendirici olduğunu ve makale özetleme, görselleştirme ve olay kodlaması gibi çeşitli alt uygulamaların geliştirildiğini göstermektedir. Kravvaris ve Kermanidis (2016), kişilerin eğitimleri ve videoları seçerken, incelerken daha iyi bir bilgiye sahip olabilmeleri için yapılan yorumları olumlu ve olumsuz olarak gruplandırmaktadır. Böylece eğitim ve videoları arayan kişiler için daha net bir bilgi sağlamaktadır. Singh vd. (2017), Twitter sosyal medya platformunda Donald Trump ile ilgili tweetleri toplamış ve weka adlı programda bir model oluşturmuştur. Weka'da oluşturulan modelde makine öğrenmesi algoritmalarında Destek Vektör Makinaları algoritmalarını kullanmışlardır. Yapılan çalışmada algoritma başarılı bir sonuç vermiş ve Donald Trump'ın sonuçları kazandığını tahmin etmiştir. Kocich (2017), Fransızca, İspanyolca ve Almanca metinleri işleyebilen az sayıda açık kaynak olduğuna dikkat çekmiştir. Farklı dillerdeki verilerin duygu analizini yapmak için bir çalışma önermiştir. Yaptığı çalışmada farklı dillerdeki tweet verilerini alarak, eş zamanlı olarak Google api tarafından İngilizceye çevirmiştir. Çevrilen verileri daha sonra olumlu ve olumsuz olarak gruplamıştır.

Kassraie vd. (2017), 2016 ABD Seçimlerinde 370.000'den fazla tweet içeren ve Google arama verileri ile büyük bir veri kümesi oluşturmuştur. Seçim sonucunu tahmin etmek için bir Gauss işlem regresyon modeli kullanılmıştır. Diğer yaklaşımlarda da görüldüğü gibi, doğrudan seçmen galibini öngörmek yerine, adayların oylarını tahmin etme fikri ortaya konulmuştur. Trump taraftarlarının başka yerde bildirilen davranışlarındaki yüksek varyans, oy tahmininde yüksek hata oranı olarak yansımıştır. Wang'a göre (2017), haber makalelerinden olaylarla ilgili bilgilerin elde edilmesinde üç duruma odaklanılmaktadır. İlk etapta, mevcut büyük ölçekli sistemlerin kodlamasımda performans ve zorlukları kapsamlı bir şekilde analiz edilmektedir. İkinci etapta, haber makalelerinden olay tespit ve kritik bilgi çıkarmaları incelenmektedir. Üçüncü etapta ise e-metinler olayların ve argümanlarının metinlerden çıkarılmasını amaçlayan olay-kodlama üzerinde incelemeler yapılmaktadır. Siyaset bilimi alanında iki büyük ölçekli olay çıkarma sistemi (ICEWS ve GDELT) araştırılmaktadır. Çok Boyutlu Konvolüsyel Sinir Ağı (MI-CNN) modeli önerilmektedir. Modeli değerlendirmek için veri kümesinde bir dizi deney yapılmıştır. Modelde anlamlı sonuçlar elde edilmiştir. Çok örnekli öğrenme modeli, sistemin kuruluş seviyesindeki temel etiketleri gerektirmediğini ve olay çıkarımı için uzaktan denetim ile birlikte çalışmayı mükemmel hale getirdiğini göstermektedir. Larsson ve Nılsson (2017), cümleler için bir konvolüsyonel nöral ağ ve önceden eğitilmiş kelime vektörleri kullanmaktadır. Cümleler, karşıt duygularla cümlenin yapısına benzeyecek şekilde bir test istatistiğinin optimizasyonu yoluyla geçilmektedir. Sonunda vektör yapısını çözmek ve yeni cümleler üretmek için tekrarlayan bir sinir ağı (RNN) kullanılmaktadır. Modeldeki kodlayıcı, duygu sınıflandırma görevinde %80 doğruluk elde etmekte ve 300 boyutta cümle temsili üretmektedir. PCA kullanarak bu gösterimlerin görselleştirilmesi hem duygu hem de farklı konulara göre kümelenmeyi göstermekte, bu da temsillerin hem duygu hem de metinsel içerik hakkında bilgi sahibi olduğuna işaret etmektedir. RNN dil modelini kullanarak çaprazlanmış özellik vektörlerinin çözümlenmesi, çoğu durumda, orijinal cümleyle karşılaştırıldığında duyguların değiştiği anlaşılır cümleler oluşturmaktadır. Şeker ve Yeşilyurt (2017), Knime adlı yazılım ile Twitterdan verileri çekmiş, ardından veri seti üzerinde temizlik işlemi yaptıktan sonra csv formatında

veriyi kullanılmak üzere kaydetmiştir. Rapid Miner adlı veri madenciliği programında makine öğrenmesi kullanılarak birçok algoritma üzerinde çalışmalar yapılmıştır. %52 ile en başarılı algoritma Naive Bayes çıkmıştır. Çalışmada KNN, Destek vektör makinaları, karar ağaları algoritmaları testlerde donanım yetersizliğinden dolayı kullanılamamıştır. Onan (2017), yaptığı çalışmada en iyi tahmini yapan algoritma olarak Naive Bayes'i belirlemiştir. Türkçe tweetler üzerinde işlem yaptığı için zemberek kullanmıştır. Çalışmada Destek Vektör Makinaları, Lojistik Regresyon algoritmalarını da kullanmıştır. Naive Bayes algoritmasını; basit yapısı, hesaplama etkinliği ve yüksek doğru sınıflandırma başarısından dolayı, Destek Vektör Makinalarını ise hem doğrusal hem de doğrusal olmayan verileri sınıflandırabileceği için tercih ettiğini belirtmiştir. Çoban ve Özyer (2017), öz nitelikler çıkartılırken kelime torbası ve n-gram olmak üzere 2 farklı model üzerinde çalışma yapmıştır. Çalışmasında 2 farklı Türkçe ve İngilizce veri seti kullanmışlardır. Twitter mesajlarını duygu sınıflandırmasında konu tabanlı Gizli Dirichlet ataması kullanmıştır. Sınıflandırmada ise Destek Vektör Makinesi algoritması kullanmıştır. Alkowaileet vd. (2018) yaptığı çalışmada, açık kaynaklı bir büyük veri yönetim sistemi olan Apache AsterixDB'ye dikkat çekmektedir. Çalışmada, AsterixDB Big Data analitiğinde makine öğrenimi algoritmalarını kullanmadaki yükü azaltmaya nasıl yardımcı olabileceği açıklanmaktadır. AsterixDB'nin kullanıcı tanımlı işlevler (UDF'ler) için yerleşik desteği, veri toplama kanalları ve sorgularındaki UDF'lerin kullanılabilirliğini ve makine öğrenme platformu ile dizüstü bilgisayarlardaki işlemlerinin çalışmasını aktarmaktadır. Uçtan uca analitik veri akışlarını daha kolay bir şekilde oluşturulması ve yönetilmesi de çalışmada yer almaktadır. Chen (2018)', sinir ağları eğitimi sırasında çok modlu hata dağılımına göre gürültülü verilerin kaldırılması için bir yöntem açıklamaktadır. Bu yöntem, mevcut çalışmaları çift modlu çıkarmadan çok modlu çıkarmaya kadar genişletmektedir. Yüksek hata oranlarının içerisinde bulunan verileri tanımlamakta ve bunları tekrarlayan eğitim programlarından kaldırmaktadır. Metodun etkinliğini, tek katlamalı sinir ağları ve en küçük kareler metodu ile karşılaştırarak doğruluğunu kontrol etmektedir. Ayrıca, metod ile duygu analizi problemi çalışması arasında performans karşılaştırmaları sunmaktadır. Wang vd. (2018), metin sınıflandırmasını

geliştirmek için yardımcı temel özellik seçme yöntemini incelemiştir. Mevcut özellik seçim yöntemlerinden farklı olarak önerilen yöntem, yazarların metin belgesindeki konuları ifade etmek için sözcükleri nasıl kullandıklarına bakılarak anlamsal sözcükleri seçmektedir. Altı makine öğrenme deposundaki veri setine dayanan deneysel testler, önerilen yöntemin metin sınıflandırmasında diğer özellik seçme yöntemlerinden daha iyi performans gösterdiği tespit edilmiştir. Moussa vd. (2018), geleneksel sözlük-temelli yaklaşımlar ölçeklenebilirlik yönünden sıkıntılarının olduğu ve genellikle manuel olarak oluşturulan güvenilmez duygu ifadeleriyle sınırlı olduğunu ifade etmektedir. Bu nedenle, ölçeklenebilirlik, sınırlılık ve mevcut modelleri doğrulukla ilgili duygu analizi için genel, genişletilebilir, alandan bağımsız bir lexikon tabanlı model önermektedir. Önerilen model, farklı sözcüklerin listesini temel alarak duygu yoğunluğunu özetlemekte, nötr ve nötr olarak yanlış yorumlanmış kelimeleri ele almaktadır. Beş farklı sözlük üzerinde yapılan kapsamlı deneyler ve örnek bir duyarlılık analizi, geleneksel sözlük-temelli duygu yaklaşımlarına göre doğrulukta önemli gelişmeler göstermiştir.

Wijayanto ve Sarno (2018), Yelp (gıda), IMDb (film) ve Amazon (ürünler) müşteri yorumlarını ile duygu analizi çalışması yapmıştır. Özellikleri denetlemek için TF-IDF kullanmıştır. Veri eğitimi kalitesini iyileştirmek için CHI2 yöntemini belirlemiştir. Çalışmada Destek Vektör Makinaları ile Naive Bayes algoritması ile sonuçlar bulunmuştur. Sonuçları normalize etmek için K Katlamalı Çapraz Doğrulama (k-fold cross validation) yöntemi kullanılmıştır. Sonuç çıktılarında tahmin, doğruluk, kesinlik ve f1-skor değerlerini veri setlerine göre ayrı ayrı oranları tespit edilmiştir. Değerler her bir veri kümesi için karşılaştırılmış ve hangisinde daha iyi sonuç verdiği belirlenmiştir.

Tongman ve Wattanakitrungroj (2018), yorumları olumlu ve olumsuz sınıflandırmak için En Yakın Komşu ve Destek Vektör Makinaları algoritmalarını kullanmıştır. İlk aşamada kelimeleri olumlu ve olumsuz olarak gruplamıştır. Grupladığı kelime listesinin terim frekansını ve ters terim frekansını (IDF) bulmuştur. Veri seti olarak makine öğrenmesi deposunda bulunan: Amazon, Yelp ve IMDB veri setlerini kullanmıştır. Yaptığı çalışmada en iyi sonucu Destek Vektör Makinalarından bulmuştur. Ribeiro vd. (2018), Doğal Dil İşleme (NLP) için makine öğrenmesinde modellerin anlamsal bozulmalara işaret etmektedir.

Bunun için Anlamsal Koruma Permütasyonu (Symantec Protection permutation) ve Eş Değer Değişken kurallara dikkat çekmiştir. Çalışmada, insan ile makine arasındaki hata farkları yer almıştır. Ayrıca anlamsal koruma ile verileri büyütme ve genişletmeler aktarılmıştır. Bari ve Saatçioğlu (2018), Duygu analizine ilişkin Amazon, Cornell, IMDB, Twitter, Yelp, Reviews ve Kaggle'a ait veri setleri üzerinde çalışmalar yapmıştır. Algoritma olarak Textblob, Opinion Finder VE Stanford NLP üzerinde testler yapmıştır. Çalışmada Textblob ve Stanford NLP iyi sonuç vermiştir. Opinion Finder ise en kötü sonucu vermiştir. Heyong ve Ming (2019), Hebb kural tabanlı özellik seçimi (HRFS) olarak adlandırılan bir yöntem önermektedir. HRFS, denetlenen Hebb kuralına dayanmaktadır. HRFS'yi diğer yedi özellik seçim yöntemiyle karşılaştırmak için altı kıyaslama veri kümesi kullanılmıştır. Deneysel sonuçlar, HRFS'nin kıyaslanan yöntemlerden daha iyi performans elde etmek için etkili olduğunu göstermektedir. HRFS, nöronlar arasındaki sinaps görüşüne göre ayırt edici terimleri belirleyebilmektedir. Ayrıca, HRFS ayrıca, özellik seçiminin karmaşıklığını azaltmak için matris işlemi görünümünde tarif edilebildiği için etkilidir. Hartmann vd. (2019), sosyal medya platformlarından çeşitli örneklem büyüklüklerini ve dilleri kapsayan 41 sosyal medya veri setinde on farklı yaklaşımın (beş sözlük tabanlı, beş makine öğrenme algoritması) performansını karşılaştırmıştır. Pazarlama araştırması ağırlıklı olarak destek vektörü ve Dil Sorgulama ve Kelime Sayımına (LIWC) dayanmaktadır. Çalışmalarda Rastgele orman (RF) veya Naive Bayes, insan sezgisini doğru bir şekilde ortaya çıkarmak açısından en iyi performansı göstermiştir. Özellikle, RF küçük örnek büyüklükleri için NB, üç sınıf duygular için sürekli yüksek performans sergilemiştir. Lexicon tabanlı yaklaşımlar, LIWC, makine öğrenimine kıyasla kötü performans gösterdiğinden bahsetmiştir.

3. MAKİNE ÖĞRENMESİ

Makine öğrenmesi adı son yıllarda sıkça duyulmaktadır. Makine öğrenmesi, çeşitli algoritmalar ve yöntemler ile veride bazı kalıpları aramakta ve bu kalıplara karşılık gelen etiketlere bakarak önce öğrenmekte, daha sonra (öğrendiklerine benzer durumla karşılaştığında) deneyimlerinden yararlanarak çıkarım yapabilen sistemler geliştirmeye imkân sağlamaktadır. Bu imkânı, çeşitli matematiksel ve istatistiksel yöntemlerin kullanıldığı birçok algoritma ile sağlamaktadır. Bu yöntem ve algoritmaların bir veya birkaçı bir arada kullanılarak model oluşturulmaktadır ve bu model, tahmin edilmesi istenilen şeyi, en verimli, en kesin en hızlı biçimde tahmin etmeyi amaçlamaktadır. Makine öğreniminin başlıca uygulamaları; makine algılaması, bilgisayarlı görme, veri madenciliği, doğal dil işleme, söz dizimsel örüntü tanıma, arama motorları, tıbbi tanı koymak, biyoinformatik, beyin-makine arayüzleri ve kiminformatik, kredi kartı dolandırıcılığı denetimi, borsa çözümlemesi, DNA dizilerinin sınıflandırılması, konuşma ve elyazısı tanıma, bilgisayarlı görmede nesne tanıma, oyun oynama ve robotik örnek verilebilir (Wikipedia,2019).

3.1. Gözetimli Öğrenme

Gözetimli öğrenmede, öğrenilmek istenen kavram ile ilgili toplanan gözlemler bir eğitim kümesi olarak modele verilmektedir. Eğitim kümesinde her örnek için istenen çıktı değerleri de verilmektedir. Bu bilgiler kullanılarak giriş ve çıkış arasında bir ilişki oluşturulmaktadır. Oluşturulan ilişki kullanılarak gelecekte karşılaşılacak girdinin karşılık geldiği çıkış değerlerini tahmin etmektedir.

3.2. Gözetimsiz Öğrenme

Gözetimsiz öğrenmede modele girdi verilmekte ve sonuç istenmektedir. Modele eğitim verisi verilmemekte, sadece test verisi verilmektedir. Ham veriden sistem bir anlam çıkarmaya çalışmaktadır. Bu öğrenme şeklinde genellikle

kümeleme yapılmaktadır. Gözetimsiz öğrenmede K-Means, K-Medoids veya CLARANS yöntemleri kullanılmaktadır (Aydın vd., 2018).

3.3. Makine öğrenme Algoritmaları

3.3.1. KNN algoritması

K-en yakın komşu algoritması (K-NN), gerçekleştiriminin basit ve kolay, öğrenme sürecinin güçlü ve kullanışlı olmasından dolayı sınıflandırmada yaygın biçimde kullanılmaktadır. Makine öğrenmesi, veri madenciliği gibi çok çeşitli alanlarda uygulanmaktadır. Bu algoritmanın uygulanması kolaydır, gürültülü eğitim verisine (noisy training data) dayanıklıdır ve eğitim verileri büyükse oldukça etkilidir. K-NN algoritması, eğitiminin olmaması, gerçekleştiriminin kolay, analitik olarak izlenebilir, yerel bilgilere uyarlanabilir, paralel gerçekleştirime uygun, gürültülü eğitim verilerine karşı dirençli olması gibi avantajları ile sınıflandırma uygulamalarında özellikle tercih edilmektedir. Uzayda K adet komşusuna bakılmakta ve hangi gruptan daha fazla ise, o gruba dahil edilmektedir. K sayısı tek sayı alınmaktadır. Nedeni ise grupların eşit çıkmaması içindir (Taşcı ve Onan, 2014).

3.3.2. Naive Bayes algoritması

Verileri olasılık ilkeleri ile hesaplayarak sınıflandıran bir sınıflandırma algoritmasıdır. Basit bir ifadeyle, bir Naive Bayes sınıflandırıcı, bir sınıftaki belirli bir özelliğin varlığının başka herhangi bir özelliğin varlığına bağlı olmadığını varsaymaktadır. Makine öğrenmesinde en çok tercih edilen algoritmalar arasında yer almaktadır. Literatürde birçok kullanıldığı çalışma vardır (Wikipedia, 2019).

Naive Bayes algoritmasının formülü:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

P(A|B); B olayı gerçekleştiği durumda A olayının meydana gelme olasılığıdır

P(B|A); A olayı gerçekleştiği durumda B olayının meydana gelme olasılığıdır

P(A) ve P(B); A ve B olaylarının önsel olasılıklarıdır.

3.3.3. Destek Vektör makineleri algoritması

Örüntü tanıma ve sınıflandırma problemlerinin çözümü için Vapnik tarafından geliştirilmiştir. Temelleri istatistiksel öğrenme teorisine diğer bir ifadeyle Vapnik-Chervonenkis (VC) teorisine dayanmaktadır. İki sınıflı ve çok sınıflı sınıflandırma probleminin çözümü için geliştirilmiş makine öğrenmesi algoritmasıdır. Doğrusal ve doğrusal olmayan problemlerin sınıflandırılmasında kullanılmaktadır (Ayhan ve Erdoğmuş, 2014).

3.3.4. Rastgele orman (Random Forest) algoritması

Breiman tek bir karar ağacı üretmek yerine her biri farklı eğitim kümeleriyle eğitilen çok sayıda, çok değişkenli ağacın kararlarının birleştirilmesini önermiştir. Farklı eğitim kümeleri önyükleme (bootstrap) ve rasgele özellik seçimi ile orijinal eğitim setinden oluşturulmaktadır. Çok değişkenli karar ağaçları CART algoritmasıyla elde edilmektedir. Önce her karar ağacı kendi kararını vermekte, karar ormanı içerisinde maksimum oyu olan sınıf son karar olarak kabul edilmekte ve gelen test verisi o sınıfa dahil edilmektedir (Diri ve Kaban, 2008).

3.3.5. Eğri uydurma (Regression) yöntemi

Regresyon problemleri, üretilen çıktının sürekli sayılardan oluştuğu durumlar için kullanılmaktadır. Örneğin bir kişinin performansı analiz edilirken, çalıştığı gün, kişinin ürettiği ürün adedine göre verimlilik hesabı yapılacağı zaman regresyon yönteminden faydalanılmaktadır. En çok regresyon yöntemlerinden Lineer regresyon ve Lojistik regresyon tercih edilmektedir. Lineer Regresyonda girdi geldiği zaman, fonksiyonda çıkış değeri hesaplanabilmektedir. Örneğin: “Kuzey Amerika’nın en yüksek enlemlerinde yaşayanlar güneş ışığının zararlı

ışınlarına daha az maruz kalır ve bu nedenle orada yaşayanların cilt kanseri nedeniyle ölüm riskinin az olduğunu tahmin edebilirsiniz ". Lojistik regresyon, bir sonucu belirleyen bir veya daha fazla bağımsız değişken bulunan bir veri kümesini analiz etmek için kullanılan istatistiksel bir yöntemdir (Veribilimci, 2019).

3.3.6. Yapay sinir ağları

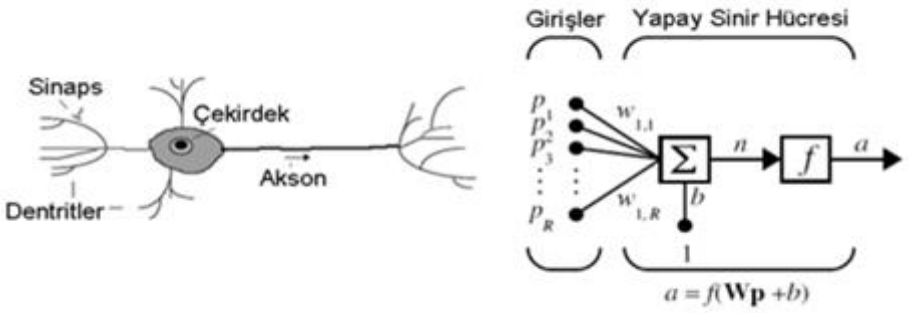
Yapay sinir ağları; insan beyninden esinlenerek, öğrenme sürecinin matematiksel olarak modellenmesi uğraşı sonucu ortaya çıkmıştır. Örneklerden olaylar arasındaki ilişkileri öğrenerek, sonraki görmediği örnekler hakkında öğrendikleri bilgileri kullanarak karar veren sistemlerdir. Uygulamada ve pratikte tek başına kullanılmaz. Birden fazla yapay sinir birbiri ile bağlantılı olarak çalışmaktadır. Dendrites bilgiyi diğer sinir hücrelerinden alır ve çekirdeğe iletilir. Çekirdek (Soma) gelen bilgiyi işler ve bilgi önemli ise Akson vasıtasıyla diğer sinir hücrelerine iletmektedir. Benzer şekilde yapay sinir hücreleri dışarıdan gelen bilgileri bir toplama fonksiyonu ile toplamakta ve aktivasyon fonksiyonundan geçirerek çıktıyı üretip ağın bağlantılarının üzerinden diğer hücrelere (proses elemanlarına) göndermektedir. Değişik toplama ve aktivasyon fonksiyonları vardır. Yapay sinir ağlarını birbirlerine bağlayan bağlantıların değerlerine ağırlık değerleri denmektedir. Proses elemanları birbirlerine paralel olarak 3 katman halinde bir araya gelerek bir ağ oluştururlar.

Bunlar;

- Girdi katmanı
- Ara katmanlar
- Çıktı katmanı

Bilgiler ağa girdi katmanından iletilmektedir. Ara katmanlarda işlenerek oradan çıktı katmanına gönderilmektedir. Bilgi işlemeden kasıt ağa gelen bilgilerin ağın ağırlık değerleri kullanılarak çıktıya dönüştürülmesidir. Ağın girdiler için doğru

çıktıları üretebilmesi için ağırlıkların doğru değerlerinin olması gerekmektedir. Doğru ağırlıkların bulunması işlemine ağın eğitilmesi denmektedir.



Şekil 3.1. Sinir ağları

3.4. Derin Öğrenme Nedir?

Klasik makine öğrenmesinde; teknikler ile modelin oluşturulması, verinin temizlenmesi ve yazıların vektörlere çevrilmesi işlemleri vardır. İncelendiğinde makine öğrenmesinde bu adımları yapacak kişilere ihtiyaç duyulmaktadır. Burda zaman ve maliyet artmaktadır. Derin öğrenmede; klasik makine öğrenmesi ve görüntü işleme tekniklerinin aksine, öğrenme işlemini ham veri üzerinde yapmaktadır. GPU'lar, çok daha büyük eğitim setleri kullanarak bu derin nöral ağları çok daha kısa sürelerde ve çok daha az veri merkezi altyapısı kullanarak eğitmek için kullanılmaktadır. Ham veriyi işlerken gerekli bilgiyi farklı katmanlarda oluşturmuş olduğu temsillerle elde etmektedir (İnik ve Ülker, 2017).

3.4.1. Konvolüsyonel sinir ağları (Convolution Neural Network-CNN)

Konvolüsyonel sinir ağları, çok katmanlı sinir ağlarının bir çeşididir. Görme merkezindeki hücreler tüm görseli kapsayacak şekilde alt bölgelere ayrılmış, basit hücreler, kenarbenzeri özelliklere, karmaşık hücreler ise daha geniş alıcılarla, tüm görsele yoğunlaştığı düşünülmektedir. İleri yönlü bir sinir ağı olan CNN algoritması da, hayvanların görme merkezinden esinlenilerek ortaya atılmıştır. Buradaki matematiksel konvolüsyon işlemi, bir nöronun kendi uyarı alanından uyaranlara verdiği cevap olarak düşünülebilir. CNN, bir veya daha

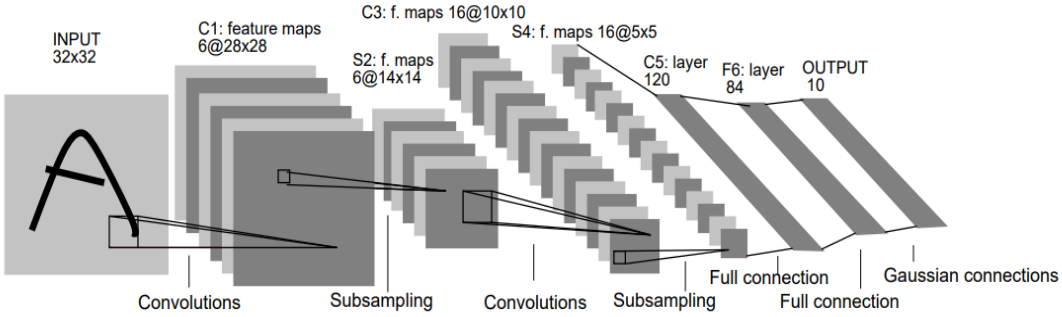
fazla konvolüsyonel katman, altörnekleme (subsampling) katmanı ve bunun ardından standart çok katmanlı bir sinir ağı gibi bir veya daha fazla tamamen bağlı katmandan oluşmaktadır (Şeker vd., 2017). Google'nin 2015 yılında derin öğrenme işlemlerinde kullanılmak üzere, açık kaynak kodlu makine öğrenmesi kütüphanesidir. Makine öğrenme kütüphanesi olarak da adlandırılan sistem, verilen verilerin kodlanması ve ayırt edilmeleri sağlanmaktadır. Google Brain ekibi tarafından oluşturulan. TensorFlow, bir dizi makine öğrenimini ve derin öğrenmeyi (nöral ağ oluşturma) modelleri ve algoritmaları bir araya getirerek, ortak bir metafor yoluyla faydalı hale getirir. Bu uygulamaları, yüksek performanslı C++ ile yürütürken, framework ile uygulamalar oluşturmak için kullanışlı bir ön uç API'si sağlamak için Python'u kullanır.

TensorFlow, el yazısı haneli sınıflandırma, görüntü tanıma, kelime toplama, tekrarlayan sinir ağları, makine çevirisi için dizi-sıralı modeller, doğal dil işleme ve PDE (kısmi diferansiyel denklem) tabanlı similasyonlar için derin sinir ağlarını çalıştırabilir ve çalıştırabilir. TensorFlow, eğitim için kullanılan aynı modellerle, ölçekli üretim tahminini destekler. Tensorflow tek bir API ile masaüstü, sunucu veya mobil cihazdaki bir veya daha fazla CPU'ya veya GPU'ya dağıtılması olanağını sunmaktadır. CNN daha düşük işlem gereksinimleri için tasarlanmış birçok katmanlı bir yapay sinir ağı sistemidir. CNN katmanları olarak giriş katmanı, çıkış katmanı, birden fazla konvolüsyon katmanları içeren gizli katman, havuz katmanı, tam bağlı katman ve normalleştirme katmanlarından oluşur. Giriş katmanından giriş, sonra gizli katmana gelir. Bizim modelimize ve veri boyutuna bağlı olarak birçok gizli katman olabilir. Her gizli katman genellikle özellik sayısından daha büyük sayıda olabilir. Gizli katmandaki öğrenilen ağırlıklar ile önceki katmanın çıktısının matris çarpımı ile hesaplanır ve sonra öğrenilebilir hata payı ek olarak ağ doğrusal olmayan sonuçlar hesaplanır. Gizli katmandan çıktı sonra her sınıfın olasılık değerine göre sınıfın çıktısını dönüştürür ve sigmoid fonksiyonuna gönderilir. Maksimum havuzlama, örnekleme dayalı bir ayrıklaştırma sürecidir. Amaç, bir giriş gösterimini aşağıdan örneklemektir (görüntü, gizli katman çıktı matrisi, vb.), Boyutluluğunu düşürür ve bindirilmiş alt bölgelerde bulunan özellikler hakkında varsayımların yapılmasını sağlar. Bu kısmen temsilin soyutlanmış bir

x”şeklini sunarak, uydurmaya yardımcı olmak için yapılır. Aynı zamanda, öğrenilecek parametre sayısını azaltarak hesaplama maliyetini düşürür. Maksimum havuzlama, ilk gösterimin örtüşmeyen alt bölgelerine (genellikle) bir maksimum filtre uygulanarak yapılır.

Sigmoid işlevini kullanmamızın ana nedeni, (0-1) arasında olmasıdır. Bu nedenle, özellikle bir olasılığı çıktı olarak tahmin etmemiz gereken modeller için kullanılır. Her şeyin olasılığı sadece 0 ile 1 arasındadır, sigmoid doğru seçimdir. SoftMax fonksiyonu çok sınıflı sınıflandırma için kullanılan daha genel bir lojistik aktivasyon fonksiyonudur.

Konvolüsyon ve pooling katmanlarının ardından gelen katmandır. Bu katman aslında prensipte multi layer perceptron'a benzer. Her bir girdinin gizli katmanda ki bütün nöronlar ile bağlantısı vardır. Gizli katmanlar arasındaki bütün nöronlar da birbiriyle bağlantılıdır. İnput ağırlıkları gizli katmanlara dağıtılır. bu sayede yük hafifler, fakat bu noktada işin içine overfitting işlemi girer. Eğer modelimiz, eğitim için kullandığımız veri setimiz üzerinde gereğinden fazla çalışıp ezber yapmaya başlamışsa ya da eğitim setimiz tek düze ise overfitting olma riski büyük demektir. Eğitim setinde yüksek bir skor aldığımız bu modele, test verimizi gösterdiğimizde muhtemelen çok düşük bir skor elde edeceğiz. Çünkü model eğitim setindeki durumların bire bir kopyalarını arar hale gelmiştir. En küçük değişiklikler bile modelimizi çaresiz bırakıp kötü skor almamıza sebep olur. Modelimiz veri için yeterince karmaşık olmadığından önyargılı bir şekilde yanlış tahminler yapıyordur. Burada model elinden geleni yapıyor aslında ama veri seti için yeterince karmaşık bir yapısı olmadığı için yetersiz kalıyor ve underfitting etkisi veriyor. Parabolik bir şekilde dağılan bir veri setine bir dereceli lineer regresyon uygulamak buna bir örnek olabilir.



Şekil 3.2. Lenet CNN mimarisi

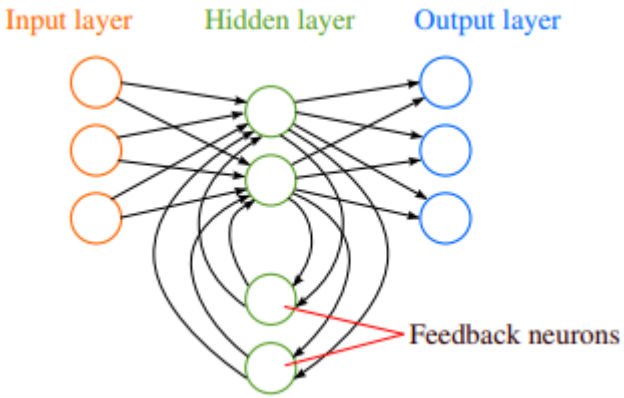
CNN ile ilgili literatür taramasında az miktarda etiketlenen bilgi ile çok iyi sonuçlar verdiği bilinmektedir. Rusça metinlerde, metin yazarlarının cinsiyetinin belirlenmesi çalışmasında CNN kullanılmış %86 başarı oranı sağlamıştır. (Sboev vd., 2016)

Gao vd. (2014) okuduğu kaynak belgeleri inceleyerek, CNN tabanlı bir tavsiye sistemi geliştirmişlerdir. Web üzerinde milyonlarca veriyi inceleyerek, kaynak-hedef belge çiftlerini vektörler ile eşleştirmiştir. Bu sayede ilgi duyulabilecek belgeler arasındaki mesafeler azaltılmıştır. Oluşturdukları model ilgililik alanında başarılı olmamış, konu modelleme alanında etkili sonuçlar üretmiştir.

3.4.2. Uzun kısa vadeli hafıza ağları (Recurrent Neural Network -RNN)

Bu mimaride yapı; inputu bir önceki nputlara bağlı olarak olarak değerlendirmesidir. Örneğin "Ağaç toprakta yetişir" cümlesinde "toprak" kelimesini tahminetmek kolaydır. Fakat bağlamlar arası boşluk arttığında RNN modelin geçmişten gelen bir bilgiyi kullanması oldukça zordur. İngilizce konuşurum." gibi bir metinde "İngilizce" kelimesini tahmin ederken, içinde bulunduğu cümleden yola çıkarak bir dil adı olacağını tahmin edilebilir, ancak doğru kelimenin "İngilizce" olduğunu tahmin etmek için, metnin başındaki cümledeki yer bilgisini hafızada tutmak gerekmektedir. Teoride mümkün olan "uzun-vadeli bağımlılıklar", pratikte büyük problemlere yol açtığı görülmüştür (Şeker vd., 2017). Literatür taraması bölümünde yer alan Larsson ve Nılsson (2017) yaptığı çalışmada %80 başarı oranı görülmüştür. Graves ve

Schmidhuber (2005)'de yaptığı çalışmada, LSTM mimarileri konuşma/metin işleme konularında oldukça iyi sonuçlar vermektedir. Çerçeve tabanlı ses sınıflandırma çalışmasında farklı lehçelerdeki zengin içerikler barındıran TIMIT veri kümesi kullanılmış, %70 başarı oranı elde edilmiştir.

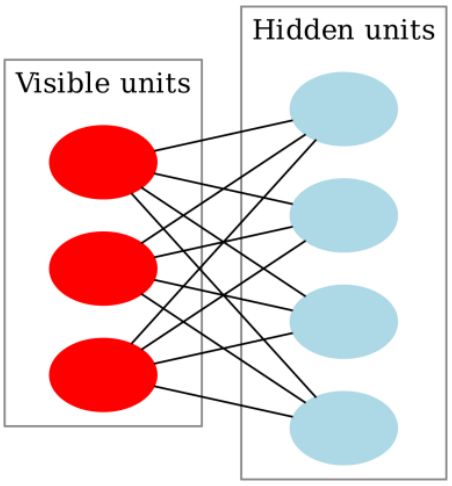


Şekil 3.3. RNN yapısı

3.4.3. Sınırlı Boltzmann makineleri (Restricted Boltzmann Machines-RBM)

Sınırlı Boltzmann makineleri özellik çıkarımı, sınıflandırma ve regresyon alanlarında kullanılmaktadır. Giriş ağı üzerinde olasılık dağılımlarını öğrenebilen bir yapay sinir ağıdır. Girdi katmanı ve gizli katman olmak üzere iki katmandan oluşur. Düğümlerde girdiler hesaplanmaktadır ve sonraki düğüme o girdiyi iletip iletmeyeceğine ilişkin rastgele kararlar vermektedir.

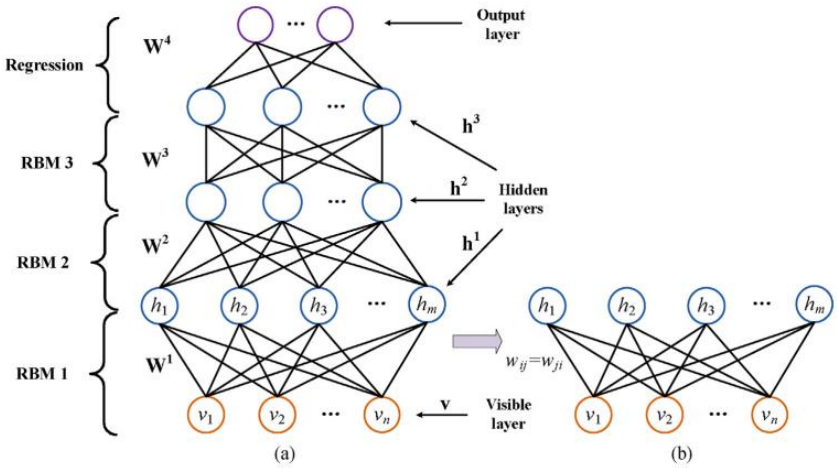
100 milyondan fazla kullanıcı / film değerlendirmesi içeren Netflix veri setine Sınırlı Boltzmann Makinelerinin başarıyla uyguladıklarını göstermişlerdir Salakhutdinov vd., 2007).



Şekil 3.4. Sınırlı Boltzmann makineleri mimarisi

3.4.4. Derin inanç ağları (Deep belief network-DBF)

Çok katmanlı yapay sinir ağlarından oluşmaktadır. Sınırlı Boltzmann makinelerinin yığını olarak ifade edilmektedir. Greedy öğrenme algoritması kullanılarak önceden eğitilmiştir. Açgözlü öğrenme algoritması yukarıdan aşağıya, üretim ağırlıklarını öğrenmek için katman-katman yaklaşımını kullanır. Bu üretken ağırlıklar bir katmandaki değişkenlerin yukarıdaki katmandaki değişkenlere nasıl bağlı olduğunu belirlemektedir.



Şekil 3.5. Derin inanç ağları mimarisi

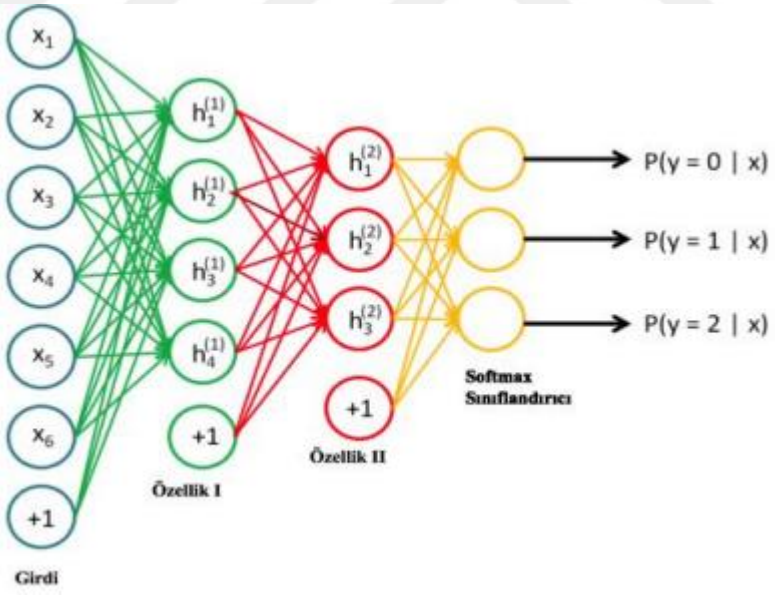
Derin inanç ağları görüntü tanıma ve üretme konuları üzerine çalışmalar yapılmıştır (Hilton vd., 2006). Salakhutdinov ve Hinton, bütün dokümana TF-

IDF uygulamak yerine DBN kullanan bir sistem ile büyük bir belge veri kümesinde kelime-sayısı vektörleri elde etmişlerdir

3.4.5. Derin Oto-Kodlayıcılar (Deep Autoencoders-AE)

Oto-kodlayıcılar (Autoencoders-AE) Yapay sinir ağına verilen veri kodlanır ve boyutu indirgenir. Daha az yer kapladığı için daha fazla veri depolanması sağlanır. Şifreli veri çıktı değerinde, giriş değeriyle aynı olana kadar yapay sinir ağları eğitilir (VeriBilimcisi, 2019). Örneğin şifreyi yazdığımız zaman şifrelenir, sonrasında şifrenin açık hali istendiğinde aynısını bulana kadar sinir ağlarının ağırlıklandırılması değiştirilir (Şeker vd., 2017).

Derin veya yığınlı oto-kodlayıcılar ise (Deep/Stacked Autoencoder-DAE), her birkatmandaki çıktıların ardışık katmanın girişlerine bağlandığı AE'lerin çok katmanlarından oluşan sinir ağıdır (Şeker vd., 2017).



Şekil 3.6. Derin Oto-Kodlayıcıların yapısı

4. DOĞAL DİL İŞLEME

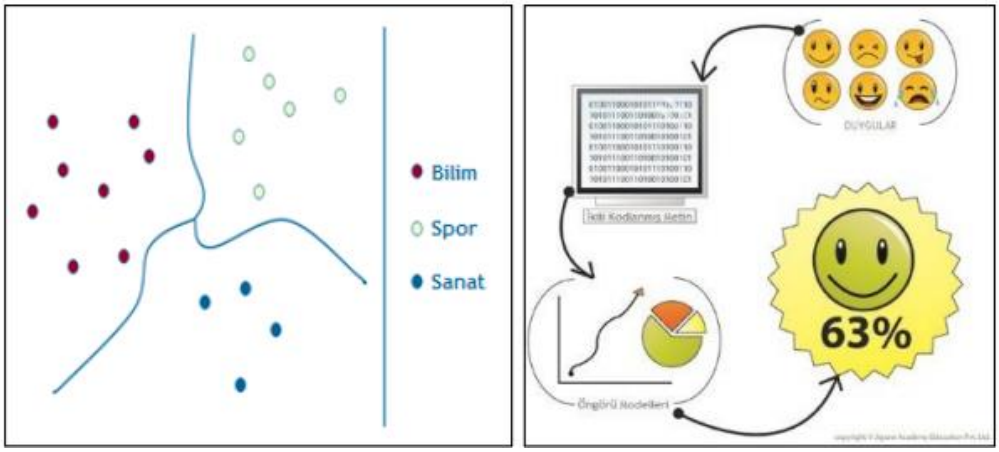
Yeni iletişim sistemlerinden biri olan internetin aktif kullanılmasından dolayı, sürekli Yapay zekâ ve dil biliminin gelişimiyle birlikte doğal dil işleme meydana gelmiştir. Yapay zekânın bir alt dalıdır. Türkçe, İngilizce, Fransızca gibi dillerin işlenmesi ve kullanılması amacı ile araştırma yapan bilim dalıdır. Doğal dillerin kurallı yapısının çözümlenerek anlaşılması veya yeniden üretilmesi amacını taşır. Bu çözümlemenin insana getireceği kolaylıklar, yazılı dokümanların otomatik çevrilmesi, soru-cevap makineleri, otomatik konuşma ve komut anlama, konuşma sentezi, konuşma üretme, otomatik metin özetleme, bilgi sağlama gibi birçok başlıkla özetlenebilir. Bilgisayar teknolojisinin yaygın kullanımı, bu başlıklardan üretilen uzman yazılımların gündelik hayatımızın her alanına girmesini sağlamıştır. Örneğin, tüm kelime işlem yazılımları birer imlâ düzeltme aracı taşır. Bu araçlar aslında yazılan metni çözümleyerek dil kurallarını denetleyen doğal dil işleme yazılımlarıdır (Wikipedia,2019). Doğal dil işleme alanında çalışan kişiler dillerin özelliklerini kendi bakış açılarından değerlendirmeye başlamışlardır. Doğal dil işleme konularında ses bilimi, biçim bilimi, söz dizimi ve anlam bilimi konularında çalışmalarına ağırlık vermektedirler.

4.1. Metin Sınıflandırma ve Duygu Analizi

Metin sınıflandırmanın başlangıcını 1960'lı yıllara gitsede, yoğun olarak çalışmaların yapılması 1980'lerin sonu ile 1990'lı yıllar olmuştur. İlk yapılan çalışmaların çoğunluğu, uzman sistemler yaklaşımını örnek almaktaydı ve kural tabanlı sistemler tasarlanmaktaydı. 1990'lı yılların sonunda donanımsal ve teknolojik gelişmelerle birlikte bu alanda makine öğrenmesi yöntemleri kullanılmaya başlanmıştır (Tantuğ, 2012).

Metin sınıflandırması daha önceden tanımlanmış kategorilere, doğal dil işleme yöntemleri kullanılarak yeni gelen metnin kategorize edilmesidir. Duygu analizi ile kişilerin ürünler veya herhangi bir konu üzerinden ne düşündükleri olumlu, olumsuz, nötr olarak gruplandırabiliyoruz Facebook, Twitter, Linkedin,

Instagram, gibi micro blogların yanı sıra, web tabanlı haber siteleri ve hatta alışveriş siteleri dahil olmak üzere birçok platform, anlık mesaj ve yorum ile kullanıcıyı aktif kılmaktadır. Çok yönlü ileti paylaşımına ve interaktif etkileşime olanak sağlayan kullanıcı tabanlı bu platformlarda, kullanıcının oluşturduğu iletiler ile hızlı bir veri akışı olmakta ve bu veriler birer analiz nesnesine dönüşmektedir. Platformlar depolanan verileri analizini kendi bünyesinde veya farklı firmalar aracılığıyla analiz edebilmektedir. Veri analizinin kullanıldığı alanlardan biri de siyasal alandır. Siyasi partiler, özellikle seçim dönemlerinde sosyal medya üzerinden yapılan yorum, beğeni veya yanıtlardan bir analiz çalışması yapabilmekte, seçim sonucuna dair çeşitli öngörülere ulaşabilmektedir.



Şekil 4.1. Metin Sınıflandırma

4.2. Karışıklık Matrisi (Confusion Matrix)

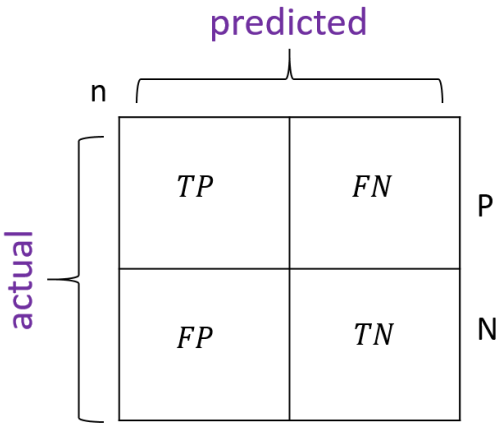
Karışıklık matrisi, örnek veri kümesindeki gerçek sınıf ile tahmin edilen sınıf etiket sayılarını içerir. Karmaşıklık matrisinde bulunan değerleri açıklamaları aşağıdaki gibidir;

-TP (True Pozitif) kısaltması gerçek değeri ile tahmin ettiğimiz değeri pozitif olanı ifade etmektedir.

-TN (True Negatif) gerçek değeri ile tahmin ettiğimiz değeri olumsuz olanları ifade etmektedir.

-FN (False Negatif) gerçek değeri pozitif olan ve tahmin ettiğimiz değeri negatif olanları ifade etmektedir.

-FP (False Pozitif) gerçek değeri negatif iken bizim tahmin ettiğimiz pozitif olanları ifade etmektedir



Şekil 4.2. Karışıklık Matrisi

4.3. Tweet Nedir?

Tweet, Twitter üzerindeki kayıtlı kullanıcıların gönderdikleri kısa sosyal mesajlara verilen kısa addır. Twitter üzerindeki kullanıcıların yaptıkları paylaşımların tümü "Tweet" adı verilen gönderi biçimiyle sınırlıdır.

"Tweet atma" veya "Tweetlemek" Twitter üzerinde yer alan üyelerin gönderi yapması anlamına gelmektedir. Twitter üzerinde kullanıcıların birçok faaliyeti olmasına karşın gönderi biçimi olarak tek biçim olarak kabul edilen Tweet bulunmaktadır. Bir Tweet'in maksimum uzunluğu 140 karakter olabilir ve bu Tweet bir görsel dosyayla ya da video ile birlikte kullanılabilir. Tweet'ler içerisinde diğer siteler için link içerebileceği gibi Twitter'ın başka üyelerini etiket olarak barındırabilir.

Şekil 4.3. Örnek Tweet

4.4. Kaggle Nedir ve Veri Seti

Veri bilimi ve makine öğrenmesi üzere çalışan kişilerin paylaşım yaptıkları, veri seti, kodların paylaşılması ve ödüllü projelerin yer aldığı topluluktur. Haziran 2017'de, Kaggle üye sayısının 1.000.000 kullanıcıyı geçtiğini duyurmuştur. Topluluk 194 ülkeyi kapsamaktadır. Dünyadaki en iyi bilinen araştırmacıların çoğundan yeni başlayanlara kadar, dünyanın en büyük ve en çeşitli veri topluluğudur (Wikipedia,2019). Çalışmamızda kullanılan veri setleri Kaggle'dan temin edilmiştir.

```
ItemID,Sentiment,SentimentText
1,0,                    is so sad for my APL friend.............
2,0,                 I missed the New Moon trailer...
3,1,              omg its already 7:30 :O
4,0,           .. Omgaga. Im sooo  im gunna CRy. I've been at this dentist since 11.. I was suposed 2 just get a crown put on (30mins)...
5,0,         i think mi bf is cheating on me!!!       T_T
6,0,         or i just worry too much?
7,1,       Juuuuuuuuuuuuuuuussssst Chilllin!!
8,0,       Sunny Again        Work Tomorrow  :-|       TV Tonight
9,1,      handed in my uniform today . i miss you already
10,1,      hmmmm.... i wonder how she my number @-)
11,0,      I must think about positive..
12,1,      thanks to all the haters up in my face all day! 112-102
13,0,      this weekend has sucked so far
14,0,     jb isnt showing in australia any more!
```

Şekil 4.4. Eğitim Veri Seti

```
"0","2329204790","Thu Jun 25 10:28:27 PDT 2009","NO_QUERY","CJROSE218","@koolgirl37 read my tweet below "
"0","2329204835","Thu Jun 25 10:28:27 PDT 2009","NO_QUERY","mattfca","My life  http://mattf.ca/2009/06/24/vav-me/"
"0","2329204987","Thu Jun 25 10:28:28 PDT 2009","NO_QUERY","360cookie","Tried to get the mutant Fawkes to follow me but he wouldn't  it's so lonely without
"0","2329205009","Thu Jun 25 10:28:28 PDT 2009","NO_QUERY","dandykim","Sick  Spending my day laying in bed listening to @taylorswift13"
"0","2329205038","Thu Jun 25 10:28:28 PDT 2009","NO_QUERY","bigenya","Gmail is down? "
"0","2329205473","Thu Jun 25 10:28:30 PDT 2009","NO_QUERY","LeeLHoke","rest in peace Farrah! So sad "
"0","2329205574","Thu Jun 25 10:28:30 PDT 2009","NO_QUERY","davidlmulder","@Eric_Urbane Sounds like a rival is flagging your ads. Not much you can do about
"0","2329205794","Thu Jun 25 10:28:31 PDT 2009","NO_QUERY","tpchandler","has to resit exams over summer...  wishes he worked harder at first year of uni..
"1","1467822272","Mon Apr 06 22:22:45 PDT 2009","NO_QUERY","ersle","I LOVE @Health4UandPets u guys r the best!! "
"1","1467822273","Mon Apr 06 22:22:45 PDT 2009","NO_QUERY","becca210","im meeting up with one of my besties tonight! Cant wait!!  - GIRL TALK!!"
"1","1467822283","Mon Apr 06 22:22:46 PDT 2009","NO_QUERY","Wingman29","@DaRealSunisaKim Thanks for the Twitter add, Sunisa! I got to meet you once at a H
"1","1467822287","Mon Apr 06 22:22:46 PDT 2009","NO_QUERY","katarinka","Being sick can be really cheap when it hurts too much to eat real food  Plus, your
"1","1467822293","Mon Apr 06 22:22:46 PDT 2009","NO_QUERY","_EmilyYoung","@LovesBrooklyn2 he has that effect on everyone "
```

Şekil 4.5. 1. Test Veri Seti

```
ItemID,Sentiment,SentimentText;;
5703010;1;@VirginAmerica plus you've added commercials to the experience... tacky.
5703010;0;"@VirginAmerica it's really aggressive to blast obnoxious ""entertainment"" in your guests' faces &amp;"
5703010;0;@VirginAmerica and it's a really big bad thing about it
5703010;0;@VirginAmerica seriously would pay $30 a flight for seats that didn't have this playing.
5703010;1;@VirginAmerica yes, nearly every time I fly VX this “ear worm” won’t go away :)
5703000;1;@virginamerica Well, I didn't…but NOW I DO! :-D
5702950;1;@VirginAmerica it was amazing, and arrived an hour early. You're too good to me.
5702900;1;@VirginAmerica I &lt
5702900;1;@VirginAmerica This is such a great deal! Already thinking about my 2nd trip to @Australia &amp
5702870;1;@VirginAmerica @virginmedia I'm flying your #fabulous #Seductive skies again! U take all the #stress away from travel http://t.co/ahlXHhKiyn
5702860;1;@VirginAmerica Thanks!
```

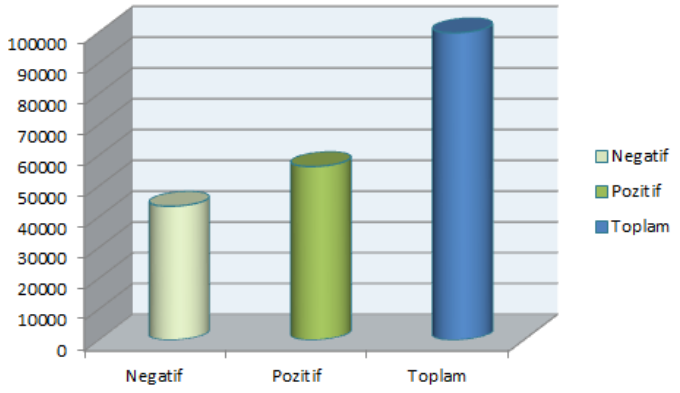
Şekil 4.6. 2. Test veri seti

```
id,sentiment,text
2,1,@ScottWalker: Didn't catch the full #GOPdebate last night. Here are some of Scott's best lines in 90 seconds. #Walker16 http://t.co/ZSfF…
4,1,@RobGeorge: That Carly Fiorina is trending -- hours after HER debate -- above any of the men in just-completed #GOPdebate says she's on …
5,1,@DanScavino: #GOPDebate w/ @realDonaldTrump delivered the highest ratings in the history of presidential debates. #Trump2016 http://t.co…
6,1,"@GregAbbott_TX: @TedCruz: ""On my first day I will rescind every illegal executive action taken by Barack Obama."" #GOPDebate @FoxNews"
12,1,@WayneDupreeShow: Just woke up to tweet this out #GOPDebate
15,1,@pattonoswalt: I loved Scott Walker as Mark Harmon's romantic rival in SUMMER SCHOOL. Look it up. #GOPDebate
21,1,@ChuckNellis: Cruz has class &amp
49,1,"@DanScavino: #GOPDebate w/ @realDonaldTrump delivered the highest ratings in the history of presidential debates. #Trump2016 Fox say thanks
52,1,@megynkelly: .@ChrisStirewalt: Big moments were @RandPaul vs. @ChrisChristie, and @realDonaldTrump reaction to women question. #KellyFil…
60,1,@KarrattiPaul: Join ISIS and sign your death warrant,
63,1,@FrankLuntz: Before the #GOPDebate, 14 focus groupers said they had favorable view of Trump.
64,1,@ Holly Renee: Just made my first donation to @CarlyFiorina.
```

Şekil 4.7. 3. Test veri seti

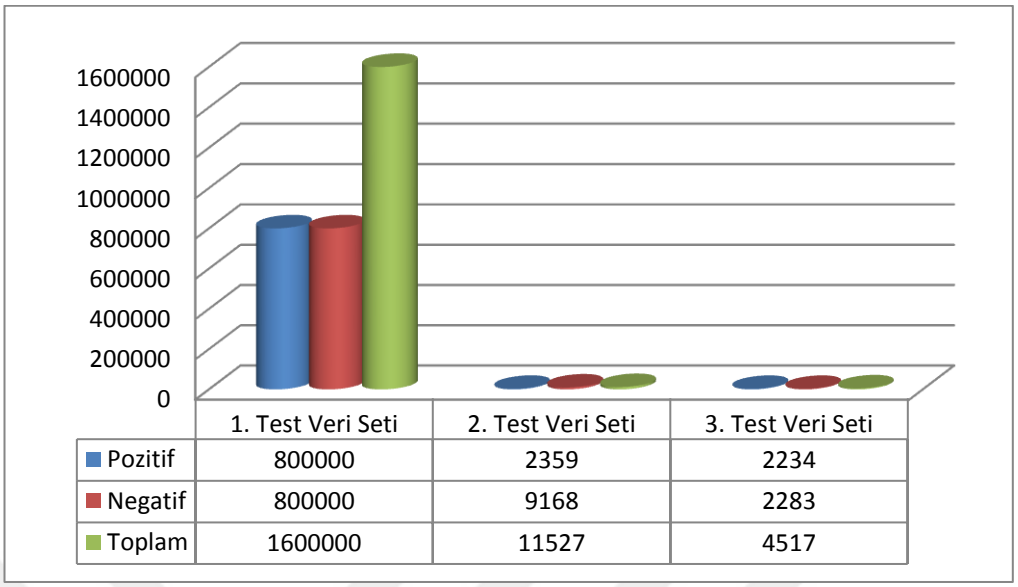
```
Name
Aaliyah
Abagail
Abbey
Abbie
Abbigail
Abby
Abigail
Abigale
Abigayle
Abril
Addison
Adeline
Adriana
Adrianna
Adrienne
```

Şekil 4.8. İsim listesi



Şekil 4.9. Eğitim veri setinde bulunan duygu dağılımı

Tablo 4.1. Test veri setinde bulunan duygu dağılımı



	1. Test Veri Seti	2. Test Veri Seti	3. Test Veri Seti
Pozitif	800000	2359	2234
Negatif	800000	9168	2283
Toplam	1600000	11527	4517

4.5. Düzenli ifadeler

Hesaplama alanında belirli yazım kurallarına göre düzenlenmiş, bir dizge (yazı karakteri) setini tanımlayan veya onunla uyuşan dizgelerdir. Düzenli ifadeler birçok metin düzenleyici, arama araçları ve metin tabanlı belirli desenlere ifade etme araçları tarafından kullanılır. Birçok programlama dili dizgeleri idare etmek için düzenli ifadeleri destekler. Yaygın uygulamaları veri doğrulama, veri ayıklama (özellikle ağ ayıklama), veri dönüştürme, basit (metin) ayrıştırma, sözdizimi vurgulama sistemlerinin hazırlanması ve daha pek çok kullanım alanı vardır (Wikipedia,2019).

Veri seti içerisinde bulunan olmaması gereken ifadeleri temizleyerek, modelin daha iyi sonuç vermesi sağlanabilir. Düzenli ifadeler ile yapılacak temizleme işleminde url (www., http, https) , kullanıcı adı(username), etiket (hashtag), işaretler (- ve /) ve küçük harfe çevirerek veri düzenlenebilir. Örneğin: tweet içerisinde @mesut ifadesi geçiyorken model veri temizlemeden eğitildiği takdirde, modelin yanlış sınıflandırma ihtimali olacaktır ve başarı oranı düşecektir.

```
REPLACE_URL = re.compile('((www\.[^\s]+)|(https?://[^\s]+))')
REPLACE_USERNAME = re.compile('@[^\s]+')
REPLACE_HASHTAG = re.compile(r'#([^\s]+)')
REPLACE_WITH_SPACE = re.compile("(-)|(/)")
REPLACE_NO_SPACE = re.compile("[^a-z' ]+")
```

Şekil 4.10. Veri temizleme için düzenli ifadeler

4.6. Tensorflow Nedir

Google'nin 2015 yılında derin öğrenme işlemlerinde kullanılmak üzere, açık kaynak kodlu makine öğrenmesi kütüphanesidir. Makine öğrenme kütüphanesi olarak da adlandırılan sistem, verilen verilerin kodlanması ve ayırt edilmeleri sağlanmaktadır. Google Brain ekibi tarafından oluşturulan. TensorFlow, bir dizi makine öğrenimini ve derin öğrenmeyi (nöral ağ oluşturma) modelleri ve algoritmaları bir araya getirerek, ortak bir metafor yoluyla faydalı hale getirir. Bu uygulamaları, yüksek performanslı C ++ ile yürütürken, framework ile uygulamalar oluşturmak için kullanışlı bir ön uç API'si sağlamak için Python'u kullanır.

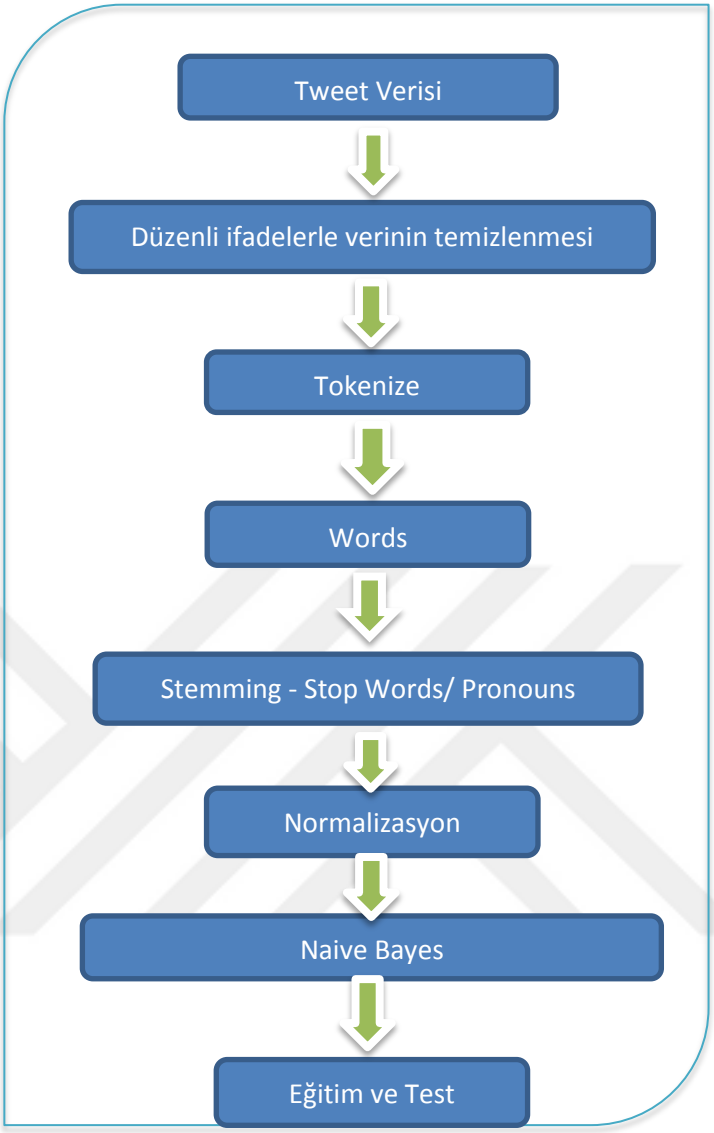
TensorFlow, el yazısı haneli sınıflandırma, görüntü tanıma, kelime toplama, tekrarlayan sinir ağları, makine çevirisi için dizi-sıralı modeller, doğal dil işleme ve PDE (kısmi diferansiyel denklem) tabanlı similasyonlar için derin sinir ağlarını çalıştırabilir ve çalıştırabilir. TensorFlow, eğitim için kullanılan aynı modellerle, ölçekli üretim tahminini destekler. Tensorflow tek bir API ile bir masaüstü, sunucu veya mobil cihazdaki bir veya daha fazla CPU'ya veya GPU'ya dağıtma olanağı sunmaktadır.

4.7. Modelin Oluşturulması

Makine öğrenmesi ile tweetlerden duygu analizi yaparken kullanacağımız yapı şeması Şekil 3.5.'te belirtilmiştir. İki veri setinden kelimeler için ön aşama olarak veri temizlemesi yapılır. Veri temizleme ile url, isim, etiket ayrıştırma ve hepsini küçük harfe çevirme işlemleri uygulanır. Küçük harfe çevirmenin amacı listeye attığımız bu kelimelerin listede tek bir yer kaplamasını isteğinden dolayıdır. Veri temizleme aşaması uygulanmadığı takdirde, model düzgün eğitilemediği için sistemin başarı oranı düşmektedir ve çalışma zamanı artmaktadır. Etkisiz kelimeleri ve zamirleri çıkarınca başarı oranında artış olmaktadır. N gram dil modeline göre cümledeki kelimeler ayrıştırılmaktadır. Örneğin: "Ben Bilgisayar Mühendisliği Öğrencisiyim." Cümlesini "Ben", "Bilgisayar, "Mühendisliği," Öğrencisiyim" şeklinde kelimelere ayrıştırılmaktadır. Benzer kelimeler bulunarak, kelime listesini normalize edilmektedir. Böylece listede daha az ve doğru kelime ile işlem yapılmaktadır. Algoritma, çalışma şeklinden dolayı her kelime için her durumun olasılığını hesaplar ve olasılık değeri en yüksek olana göre sınıflandırır. Model eğitildikten sonra herhangi bir veri setine göre testler yapılabilmektedir.

Tablo 4.2. CNN modelimizde kullanılan parametre değerleri

Parametre Adı	Parametre Değeri
embedding_dim	128
filter_sizes	3
num_filters	128
batch_size	64
num_epochs	200
fc_hidden_size	1024



Şekil 4.11. Önerilen Sistem Modeli

5. SONUÇ VE ÖNERİLER

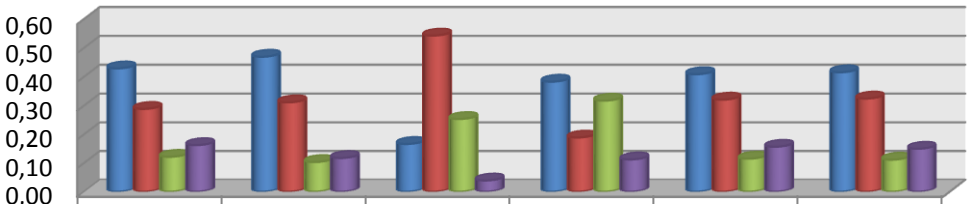
Çalışmamızda ilk olarak Naive Bayes algoritması kullanılmıştır. Yapılan literatür taramasında diğer makine öğrenmesi algoritmaları ile karşılaştırılmada en iyi sonuçları Naive Bayes verdiği için, yapılan çalışmamızın ilk adımında Naive Bayes algoritması tercih edilmiştir. Naive Bayes ile ilk eğitimizde veri setinin %75 lik kısmı eğitim için, kalan %25'lik kısmı test için kullanılmıştır. Sonuçlarını inceleyince %73 başarı oranı sağlanmıştır. Model eğitilirken sonrasında başarı oranını arttırmak için zamirlerin ve isimlerin etkisi kaldırıldı ve kodda iyileştirilmeler yapılmıştır. Kodları dizüstü bilgisayarda çalıştırdığımız için, donanımın yeterli olmadığından kaynaklı çalışma süresi çok fazla olmuştur. Kodlarımızın daha kısa sürede sonucu vermesi için ubuntu işletim sistemine sahip bir sunucu kullanılmıştır. İkinci bağımsız bir veri setinde %80 başarı oranı sağlanmıştır. Karmaşıklık matrisinde (Confusion matrix) ilk ve ikinci yapılan testlerin değerleri normalize edilerek karşılaştırılmıştır. Karşılaştırmayı Tablo 4.2.'de sayısal ve grafik olarak görebilirsiniz. Karmaşıklık matrisinde bulunan TP kısaltması gerçek değeri ile tahmin ettiğimiz değeri pozitif olanı ifade etmektedir. TP (True Pozitif) değerine baktığımızda doğru bilme oranı artmıştır. İlk veri setinde doğru tahmin etme yaklaşık 0,43 iken, ikinci veri setinde 0,47'ye yükselmiştir. TN (True Negatif) gerçek değeri ile tahmin ettiğimiz değeri olumsuz olanları ifade etmektedir. TN Değerine bakınca ilk veri setinde yaklaşık 0,29 iken, ikinci veri setinde 0,31 olmuştur. Her iki başarı ile tahmin etme de oran daha yükselmiştir. FN (False Negatif) gerçek değeri pozitif olan ve tahmin ettiğimiz değeri negatif olanları ifade etmektedir. Yanlış tahmin etmesinde FN değerine bakınca ilk veri setinde 0,12 iken, ikinci veri setinde 0,10 olarak azalmıştır. FP (False Pozitif) gerçek değeri negatif iken bizim tahmin ettiğimiz pozitif olanları ifade etmektedir. FP değerine bakınca 0,16'dan 0,11'e azalmıştır. Doğruluk (Accuracy) genel olarak, sınıflayıcının ne sıklıkta doğru tahmin ettiğinin bir ölçüsüdür. Tablo 5.1.'te görüldüğü üzere, başarı oranında artış gözlemlenmiştir. Başarı oranında doğruluk değerini incelediğimizde ilk veri seti ile test ettiğimizde yaklaşık %72 sonuç bulunmuştur. İkinci veri seti ile yapılan testte %78 başarı oranı bulunmuştur. Bakınca %6 oranında doğruluk değerinin arttığını görebiliyoruz. Tutarlılık

(Precision) tüm sınıflardan, doğru olarak ne kadar tahmin edildiğinin bir ölçüsüdür. Mümkün olduğu kadar yüksek olmalıdır. Pozitif tahmin edici değer olarak da bilinir. Sonuçlar incelendiğinde Eğitim veri setinde %73 tahmin oranı bulunurken, ikinci veri seti ile test ettiğimizde değer %80'e ulaşmıştır. Yaklaşık %7 artış olmuştur.

Farklı veri setinde modelin başarısını gözlemlemek için, 2 adet yeni test veri seti ile çalışmalara devam edilmiştir. Test için kullandığımız hava yolu firması için atılan tweetlerden oluşan veri setinden %70 doğruluk, %81 oranında tutarlılık değerleri bulunmuştur. Veri setinde olumlu tweet sayısı az olduğu için TP değerinin az, TN değerinin ise fazla olduğu gözlemlenmiştir. Olumsuz tweet sayısının fazla olduğundan, FP değerinde bir artış olmuştur. Diğer sonuçlarla kıyasladığımızda, bu veri setinde tahmin oranının en yüksek olduğu görülmektedir. Test için kullandığımız 3. Veri seti ile yapılan çalışmada olumlu ve olumsuz tweet sayıları birbirine yakın olduğu görülmektedir. Karmaşıklık matrisi incelendiğinde 0,38 ile diğer 3 testten düşük olduğu, TN değerinin ise en düşük olduğu gözlemlenmiştir. FP değeri 0,11 ile diğer testlerdeki sonuçlara göre gerçek değeri negatif iken bizim pozitif olarak tahmin ettiği en düşük sonuç bulunmuştur. 0,31 FN değerinde, diğer testlerdeki karşılaştığımız en yüksek değer görülmüştür. Olumlu tweetleri, olumsuz olarak gruplandırdığı en yüksek değer görülmüştür. Sonuçlarında ise, %57 doğruluk oranıyla diğer veri setlerinden düşük sonuç bulunmuştur. Tahmin oranında ise %77 başarı oranı görülmektedir. 4.Testte; modeli test için kullanılan 1.veri setinde bulunan ilk 50.000 tweet ile eğitildi. Veri setinde tweet sayısı fazla olduğu için veri setinin tamamı eğitimde kullanılamadı. Test için ise önceki çalışmalarda kullanılan, eğitim veri seti kullanılmıştır. Karmaşıklık matrisi ve başarı oranı incelendiğinde, değerlerin eğitim verisi ile yapılan ilk teste yakın olduğu gözlemlenmiştir. Başarı oranını arttırmak için eğitimde kullanılan tweet sayısı 100.000 olarak güncellenmiştir ve 5.test yapılmıştır. Doğruluk ve tahmin değeri %74 olduğu gözlemlenmiştir. 4.Test ile arasında %2 değerinde bir başarı artışı olduğu görülmüştür.

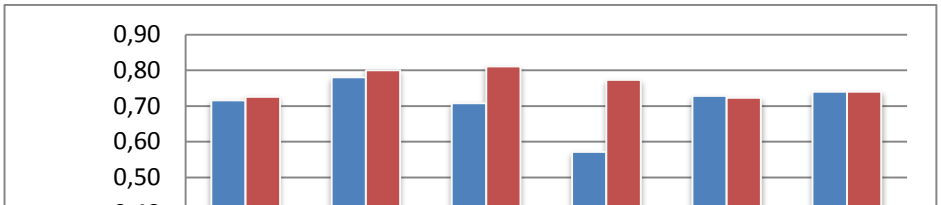
Naive Bayes ile yapılan çalışmada modeli geliştirerek, başarı oranında artış sağlandığı gözlemlenmiştir. Diğer yapılan çalışmalar ile karşılaştırıldığında Akgül vd. Yaptığı çalışmada %70 başarı oranı, Onan (2017) yaptığı çalışmasında %76, Şeker ve Yeşilyurt (2017) yaptığı çalışmada %52, Gamal vd. (2018)'de %74 başarı oranı bulunmuştur. Oluşturduğumuz modelimizdeki başarı sonucu diğer çalışmalardan daha yüksek başarı oranı vermiştir.

Tablo 5.1. Confusion Matrix



	Eğitim Veri Seti	1. Test	2. Test	3. Test	4. Test	5.Test
TP	0,429	0,469	0,166	0,383	0,408	0,415
TN	0,288	0,311	0,542	0,188	0,321	0,324
FN	0,121	0,103	0,253	0,317	0,115	0,111
FP	0,162	0,117	0,039	0,112	0,156	0,149

Tablo 5.2. Başarı Oranı

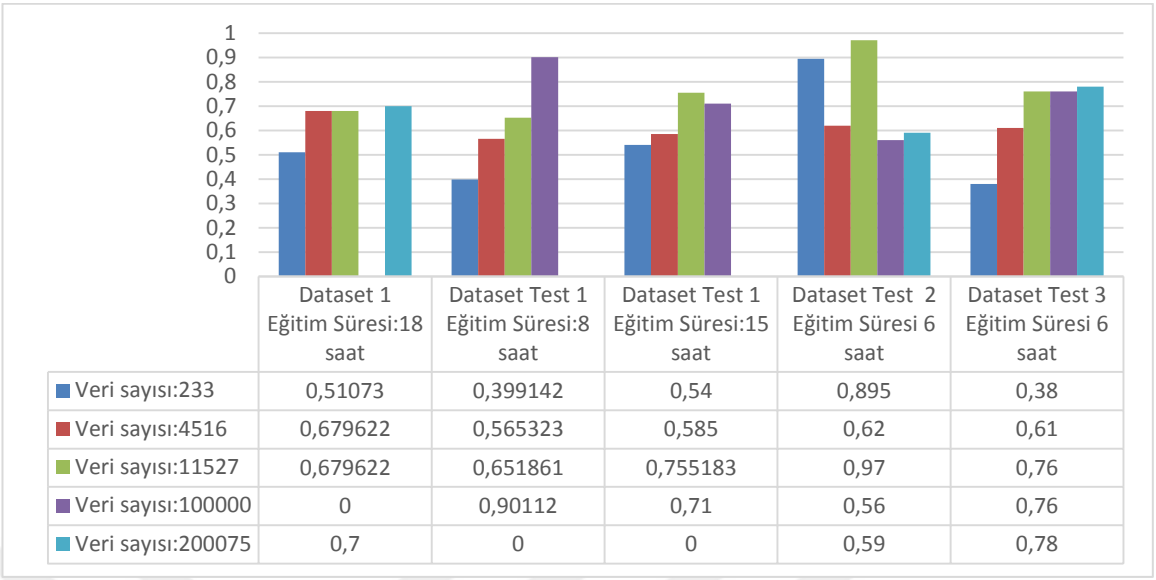


	Eğitim Veri Seti	1. Test	2. Test	3. Test	4. Test	5.Test
Accuracy	0,716	0,78	0,708	0,571	0,728	0,74
Precision	0,7257	0,8003	0,8109	0,773	0,723	0,74

Klasik makine öğrenmesi algoritmasıyla bulduğumuz sonuçları, aynı veri seti derin öğrenme ile testler yapılmıştır. Chen vd. (2016), yaptıkları çalışmada klasik makine öğrenmesi algoritmaları ve cnn ile çalışma yaptı. Yaptığı çalışmada Cnn klasik makine öğrenmeleri algoritmalarına göre çalışmada en iyi

sonucu verdiğini belirtmiştir. Yapılan literatür taramasında derin öğrenme yöntemleri arasından en çok konvolüsyonel nöral ağların (CNN) tercih edildiği görülmüştür. İlk aşamada 100.000 verinin bulunduğu veri setimiz ile modelimiz 18 saat eğitilmiştir. Eğitilen model ile testler yapılmaya başlandığında veri setinde bulunan veri miktarı ne kadar fazla ise, doğruluk oranı daha yüksek olduğu gözlemlenmiştir. 1. Test veri setinden alınan 200.000 verinin olduğu tweet veri seti ile model test edildiğinde %70 başarı oranı olduğu görülmektedir. Diğer yapılan testlerde oranın daha düşük olduğu Tablo 5.2.'te görülmektedir. 1.Test veri seti ile model 8 saat eğitilmiştir. İlk aşamada kullanılan eğitim veri seti ile test edilmiştir ve %90 başarı oranı verilmiştir. Diğer veri setleri ile yapılan testlerde ise veri miktarı ile başarı oranının değiştiği görülmüştür. Test veri setinde bulunan veri miktarı azaldığı zaman başarı oranınında azaldığı Tablo 5.2.'te yer almaktadır. Modelin daha iyi eğitilmesi için 15 saat eğitim verilmiştir. Eğitimin sonunda model veri kümeleri ile test edildiğinde başarı oranın arttığı görülmüştür. 8 saat eğitimin sonunda ki model ile 15 saat eğitilmiş model arasında; eğitim veri setinde bir %19 oranında bir azalma olmuştur ve %71 olarak sonuçlanmıştır. Test veri seti 2 ile modeli eğitirken, aynı veri setinin bir bölümü ile yapılan testlerde %97 başarı oranı gözlemlendi. Veri setinde olumsuz verilerin ağırlıklı olması ve sadece bir kategoriye ait olan mesajlardan dolayı eğitimdeki başarı oranı yüksek olmuştur. Veri sayısı az olan veri setinde %89 başarı oranı bulunurken, veri sayısı fazla olan diğer iki veri setinde yapılan testlerde %62, %59 ve %56 başarı oranı bulunmuştur. 3. Veri seti setimizde bulunan veri sayımızın yüksek olmasından dolayı ve modeli eğitirken yaptığımız testlerde %80,5 başarı oranı bulunmuştur. Farklı veri setlerimiz ile yaptığımız testlerde %78, %78, %76, %61 ve %38 başarı oranı bulunmuştur. 3. Veri setimiz ile modeli eğittimizde en iyi sonucu bulduğumuz gözlemlenmiştir. Sonuçları incelediğimizde farklı veri setlerine göre başarı oranının etkisindeki değişmenin çok az olduğu görülmüştür. Diğer testlerde bu başarı oranındaki değişmelerin çok fazla olduğu görülmektedir. Modeli eğitirken veri kümeside bulunan verilerin temizlenmesinin ve veri sayısının yüksek olmasının başarının yükselmesinde önemli bir etken olduğu görülmüştür. Az veri seti ile yapılan testlerde ise, başarı oranının çok düşük olduğu görülmüştür.

Tablo 5.3. CNN ile başarı oranı verileri



	Dataset 1 Eğitim Süresi:18 saat	Dataset Test 1 Eğitim Süresi:8 saat	Dataset Test 1 Eğitim Süresi:15 saat	Dataset Test 2 Eğitim Süresi 6 saat	Dataset Test 3 Eğitim Süresi 6 saat
Veri sayısı:233	0,51073	0,399142	0,54	0,895	0,38
Veri sayısı:4516	0,679622	0,565323	0,585	0,62	0,61
Veri sayısı:11527	0,679622	0,651861	0,755183	0,97	0,76
Veri sayısı:100000	0	0,90112	0,71	0,56	0,76
Veri sayısı:200075	0,7	0	0	0,59	0,78

Naive Bayes ile konvolüsyonel nöral ağlarının ürettiği sonuçları karşılaştırdığımızda en iyi sonucu konvolüsyonel nöral ağlarının verdiği görülmektedir.

KAYNAKLAR

Aizenberg I., 2019. Erişim Tarihi: 09.05.2019. https://www.import.io/post/history-of-deep-learning/ .

Akgül E. S., Ertano C., Diri B., 2016. Twitter Verileri ile Duygu Analizi, Pamukkale üniversitesi Mühendislik Bilimleri Dergisi, 22(2), 106-110.

Alkowaileet W., Alsubaiee S., Carey M., Li C., Ramampiaro H., Sinthong P., Wang X., 2018. End-to-End Machine Learning with Apache AsterixDB, DEEM'18 Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning.

Aydın İ., Salur M. U., Başkaya F., 2018. Duygu Analizi İçin Çoklu Popülasyon Tabanlı Parçacık Sürü Optimizasyonu, Türkiye Bilişim Vakfı Bilgisayar Bölümleri ve Mühendisleri Dergisi, 11, 52-64.

Ayhan S., Erdoğmuş Ş., 2014. Destek Vektör Makineleriyle Sınıflandırma Problemlerinin Çözümü İçin Çekirdek Fonksiyonu Seçimi, Eskişehir Osmangazi Üniversitesi İibf Dergisi, 9(1), 175-201.

Bari A., Saatçioğlu G., 2018. Emotion Artificial Intelligence Derived from Ensemble Learning, 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering.

Chen X., 2018. Sentiment Analysis Using Convolutional Neural Network and Modal Distribution Removal, Research Scholl of Computer Science Australian National University.

Chen T., Xu R., HE Y., Wang X., 2016. Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN, Elsevier, 72, 221-230.

Ciresan D., Meier U., Schmidhuber J., 2012. Flexible, High Performance Convolutional Neural Networks for Image Classification, in Proceedings of the TwentySecond international joint conference on Artificial Intelligence, 1, 1237-1242.

Çoban Ö., Özyer-Tümüklü G., 2018. Twitter duygu analizinde terim ağırlıklandırma yönteminin etkisi, Pamukkale üniversitesi Mühendislik Bilimleri Dergisi 24(2), 283-291.

Gao J., Deng L., Gamon M., He X., 2014. Modeling interestingness with deep neural networks, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar.

Graves A., Schmidhuber J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Networks, 18(5), 602–610.

Hartmann J., Huppertz J., Schamp C., Heitmann M., 2019. Comparing automated text classification methods, International Journal of Research in Marketing, 36(1), 20-38.

Heyong W., Ming H., 2019. Supervised Hebb rule based feature selection for text classification, Information Processingand Management, 56(1), 167-191.

Hinton E. G., Osindero S., Teh Y.W., 2006, A Fast Learning Algorithm for Deep Belief Nets, Neural Computation 18(7), 1527-1554.

Hinton G. E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R. R., 2014. Improving neural networks by preventing co-adaptation of feature detectors, Toronto University, arXiv preprint arXiv:1207.0580.

İnik Ö., Ülker E., 2017. Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri, Gaziosmanpaşa Bilimsel Araştırma Dergisi, 6(3), 85-104.

Kaban Z., Diri B., 2008. Yapay Bağışıklık Sistemleri ile Türkçe Metinlerde Tür ve Yazar Tanıma, IEEE, 1-4.

Kassraie P., Modirshanechi A., Aghajan H., 2017. Election Vote Share Prediction using a Sentiment-based Fusion of Twitter Data with Google Trends and Online Polls, In Proceedings of the 6th International Conference on Data Science, 363-370.

Kocich D., 2017. Multilingual Sentiment Mapping Using Twitter, Open Source Tools, and Dictionary Based Machine Translation Approach, Symposium GIS Ostrava, Springer, 223-238.

Kravvaris D., Kermanidis K. L., 2016. Opinion Mining for Educational Video Lectures, 235-243.

Larsson M., Nılsson A., 2017. Manifold Traversal for Reversing the Sentiment of Text, Chalmers University of Technology and University of Gothenburg, Swedish.

Lojistik Regresyon nedir, 2019. Erişim Tarihi:11.05.2019. https://veribilimcisi.com/2017/07/18/lojistik-regresyon/

Makine Öğrenimi, 2019. Erişim Tarihi: 01.05.2019. https://tr.wikipedia.org/wiki/Makine_%C3%B6%C4%9Frenimi

Moussa M. E., Mohamed E., Haggag M. H., 2018. A generic lexicon-based framework for sentiment analysis, International Journal of Computers and Applications, 1-11.

Onan A., 2017, Twitter Mesajları Üzerinde Makine Öğrenmesi Yöntemlerine Dayalı Duygu Analizi, Yönetim Bilişim Sistemleri Dergisi, 3(2), 1-14.

Onan A., Kurukoglu S., 2016. Makine öğrenmesi yöntemlerinin görüş madenciliğinde kullanılması üzerine bir literatür araştırması, Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 22(2), 111-122.

Regresyon nedir, 2019. Erişim Tarihi:11.05.2019. https://veribilimcisi.com/2017/07/13/basit-lineer-regresyon-nedir/

Ribeiro M.T., Singh S., Guestrin C., 2018. Semantically Equivalent Adversarial Rules for Debugging NLP models, Association for Computational Linguistics, 1, 856-865.

Salakhutdinov R., Hinton G., Semantic Hashing, 2019. International Journal of Approximate Reasoning, 50, 969– 978.

Salakhutdinov R., Mnih A., Hinton G., 2007. Restricted Boltzmann machines for collaborative filtering, in Proceedings of the 24th international conference on Machine learning.

Sboev, T. Litvinova, D. Gudovskikh, R. Rybka, and I. Moloshnikov, 2016. Machine Learning Models of Text Categorization by Author Gender Using Topic-independent Features, Procedia Comput. Sci., 101, 135–142

Singh P., Sawhney R. S., Kahlon S. K., 2017. Forecasting the 2016 US Presidential Elections Using Sentiment Analysis, Digital Nations – Smart Cities, Innovation, and Sustainability, 1, 412-423.

Şeker A., Diri B., Balık H. H., 2017. Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme, Gazi Mühendislik Bilimleri Dergisi, 3(3), 47-64.

Şeker, Ş. E., Yeşilyurt A., 2017. Metin Madenciliği Yöntemleri ile Twitter Duygu Analizi, YBS Ansiklopedisi, 2(3), 30-32.

Tantuğ A. C., 2012. Metin Sınıflandırma, Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 5(2), 1-12.

Taşcı E., Onan A., 2014. K-En Yakın Komşu Algoritması Parametrelerinin Sınıflandırma Performansı Üzerine Etkisinin İncelenmesi, Akademik Bilişim Konferansları.

Tongman S., Wattanakitrungroj N., 2018. Classifying Positive or Negative Text Using Features Based on Opinion Words and Term Frequency- Inverse Document Frequency, 2018 5th International Conference on Advanced Informatics: Concept Theory and Applications.

Wang H., Hong M. Lau R. Y. K., 2018. Utility-based feature selection for text classification, Knowledge and Information Systems 2018, 1-30.

Wang W., 2017. Event Detection and Encoding from News Articles, Virginia Polytechnic Institute and State University, Arlington, Virginia.

Wang W., Ning Y., Rangwala H., Ramakrishnan N., 2016. A Multiple Instance Learning Framework for Identifying Key Sentences and Detecting Events, CIKM '16 Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.

Wijayanto U. W., Sarno R., 2018. An Experimental Study of Supervised Sentiment Analysis Using Gaussian Naïve Bayes, 2018 International Seminar on Application for Technology of Information and Communication, 476-481.

Sınırlı Boltzmann Makineleri, 2019. Erişim Tarihi: 12.05.2019. https://upload.wikimedia.org/wikipedia/commons/thumb/e/e8/Restricted_Boltzmann_machine.svg/1200px-Restricted_Boltzmann_machine.svg.png

Doğal dil işleme, 2019. Erişim Tarihi:12.05.2019. https://tr.wikipedia.org/wiki/Do%C4%9Fal_dil_i%C5%9Fleme

Kaggle Nedir, 2019. Erişim Tarihi:12.05.2019. https://en.wikipedia.org/wiki/Kaggle

Düzenli ifadeler, 2019. Erişim Tarihi:12.05.2019. https://tr.wikipedia.org/wiki/D%C3%BCzenli_ifade

Veri kümesi: https://www.kaggle.com/c/twitter-sentiment-analysis2/data
Veri kümesi: https://www.kaggle.com/arjunrao2000/baseline-keras-model/data

Veri kümesi: https://www.kaggle.com/crowdflower/twitter-airline-sentiment#Tweets.csv

Veri kümesi: https://www.kaggle.com/crowdflower/first-gop-debate-twitter-sentiment/downloads/first-gop-debate-twitter-sentiment.zip/2

Naive Bayes Kaynak Kodu

```
# coding=utf-8
import csv
import re

from random import shuffle

import multiprocessing
from Levenshtein._levenshtein import ratio, distance

REPLACE_URL = re.compile('((www\.[^\s]+)|(https?://[^\s]+))')
REPLACE_USERNAME = re.compile('@[^\s]+')
REPLACE_HASHTAG = re.compile(r'#([^\s]+)')
REPLACE_WITH_SPACE = re.compile("(-)|(/)")
REPLACE_NO_SPACE = re.compile("[^a-z' ]+")

TOKENIZE = re.compile(r'([a-z])\1\1+')

STOP_WORDS = ['ourselves', 'hers', 'between', 'yourself', 'but',
'again', 'there', 'about', 'once', 'during', 'out',
              'very', 'having', 'with', 'they', 'own', 'an', 'be',
'some', 'for', 'do', 'its', 'yours', 'such', 'into',
              'of', 'most', 'itself', 'other', 'off', 'is', 's',
'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the',
              'themselves', 'until', 'below', 'are', 'we', 'these',
'your', 'his', 'through', 'don', 'nor', 'me',
              'were', 'her', 'more', 'himself', 'this', 'down',
'should', 'our', 'their', 'while', 'above', 'both',
              'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when',
'at', 'any', 'before', 'them', 'same', 'and',
              'been', 'have', 'in', 'will', 'on', 'does',
'yourselves', 'then', 'that', 'because', 'what', 'over',
              'why', 'so', 'can', 'did', 'not', 'now', 'under',
'he', 'you', 'herself', 'has', 'just', 'where', 'too',
              'only', 'myself', 'which', 'those', 'i', 'after',
'few', 'whom', 't', 'being', 'if', 'theirs', 'my',
              'against', 'a', 'by', 'doing', 'it', 'how', 'further',
'was', 'here', 'than']

PRONOUNS = [
"it",
"i",
"you",
"he",
"they",
"we",
"she",
"who",
"them",
"me",
"him",
"one",
"her",
"us",
"something",
```

```
"nothing",
"anything",
"himself",
"everything",
"someone",
"themselves",
"everyone",
"itself",
"anyone",
"myself"
]



def tokenize(s):
    # return "".join(OrderedDict.fromkeys(s))
    return re.sub(TOKENIZE, r'\1\1', s)


def pre_process(text):
    text = text.lower()
    text = re.sub(r'&quot;', '', text)
    text = re.sub(REPLACE_URL, 'URL', text)
    text = re.sub(REPLACE_USERNAME, 'AT_USER', text)
    text = re.sub(REPLACE_HASHTAG, r'\1', text)

    text = re.sub(';\)', 'HAPPY_SMILEY', text);

    text = re.sub(REPLACE_WITH_SPACE, ' ', text)
    text = re.sub(REPLACE_NO_SPACE, '', text)
    return text


classes = ['0', '1']
data = []
data_test = []

with open('data/train2.csv', 'rb') as f:
    reader = csv.reader(f)
    csv_data = list(reader)
    shuffle(csv_data)
    csv_data = csv_data[:400000]

NAME = []
with open('data/name.csv', 'rb') as f:
    reader = csv.reader(f)
    for item in list(reader):
                NAME.append(tokenize(pre_process(item[0].lower())))


with open('data/train.csv', 'rb') as f:
    reader = csv.reader(f)
    csv_data2 = list(reader)
    shuffle(csv_data2)
   #csv_data2 = csv_data2[:1000]

print 'Starting to load data...'

# shuffle(csv_data)
# csv_data += csv_data[random.randint(0, (len(csv_data) - 1) /
2):random.randint(((len(csv_data) - 1) / 2), len(csv_data) - 1)]
```

```
# csv_data = csv_data[:2000]

for i in xrange(len(csv_data)):
    item = csv_data[i]
    if item[0] in classes:
        __text = item[5]
        if len(__text) > 100:
            data.append({"class": item[0], "text":
pre_process(__text)})

shuffle(csv_data2)
for i in xrange(len(csv_data2)):
    item = csv_data2[i]
    if item[1] in classes:
        data_test.append({"class": item[1], "text":
pre_process(item[2])})


# for i in xrange(100000):
#     item = csv_data2[i]
#     if item[0] in classes:
#         __text = item[5]
#         if len(__text) > 100:
#             data.append({"class": item[0], "text":
pre_process(__text)})
#
# test_index = i
# for i in xrange(test_index, test_index + 2000):
#     item = csv_data2[i]
#     if item[0] in classes:
#         __text = item[5]
#         data_test.append({"class": item[0], "text":
pre_process(__text)})

print 'Train data: {0} | Test data: {1}'.format(len(data),
len(data_test))


print 'Starting process...'
vocabulary = []
class_t_verbs = {}
verbs_freq = {}

for item in classes:
    class_t_verbs[item] = dict()

for item in data:
    words = pre_process(item['text']).split(' ')

    for word in words:
        word = tokenize(word)
        if len(word) > 0 and word not in STOP_WORDS and word not in
PRONOUNS and word not in NAME:
            class_t_verbs[item['class']][word] =
class_t_verbs[item['class']].get(word, 0) + 1
            verbs_freq[word] = verbs_freq.get(word, 0) + 1

            if word not in vocabulary:
                vocabulary.append(word)
```

```
# print vocabulary
# print class_t_verbs
print 'Vocabulary count : {0}'.format(len(vocabulary))

# remove_keys = {}
# for class_name, class_verbs in class_t_verbs.items():
#     for key, val in class_verbs.items():
#         if val < 2:
#             class_t_verbs[class_name].pop(key, None)
#             remove_keys[key] = remove_keys.get(key, 0) + 1
#
# for key, val in remove_keys.items():
#     if val == len(classes) and key in vocabulary:
#         vocabulary.remove(key)

# for p in PRONOUNS:
#     for verb in vocabulary:
#         if len(p) > 0 and len(verb) > 0:
#             if ratio(p, verb) > 0.83 or (p[0] == verb[0] and
(ratio(p, verb) > 0.65 or (len(p) == len(verb) and distance(p, verb)
< 3))):
#                 print p, '-', verb
#                 vocabulary.remove(verb)
#
#                 for c in classes:
#                     if class_t_verbs[c].get(verb):
#                         class_t_verbs[c].pop(verb)

selected_verbs = []

def clear_similars():
    similars = 0
    shuffle(vocabulary)
    for key1 in vocabulary[:1500]:
        for key2 in vocabulary:
            if key1 != key2:
                if ratio(key1, key2) > 0.83 or (len(key1) > 4 and
key1[0] == key2[0] and distance(key1, key2) == 1):
                    # print key1, verbs_freq[key1], ' -> ',
verbs_freq[key2], key2
                    similars += 1

                    if verbs_freq[key1] > verbs_freq[key2]:
                        selected_verbs.append(key1)

                        verbs_freq[key1] += verbs_freq[key2]
                        if key2 in vocabulary:
                            vocabulary.remove(key2)

                            for c in classes:
                                verbs = class_t_verbs[c]
                                f1 = verbs.get(key1, 0)
                                f2 = verbs.get(key2, 0)

                                if f1 > 0:
                                    class_t_verbs[c][key1] = f1 + f2
                                if f2 > 0:
                                    class_t_verbs[c].pop(key2)
```

```
                    elif verbs_freq[key1] < verbs_freq[key2]:
                        selected_verbs.append(key2)

                        verbs_freq[key2] += verbs_freq[key1]
                        if key1 in vocabulary:
                            vocabulary.remove(key1)

                            for c in classes:
                                verbs = class_t_verbs[c]
                                f1 = verbs.get(key1, 0)
                                f2 = verbs.get(key2, 0)

                                if f2 > 0:
                                    class_t_verbs[c][key2] = f1 + f2
                                if f1 > 0:
                                    class_t_verbs[c].pop(key1)

    return similars

# x = True
# while x:
#     sim = clear_similars()
#     print sim
#     if sim < 300:
#         x = False

# print class_t_verbs
print 'Vocabulary count (after cleaning) :
{0}'.format(len(vocabulary))

selected_verbs = list(set(selected_verbs))
print selected_verbs


# STEP 1: Prior Probability
#   P(ham)  = Nham / (Nham+Nspam) = 2 / (2+3) = 0.40
#   P(spam) = Nspam / (Nham+Nspam) = 3 / (2+3) = 0.60

prior_probability = {}
for __class in classes:
    prior_probability[__class] = 1.0 * len([item for item in data if
item['class'] == __class]) / len(data)

print 'STEP 1: Prior Probability'
print prior_probability

# STEP 2: Constructing the Unigram Language Model
#   Add-One Smoothing
#   P(very|ham) = (Tvery|ham+1) / ((Tvery|ham+1) + (Tgood|ham+1) +
(Tbad|ham+1))

unigram_modal = {}


def unigram_worker(args):
    __class = args[0]
    __unigram_modal = dict()

    for verb in vocabulary:

        others = 1.0
```

```
        for verb2 in vocabulary:
            others += (class_t_verbs[__class].get(verb2, 0) + 1.0)

        __unigram_modal[verb] = (class_t_verbs[__class].get(verb, 0)
+ 1.0) / others

    print '  Completed unigram modal for class:{0}, items:
{1}'.format(__class, __unigram_modal.items())
    return {__class: __unigram_modal}

worker_data = ()
worker_pool = multiprocessing.Pool(processes=8)
for __class in classes:
    worker_data += ((__class, ),)

worker_results = ((),)
worker_results = worker_pool.map(unigram_worker, worker_data)

for item in worker_results:
    unigram_modal = dict(item.items() + unigram_modal.items())


print 'STEP 2: Constructing the Unigram Language Model'
print unigram_modal

# TESTING --------
true_positive = 0
true_negative = 0
false_positive = 0
false_negative = 0

for test_item in data_test:
    test_sentiment = test_item['class']
    test_data_verbs = test_item['text'].split(' ')

    new_verbs = []
    for verb in test_data_verbs:
        if len(verb) > 0 and verb not in STOP_WORDS and verb not in
PRONOUNS:
            verb = tokenize(verb)
            __s = None
            for similar in selected_verbs:
                if ratio(verb, similar) > 0.8:
                    __s = similar

            if __s:
                new_verbs.append(__s)
            else:
                new_verbs.append(verb)


    test_data_verbs = new_verbs


    # STEP 3: Likelihood
    #   Applying the Unigram Language Model
    #   P(d6|ham) = P(good|ham)×P(bad|ham)×P(very|ham)×P(bad|ham)

    likelihood = {}
    for __class in classes:
```

```
            __like = 1
            __has_verb = False
            for verb in test_data_verbs:
                verb = tokenize(verb)
                if unigram_modal[__class].get(verb):
                    __has_verb = True
                    __like *= unigram_modal[__class].get(verb)

            if __has_verb:
                likelihood[__class] = __like

        print 'STEP 3: Likelihood'
        print likelihood
        print test_item['text']

        # STEP 4: Posterior Probability
        #   P(ham| d6) = P(d6|ham)×P(ham)/P(d6)

        __sentiment = ''
        __sentiment_rank = 0.0
        for __class in classes:
            rank = likelihood.get(__class, 0.0) *
prior_probability[__class]
            if rank > __sentiment_rank:
                __sentiment_rank = rank
                __sentiment = __class

            print '{0} -> {1}'.format(__class, likelihood.get(__class,
0.0) * prior_probability[__class])

        print test_sentiment, __sentiment

        if test_sentiment == '1':
            if __sentiment == '1':
                true_positive += 1
            else:
                false_positive += 1
        else:
            if __sentiment == '0':
                true_negative += 1
            else:
                false_negative += 1

    print 'Test n={0}'.format(len(data_test))

    print 'Accuracy: {0}'.format(1.0 * (true_positive + true_negative) /
(true_positive + true_negative + false_negative + false_positive))
    print 'Precision: {0}'.format(1.0 * true_positive / (true_positive +
false_positive))

    print 'Confusion Matrix:\nTP:{0} FN:{1}\nFP:{2} TN:{3}
'.format(true_positive, false_negative, false_positive,
true_negative)
```

Konvolüsyonel sinir ağları kodu

Data_helper.py

```
import numpy as np
import re


def clean_str(string):
    string = re.sub(r"@[^\s]+", "", string)
    string = re.sub(r"#([^\s]+)", " ", string)
    string = re.sub(r"((www\.[^\s]+)|(https?://[^\s]+))", "",
string)
    string = re.sub(r"(-)|(/)", " ", string)
    string = re.sub(r"[^A-Za-z0-9(),!?\'\`]", " ", string)
    string = re.sub(r"\'s", " \'s", string)
    string = re.sub(r"\'ve", " \'ve", string)
    string = re.sub(r"n\'t", " n\'t", string)
    string = re.sub(r"\'re", " \'re", string)
    string = re.sub(r"\'d", " \'d", string)
    string = re.sub(r"\'ll", " \'ll", string)
    string = re.sub(r",", " , ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"\(", " \( ", string)
    string = re.sub(r"\)", " \) ", string)
    string = re.sub(r"\?", " \? ", string)
    string = re.sub(r"\s{2,}", " ", string)

    return string.strip().lower()

STOP_WORDS = ['ourselves', 'hers', 'between', 'yourself', 'but',
'again', 'there', 'about', 'once', 'during', 'out',
              'very', 'having', 'with', 'they', 'own',   'some',
'for',  'its', 'yours', 'such', 'into',
             'most', 'itself', 'other', 'off', 'who',  'from',
'him', 'each', 'the',
              'themselves', 'until', 'below', 'are', 'we', 'these',
'your', 'his', 'through', 'don', 'nor',
              'were', 'her', 'more', 'himself', 'this', 'down',
'should', 'our', 'their', 'while', 'above', 'both',
               'ours', 'had', 'she', 'all',   'when',   'any',
'before', 'them', 'same', 'and',
              'been', 'have',  'will',  'does', 'yourselves',
'then', 'that', 'because', 'what', 'over',
              'why',  'can', 'did', 'not', 'now', 'under', 'you',
'herself', 'has', 'just', 'where', 'too',
              'only', 'myself', 'which', 'those', 'after', 'few',
'whom',  'being',  'theirs', 'my',
              'against',  'by', 'doing','how', 'further', 'was',
'here', 'than']

PRONOUNS = [
"you",
"they",
"she",
"who",
"them",
"him",
"one",
"her",
```

```
"something",
"nothing",
"anything",
"himself",
"everything",
"someone",
"themselves",
"everyone",
"itself",
"anyone",
"myself"
]



def load_data_and_labels(data_file):
    """
    Loads MR polarity data from files, splits the data into words
and generates labels.
    Returns split sentences and labels.
    """
    # Load data from files

    positive_examples = []
    negative_examples = []
    positive_examples_temp = []
    negative_examples_temp = []
    positive_examples_temp1 = []
    negative_examples_temp1 = []
    with open(data_file, "rb") as f:
        for line in f:
            if line.decode().split(",")[1] == '1':
                x=line.decode().split(",")[2].split('\n')[0]
                positive_examples_temp.append(x)

            if line.decode().split(",")[1] == '0':
                    y=line.decode().split(",")[2].split('\n')[0]
                    negative_examples_temp.append(y)

    print("Aşama 1")

    for words in positive_examples_temp:
        x="";
        x=words;
        for xx in PRONOUNS:
               x=x.replace(xx,"")
        if x not in positive_examples_temp1:
              positive_examples_temp1.append(x)

    for words in negative_examples_temp:
        x="";
        x=words;
        for xx in PRONOUNS:
                x=x.replace(xx,"")
        if x not in negative_examples_temp1:
            negative_examples_temp1.append(x)

    print("Aşama 1.1")
    for words in positive_examples_temp1:
        x="";
```

```
            x=words;
            for xx in STOP_WORDS:
                   x=x.replace(xx,"")
            if x not in positive_examples:
                positive_examples.append(x)

        for words in negative_examples_temp1:
            x="";
            x=words;
            for xx in STOP_WORDS:
                    x=x.replace(xx,"")
            if x not in negative_examples:
                negative_examples.append(x)
        print("Aşama 1.2")
        # Split by words
        x_text = positive_examples + negative_examples
       # print(x_text)
       # for words in STOP_WORDS:
            #y=x_text.replace(words,"")

         #for word in PRONOUNS:
            #  x_text=string.replace(word, "",x_text)

        x_text = [clean_str(sent) for sent in x_text]
        x_text = [clean_str(sent) for sent in x_text]


        print("Aşama 2")

        # Generate labels
        positive_labels = [[0, 1] for _ in positive_examples]

        negative_labels = [[1, 0] for _ in negative_examples]
        y = np.concatenate([positive_labels, negative_labels], 0)



        return [x_text, y]

    def batch_iter(data, batch_size, num_epochs, shuffle=True):
        """
        Generates a batch iterator for a dataset.
        """
        data = np.array(data)
        data_size = len(data)
        num_batches_per_epoch = int((len(data)-1)/batch_size) + 1
        for epoch in range(num_epochs):
            # Shuffle the data at each epoch
            if shuffle:
                shuffle_indices =
    np.random.permutation(np.arange(data_size))
                shuffled_data = data[shuffle_indices]
            else:
                shuffled_data = data
            for batch_num in range(num_batches_per_epoch):
                start_index = batch_num * batch_size
                end_index = min((batch_num + 1) * batch_size, data_size)
                yield shuffled_data[start_index:end_index]
```

Text_CNN.py

```
import tensorflow as tf
import numpy as np


class TextCNN(object):
    """
    A CNN for text classification.
    Uses an embedding layer, followed by a convolutional, max-
pooling and softmax layer.
    """
    def __init__(
      self, sequence_length, num_classes, vocab_size,
      embedding_size, filter_sizes, num_filters,fc_hidden_size,
l2_reg_lambda=0.0):

        # Placeholders for input, output and dropout
        self.input_x = tf.placeholder(tf.int32, [None,
sequence_length], name="input_x")
        self.input_y = tf.placeholder(tf.float32, [None,
num_classes], name="input_y")
        self.dropout_keep_prob = tf.placeholder(tf.float32,
name="dropout_keep_prob")
        fc_hidden_size=1024
        self.is_training = tf.placeholder(tf.bool,
name="is_training")
        # Keeping track of l2 regularization loss (optional)
        l2_loss = tf.constant(0.0)
        def _highway_layer(input_, size, num_layers=1, bias=-2.0,
f=tf.nn.relu):
                """
                Highway Network (cf.
http://arxiv.org/abs/1505.00387).
                t = sigmoid(Wy + b)
                z = t * g(Wy + b) + (1 - t) * y
                where g is nonlinearity, t is transform gate, and (1
- t) is carry gate.
                """

                for idx in range(num_layers):
                    g = f(_linear(input_, size,
scope=("highway_lin_{0}".format(idx))))
                    t = tf.sigmoid(_linear(input_, size,
scope=("highway_gate_{0}".format(idx))) + bias)
                    output = t * g + (1. - t) * input_
                    input_ = output

                return output
        def _linear(input_, output_size, scope="SimpleLinear"):
            """
            Linear map: output[k] = sum_i(Matrix[k, i] * args[i] ) +
Bias[k]
            Args:
                input_: a tensor or a list of 2D, batch x n,
Tensors.
                output_size: int, second dimension of W[i].
                scope: VariableScope for the created subgraph;
defaults to "SimpleLinear".
            Returns:
                A 2D Tensor with shape [batch x output_size] equal
to
```

```
                sum_i(args[i] * W[i]), where W[i]s are newly created
matrices.
            Raises:
                ValueError: if some of the arguments has unspecified
or wrong shape.
            """

            shape = input_.get_shape().as_list()
            if len(shape) != 2:
                raise ValueError("Linear is expecting 2D arguments:
{0}".format(str(shape)))
            if not shape[1]:
                raise ValueError("Linear expects shape[1] of
arguments: {0}".format(str(shape)))
            input_size = shape[1]

            # Now the computation.
            with tf.variable_scope(scope):
                W = tf.get_variable("W", [input_size, output_size],
dtype=input_.dtype)
                b = tf.get_variable("b", [output_size],
dtype=input_.dtype)

            return tf.nn.xw_plus_b(input_, W, b)
        # Embedding layer
        with tf.device('/cpu:0'), tf.name_scope("embedding"):
            self.W = tf.Variable(
                tf.random_uniform([vocab_size, embedding_size], -
1.0, 1.0),
                name="W")
            self.embedded_chars = tf.nn.embedding_lookup(self.W,
self.input_x)
            self.embedded_chars_expanded =
tf.expand_dims(self.embedded_chars, -1)

        # Create a convolution + maxpool layer for each filter size
        pooled_outputs = []
        for i, filter_size in enumerate(filter_sizes):
            with tf.name_scope("conv-maxpool-%s" % filter_size):
                # Convolution Layer
                filter_shape = [filter_size, embedding_size, 1,
num_filters]
                W = tf.Variable(tf.truncated_normal(filter_shape,
stddev=0.1), name="W")
                b = tf.Variable(tf.constant(0.1,
shape=[num_filters]), name="b")
                conv = tf.nn.conv2d(
                    self.embedded_chars_expanded,
                    W,
                    strides=[1, 1, 1, 1],
                    padding="VALID",
                    name="conv")

                #conv = tf.contrib.layers.batch_norm(conv,
is_training=True, scope='cnn_bn_')

                # Apply nonlinearity
                h = tf.nn.sigmoid(tf.nn.bias_add(conv, b),
name="sigmoid")
                 #h = tf.nn.relu(tf.nn.bias_add(conv, b),"relu")
```

```
                #h = tf.nn.relu(tf.nn.bias_add(conv, b),
name="relu")
                # Maxpooling over the outputs
                pooled = tf.nn.max_pool(
                    h,
                    ksize=[1, sequence_length - filter_size + 1, 1,
1],
                    strides=[1, 1, 1, 1],
                    padding='VALID',
                    name="pool")
                pooled_outputs.append(pooled)

        # Combine all the pooled features
        num_filters_total = num_filters * len(filter_sizes)
        self.h_pool = tf.concat(pooled_outputs, 3)
        self.h_pool_flat = tf.reshape(self.h_pool, [-1,
num_filters_total])

        ##mesut ADD
        num_filters_total = num_filters * len(filter_sizes)
        self.pool = tf.concat(pooled_outputs, axis=3)
        self.pool_flat = tf.reshape(self.pool, shape=[-1,
num_filters_total])

        # Fully Connected Layer
        with tf.name_scope("fc"):
            W =
tf.Variable(tf.truncated_normal(shape=[num_filters_total,
fc_hidden_size],
                                                    stddev=0.1,
dtype=tf.float32), name="W")
            b = tf.Variable(tf.constant(value=0.1,
shape=[fc_hidden_size], dtype=tf.float32), name="b")
            self.fc = tf.nn.xw_plus_b(self.pool_flat, W, b)

            # Batch Normalization Layer
            self.fc_bn = tf.layers.batch_normalization(self.fc,
training=self.is_training)

            # Apply nonlinearity
            self.fc_out = tf.nn.relu(self.fc_bn, name="relu")

        # Highway Layer
        with tf.name_scope("highway"):
            self.highway = _highway_layer(self.fc_out,
self.fc_out.get_shape()[1], num_layers=1, bias=0)



        # Add dropout
        with tf.name_scope("dropout"):
            self.h_drop = tf.nn.dropout(self.h_pool_flat,
self.dropout_keep_prob)

        # Final (unnormalized) scores and predictions
        with tf.name_scope("output"):
            W = tf.get_variable(
                "W",
```

```
                shape=[num_filters_total, num_classes],
                initializer=tf.contrib.layers.xavier_initializer())
            b = tf.Variable(tf.constant(0.1, shape=[num_classes]),
name="b")
            l2_loss += tf.nn.l2_loss(W)
            l2_loss += tf.nn.l2_loss(b)
            self.scores = tf.nn.xw_plus_b(self.h_drop, W, b,
name="scores")
            self.predictions = tf.argmax(self.scores, 1,
name="predictions")

        # Calculate mean cross-entropy loss
        with tf.name_scope("loss"):
            losses =
tf.nn.softmax_cross_entropy_with_logits(logits=self.scores,
labels=self.input_y)
            self.loss = tf.reduce_mean(losses) + l2_reg_lambda *
l2_loss

        # Accuracy
        with tf.name_scope("accuracy"):
            correct_predictions = tf.equal(self.predictions,
tf.argmax(self.input_y, 1))
            self.accuracy =
tf.reduce_mean(tf.cast(correct_predictions, "float"),
name="accuracy")
```

Train.py

```
import tensorflow as tf
import numpy as np
import os
import time
import datetime
import data_helpers
from text_cnn import TextCNN
from tensorflow.contrib import learn

# Parameters
# ==================================================

# Data loading params
tf.flags.DEFINE_float("dev_sample_percentage", .15, "Percentage of
the training data to use for validation")
tf.flags.DEFINE_string("data_file", "./train.csv", "Data source.")

# Model Hyperparameters
tf.flags.DEFINE_integer("embedding_dim", 128, "Dimensionality of
character embedding (default: 128)")
tf.flags.DEFINE_string("filter_sizes", "3,4,5", "Comma-separated
filter sizes (default: '3,4,5')")
tf.flags.DEFINE_integer("num_filters", 128, "Number of filters per
filter size (default: 128)")
tf.flags.DEFINE_float("dropout_keep_prob", 0.5, "Dropout keep
probability (default: 0.5)")
tf.flags.DEFINE_float("l2_reg_lambda", 0.0, "L2 regularization
lambda (default: 0.0)")
```

```
# Training parameters
tf.flags.DEFINE_integer("batch_size", 64, "Batch Size (default:
64)")
tf.flags.DEFINE_integer("num_epochs", 200, "Number of training
epochs (default: 200)")
tf.flags.DEFINE_integer("evaluate_every", 100, "Evaluate model on
dev set after this many steps (default: 100)")
tf.flags.DEFINE_integer("checkpoint_every", 100, "Save model after
this many steps (default: 100)")
tf.flags.DEFINE_integer("num_checkpoints", 5, "Number of checkpoints
to store (default: 5)")
# Misc Parameters
tf.flags.DEFINE_boolean("allow_soft_placement", True, "Allow device
soft device placement")
tf.flags.DEFINE_boolean("log_device_placement", False, "Log
placement of ops on devices")

FLAGS = tf.flags.FLAGS
# FLAGS._parse_flags()
# print("\nParameters:")
# for attr, value in sorted(FLAGS.__flags.items()):
#     print("{}={}".format(attr.upper(), value))
# print("")

def preprocess():
    # Data Preparation
    # ==================================================

    # Load data
    print("Loading data...")
    x_text, y = data_helpers.load_data_and_labels(FLAGS.data_file)

    # Build vocabulary
    max_document_length = max([len(x.split(" ")) for x in x_text])
    vocab_processor =
learn.preprocessing.VocabularyProcessor(max_document_length)
    x = np.array(list(vocab_processor.fit_transform(x_text)))

    # Randomly shuffle data
    np.random.seed(10)
    shuffle_indices = np.random.permutation(np.arange(len(y)))
    x_shuffled = x[shuffle_indices]
    y_shuffled = y[shuffle_indices]

    # Split train/test set
    # TODO: This is very crude, should use cross-validation
    dev_sample_index = -1 * int(FLAGS.dev_sample_percentage *
float(len(y)))
    x_train, x_dev = x_shuffled[:dev_sample_index],
x_shuffled[dev_sample_index:]
    y_train, y_dev = y_shuffled[:dev_sample_index],
y_shuffled[dev_sample_index:]

    del x, y, x_shuffled, y_shuffled

    print("Vocabulary Size:
{:d}".format(len(vocab_processor.vocabulary_)))
    print("Train/Dev split: {:d}/{:d}".format(len(y_train),
len(y_dev)))
    return x_train, y_train, vocab_processor, x_dev, y_dev
```

```
def train(x_train, y_train, vocab_processor, x_dev, y_dev):
    # Training
    # ==================================================

    with tf.Graph().as_default():
        session_conf = tf.ConfigProto(
          allow_soft_placement=FLAGS.allow_soft_placement,
          log_device_placement=FLAGS.log_device_placement)
        sess = tf.Session(config=session_conf)
        with sess.as_default():
            cnn = TextCNN(
                sequence_length=x_train.shape[1],
                num_classes=y_train.shape[1],
                vocab_size=len(vocab_processor.vocabulary_),
                embedding_size=FLAGS.embedding_dim,
                filter_sizes=list(map(int,
FLAGS.filter_sizes.split(","))),
                num_filters=FLAGS.num_filters,
                l2_reg_lambda=FLAGS.l2_reg_lambda)

            # Define Training procedure
            global_step = tf.Variable(0, name="global_step",
trainable=False)
            optimizer = tf.train.AdamOptimizer(1e-3)
            grads_and_vars = optimizer.compute_gradients(cnn.loss)
            train_op = optimizer.apply_gradients(grads_and_vars,
global_step=global_step)

            # Keep track of gradient values and sparsity (optional)
            grad_summaries = []
            for g, v in grads_and_vars:
                if g is not None:
                    grad_hist_summary =
tf.summary.histogram("{}/grad/hist".format(v.name), g)
                    sparsity_summary =
tf.summary.scalar("{}/grad/sparsity".format(v.name),
tf.nn.zero_fraction(g))
                    grad_summaries.append(grad_hist_summary)
                    grad_summaries.append(sparsity_summary)
            grad_summaries_merged = tf.summary.merge(grad_summaries)

            # Output directory for models and summaries
            timestamp = str(int(time.time()))
            out_dir = os.path.abspath(os.path.join(os.path.curdir,
"runs", timestamp))
            print("Writing to {}\n".format(out_dir))

            # Summaries for loss and accuracy
            loss_summary = tf.summary.scalar("loss", cnn.loss)
            acc_summary = tf.summary.scalar("accuracy",
cnn.accuracy)

            # Train Summaries
            train_summary_op = tf.summary.merge([loss_summary,
acc_summary, grad_summaries_merged])
            train_summary_dir = os.path.join(out_dir, "summaries",
"train")
            train_summary_writer =
tf.summary.FileWriter(train_summary_dir, sess.graph)

            # Dev summaries
```

```
            dev_summary_op = tf.summary.merge([loss_summary,
acc_summary])
            dev_summary_dir = os.path.join(out_dir, "summaries",
"dev")
            dev_summary_writer =
tf.summary.FileWriter(dev_summary_dir, sess.graph)

            # Checkpoint directory. Tensorflow assumes this
directory already exists so we need to create it
            checkpoint_dir = os.path.abspath(os.path.join(out_dir,
"checkpoints"))
            checkpoint_prefix = os.path.join(checkpoint_dir,
"model")
            if not os.path.exists(checkpoint_dir):
                os.makedirs(checkpoint_dir)
            saver = tf.train.Saver(tf.global_variables(),
max_to_keep=FLAGS.num_checkpoints)

            # Write vocabulary
            vocab_processor.save(os.path.join(out_dir, "vocab"))

            # Initialize all variables
            sess.run(tf.global_variables_initializer())

            def train_step(x_batch, y_batch):
                """
                A single training step
                """
                feed_dict = {
                  cnn.input_x: x_batch,
                  cnn.input_y: y_batch,
                  cnn.dropout_keep_prob: FLAGS.dropout_keep_prob
                }
                _, step, summaries, loss, accuracy = sess.run(
                    [train_op, global_step, train_summary_op,
cnn.loss, cnn.accuracy],
                    feed_dict)
                time_str = datetime.datetime.now().isoformat()
                print("{}: step {}, loss {:g}, acc
{:g}".format(time_str, step, loss, accuracy))
                train_summary_writer.add_summary(summaries, step)

            def dev_step(x_batch, y_batch, writer=None):
                """
                Evaluates model on a dev set
                """
                feed_dict = {
                  cnn.input_x: x_batch,
                  cnn.input_y: y_batch,
                  cnn.dropout_keep_prob: 1.0
                }
                step, summaries, loss, accuracy = sess.run(
                    [global_step, dev_summary_op, cnn.loss,
cnn.accuracy],
                    feed_dict)
                time_str = datetime.datetime.now().isoformat()
                print("{}: step {}, loss {:g}, acc
{:g}".format(time_str, step, loss, accuracy))
                if writer:
                    writer.add_summary(summaries, step)
```

```
            # Generate batches
            batches = data_helpers.batch_iter(
                list(zip(x_train, y_train)), FLAGS.batch_size,
FLAGS.num_epochs)
            # Training loop. For each batch...
            for batch in batches:
                x_batch, y_batch = zip(*batch)
                train_step(x_batch, y_batch)
                current_step = tf.train.global_step(sess,
global_step)
                if current_step % FLAGS.evaluate_every == 0:
                    print("\nEvaluation:")
                    dev_step(x_dev, y_dev,
writer=dev_summary_writer)
                    print("")
                if current_step % FLAGS.checkpoint_every == 0:
                    path = saver.save(sess, checkpoint_prefix,
global_step=current_step)
                    print("Saved model checkpoint to
{}\n".format(path))

def main(argv=None):
    x_train, y_train, vocab_processor, x_dev, y_dev = preprocess()
    train(x_train, y_train, vocab_processor, x_dev, y_dev)

if __name__ == '__main__':
    tf.app.run()
```

Evulacation.py

```
import tensorflow as tf
import numpy as np
import os
import time
import datetime
import data_helpers
from text_cnn import TextCNN
from tensorflow.contrib import learn
import csv
import sys

# Parameters
# ==================================================

# Data Parameters
data_file = "./data/train7.csv"
# tf.flags.DEFINE_string("data_file", "./train.csv", "Data source.")

# Eval Parameters
batch_size = 64
checkpoint_dir = "./runs/1558379363"
eval_train = True
# tf.flags.DEFINE_integer("batch_size", 64, "Batch Size (default:
64)")
# tf.flags.DEFINE_string("checkpoint_dir", "./runs/1558171973",
"Checkpoint directory from training run")
# tf.flags.DEFINE_boolean("eval_train", True, "Evaluate on all
training data")

# Misc Parameters
allow_soft_placement_ = True
```

```
log_device_placement_ = False
# tf.flags.DEFINE_boolean("allow_soft_placement", True, "Allow
device soft device placement")
# tf.flags.DEFINE_boolean("log_device_placement", False, "Log
placement of ops on devices")

FLAGS = tf.flags.FLAGS


print("\nParameters:")
for attr, value in sorted(FLAGS.__flags.items()):
    print("{}={}".format(attr.upper(), value))
print("")

# CHANGE THIS: Load data. Load your own data here
if eval_train:
    x_raw, y_test = data_helpers.load_data_and_labels(data_file)
    y_test = np.argmax(y_test, axis=1)
else:
    x_raw = ["a masterpiece four years in the making", "everything
is off."]
    y_test = [1, 0]

# Map data into vocabulary
vocab_path = checkpoint_dir + "/vocab"
vocab_processor =
learn.preprocessing.VocabularyProcessor.restore(vocab_path)
x_test = np.array(list(vocab_processor.transform(x_raw)))

print("\nEvaluating...\n")

# Evaluation
# ==================================================
checkpoint_file = tf.train.latest_checkpoint(checkpoint_dir +
"/checkpoints")
graph = tf.Graph()
with graph.as_default():
    session_conf = tf.ConfigProto(
      allow_soft_placement=allow_soft_placement_,
      log_device_placement=log_device_placement_)
    sess = tf.Session(config=session_conf)
    with sess.as_default():
        # Load the saved meta graph and restore variables
        saver =
tf.train.import_meta_graph("{}.meta".format(checkpoint_file))
        saver.restore(sess, checkpoint_file)

        # Get the placeholders from the graph by name
        input_x = graph.get_operation_by_name("input_x").outputs[0]
        # input_y =
graph.get_operation_by_name("input_y").outputs[0]
        dropout_keep_prob =
graph.get_operation_by_name("dropout_keep_prob").outputs[0]

        # Tensors we want to evaluate
        predictions =
graph.get_operation_by_name("output/predictions").outputs[0]

        # Generate batches for one epoch
        batches = data_helpers.batch_iter(list(x_test), batch_size,
1, shuffle=False)
```

```
        # Collect the predictions here
        all_predictions = []

        for x_test_batch in batches:
            batch_predictions = sess.run(predictions, {input_x:
x_test_batch, dropout_keep_prob: 1.0})
            all_predictions = np.concatenate([all_predictions,
batch_predictions])

# Print accuracy if y_test is defined
if y_test is not None:
    correct_predictions = float(sum(all_predictions == y_test))
    print("Total number of test examples: {}".format(len(y_test)))
    print("Accuracy:
{:g}".format(correct_predictions/float(len(y_test))))

# Save the evaluation to a csv
predictions_human_readable = np.column_stack((np.array(x_raw),
all_predictions))
out_path = os.path.join(checkpoint_dir, "prediction.csv")
print("Saving evaluation to {0}".format(out_path))
with open(out_path, 'w') as f:
    csv.writer(f).writerows(predictions_human_readable)
```

ÖZGEÇMİŞ

Adı Soyadı : Mesut PEK

Doğum Yeri ve Yılı : Sivas, 17/03/1993

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta : mesut.pek@gmail.com

Eğitim Durumu

Lise : Taşdelen İMKB Anadolu Teknik Lisesi, 2011

Lisans : İstanbul Ticaret Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü

Yüksek Lisans : İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı

Mesleki Deneyim

Opus Bilgi Teknolojileri, Yazılım Geliştirme ve Destek	2013-2014
İstanbul Ticaret Üniversitesi, Bilgi İşlem Birimi	2017-2018
İstinye Üniversitesi, Öğrenci İşleri Birimi	2018-2019
Şişli Meslek Yüksekokulu, Bilgi İşlem Birimi	2019-…(devam ediyor)

Yayınları

PEK M., TURAN M., “Sentiment Analysis Of Tweets Using Machine Learning”, Data Science Machine Learning and Statistics, Van, Türkiye, 2019