GEREKSİNİM RAPORU

Giriş:

Günümüzde internet ağları ve sosyal medya kullanımı, insanların başka kişi ya da kurumlar

hakkındaki görüşlerini kolay ve açık şekilde ifade edebilmesine ve yorumlamasına imkân

sağlamıştır. Bu imkânın giderek yaygınlaşması ile büyüyen zengin bir içerik ortaya çıkmaktadır.

Bunun sonucunda internetten elde edilen büyük verinin analiz edilerek anlamlı bilgiye

dönüştürülmesi ve kullanılması son yıllarda yoğun olarak çalışılan bir konudur. Bu bağlamda, son

on yılda, sosyal medya içeriğini analiz etmek için yapılan çalışmalar hızlı bir artış göstermiş ve

netnografi çalışmalarından (Kozinets, 2009; Hine, 2005) fikir madenciliğine (Dave vd., 2003) ve

doğal dil işlemeye (Nasukawa ve Yi, 2003) kadar çok sayıda yöntem önerilmiştir. Duygu analizi

ise büyük veri içindeki öznel bilginin sistematik olarak analiz edilmesini sağlayan bir doğal dil işleme

yöntemidir. Metin madenciliği ve hesaplamalı dilbilim araçlarını kullanarak,

araştırmacıların metin içindeki duyguları belirlemelerine, duygu kategorilerine göre

sınıflandırmalarına ve kategorilerin duygusal polaritelerini bulmalarına yardımcı olur (Arazy ve

Woo, 2007; Jacobson, 2009).

Python ile Film Yorumlarının Duygu Analizi

Duygu Analizi asıl olarak, insani duyguları makineye öğretmeyi amaçlamaktadır. Çoğunlukla pozitif,

negatif ve nötr olarak sınıflandırma yapılır. Fakat başka birçok çeşitleri de mevcuttur. Mesela mutlu,

üzgün, şaşkın, kızgın vb. şeklinde de sınıflandırma yapılabilir.

Son zamanlarda çok fazla önem kazanan bir alan oldu. Özellikle pandemi süreci içerisinde yorumların

önemi kat kat arttı. İnsanlar sosyal medya, film/dizi platformları, e-ticaret vb. sitelerde oldukça fazla

vakit geçirmeye başladı. Bunlardan birisi için duygu analizi örneği verecek olursak; e-ticaret

sitelerinde herhangi bir ürünün alışverişinden sonra yapılan yorumlar alıcının o ürün hakkındaki

duygularını, tutumunu belirlemeye fayda sağlar. Bu sayede satıcı müşteri tutumunu tespit edip

yapılması gereken herhangi bir şey varsa yapmak ya da düzeltilmesi gereken herhangi bir durum

varsa düzeltmek konusunda girişimlerde bulunur. Aynı zamanda bahsettiğimiz üzere özellikle

pandemi sürecinde diğer alıcı adaylara büyük fayda sağlamıştır. Çünkü müşteriler ürünü canlı

göremiyorlar ve bu yüzden de tereddüt içerisinde kalıyorlar. Mesela kıyafet alırken tam olarak beden

yapısı hakkında öngörüşlü olamıyorlar ya da ev eşyası alacaklarsa kalitesi hakkında bir çıkarımda

bulunamıyorlar. Bu yorumlar sayesinde ürünün kalitesi, tipi, yapısı, faydası, fiyatı vb. özellikleri

hakkında fikir edinip ürünü alıp almamaları konusunda daha kolay karara varabiliyorlar. Bu da yine

satıcının satış miktarını büyük oranda etkilemektedir.

Film Yorumlarının Duygu Analizi

Kaggle üzerinden aldığım veri seti içerisinde 83227 gözlem ve 3 değişken mevcuttur. Bu veri; film

yorumları (comments), film adları (film\_name) ve yorumların duygu derecelerinden (points)

oluşmaktadır.

Öncelikle veriyi ön işlemlerden geçirip modele uygun hale getireceğiz ve ardından duygu analizini

gerçekleştireceğiz.

Gerekli kütüphaneleri import edelim.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Veri setimizi okutalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görüldüğü üzere yorumların hepsinin başında “\n” böyle bir karakter mevcuttur. Biz de bir fonksiyon

yazarak bu karakterleri kaldıracağız.

Birkaç satır ile kontrol ettiğimde verinin başından 23 karakter ileri, sonundan da 24 karakter geri

gittiğimde oluşmuş boşluklar ve “\n” karakterleri ortadan kalkıyor. O yüzden bunu tüm veriye

uygulayabileceğimiz bir fonksiyon yazmamız gerekiyor.

Öncelikle comments adlı bir fonksiyon oluşturuyorum. Bu fonksiyon istediğimiz değişken içerisindeki

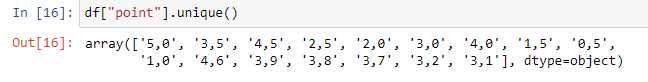
gözlemlerin başından 23, sonundan 24 karakter kaldıracak. Ardından bu fonksiyonu comment

değişkenindeki gözlemlere apply modülü ile tek tek uyguluyorum ve tekrar aynı değişken içerisine

atıyorum.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



Filmlere 1–5 arası puanlandırma yapılmıştır. Bizim duygu sınıflandırmamız pozitif ve negatif olarak

ayrılacağı için bunu ikili sınıflandırmaya dönüştürmem gerekiyor. Fakat ondan önce bir sorun daha

mevcut, veri içerisinde numaralar virgül ile ayrılmış. Biz sadece ilk karakteri alacağız ve ardından float

tipine dönüştüreceğiz.

Bir önceki kodda yaptığımıza benzer floatize fonksiyonu oluşturuyoruz ve point içerisindeki gözlemleri

apply modülünü kullanarak düzeltiyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Düzelttiğimiz puanları, 0 ve 1 olarak sınıflandıracağız. 3 puanı bizim için nötr oluyor. Ne pozitif ne de

negatif diyebiliyoruz. Bu satırları veriden çıkaracağız. Ardından 3'ten küçüklere 0 (negatif), 3'ten

büyüklere 1 (pozitif) diye atama yapacağız.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Veri içerisinden sildiğimiz satırlar olduğu için index yapısı bozuldu. İndexleri resetliyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yorumların ön işlemlerine geçiş yapıyoruz. İlk önce comments altındaki bütün karakterleri

küçültüyoruz. Bunu lower modülü ile gerçekleştireceğiz. Apply modülü ile tüm veri içerisine

uyguluyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şimdi ise noktalama işaretlerini kaldıracağız. remove\_punctuation adlı bir fonksiyon yazıyoruz. Bu

fonksiyon yorumlar içerisinde dolanacak ve eğer gezindiği karakter noktalama işareti değil ise bunu

word\_wo\_punc değişkenine atacak. Ardından da biz bu apply modülü ile bunu veriye uygulayacağız

ve bize veriyi noktalama işaretleri kaldırılmış bir şekilde verecek. Fakat bunu tek başına yaptığımızda

içerisinde hala kalkmamış olan “\n” ve “\r” karakterleri yer alıyor. Bunu da kendimiz manuel olarak

değiştiriyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sırada numerik karakterleri kaldırmamız gerekiyor. Bunun içinde remove\_numeric isimli bir fonksiyon

yazıyoruz. Bu fonksiyon da yorumlar içerisinde dolanacak ve numerik olmayan her şeyi output

değişkeninin içerisine atacak. Ardından bize de numerik karakterler kaldırılmış olarak bir çıktı verecek.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yorumlar Word Cloud

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Artık modeli kurmak için girişimlerde bulunabiliriz. Etiketleri (points) ve yorumları (comments)

ayırarak listeler içerisine alıyoruz.

Elimizdeki verilerin %80'i ile modeli kurup, %20'si ile modelimizi test edeceğiz. Bu sebepten dolayı

öncelikle verinin boyutunun %80'inini alarak cutoff değişkenine atıyoruz. Ardından da cutoff

değişkenine göre oluşturduğumuz target ve data listelerini, train ve test olarak ayırıyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Artık elimizdekileri tokenleştirebiliriz. Tokenleştirme işlemi yapılırken her yorum kelimelere ayrılacak.

Kelime hazinesindeki her kelimeye karşılık farklı bir sayı gelecek. İlk önce kelime haznemizde en fazla

kaç tane kelime oluşsun bunu belirleyeceğiz. Bunun için num\_words değişkenine 10.000 değerini

veriyoruz. Bununla beraber en sık geçen 10.000 kelimeyi alacağız. Duygu Analizi yapacağımız için veri

içerisindeki tüm kelimeleri almamıza gerek yok. Tokenleştirme işlemini “keras” kullanarak

gerçekleştiriyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Tokenizer tanımladığımıza göre elimizdeki veriyi tokenleştirebiliriz.

Train ve test verisindeki tüm yorumları tokenler halinde bir değişken içerisinde saklayalım. Ardından

da 1000. satırın nasıl gözüktüğünü kontrol edelim.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kelime haznesini sınırlandırdığımız için her kelimeye karşılık bir token bulunmayacak. Eğer kelime

haznesinde değilse o kelime, yok sayılayacak.

Sinir ağlarında genellikle RNN ile oluşturduğumuz modellere belli boyutlarda inputlar veririz. Bizim

verimizdeki yorumların hepsi farklı sayılardaki kelimelerden oluşmaktadır. Farklı sayılardaki kelimeleri

RNN içerisine input olarak veremeyiz. Tüm yorumları aynı boyuta getirmemiz gerekir. Eğer yorum

belirlediğimiz boyuttan düşük boyutta olursa 0 eklenecek. Eğer yorum belirlediğimiz boyuttan

yüksekse belirli kısımları silerek kendi belirlediğimiz boyuta getireceğiz.

Öncelikle for döngüsü oluşturup veri setimizdeki her yorumun üzerinden tek tek geçeceğiz ve token

sayısını alacağız. Liste üzerinde işlem yapmayı kolaylaştırmak amacıyla da listeyi numpy array’e

dönüştüreceğiz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ortalama olarak bir yorumda kaç tane token olduğuna bakalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

En fazla token bulunan yorumun token sayısına bakalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şimdi yorumların boyutlarını eşitleyeceğiz. Bunun için kendimiz bir boyut belirliyoruz. Kendimiz değer

verebiliriz, en yüksek tokenli yorumun boyutunu kullanabiliriz ya da ortalamayı alabiliriz. Biz ise

ortalamayı alıp 2 standart sapma ekleyerek elde edeceğiz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Elde ettiğimiz boyut yorumların yüzde kaçını kapsıyor bakalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Train setindeki her yorumun boyutunu elde ettiğimiz boyuta getiriyoruz. İlk önce padding eklenecek

veriyi giriyoruz. Ardından belirlediğimiz boyutu giriyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Padding’i daha iyi görebilmek 800. yoruma bakalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Aynı yorumun padding uygulanmış haline bakalım.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Tokenleri verip, stringi alabilmek için fonksiyon yazacağız. Bu şekilde elimizdeki tokenleri tekrar

yoruma çevireceğiz. word\_index içerisinde sözlük olarak kelimeler, kelimelerin sayısal olarak

tokenleri bulunuyordu. word\_index ile kelimeler ve sayıların yerini tersine çeviriyoruz. Artık bir sayıyı

verdiğimiz zaman o sayıya karşılık gelen kelimeyi bulabileceğiz. Fonksiyon içerisinde sıfırlar kelimeye

karşılık gelmediği için dahil etmiyoruz. Diğerlerini de liste içerisinde topluyoruz. Son olarak da bize

yorumu çıktı olarak verecek.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sıra modeli kurmaya geldi. Embedding matrisini oluşturmadan önce, embedding matrisinin

büyüklüğünü belirliyoruz. Her kelimeye karşılık gelen 50 uzunluğunda vektör oluşturuyoruz. Modele

bir şey ekleyebilmek model.add kullanıyoruz.

num\_words (10 bin) kelime sayılarıdır. 10 bin kelimeye karşılık gelen 50 uzunluğunda rastgele

vektörler oluşturalım. Layer input aldığı zaman, input içerisindeki kelimelerin vektörlerini bir sonraki

layer’a gönderecektir. embedding\_layer’ın output’u, bir sonraki layer’ın input’u olacak.

Yinelenen sinir ağımızı (RNN) oluşturuyoruz. Yinelenen sinir ağını oluşturmak GRU kullanacağız.

model.add(GRU()) bu şekilde de kullanılabilir. units, GRU değerimizin nöron sayısıdır. Bu layer’da 16

tane output verilecek. return\_sequence = True, output olarak sequences tamamı döndürülecek. Eğer

False olsaydı sadece son output döndürülürdü. Bir sonraki layer’a GRU ekleyeceğimiz için buna True

demek zorundayız. Eğer bir sonraki layer bir nörondan oluşsaydı False girerdik. Sigmoid ile output’u

[0,1] arasına sıkıştırıyoruz.

metin içeren bir resim

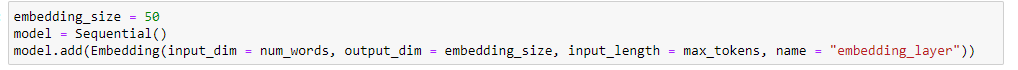
Açıklama otomatik olarak oluşturuldu

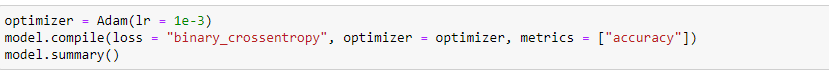
Modeli kurduk. Modelin eğitimine dair optimizasyon algoritması belirleyeceğiz. Bunun için Adam

algoritmasını kullanacağız.

Optimizasyon algoritmasını belirledikten sonra, modelimizi derliyoruz. Sadece iki sınıf olduğu için de

“binary\_crossentropy” kullanacağız.





tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eğitime başlıyoruz. Eğitimden kaç defa geçeceğine epochs parametresi ile karar veriyoruz.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Test setindeki ilk 1000 yorum için tahminde bulunuyoruz. Bu tahminler bir matrisin sütunu olarak

atanacaktır. Her satırda bir yorumun tahmin değeri bulunacaktır. Bu elemanlarla işlem yapmayı

kolaylaştırabilmek için sütunu satıra çevireceğiz. Böylece elimizde tahmin değerlerini gösterecek bir

vektör elde edeceğiz. Vektör üzerinde işlem yapmak daha kolaydır. Sütunu satıra çevirebilmemiz için

matrisin transpozunu alıyoruz.

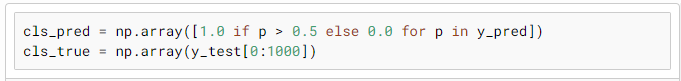
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Elimizdeki tahmin değerlerini gerçekleri ile karşılaştırarak hangi yorumlarda yanlış olduğunu

göreceğiz. Elimizdeki tahmin değerleri 0.5'ten büyükse 1, küçükse 0 yapacağız. Böylece 0 ve 1'lerden

oluşan bir vektör elde edeceğiz. Elde ettiğimiz vektörü etiketlerle karşılaştıracağız.



Elimizdeki iki vektörü karşılaştıracağız. Yanlış tahminlerin indexlerini alalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

1000 tane yorumdan kaç tanesinin yanlış tahmin edildiğine bakalım.



Yanlış bilinen yorumlardan bir yorumun indexini alarak yoruma bakalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Modelimizin bu yoruma nasıl tahminde bulunduğuna bakalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Modelimiz yorumu olumlu tahmin etmiş, doğrusuna bakalım.



Kütüphaneler:

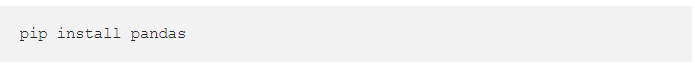
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Install NumPy with pip:



Install pandas with pip:



Install scikit-learn with pip:



Install matplotlib with pip:



Recurrent Neural Network Nedir?

Recurrent Neural Networks (RNN) ‘ü anlayabilmek için önce feedforward (ileri doğru çalışan) bir ağın

çalışma prensibini tekrar inceleyelim. Kısaca katmanlar üzerindeki nöronlara gelen bilgilere bir takım

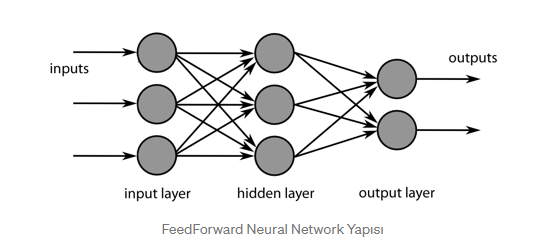
matematiksel işlemler uygulayarak çıktı üreten bir yapı diyebiliriz.

Feedforward çalışan yapıda gelen bilgi sadece ileri doğru işlenir. Bu yapıda kabaca, input verileri

ağdan geçirilerek bir output değeri elde edilir. Elde edilen output değeri doğru değerler ile

karşılaştırılarak hata elde edilir. Ağ üzerindeki ağırlık değerleri hataya bağlı olarak değiştirilir ve bu

şekilde en doğru sonucu çıktı verebilen bir model oluşturulmuş olur.



Feedforward bir ağın eğitiminde hatanın yeterince düşürülmesi gerekir. Böylece nöronlara giden

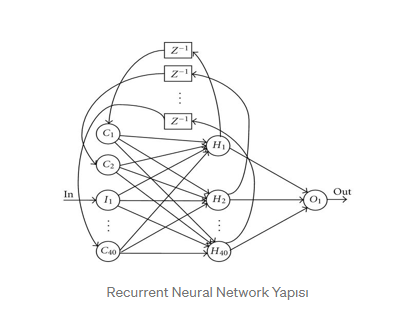
ağırlıklar yenilenerek girilen inputa uygun output verecek bir yapı oluşmuş olur.

Örneğin, bir fotoğraf üzerindeki nesneleri kategorize etmek için eğitilen feedforward bir ağ

düşünelim. Verilen fotoğraf rastgele bir sıra ile de olsa hem eğitim hem de test için kullanılabilir. Bir

önceki veya bir sonraki fotoğraf ile herhangi bir bağının olması gerekmez. Yani zamana veya sıraya

bağlı bir kavram yoktur, ilgilendiği tek input o andaki mevcut örnektir.



Recurrent(yinelenen) yapılarda ise sonuç, sadece o andaki inputa değil, diğer inputlara da bağlı olarak

çıkarılır. Şekilde de görüleceği gibi RNN’de t anındaki input verilerinin yanında, t-1 anından gelen

hidden layer sonuçları da hidden layer’ın t anındaki girdisidir. t-1 anındaki input için verilen karar, t

anında verilecek olan kararı da etkilemektedir. Yani bu ağlarda inputlar şimdiki ve önceki bilgilerin

birleştirilmesi ile output üretirler.

Recurrent yapılar, outputlarını sonraki işlemde input olarak kullandıkları için feedforward yapılardan

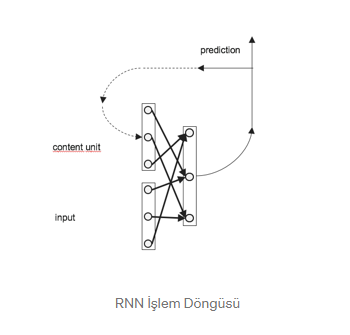
ayrılmış olurlar. Recurrent ağların bir belleğe sahip olduğunu söyleyebiliriz. Bir ağa memory

eklemenin sebebi ise, belli bir düzende gelen input setinin, çıktı için bir anlamı olmasıdır. Bu çeşit

data setleri için feedforward ağlar yeterli olmaz.

Tam bu noktada RNN’ler devreye girer. Yazı, konuşma, zamana bağlı çeşitli sensör veya istatiksel

veriler gibi belli bir sıra ile gelen verilerin yapısını anlamada recurrent ağlar kullanılır.



Diagramda görüldüğü gibi , RNN’in işlem döngüsünde hidden layer’dan çıkan sonuç hem output

üretir, hem content unitlere yazılır. Bu şekilde, her yeni input, önceki inputların işlenmesi sonucu

üretilmiş content unitlerle birlikte işlenir. Farklı zamanlarda belleğe alınan veriler arasında korelasyon

bulunuyorsa buna “long term” bağımlılık denir. RNN, bu long-term bağımlılıkların arasındaki ilişkiyi

hesaplayabilen bir ağdır.

İnsanların davranışlarında, konuşma ve düşüncelerinde de bu yapıdaki gibi önceden memory’de olan

bilgi tıpkı bir hidden layer’da yeni veri ile döngüye girerek işlenir. Bu işlemi yaparken kullanılan

matemetiksel formül aşağıdaki gibidir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

ht, t anındaki hidden layer’ın sonucudur. xt inputu W ağırlığı ile çarpılır. Daha sonra t-1 anında,

content unit ‘te tutulan h(t-1) değeri U ağırlığı ile çarpılır ve Wxt ile toplanır. W ve U değerleri girdi ile

katman arasındaki ağırlıklardır. Burada ağırlık matrisi önceki ve şimdiki verinin hangisinin sonuca

etkisi daha çok veya az ise ona göre değerler alır. Bu işlemler sonucunda oluşan hata hesaplanır ve

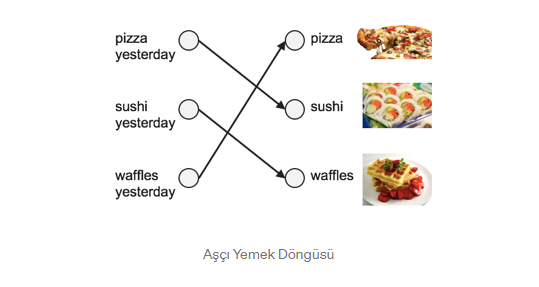
backpropagation ile yeni ağırlık değerleri tekrar düzenlenir. Backprop işlemi hata yeterince minimize

edilene kadar devam eder. Wxt + Uh(t-1) toplamı sigmoid, tanh gibi aktivasyon fonksiyonuna sokulur.

Böylece çok büyük veya çok küçük değerler mantıklı bir aralığa alınır. Bu şekilde non-lineerlik de

sağlanmış olur.

Örnek Bir Recurrent Neural Network



Bir aşçının 3 farklı yemeği sırasıyla yaptığını düşünelim. 1. gün pizza yaptıysa 2. gün sushi 3. gün

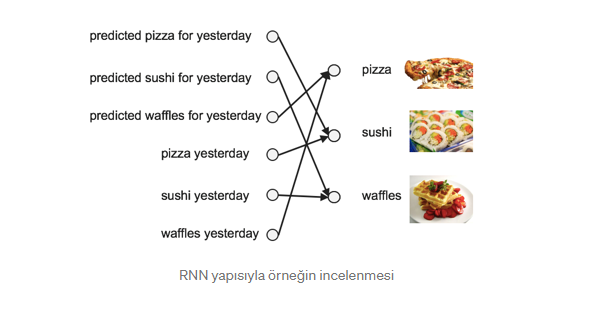
waffle yapıyor olsun. Bir sonraki gün ne yapacağını tahmin etmemiz gerekiyorsa öncelikle nasıl bir

problemle karşı karşıya olduğumuz anlayarak, bu probleme uygun bir metot kullanmamız gerekir.

Burada yemek bir sıraya göre geldiği için, yani önceki gün yapılan yemeğin sonraki günde etkisi

olacağı için uygulanması gereken metot recurrent neural network metodudur. Bu sayede elimizdeki

bilgilerle yeni bilgi olmasa dahi birkaç hafta sonraki günde yapılacak yemeği dahi tahmin edebiliriz.



Backpropagation Trough Time(BPTT)

Recurrent ağların amacı sıralı inputları doğru bir şekilde sınıflandırmak diyebiliriz. Bu işlemleri

yapabilmek için hatanın backpropunu ve gradient descentini kullanırız. Backprop, feedforward

ağlarda sonda outputtaki hatayı geriye hatanın türevini ağırlıklara dağıtılarak yapılır. Bu türev

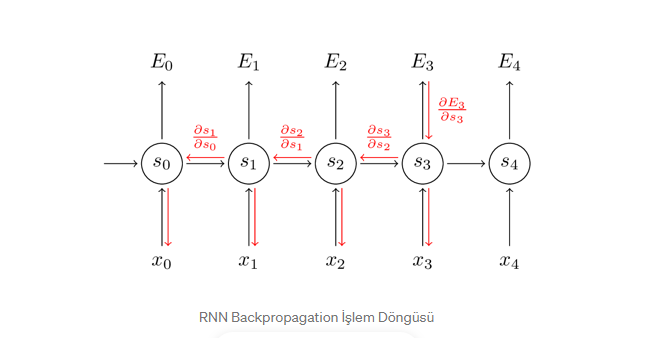
kullanılarak öğrenme katsayısı, gradient descent düzenlenerek hatayı düşürecek şekilde ağırlıklar

düzenlenir.

RNN için kullanılan yöntem ise BPTT diye bilinen zamana bağlı sıralı bir dizi hesaplamanın tümü için

backprop uygulamasıdır. Yapay ağlar bir dizi fonksiyonu içiçe f(h(g(x))) şeklinde kullanır. Buraya

zamana bağlı değişken eklendiğinde türev işlemi zincir kuralı ile çözümlenebilir.



Yukardaki RNN’in çalışma mantığı anlatılmıştır. Üstteki figürde ise sıralı 5 girdili bir recurrent ağ yapısı

gösterilmektedir. E burada oluşan hatayı ifade etmektedir. Örneğin, E3 için backpropagation

yaparken yaptığımız işlemde w ağırlığına göre türevi kullanılmaktadır. Bu türevi çözebilmek için zincir

kuralı ile birkaç türevin çarpımını kullanırız.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu çarpım yukarıda gösterildiği gibidir. Burada s3’ün açılımında s2’ye bir bağımlılık bulunur. Bunu

çözebilmek için yine zincir kuralını kullanarak s3’ün s2’ye türevini de ekleyerek sonucu bulabiliriz.

Böylece formüldeki gibi gradient zamana bağlı şekilde dağıtılmış olur.

Truncated BPTT

Truncated BPTT ise BPTT ‘nin uzun bir sıralı veri için bir çok ileri, geri işlemler çok masraflı

olacağından, BPTT için yaklaşım yapar. Bunun kötü yanı gradient kesilmiş hesaplandığından ağ “long

term” bağımlılıkları full BPTT kadar iyi öğrenemez.

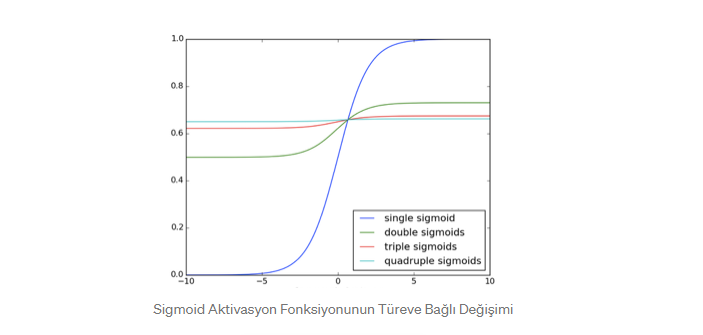
Vanishing/Exploding Gradient ( Gradient yokolması/uçması)

Gradient tüm ağırlıkları ayarlamamızı sağlayan bir değerdir. Ancak birbirine bağlı uzun ağlarda

hatanın etkisi oldukça düşerek gradient kaybolmaya başlayabilir. Bu da doğru sonucu bulmayı

olanaksızlaştırır. Bütün katmanlar ve zamana bağlı adımlar birbirine çarpımla bağlı olduğundan,

türevleri yokolma veya uçma yani aşırı yükselme tehlikesindedir.



Gradient exploding yani aşırı büyümesi ağın çok büyük değerler üretmesini sağlayarak doğru

sonuçtan uzaklaştıracaktır. Bunun için threshold koyarak çok yüksek değerli gradientleri kesmek basit

ve etkili yollardan biridir. Gradientlerin aşırı küçülerek yok olması ise çok daha zor bir problemdir.

Nerde, ne zaman durdurulması gerektiği çok açık değildir.

Neyse ki bu sorunu çözmek için de birkaç çözüm bulunmaktadır. W için uygun başlangıç değerleri

seçmek yok olma etkisini azaltacaktır. Bir diğer çözüm ise sigmoid ve tanh aktivasyon fonksiyonları

yerine ReLU kullanmaktır. ReLU fonksiyonunun türevi 0 veya 1’dir. Bu sebepten böyle bir problem

içerisine girmeyecektir. Bir diğer yöntem ise bu problemi çözmek için dizayn edilmiş olan LSTM

metodudur.