Mert Akçay 05200000002

İşlemsel Zeka Rapor

1.Literatür Taraması

1. https://en.wikipedia.org/wiki/BERT_(language_model)
2. https://huggingface.co/blog/bert-101
3. https://huggingface.co/models?language=en&sort=downloads&search=bert
4. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
5. https://arxiv.org/pdf/2010.15296.pdf
6. https://arxiv.org/pdf/1810.04805.pdf
7. https://www.section.io/engineering-education/classification-model-using-bert-and-tensorflow/ (Özellikle PyTorch ile implementasyon edilirken kullanıldı)
8. https://arxiv.org/abs/1905.05583
9. https://arxiv.org/abs/2111.15379
10. https://arxiv.org/pdf/2202.00795.pdf

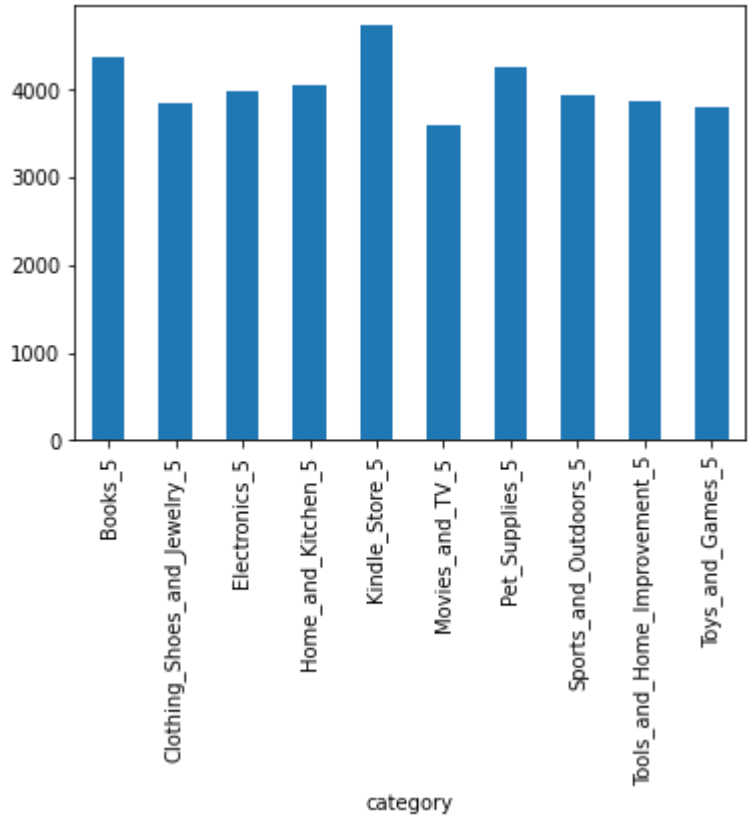
2.Beyin Fırtınası Aşaması

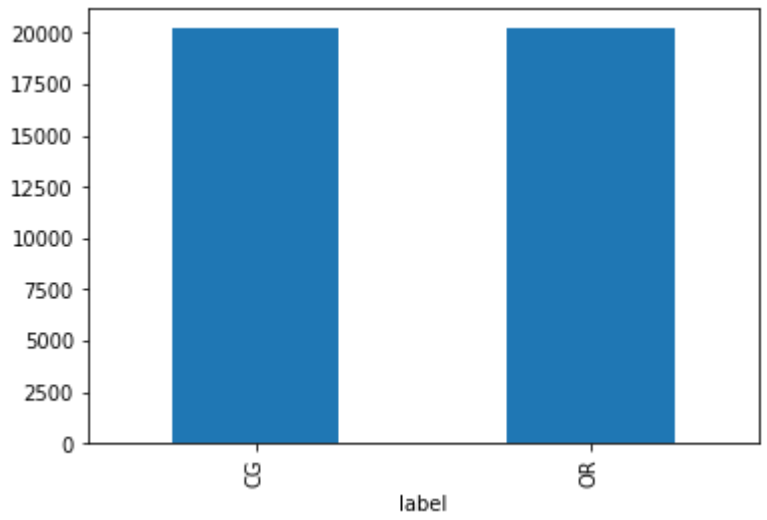
Genellikle kullanıcılar ürün alırken yorumları inceleyerek alışveriş yaparlar ancak son zamanlarda(özellikle yurt dışında anonim kişiler) para karşılığı 3.parti satıcıların dükkanlarında sahte yorumların çok fazla arttığını fark ettim. Genellikle tek düze noktalama işaretleri yok(text preprocessing işleminden çıkmış gibi) ve sadece sıfatları kullanarak yorumlar olduğunu gördüm. Bunu çözmek için ise text classification algoritması geliştirmem gerektiğini fark ettim. Özellikle son zamanlarda NLP üzerine çok fazla model geliştiren HuggingFace üzerinden çeşitli NLP algoritmalarının performans metriklerini inceledikten sonra BERT Modelini kullanmaya karar verdim.

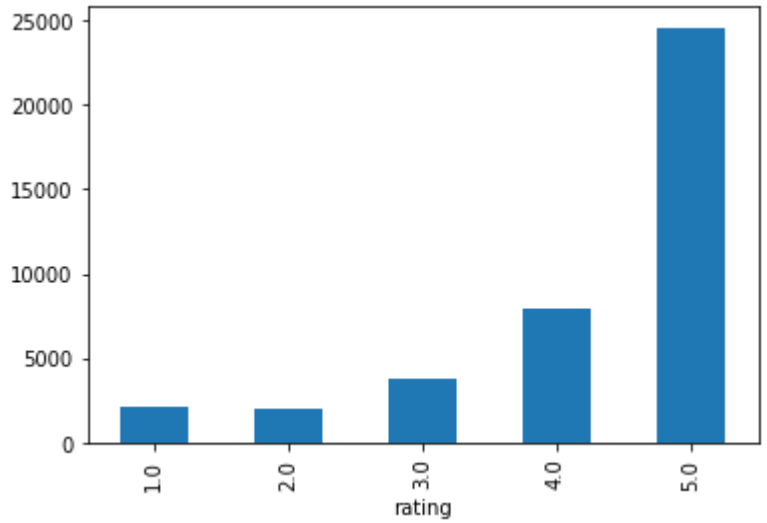
3.Veri Seti

Dataset imbalance bir yapıya sahiptir değildir bundan dolayı bir upsample ve weighted classification işlemi gerçekleştirilmemiştir. Verinin inceleniceği nokta review üzerinden olucağı için bu category üzerinden değelerndirmek daha sağlıklı olduğunu düşünmekteyim.
Train Size : 32345
Validation Size : 4043
Test Size : 4044
Dataset Link: https://osf.io/tyue9/

3.1 Kategori Dağılımları







4.Yazılım Ortamı ve Kütüphaneler

- Pytorch
- Python
    - Numpy
    - Pandas
    - Matplotlib
- HuggingFace
    - BERT

5.Deneysel Çalışmalar

5.1 Hazırlık Ve Inceleme Aşaması

Çeşitli datasetlerini ve farklı sitelerdeki yorumları inceledikten sonra web scraping ile kolayca tüm yorumları toplayabileceğimi fark ettim. Burada yaşanılan en önemli sıkıntı alınan veriler label'a sahip değildi. Çeşitli unsupervised learning algoritmaları ile kelime uzaklıklarını kümelemeye çalıştım fakat bir sonuç alamadım. Bunun yerine mevcut labeled datasetler ile text sınıflandırma algoritmalarına yöneldim.

5.2 Hiperparametre Araması

Newbie modeli oluşturmak amacıyla farklı optimzerlar ile en hızlı öğrenme ivmesini yakalayabileceğim optimizerı bulmaya çalıştım. AdamW burada en optimum çalışan optimizerdı. Daha sonraki adımda LR hiperparametrelerin araması ve modeli dahada iyileştirmeye çalıştım.

5.3 İlk model için Optimizer ve LR

32 Epoch ile eğittiğim modelde 4. Epochtan sonra modelin ayrıştığını buldum. Optuna ile hiperparametre araması sonucunda AdamW ile Learning Rate 0.01-0.0000001 Arası arama yaptıktan sonra optimal LR değeri 0.000001 olduğunu keşfettim.

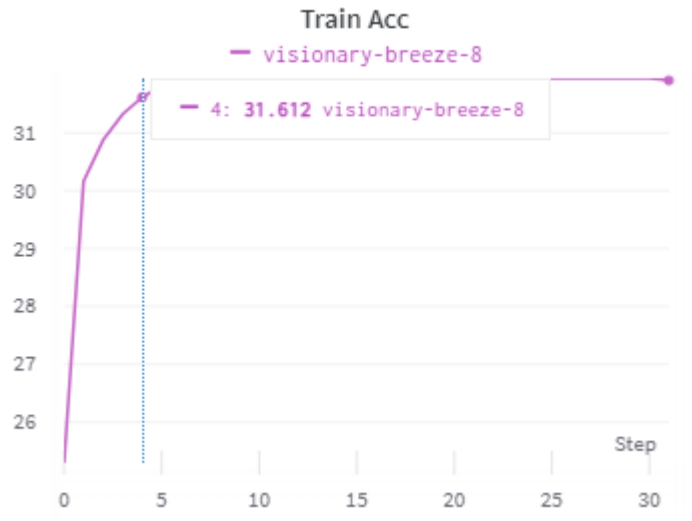
5.4 BERT Modeli Katkısı

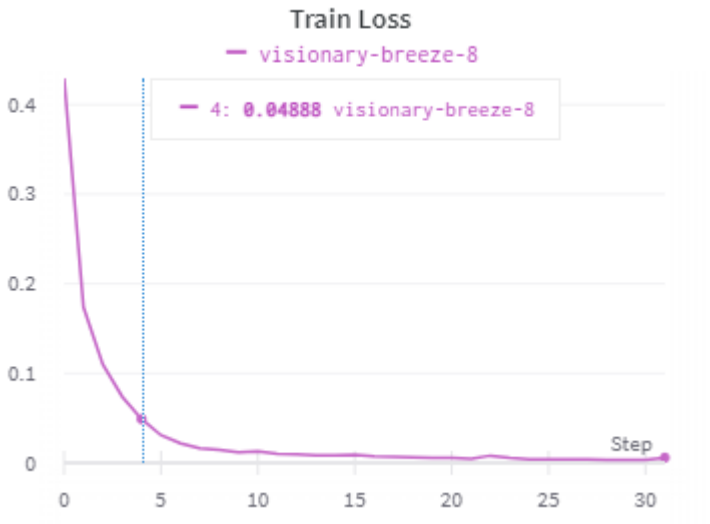
https://arxiv.org/pdf/2010.15296.pdf paperda farklı model yapıları üzerine incelenmiştir. Benim burada sağladığım katkı ise hem farklı optimizerlar altında BERT üzerinden çıkan featureların nasıl bir davranış gösterdiğidir.

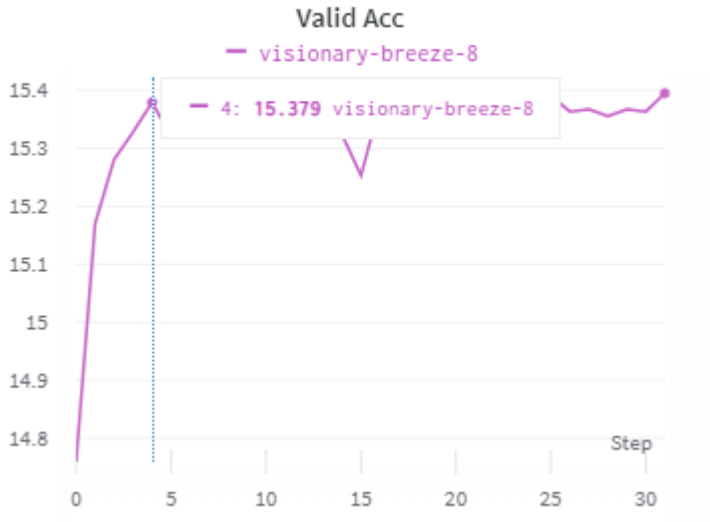
5.5 Model Değerlendirmesi

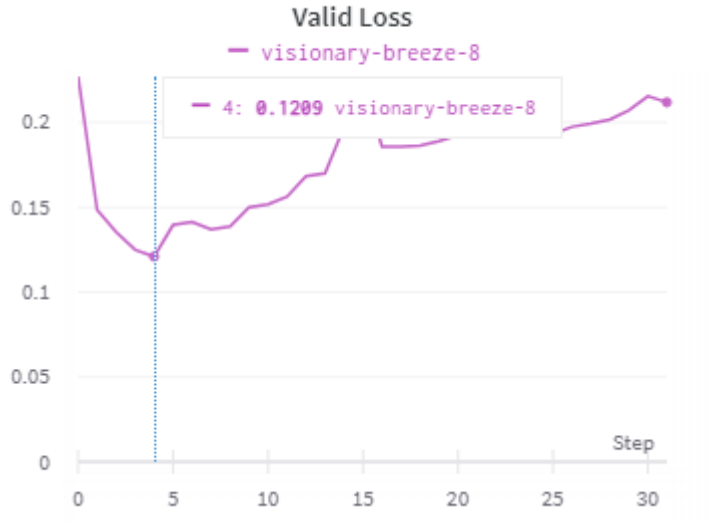
İkili sınıflandırma yapıldığı için Eğitim verisinde %98,7875 başarı oranı Validasyon veri setinde ise %96,11875 başarı oranı yakalanmıştır(Referans değerler her bir batch başına olan item doğruluk oranına denk gelmektedir).
Train Batch: 32
Validation Batch: 16

Ayrıca Loss fonksiyonunu incelediğimizde 4. Epochta ayrıştığını net bir şekilde görmekteyiz.




Train Loss
visionary-breeze-8
4: 0.04888 visionary-breeze-8
0.4
0.3
0.2
0.1
0
Step
0
5
10
15
20
25
30


Valid Acc
visionary-breeze-8
4: 15.379 visionary-breeze-8
15.4
15.3
15.2
15.1
15
14.9
14.8
Step
0
5
10
15
20
25
30


Valid Loss
visionary-breeze-8
4: 0.1209 visionary-breeze-8
0.2
0.15
0.1
0.05
0
Step
0
5
10
15
20
25
30

6.Ekler

6.1 EK

Kaggle ve github üzerinden dataset araştırması yaptığımda hiç bu dataset ile çalışıldığını görmedim. Bu sebepten dolayı domain bakımından nasıl bir katkı yaptığımı net bir şekilde açıklamam doğru olmaz. Ancak Modelin gerçekleştirilmesi ve optimizasyonu noktasında BERT-Base token size'ı ile hangi optimizerın nasıl bir tepki verdiği ve bu optimizera bağlı olarak optimal LR değerini bulmaya çalıştığımı söyleyebilirim.

6.2 EK

Bir önceki ekte bahsettiğim gibi kullandığım dataset ile bir çalışmayla karşılaşmadım. Ancak mevcut başarı değerleri (test: %98,7875 validation %96,11875) ile sağlıklı bir başarı yakaldığımı düşünmekteyim.

6.3 EK

https://www.section.io/engineering-education/classification-model-using-bert-and-tensorflow/ da kullanılan yapıyı PyTorch ortamına çektim ve otomatik hiperparametre araması için optuna,optimizerların model aşamasındaki tepkilerini ölçtüm, trace etmek içinse Weight and Biases ortamını dahil ettim.

6.4 EK

Q-Learning: Mevcut dünyadaki mantığa en yakın durum olduğunu düşünmekteyim. Algoritmanın aldığı her bir karar pozitif ve negatif durumları düşünerek karar verdiği yapıyı temsil etmektedir.

Deep Q-Learning: NN lerin gücünün kullanıldığı, impilicit knowledge ın da aktif olarak kullanıldığı ve bir domino taşı gibi küçük bir zararlı impulsun sonuçlarını bile değerlendirebileceğini düşündüğüm yapıdır. NN ile Q learning in state mantığı birleştirip çok daha güçlü ama çok daha israfkar bir modeldir.

GlooVe Öncesinde çok fazla karşılaştığım fakat kullanmadığım yapıdır. Text çeşitli optimizasyon yöntemleri ile vectör haline getirip bunları anlamlandırılmasında kullanılan algoritmadır.

Fuzzy Logic: Bankacılık, Finans gibi kritik alanlarda bulunan gri alanların temsili ve bir kümenin birden fazla kümeye yakın bulunabileceğini ön tanımlı fonksiyonlarla tanımlayıp aitlik kümesini tanımlayan çok sevdiğim bir yapıdır.

Ant Colony Optimization: Bu dönem ilk öğrendiğimde doğadaki diğer canlıların hiyerarşisini merak ettiğim algoritmadır. Kullanılan stateler ve sorumluluklarda bir işbirliği, sonuca ulaşmak için yapılan koku gibi ipucları ve bunların dağıtılması gibi orijinalliklere sahip olan optimizasyon yöntemidir.

6.5 EK

Daha önceden BERT,Glove gibi NLP algoritmalarını okumuştum fakat hiç implementasyonu gerçekleştirmemiştim. Bunların implementasyonu ve gerçekleştirimi, hiperparametre arayışının gerçekleştirilmesi ve son olarakta implementasyonunu Weight and Biases ile trace etmeyi tecrübe etme imkanı buldum.

6.6 EK

Transformer, girdi verilerinin her bir bölümünün önemini farklı şekilde ağırlıklandıran, self-attention mekanizmasını benimseyen bir derin öğrenme modelidir.Genellikle natural language processing (NLP) ve computer vision (CV) alanlarında kullanılır.

Yinelemeli sinir ağları (RNN) gibi transformerlar, çeviri ve metin özetleme gibi görevlere yönelik uygulamalarla doğal dil gibi sıralı giriş verilerini işlemek için tasarlanmıştır.

Ancak, RNN'lerin aksine, transformatörler tüm girişi bir kerede işler. Dikkat mekanizması, giriş dizisindeki herhangi bir konum için context sağlar. Örneğin giriş verileri bir doğal dil cümlesiyse dönüştürücünün her seferinde bir kelimeyi işlemesi gerekmez. Bu, RNN'lerden daha fazla paralelleştirmeye izin verir ve bu nedenle eğitim sürelerini azaltır. Transformer modeli, TensorFlow ve PyTorch gibi standart derin öğrenme frameworklerinde uygulanmıştır.

Transformatörlerden önce son teknoloji ürünü NLP sistemleri (LSTM ve kapılı tekrarlayan birimler (GRU'lar) gibi) kapılı RNN'lere ve ek dikkat mekanizmalarına dayanıyordu. Transformatörler, bir RNN yapısı kullanılmadan bu dikkat teknolojileri üzerine inşa edilmiştir ve attention mekanizmalarının tek başına attention mekanizmalı RNN'lerin performansını yakalayabileceğini göstermiştir.

Attention

Sinir ağlarında dikkat, bilişsel dikkati taklit eden bir tekniktir. Bu etki girdi verilerinin bazı kısımlarını geliştirirken diğer kısımlarını azaltır.Motivasyon ağın verilerin küçük ama önemli kısımlarına daha fazla odaklanması gerektiğidir. Verinin hangi bölümünün diğerinden daha önemli olduğunu öğrenmek bağlama bağlıdır ve gradyan inişiyle eğitilir.

Sequential processing(Sıralı İşleme)

Kapılı RNN'ler tokenleri sırayla işler ve her tokenden sonra görülen verileri temsil eden bir durum vektörü tutar. Model n'inci tokeni işlemek için n-1 tokenine kadar olan cümleyi temsil eden durumu yeni tokenin bilgisiyle birleştirerek, n'e kadar olan cümleyi temsil eden yeni bir durum oluşturur.

Teorik olarak, eğer state her noktada token hakkındaki bağlamsal bilgiyi kodlamaya devam ederse, bir tokenden gelen bilgi, sıranın aşağısına keyfi olarak yayılabilir. Pratikte bu mekanizma kusurludur.Kaybolan gradyan sorunu modelin state'ini uzun bir cümlenin sonunda önceki tokenler hakkında kesin çıkarılabilir bilgi olmadan bırakır. Token hesaplamalarının önceki sonuçlara bağımlılığı modern derin öğrenme donanımında hesaplamayı paralel hale getirmeyi de zorlaştırır. Bu, RNN'lerin eğitimini verimsiz hale getirir.

Self-Attention

Bu sorunlar dikkat mekanizmaları tarafından ele alınmıştır. Dikkat mekanizmaları bir modelin dizi boyunca herhangi bir önceki noktada durumdan çekilmesine izin verir. Dikkat katmanı önceki tüm durumlara erişebilir ve bunları öğrenilmiş bir uygunluk ölçüsüne göre ağırlıklandırabilir ve uzaktaki tokenler hakkında ilgili bilgiler sağlar.

Attention'un öneminin bariz bir örneği bir cümledeki kelimenin anlamını atamak için bağlamın gerekli olduğu dil çevirisidir. İngilizce-Fransızca çeviri sisteminde, Fransızca çıktının ilk kelimesi büyük olasılıkla İngilizce girdinin ilk birkaç kelimesine bağlıdır. Klasik bir LSTM modelinde Fransızca çıktının ilk kelimesini üretmek için modele sadece son İngilizce kelime işlendikten sonra durum vektörü verilir. Teorik olarak bu vektör, modele gerekli tüm bilgileri vererek tüm İngilizce cümle hakkındaki bilgileri kodlayabilir. Uygulamada bu bilgi genellikle LSTM tarafından yetersiz şekilde korunur. Bu sorunu çözmek için bir dikkat mekanizması eklenebilir.Kod çözücüye yalnızca sonuncusuna değil her İngilizce giriş kelimesinin durum vektörlerine erişim izni verilir ve her bir İngilizce giriş durum vektörüne ne kadar katılacağını belirleyen dikkat ağırlıklarını öğrenebilir.

RNN'lere eklendiğinde dikkat mekanizmaları performansı artırır. Transformer mimarisinin gelişimi dikkat mekanizmalarının kendi içinde güçlü olduğunu ve RNN'lerin kalite kazanımlarını dikkatle elde etmek için sıralı tekrarlayan veri işlemenin gerekli olmadığını ortaya koydu. Transformatörler, RNN'siz bir dikkat mekanizması kullanır, tüm tokenleri aynı anda işler ve birbirini izleyen katmanlarda aralarındaki dikkat ağırlıklarını hesaplar. Dikkat mekanizması yalnızca alt katmanlardaki diğer tokenler hakkındaki bilgileri kullandığından hepsi için paralel olarak hesaplanabilir bu da eğitim hızının artmasına neden olur.

	İstenen Madde	Tahmini Not
1	**Yayın Özet, Giriş ve Sonuç Bölümleri (10)**	10
2	**Önceki Çalışmalar / Literatür Taraması (10)**	7
3	**Problem ve Önerilen Yöntem (10)**	10
4	**Deneysel Çalışmalar** ve **Ek1: Başarım İyileştirme (10)**	9
5	**Makale Biçimi, Organizasyonu, Boyutu, Kalitesi, Kaynak ve atıflar (10)**	7
6	**Ek 2 (10)**	10
7	**Ek 3 (10)**	10
8	**Ek 4 (5): Dönem başından beri derste Anlatılan / Öğrendiğiniz en önemli 5 konu / kavram (Yapay Zeka dersinden farklı olarak) isimleri ve her birine ilişkin en az birer satır açıklama.**	10
9	**Ek 5 (5): P2'de öğrendiğiniz yeni yöntem / iyileştirmeler ve varsa ilk defa yaptığınız yapay zeka ve makine öğrenmesi işlemleri / kazandığınız yetenekleri kısaca açıklayınız.**	10
10	**Ek 6 (10): Derin Öğrenmede Transformer Model nedir? Attention Mekanizması nedir? Bunları tanımlayınız ve önemlerini ve getirdikleri avantajları açıklayınız.**	10
11	**Özdeğerlendirme Tablosu (10)**	10
100 üzerinden Toplam Not:		93