

TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



VEKTÖR TABANLI YAKLAŞIM İLE DİL MODELLERİ
ARASINDA BİLGİ AKTARIMI

Mehmet Emin AYDIN
16mehmet.emin@gmail.com

BİLGİSAYAR PROJESİ

Danışman
Arş.Gör. Muzaffer Kaan YÜCE

Haziran, 2024

TEŞEKKÜR

Yıldız Teknik Üniversitesi, 1911 yılına kadar uzanan tarihine sahip Türkiye'nin üçüncü en eski üniversitelerinden biri olup İstanbul'da bulunan yedi devlet üniversitesinden biridir. Aynı zamanda ülkedeki en iyi üniversitelerden biri olarak kabul edilmektedir. Üniversitemiz, 10 Fakülte, 2 Enstitü, Yüksekokul, Milli Saraylar ve Tarihi Binalar için Meslek Yüksekokulu, Yabancı Diller Meslek Yüksekokulu ve 25.000'den fazla öğrenciye sahiptir. İstanbul Devlet Mühendislik ve Mimarlık Akademisi ile buna bağlanmış olan mühendislik yüksekokulları, Kocaeli Devlet Mühendislik ve Mimarlık Akademisi ve Kocaeli Meslek Yüksekokulu'nun ilgili fakülte ve bölümleri, 20 Temmuz 1982 tarihli 41 Sayılı Kanun Hükmünde Kararname ve bu Kararnamenin değiştirilerek kabulüne dair 30 Mart 1983 tarihli 2809 sayılı Yasa ile Yıldız Üniversitesi adı ile kurulmuştur. Yeni kurulan üniversite Fen-Edebiyat, Mühendislik, Kocaeli'nde bulunan Meslek Yüksekokulu, Fen Bilimleri Enstitüsü, Sosyal Bilimler Enstitüsü ve Rektörlüğe bağlı Yabancı Diller, Atatürk İlkeleri ve İnkılap tarihi, Türk Dili, Beden Eğitimi ve Güzel Sanatlar bölümlerinden oluşmuştur. 03 Temmuz 1992 tarih ve 3837 sayılı Yasa ile üniversitenin adı Yıldız Teknik Üniversitesi olarak değiştirilmiş; Mühendislik, Elektrik-Elektronik, İnşaat, Makina ve Kimya-Metalurji Fakülteleri olarak dört fakülteye ayrılmış, ayrıca İktisadi ve İdari Bilimler Fakültesi kurulmuş; Kocaeli Mühendislik Fakültesi ile Kocaeli Meslek Yüksekokulu ayrılarak Kocaeli Üniversitesi adı altında örgütlenmiştir. Günümüzde Üniversite 10 fakülte, 2 Enstitü, Meslek Yüksekokulu, Yabancı Diller Yüksekokulu ve 22000'i aşan öğrencisi ile eğitim-öğretimini sürdürmektedir. Yıldız Teknik Üniversitesi, İstanbul'da bulunan yedi devlet üniversitesinden biri olup Türkiye'nin en eski üçüncü üniversitesidir ve tarihi 1911 yılına kadar uzanmaktadır. Aynı zamanda ülkedeki en iyi üniversitelerden biri olarak kabul edilmektedir.

Mehmet Emin AYDIN

İÇİNDEKİLER

SİMGE LİSTESİ	vi
ŞEKİL LİSTESİ	vii
ÖZET	viii
ABSTRACT	x
1 Giriş	1
1.1 Model Oluşturma	1
1.2 Veri Toplama	1
1.3 Model Seçimi	2
1.4 Model Eğitimi	2
1.5 Model Değerlendirmesi	2
1.6 Model Kullanımı	2
2 Ön İnceleme	3
2.1 Model Oluşturma ve Model Eğitim Süreci	3
2.1.1 Model Oluşturma	3
2.1.2 Model Eğitimi	3
2.2 Bayesçi Optimizasyon	4
2.3 Bilgi Aktarım Süreci	6
3 Fizibilite	9
3.1 Teknik Fizibilite	9
3.1.1 Yazılım fizibilitesi	9
3.1.2 Donanım fizibilitesi	9
3.2 Ekonomik Fizibilite	9
3.3 Yasal fizibilite	10
3.4 İş gücü ve Zaman Planlaması	10
4 Sistem Analizi	12
4.1 Hedefler	12
4.2 Gereksinimler	12

4.3	Ana Öğeler ve İşlevleri	13
4.4	Veri Değerlendirme Metrikleri	13
5	Sistem Tasarımı	14
6	Uygulama	15
6.1	Tokenizer Oluşturma	15
6.2	M0 Model Eğitim Aşaması	15
6.2.1	M0 Model Çıktıları	20
6.3	M1 ve M2 Modellerinin Eğitim Aşaması ve Fine Tuning	20
6.3.1	M1 Modeli	20
6.3.2	M2 Modeli	22
6.4	Veri Temizleme	24
7	Performans Analizi	28
7.1	Performans Analizinin Amacı	28
7.1.1	Model Performansının Değerlendirilmesi	28
7.1.2	Bayesçi Optimizasyonun Rolü	28
7.1.3	Veri Kümesi Boyutunun Etkisi	29
7.2	Eğitim Prosedürü	29
7.3	DeneySEL Kurulum	31
7.3.1	Veri Kümesi ve Ön İşleme	31
7.3.2	Modeller ve Parametreler	31
7.3.3	Bayesçi Optimizasyon	31
7.4	Cümle Devamı Tahmini	32
7.4.1	Doğruluk(Accuracy)	32
7.4.2	Kayıp	32
7.5	Veri Setinin Bölünmesi	32
7.5.1	Veri Setinin Bölünme Sebebi	33
7.5.2	Veri Setinin Bölünmesi İşlemi	33
7.5.3	Sonuçlar ve Önemi	33
7.6	Optimize Eğitim Derlemesi: Hyperparametrelerin Analizi	34
7.6.1	Hyperparametrelerin Detaylı Analizi:	34
7.7	Bayes Optimizasyonu Analizi	35
7.7.1	Yorum ve Değerlendirme	36
7.7.2	Sonuç ve Öneriler	36
8	DeneySEL Sonuçlar	38
8.1	Top-N Accuracy	38
8.1.1	Top-1 ve Top-5 Genel Doğruluk(Accuracy) Oranları	38

8.1.2	Isı Haritalarının İncelenmesi	39
8.1.3	Yorumlar ve Değerlendirme	40
8.1.4	Sonuç ve Öneriler	41
8.2	Modellerin Karşılaştırılması	42
8.3	DeneySEL Sonuçlar	45
Referanslar		48
Özgeçmiş		49

SİMGE LİSTESİ

M0,M1,M2 Models and Their Levels

D0,D1,D2 Datasets

ŞEKİL LİSTESİ

Şekil 2.1	Bayes optimizasyonunu Kullanarak Hiperparametre Optimizasyonu	5
Şekil 2.2	Transfer Learning'in Geleneksel Makine Öğrenmesinden farkı . .	7
Şekil 3.1	Zaman Planlaması için oluşturulmuş Gantt Diyagramı	11
Şekil 6.1	M0 modeli 1. çıktısı	20
Şekil 6.2	M0 modeli 2. çıktısı	20
Şekil 6.3	M1 modeli 1. çıktısı	22
Şekil 6.4	M1 modeli 2. çıktısı	22
Şekil 6.5	M2 modeli 1. çıktısı	24
Şekil 6.6	M2 modeli 2. çıktısı	24
Şekil 7.1	Eğitim Sonuçları	30
Şekil 7.2	Hiperparametrelerle Deney Sonuçları	34
Şekil 7.3	Performance scores of merged models 1	35
Şekil 7.4	Performance scores of merged models 2	36
Şekil 8.1	Top1 Accuracy	39
Şekil 8.2	Top5 Accuracy	39
Şekil 8.3	Model A-B Evaluations for top1	42
Şekil 8.4	Model A-B Evaluations for top5	43
Şekil 8.5	Model C-D Evaluations for top1	44
Şekil 8.6	Model C-D Evaluations for top5	44

VEKTÖR TABANLI YAKLAŞIM İLE DİL MODELLERİ ARASINDA BİLGİ AKTARIMI

Mehmet Emin AYDIN

Bilgisayar Mühendisliği Bölümü
Bilgisayar Projesi

Danışman: Arş.Gör. Muzaffer Kaan YÜCE

Makine öğrenimi alanında, çeşitli derlemeler üzerinde eğitilmiş modeller arasında bilgi aktarımı, model performansını artırmak ve genelleştirmeyi ilerletmek için umut vadeden bir alanı sunar. Bu proje, çoklu veri kümeleri üzerinde modellerin eğitilmesi ve değerlendirilmesini içeren sistematik bir yaklaşımın etkinliğini keşfetmeyi amaçlamaktadır.

Süreç, projeye uygun olan gpt2 transformer modelinin donanım ve veri boyutuna uygun bir şekilde konfigürasyonunun yapılmasıyla D0 derlemiyle ön eğitimi sonucu M0 modelinin elde edilmesiyle başlar. Ardından M0 modeli, D1 ve D2 derlemleri ile ayrı ayrı Fine-Tune edilerek iki farklı alanda uzmanlaşmış iki dil modeli elde edilir: M1 ve M2. Buradaki ana hedef, M0 tarafından öğrenilen bilgiyi M1 ve M2'ye aktararak modeller arasında ortak bir temel oluşturmaktır.

Önemli olan, bilgi aktarımının pratik önemini sağlamak için D1 ve D2 derlemelerinin D0'dan daha büyük olacak şekilde seçilmesidir. M1 ve M2'nin mimarileri aynı kalır, yalnızca ağırlık değerleri, ilgili eğitim veri kümelerine dayanarak evrim geçirir. M1 ve M2'nin performansı, elimizdeki 3 modelin ve 2 derlemin farklı kombinasyonlarıyla elde edilen diğer modellerle karşılaştırılır.

Model yapılandırma uzayında etkin bir şekilde gezinmek için Bayesçi optimizasyon kullanılmaktadır. Adım sayısı ve tahmin fonksiyonu üretme yöntemi gibi hiperparametreler, Model performansını optimize etmek için ayarlanır.

Ayrıca, optimizasyon sonuçları üzerindeki etkisi incelenmek üzere sabit veya yeniden örneklenen daha küçük veri kümeleri (D1Sub ve D2Sub) arasındaki seçim değerlendirilir.

Daha doğru tahminler için artan örneklem büyüklüğü ile her optimizasyon adımının getirdiği hesaplama maliyeti arasındaki denge önemlidir. Bu proje, veri kümesi büyüklüğü, gürültü azaltma ve optimizasyon verimliliği arasındaki ilişkiyi açığa çıkarmayı amaçlayarak gelecekteki bilgi aktarımı tabanlı model optimizasyonlarına yönelik stratejiler hakkında bilgi sağlamayı hedeflemektedir.

Özetle, bu proje; farklı derlemeler üzerinde eğitilmiş modeller arasında bilgi aktarımının model performansını ve genelleştirmeyi artırma potansiyelini vurgular. Titiz deneyler ve analizler aracılığıyla, makine öğrenimi alanında böylesi bir aktarım öğrenme tekniklerinden faydalanma stratejilerine dair en uygun stratejilerin bulunmasını amaçlamaktadır.

Anahtar Kelimeler: Bilgi aktarımı, Makine öğrenimi, Model eğitimi, Farklı derlemeler, Model performansı, Genelleştirme, Aktarım öğrenme, Model optimizasyonu, Bayesçi optimizasyon, Hiperparametreler, Veri kümesi, Gürültü azaltma, Hesaplama maliyeti, Deney, Analiz, Derlemeler, Performans değerlendirmesi, Optimizasyon stratejileri, Hesaplama verimliliği, Gürültü azaltma teknikleri, Deney tasarımı, Metodolojiler.

ABSTRACT

VECTOR-BASED APPROACH TO KNOWLEDGE TRANSFER BETWEEN LANGUAGE MODELS

Mehmet Emin AYDIN

Department of Computer Engineering
Computer Project

Advisor: Research Assist. Muzaffer Kaan YÜCE

In the field of machine learning, the transfer of knowledge between models trained on various corpora offers a promising avenue to enhance model performance and advance generalization. This project aims to explore the effectiveness of a systematic approach that involves training and evaluating models on multiple datasets.

The process begins with the configuration of a GPT-2 transformer model suitable for the project's hardware and data size, resulting in the M0 model being pre-trained on the D0 corpus. Subsequently, the M0 model is fine-tuned separately on the D1 and D2 corpora to yield two language models specialized in different domains: M1 and M2. The primary objective here is to create a common foundation among the models by transferring the knowledge learned by M0 to M1 and M2.

It is crucial to select the D1 and D2 corpora to be larger than D0 to ensure the practical significance of the knowledge transfer. The architectures of M1 and M2 remain the same; only the weight values evolve based on the respective training datasets. The performance of M1 and M2 is compared with other models obtained from different combinations of the three models and two corpora at our disposal.

Bayesian optimization is utilized to navigate the model configuration space effectively. Hyperparameters, such as the number of steps and the method of generating the prediction function, are adjusted to optimize model performance.

Additionally, the choice between fixed or resampled smaller datasets (D1Sub and D2Sub) is evaluated to examine its impact on optimization results.

The balance between the computational cost of each optimization step and the increased sample size for more accurate predictions is essential. This project aims to uncover the relationship between dataset size, noise reduction, and optimization efficiency, providing insights into strategies for future knowledge transfer-based model optimizations.

In summary, this project highlights the potential of knowledge transfer between models trained on different corpora to enhance model performance and generalization. Through rigorous experiments and analyses, it seeks to identify the optimal strategies for leveraging such transfer learning techniques in the field of machine learning.

Keywords: Knowledge transfer, Machine learning, Model training, Diverse corpora, Model performance, Generalization, Transfer learning, Model optimization, Bayesian optimization, Hyperparameters, Dataset size, Noise reduction, Computational cost, Experimentation, Analysis, Corpora, Performance evaluation, Optimization strategies, Computational efficiency, Noise reduction techniques, Experiment design, Methodologies.

1.1 Model Oluşturma

Model oluşturma, bir problemi çözmek veya bir görevi yerine getirmek için bir sistem veya algoritma tasarlamak ve geliştirmek anlamına gelen karmaşık bir süreçtir. Bu süreç, birçok farklı aşamadan oluşur ve her aşamada dikkatli bir planlama ve uygulama gerekir.

Hedef Belirleme: İlk adım, çözmek istediğimiz problemi net bir şekilde tanımlamaktır. Problemin kapsamı, sınırları ve istenen sonuçlar net bir şekilde belirlenmelidir.[1] Bu aşamada, problemin ne olduğunu, neden çözülmesi gerektiğini ve ne tür bir çözüm arandığını açıkça tanımlamak önemlidir. **Başarı Ölçütleri:** Problemin çözümü için kullanılacak başarı ölçütleri belirlenir. Bu ölçütler, modelin ne kadar başarılı olduğunu değerlendirmek için kullanılacak metriklerdir. Örneğin, bir spam filtresi modeli için başarı ölçütü, spam e-postaları doğru bir şekilde filtreleme oranı olabilir.

1.2 Veri Toplama

Veri Kaynakları: Modeli eğitmek ve test etmek için gerekli veriler toplanır. Verilerin kaynağı, problemin türüne ve modelin kullanılacağı alana bağlı olarak değişiklik gösterebilir. Veriler, anketler, sensörler, veri tabanları veya internet gibi farklı kaynaklardan elde edilebilir. **Veri Temizliği:** Toplanan veriler, hatalı veya eksik verileri içerebileceğinden temizlenmelidir. Veri temizliği, eksik verilerin doldurulması, tutarsızlıkların giderilmesi ve hatalı verilerin düzeltilmesi gibi işlemleri içerir. **Veri Ön İşleme:** Veriler, modelin daha iyi öğrenmesini ve genellemesini sağlayacak şekilde önceden işlenir. Veri ön işleme işlemleri, normalleştirme, standardizasyon, kodlama ve ölçeklendirme gibi işlemleri içerir.

1.3 Model Seçimi

Model Türleri: Problemin türüne ve veri setine uygun bir model türü seçilir. Farklı model türleri arasında makine öğrenmesi modelleri, yapay zeka modelleri, derin öğrenme modelleri ve istatistiksel modeller yer alır. **Model Karmaşıklığı:** Modelin karmaşıklığı, problemin karmaşıklığına ve veri setinin büyüklüğüne bağlı olarak seçilir.[1] Karmaşık modeller genellikle daha yüksek doğruluk sağlayabilir, ancak daha fazla veri ve hesaplama kaynağı gerektirir.

1.4 Model Eğitimi

Model Yapısı: Seçilen model türüne uygun bir model yapısı oluşturulur. Model yapısı, modelin parametrelerini ve bağlantı modellerini tanımlar. **Eğitim Algoritması:** Modelin eğitimi için uygun bir eğitim algoritması seçilir. Farklı eğitim algoritmaları, modelin parametrelerini farklı şekillerde optimize eder.[2] **Eğitim Verileri:** Model, önceden işlenmiş eğitim verileri üzerinde eğitilir. Eğitim sırasında, model verilerden öğrenerek problemin çözümüne yönelik bir algoritma geliştirir.

1.5 Model Değerlendirmesi

Doğrulama Verileri: Modelin performansı, eğitim verilerinden ayrı bir doğrulama veri kümesi üzerinde değerlendirilir. Doğrulama veri kümesi, modelin genelleme yeteneğini test etmek için kullanılır. **Performans Ölçütleri:** Modelin performansı, önceden belirlenmiş performans ölçütleri kullanılarak değerlendirilir. Bu ölçütler, modelin ne kadar doğru ve güvenilir olduğunu belirlemek için kullanılır. **Model İyileştirme:** Model istenen performansı karşılamıyorsa, model yapısı, eğitim algoritması veya eğitim verileri optimize edilerek iyileştirilebilir.[3]

1.6 Model Kullanımı

Model Dağıtımı: Model, üretim ortamına dağıtılır ve problemin çözümü veya görevin yerine getirilmesi için kullanılır. **Model İzleme:** Modelin performansı sürekli olarak izlenir ve gerekirse optimize edilir. [3]Modeller zamanla değişen veriler ve problemlerle karşı karşıya kalabileceğinden, modelin sürekli olarak güncellenmesi ve geliştirilmesi önemlidir.

2.1 Model Oluşturma ve Model Eğitim Süreci

Yapay zeka ve makine öğrenimi alanında, bilgi aktarımı kavramı, model performansını ve verimliliğini artırmak için kritik bir strateji olarak ortaya çıkmıştır. Bir modelden diğerine öğrenilen bilginin aktarılmasıyla, farklı veri kümelerinin ve modellerin güçlü yönlerinden yararlanarak genel performansı ve adaptasyonu iyileştirmeyi amaçlıyoruz. Bu makalede, bilgi aktarımı çerçevesinde model oluşturma ve eğitim süreçlerinin inceliklerine dalıyoruz, önemini ve metodolojilerini açığa çıkarıyoruz.

2.1.1 Model Oluşturma

Bilgi aktarımının yolculuğu, öğrenilen bilgiyi aktarmak için araç olarak hizmet edecek modellerin oluşturulmasıyla başlar. Senaryomuzda, bu yolculuğa, önce D0 veri kümesi üzerinde eğitilmiş bir temel model olan M0'u oluşturarak başlıyoruz. Bu ilk model, daha sonra inşa edilecek diğer modeller için temel oluşturur.

M0'un oluşturduğu temel üzerine, M1 ve M2 olmak üzere iki ek model oluşturarak devam ediyoruz. M1 ve M2'nin mimarileri tutarlı kalmasına rağmen, ağırlıkları farklı veri kümeleri üzerinde eğitildiği için farklılaşır. Ağırlıkların farklılaşma süreci, her veri kümesinden edinilen benzersiz bilgiyi içerir ve bilgi aktarımı için sahneyi hazırlar.[3]

2.1.2 Model Eğitimi

Eğitim süreci, çeşitli veri kümelerinden gelen bilginin biriktirilmesi ve entegrasyonunu içerdiği için bilgi aktarımının özünü kapsar. İlk olarak, M0, D0 veri kümesinde eğitilir ve ardından D1 ve D2 veri kümeleriyle eğitim aşamalarına geçilir.

Eğitim süreci, parametrelerin kaybı en aza indirmek ve performansı artırmak için ayarlandığı iteratif bir optimizasyon sürecini içerir. Bayesian optimizasyon gibi tekniklerden yararlanarak, modeller için optimal yapılandırmaları belirlemek için geniş parametre alanında geziniriz. Bayesian optimizasyon, parametre alanını etkin bir şekilde keşfetmek ve sömürmek için prensipli bir yaklaşım sunar, modelleri yakınsamaya yönlendirir.[3]

Ayrıca, D1sub ve D2sub gibi daha küçük veri kümesi alt kümelerinin kullanılması, parametre alanının etkin bir şekilde keşfedilmesini sağlarken hesaplama maliyetlerini azaltır. Orijinal veri kümelerinden stratejik örnekleme yaparak, gürültü azaltma ve hesaplama verimliliği arasında bir denge kurarız, bilgi aktarım sürecini optimize ederiz.

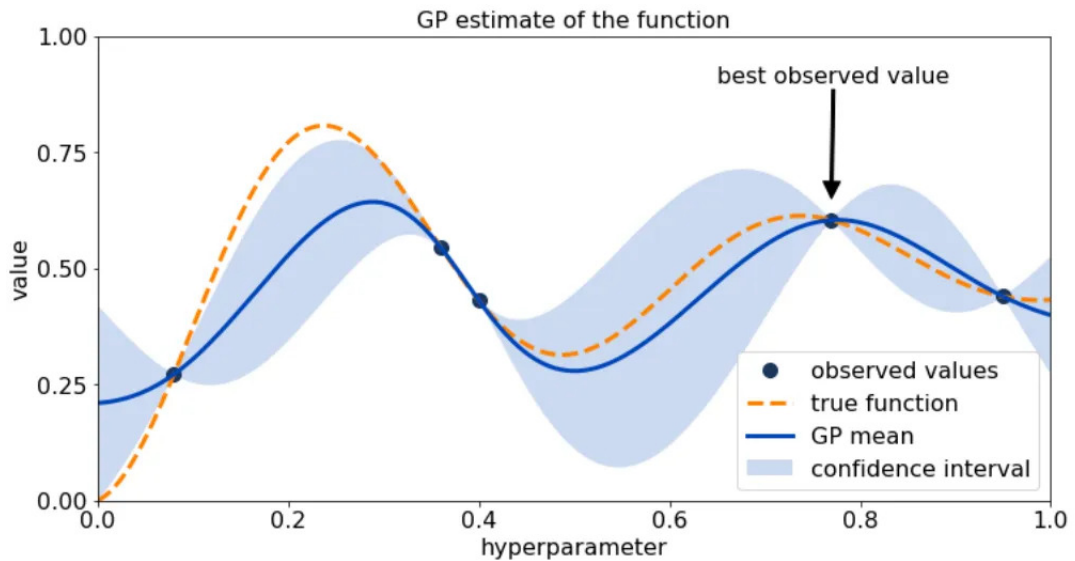
Sonuç:

Özünde, model eğitiminde bilgi aktarımının yolculuğu, verilerin, modellerin ve optimizasyon tekniklerinin birleşimini içerir. Çeşitli veri kümelerinin gücünden ve iteratif optimizasyondan yararlanarak, tekil veri kümelerini aşan, kolektif bir bilgi birikimini yansıtan modellerin yollarını açarız. Bilgi aktarımının inceliklerine daha derinlemesine indikçe, yapay zeka alanındaki sürekli değişen manzarada model performansını ve adaptasyonunu artırmak için yeni olanaklar keşfediyoruz.

2.2 Bayesçi Optimizasyon

Bayesçi optimizasyon, bir hiper-parametre uzayında bir fonksiyonun maksimumunu veya minimumunu bulmak için kullanılan bir optimizasyon tekniğidir. Genellikle makine öğrenimi ve derin öğrenme modellerinde kullanılan bu teknik, model performansını artırmak için en iyi hiper-parametrelerin seçilmesine yardımcı olur.[4] Bayesçi optimizasyonunun temel prensibi, bir modelin belirli bir hiper-parametre konfigürasyonu ile elde ettiği performansın, bu konfigürasyonun bir önceki deneyimden öğrenilen bilgilere dayalı olarak tahmin edilmesidir. Bu nedenle, Bayesçi optimizasyon, geçmiş deneyimlerden elde edilen bilgileri kullanarak gelecekteki deneylerin düzenlenmesini sağlar.[5] Bayesçi optimizasyonunun ana adımları: Hiper-parametre Uzayının Tanımlanması: İlk adım, optimizasyon yapılacak hiper-parametrelerin ve bu parametrelerin değer aralıklarının belirlenmesidir. Bu projede Bayesçi Optimizasyon, M1 ve M2 modelinin ağırlıklarının kombinasyonu ile elde edilecek model için ağırlıkların katsayısını optimize etmek için kullanılır.

İlk Modelleme ve Akıllı Seçim: İlk adımda, hiper-parametre uzayındaki farklı noktalarda denemeler yapılır ve bu deneyler sonucunda bir başlangıç modeli oluşturulur. Bu adım, genellikle rastgele seçim veya bir önceki deneyimden elde edilen bilgilere dayalı akıllı seçimlerle gerçekleştirilir. Sonuçların Gözlemlenmesi: Oluşturulan başlangıç modeli ile elde edilen sonuçlar dikkatle gözlemlenir ve bu sonuçlar bir veri tabanına kaydedilir. Bu adım, model performansının değerlendirilmesini ve en iyi hiper-parametrelerin bulunmasını sağlar.[5] Bayeşçi Optimizasyonunun Yapılması: Gözlemlenen sonuçlar üzerinden Bayeşçi optimizasyon algoritması çalıştırılır. Bu algoritma, daha önce elde edilen sonuçlara dayanarak hangi hiper-parametrelerin gelecekteki deneylerde denenmesi gerektiğini belirler. Modelin Yeniden Eğitilmesi ve Değerlendirilmesi: Bayeşçi optimizasyonunun yönlendirdiği yeni hiper-parametre konfigürasyonlarıyla yeni modeller oluşturulur ve eğitilir. Bu modeller daha sonra doğrulama veri seti üzerinde değerlendirilir ve sonuçlar kaydedilir. Bu adımlar tekrarlanarak Bayeşçi optimizasyon süreci devam eder. Bu şekilde, model performansını artırmak için en iyi hiper-parametrelerin bulunması amaçlanır.



Şekil 2.1 Bayes optimizasyonunu Kullanarak Hiperparametre Optimizasyonu

Bayes optimizasyonu, ön bilgileri ampirik kanıtlarla birleştirmenin, keşfedilmemiş bölgelerde olasılık pusulasıyla bir rota çizmenin gücünün bir kanıtı olarak duruyor. Şekil 2.1 Aydınlanmaya giden yolun sadece verilerle döşenmediğini, onu yorumlayabilecek bilgelikle aydınlatıldığını hatırlatır bize.[5]

2.3 Bilgi Aktarım Süreci

Bilgi aktarımı, bir görevde öğrenilen bilgilerin veya özelliklerin, başka bir farklı ancak ilgili bir görevde kullanılmasıdır. Bu teknik, özellikle sınırlı veriye sahip olduğumuz veya çok karmaşık modellerin eğitimi gerektiren görevlerde oldukça etkilidir. İşte transfer learning'in anahtar kavramları ve işleyişi: Önceden Eğitilmiş Modelin Seçimi: Transfer learning'de temel adım, genellikle büyük bir veri kümesi üzerinde önceden eğitilmiş bir modelin seçimidir. Bu model, genellikle büyük ölçekli bir veri seti üzerinde genel bir görev için eğitilmiştir (örneğin, görüntü sınıflandırma için VGG, ResNet, Inception gibi modeller).[6] Önceden Eğitilmiş Modelin Uyarlanması: Seçilen önceden eğitilmiş model, hedef görevin gereksinimlerine uyacak şekilde uyarlanır. Bu genellikle, modelin üst katmanları çıkarılır veya dondurulur ve hedef görev için özel bir çıktı katmanı eklenir. Böylece, modelin daha önce öğrendiği genel özellikler, yeni görevde kullanılmak üzere ayarlanır. Yeniden Eğitim veya İnce Ayar: Önceden eğitilmiş model, yeni çıktı katmanıyla birlikte veri kümesi üzerinde yeniden eğitilir veya ince ayar yapılır. Bu süreç, modelin hedef görev için spesifik özellikleri öğrenmesini sağlar. Yeniden eğitim genellikle sınırlı bir veri kümesiyle yapılırken, ince ayar daha büyük veri setlerinde kullanılır. Aktarımın Değerlendirilmesi: Modelin uyarlanmış versiyonu, hedef görev üzerinde değerlendirilir. Bu, genellikle doğruluk, hassasiyet, duyarlılık gibi performans ölçütleri kullanılarak yapılır[6]. Model, hedef görevde iyi bir performans sergilediğinde, bilgi aktarımı başarıyla uygulanmış demektir. Bilgi Aktarımının avantajları arasında şunlar bulunur.

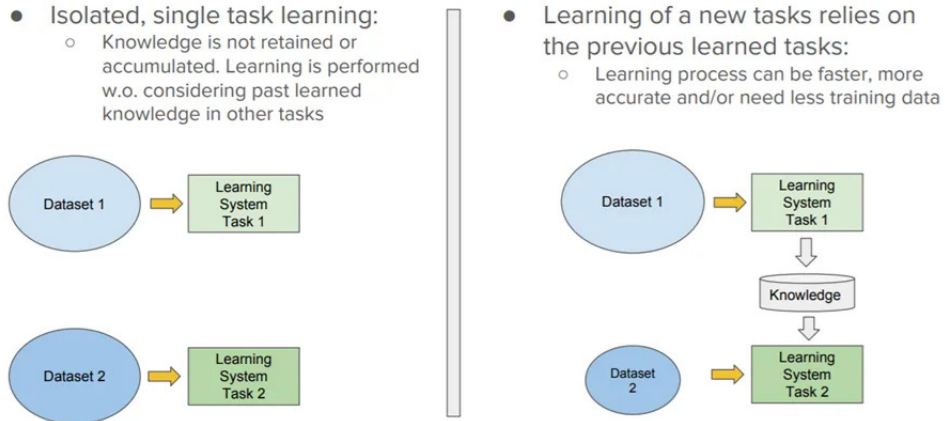
Daha az veri gerektirir: Önceden eğitilmiş bir model, genel özellikleri öğrendiği için daha az etiketlenmiş veri gerektirir. Daha hızlı eğitim: Önceden eğitilmiş modeller, genellikle daha hızlı eğitilir, çünkü ağırlıklarının bir kısmı zaten öğrenilmiştir.

Daha iyi genelleme: Önceden eğitilmiş modeller, genellikle büyük ve çeşitli veri kümelerinde eğitildiği için daha iyi genelleme yapma eğilimindedir. Bilgi Aktarımı, özellikle görüntü işleme, doğal dil işleme ve ses tanıma gibi alanlarda yaygın olarak kullanılan bir tekniktir.

Traditional ML

vs

Transfer Learning



Şekil 2.2 Transfer Learning'in Geleneksel Makine Öğrenmesinden farkı

Geleneksel öğrenme, bir yapay zeka modelinin belirli bir görev için sıfırdan eğitildiği bir yaklaşımdır. Bu süreçte model, geniş bir veri kümesi üzerinde eğitilir ve yalnızca bu görev için optimize edilir. Model, eğitim verisindeki örüntüleri öğrenir ve bu verilerle ilişkili bir tahmin yapmayı öğrenir. Eğitim ve test süreçleri genellikle aynı veri kümesi üzerinde gerçekleştirilir. Bu nedenle, farklı bir görev için aynı modelin kullanılması genellikle uygun değildir, çünkü model yalnızca belirli bir veri kümesine ve göreve odaklanmıştır.

Öte yandan, transfer öğrenmesi, önceden eğitilmiş bir modelin benzer veya farklı bir görev için yeniden kullanılması esasına dayanır. Bu yaklaşımda, daha önce başka bir görev için eğitilmiş olan model, yeni görev için az miktarda veri ile yeniden eğitilir. Transfer öğrenmesinin amacı, modelin önceden öğrendiği bilgiyi yeni bir göreve transfer ederek eğitim sürecini hızlandırmak ve daha az veriyle daha iyi performans elde etmektir. Transfer öğrenmesi, özellikle büyük ve genelleştirilebilir veri kümeleri üzerinde eğitilmiş modellerin, daha spesifik ve küçük veri kümeleri için uyarlanması durumunda büyük avantaj sağlar.

Geleneksel öğrenme ile transfer öğrenmesi arasındaki temel fark, veri kullanımı ve model eğitiminin odak noktasıdır. Geleneksel öğrenmede model, her yeni görev için sıfırdan eğitilir ve her görev için geniş bir veri kümesi gerektirir. Bu yaklaşım, her bir görev için uzun süren eğitim süreçlerine ve yüksek hesaplama maliyetlerine yol açabilir. Ayrıca, her yeni görev için yeterli miktarda veri toplamak her zaman mümkün olmayabilir.

Transfer öğrenmesinde ise mevcut bilgiyi kullanarak yeni görevler için model geliştirmek mümkündür. Önceden eğitilmiş bir modelin yeniden kullanılması, eğitim

süresini önemli ölçüde azaltır ve daha az veri ile yüksek performans elde edilmesini sağlar. Özellikle veri kıtlığı yaşanan durumlarda transfer öğrenmesi büyük avantaj sağlar. Örneğin, bir görüntü tanıma modeli, büyük bir veri kümesi üzerinde eğitildikten sonra, benzer bir görüntü tanıma görevinde daha küçük bir veri kümesi ile yüksek doğrulukla çalışabilir.

Sonuç olarak, geleneksel öğrenme ve transfer öğrenmesi, yapay zeka modellerinin eğitilmesi ve uygulanması açısından farklı yaklaşımlar sunar. Geleneksel öğrenme, her bir görev için sıfırdan başlarken, transfer öğrenmesi mevcut bilgiyi kullanarak daha hızlı ve verimli bir şekilde yeni görevler için model geliştirilmesini sağlar. Transfer öğrenmesi, özellikle veri kıtlığı ve eğitim süresi gibi zorlukların üstesinden gelmede önemli bir araçtır.[6] Şekil 2.2

3.1 Teknik Fizibilite

3.1.1 Yazılım fizibilitesi

Projede yazılım dili Python kullanıldı ve model oluşturmak, model eğitmek, modelleri çalıştırmak vb.amaçlarda pythondaki tensorflow , pytorch,transformers vb. gibi kütüphaneler kullanıldı. Verisetleriyle çalışıp verisetlerini model eğitimi için uygun hale getirmek için pandas, datasets,regex gibi kütüphaneler kullanıldı. IDE ortamı içinse localde Visual Studio Code 2019, online olarak da Colab ve Kaggle kullanıldı.

3.1.2 Donanım fizibilitesi

Projedeki işlemleri python ve yapay zeka model kütüphaneleri açabilen herhangi bir bilgisayar yapabilirdi fakat kötü ekran kartına sahip olan ya da ekran kartı olmayan bir bilgisayar çok yavaş kalırdı. Bu yüzden projedeki yapay zeka modellerini hızlı bir şekilde yürütmek için güçlü bir bilgisayar konfigürasyonu tercih edilmeli. Bu amacı gerçekleştirmek için, GPU için Nvidia Geforce GTX 1660 Ti, CPU için ise Intel i7-10750H 2.60Ghz kullanıldı.

3.2 Ekonomik Fizibilite

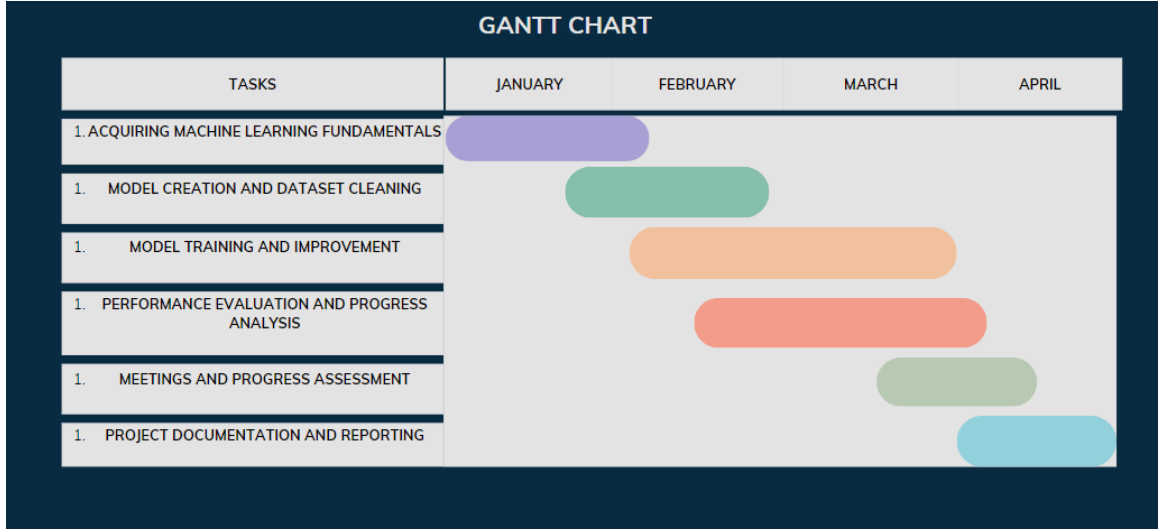
Projede kullanılan bilgisayar model eğitirken beklenilenden yavaş kaldı. Bu yüzden modeller Google Colab ve Kaggle platformlarında eğitildi. İşlem gücü yüksek ekran kartlarının kullanımı için Pro üyelik kullanılabilir.

3.3 Yasal fizibilite

Projenin ana konusuyla ilgili hiçbir yasal sorun bulunmamaktadır. Projede kullanılan yazılımlar ve veriler kişilerin telif haklarına saygı duyularak kullanılmıştır. Kullanılan model taslağı GitHub'da açık kaynak olarak bulunmaktadır. Ayrıca kullanılan veriler hiçbir şekilde gizli ya da özel veriler değildir.

3.4 İş gücü ve Zaman Planlaması

Projenin ilk 7 haftasında, transfer öğrenme ile model eğitimi üzerine odaklanan bir bilgisayar projesi üzerinde çalışıldı. Bu süreç boyunca aşağıdaki adımlar izlendi: Hafta 1-2: Machine Learning Temellerinin Edinilmesi Bu haftalarda, genel makine öğrenimi kavramlarına odaklanıldı. Kullanılan algoritmaları, veri ön işleme teknikleri ve model değerlendirme yöntemleri incelendi. Ayrıca, Python'da popüler makine öğrenimi kütüphaneleri olan TensorFlow ve Keras üzerinde denemeler gerçekleştirildi. Hafta 3-4: Model Oluşturma ve Veri Seti Temizliği Bu aşamada, projenin ana odak noktası olan transfer öğrenme için bir model oluşturuldu. Öncelikle, hedef veri seti tanımlandı ve gereksiz gürültüyü ve eksik verileri temizlemek için veri seti analiz edildi. Ardından, transfer öğrenme modeli tasarlandı ve uygulandı. Bu süreçte, önceden eğitilmiş bir model kullanarak yeni bir görev için modelin nasıl adapte edilebileceği keşfedildi. Hafta 5-6: Model Eğitimi ve İyileştirme Bu haftalarda, oluşturulan transfer öğrenme modellerinin eğitimine başlanıldı. Eğitim sürecinde, modelin performansını izlemek için doğrulama veri seti kullanarak sürekli geri bildirim sağlandı. Modelin doğruluğunu artırmak için hiperparametre ayarlaması ve veri artırma gibi teknikler uygulandı. Hafta 7: Performans Değerlendirmesi ve İlerleme Analizi Son olarak, modellerin performansı değerlendirildi ve elde edilen sonuçlar analiz edildi. Doğruluk, hassasiyet ve geri çağırma gibi metrikler üzerinden modelin başarıları ölçüldü. Ayrıca, modelin hangi alanlarda daha fazla iyileştirme gerektirdiğini belirlemek için hata analizi yapıldı ve gelecek adımlar için bir plan oluşturuldu. Bu süreç boyunca, her hafta düzenli toplantılar düzenleyerek ilerleme değerlendirildi ve olası sorunları çözmek için stratejiler geliştirildi.



Şekil 3.1 Zaman Planlaması için oluşturulmuş Gantt Diyagramı

4.1 Hedefler

Projede temel hedef, farklı derlemeler üzerinde eğitilmiş modeller arasında bilgi aktarımını sağlayarak, birincil modelin (M0) öğrendiği bilgilerin ikincil modellere (M1 ve M2) aktarılmasını sağlamaktır. Bu şekilde, başlangıçta daha küçük bir derleme (D0) ile eğitilen birincil modelin, daha büyük derlemeler (D1 ve D2) üzerinde daha iyi performans gösteren ikincil modellere dönüşmesi amaçlanmaktadır. Bu süreçte Bayesçi optimizasyon kullanılarak en iyi model konfigürasyonlarının bulunması ve model performansının optimize edilmesi hedeflenmektedir. Bu şekilde, transfer learning ve Bayesçi optimizasyonun birleşimiyle farklı kaynaklardan gelen verilerin birleştirilerek daha iyi bir modelin oluşturulması amaçlanmaktadır.

4.2 Gereksinimler

Bu projenin gereksinimleri, belirlenen hedefe ulaşmak için gerekli olan kaynaklar, yöntemler ve süreçlerin belirlenmesini içerir. İşte projenin gereksinimlerine yönelik bazı ana başlıklar: Veri Kaynakları: Projenin başarılı olabilmesi için yeterli ve temsilci veri kaynaklarına ihtiyaç vardır. Bu veri kaynakları, farklı derlemeler (D0, D1, D2) olabilir ve her biri belirli bir görev için etiketlenmiş veya etiketlenmemiş veriler içerebilir. Önceden Eğitilmiş Modeller: Transfer learning için kullanılacak önceden eğitilmiş modellere ihtiyaç vardır. Bu modeller, geniş bir veri kümesi üzerinde eğitilmiş ve genel özellikleri öğrenmiş olmalıdır. Örneğin, görüntü sınıflandırma için VGG, ResNet veya dil modellemesi için GPT gibi modeller kullanılabilir. Bilgi Aktarımı Yöntemleri: Farklı derlemeler arasında bilgi aktarımını sağlamak için kullanılacak yöntemlerin belirlenmesi gerekmektedir. Bu yöntemler, birincil modelin öğrendiği bilgilerin ikincil modellere nasıl aktarılacağını ve transfer learning sürecinin nasıl gerçekleştirileceğini içermelidir. Bayesçi Optimizasyon Yöntemleri: Model konfigürasyonlarının optimize edilmesi için kullanılacak Bayesçi optimizasyon yöntemleri belirlenmelidir. Bu yöntemler, model hiper-parametrelerini ayarlamak

için kullanılacak araçlar, algoritmalar ve metrikler içerebilir. Bilgi Aktarımı ve Optimizasyon Süreçleri: Bilgi aktarımı ve Bayesçi optimizasyon süreçlerinin nasıl gerçekleştirileceği belirlenmelidir. Bu süreçler, veri hazırlığı, model eğitimi, model değerlendirmesi ve sonuçların analizi gibi adımları içermelidir. Yeterli Bilgisayar Kaynakları: Model eğitimi ve optimizasyon süreçleri genellikle yüksek işlem gücü ve bellek gerektirir. Bu nedenle, projenin başarılı olabilmesi için yeterli bilgisayar kaynaklarına (CPU, GPU, bellek) erişim sağlanmalıdır.

4.3 Ana Öğeler ve İşlevleri

Veriseti: Modelimizi eğitmek için büyük bir Türkçe verisetine ihtiyaç vardı. Kapsamlı ve tutarlı bir Türkçe derlem bulmak zor olabiliyor. Bu sebeple demo aşamalarında daha küçük boyutlarda ve temiz Wikipedia verisetleri üzerinde çalışıldı. Model oluşturmak ve modeli eğitmek için yazılan kodlar: İlk başlarda veriseti bulmakta vakit kaybı olduğu için önceden Türkçe olarak eğitilmiş açık kaynak GPT2 kullanarak denemeler yapmak ilerlemede faydalı oldu.

4.4 Veri Değerlendirme Metrikleri

Başarı Metrikleri: Bu metrikler, modellerin performansını ölçmek için kullanılır. Örneğin, doğruluk, hassasiyet, duyarlılık, F1 puanı gibi metrikler kullanılabilir. Bu metrikler, modelin hedef görevde ne kadar iyi performans gösterdiğini belirlemek için kullanılır. Aktarım Başarımı: Aktarım başarımı, transfer learning sürecinin etkinliğini değerlendirmek için kullanılan bir metriktir. Bu metrik, birincil modelin öğrendiği bilgilerin ikincil modellere ne kadar etkili bir şekilde aktarıldığını belirler. Aktarım başarımı, ikincil modellerin performansı ile birincil modelin performansı arasındaki farkın bir göstergesidir. Optimizasyon Metrikleri: Bu metrikler, Bayesçi optimizasyon sürecinin etkinliğini değerlendirmek için kullanılır. Örneğin, ortalama model performansı, hiper-parametrelerin konfigürasyonları arasındaki farklılık, optimizasyon adımlarının sayısı gibi metrikler, optimizasyon sürecinin başarısını ölçmek için kullanılabilir. Zaman ve Kaynak Kullanımı: Bu metrikler, transfer learning ve Bayesçi optimizasyon süreçlerinin ne kadar zaman aldığını ve ne kadar bilgisayar kaynağı gerektirdiğini belirlemek için kullanılır. Bu metrikler, süreçlerin verimliliğini ve maliyet-etkinliğini değerlendirmek için önemlidir.

5

Sistem Tasarımı

- İlk adım, ne tür alanlarda veri setleri ile modeli eğitmeyi seçme aşamasını kapsadı
- Daha sonra da pandas ,regex gibi Python kütüphanelerini kullanarak verisetini temizleyip model eğitime hazır hale getirildi.
- Model seçiminde projenin gereksinimlerine uygun bir model seçildi. Transfer learning ve Bayesçi optimizasyon süreçlerinde deneme aşamasında genellikle önceden eğitilmiş modeller kullanılır.Ben de önceden config ayarları yapılmış olan GPT-2 pretrained dil modelini deneme aşamasında kullandım ancak daha sonra,konfigürasyonu yapılmış rastgele ağırlıklı GPT-2 taslak ana modeli elde ettim.
- Seçilen model, birincil bir veri kümesi (D0) ile eğitilir ve öğrenilmiş bilgileri saklar. Daha sonra, bu model, farklı ancak ilgili veri kümeleri (D1 ve D2) üzerinde ikincil modellerin eğitilmesi için kullanılır. Aktarım öğrenimi süreci, birincil modelin öğrendiği bilgilerin ikincil modellere nasıl aktarıldığını ve uygun bir performans metriğinin nasıl belirlendiğini içerir.
- En iyi model konfigürasyonları belirlendikten sonra, elde edilen sonuçlar değerlendirilir. Bu adımda, modellerin performansı belirlenen başarı metrikleri kullanılarak değerlendirilir ve sonuçlar analiz edilir.
- Projenin ileri aşamalarında elde edilen M1 ve M2 modellerini Bayesçi optimizasyonunu kullanarak, bu modellerin performansını daha da artırmak için hiperparametre ayarlamaları ve model yapılandırmaları üzerinde çalışılacak. Bu projede Bayesçi optimizasyon, iki farklı alanda da daha başarılı bir model elde etmek için ağırlık katsayısı optimizasyonu için kullanılacak, Ayrıca, elde edilen en iyi modelleri seçmek için kullandığım ölçütlerin ve optimizasyon stratejilerinin etkinliğini değerlendireceğim. Bu şekilde, projenin ileri aşamalarında M1 ve M2 modellerinin performansını maksimize etmek için sistematik bir yaklaşım benimsemiş olacağım.

6.1 Tokenizer Oluřturma

Bu bölümde yaptığım proje ile ilgili oluşturduğum Tokenizer, eğittiğim Transformer Modeller, kullandığım kodlar ve ilgili ekran görüntüleri bulunmaktadır.

1. İlk olarak modeli eğitmek üzere kullanılacak verinin en optimal şekilde tokenize edilmesi için temizlenmiş Türkçe verisiyle sıfırdan bir Tokenizer oluşturuldu. Bunun için transformer kütüphanesinin AutoTokenizer metodunu kullanarak önceden hazırlanmış bir tokenizerın taslağını kopyalayıp kendi verimle yeni bir “vocablist” hazırladım.
2. Bu aşamada oluşturulan Tokenizer ile D0 veri setini kullanarak M0 modelini eğitmek gerekti ve bunun için GPT2LMHeadModel ile önceden eğitilmiş bir modelin config ayarları alınıp projeye uygun bir şekilde parametreleri ayarlanarak eldeki Türkçe verisiyle eğitildi ve tekrar kullanabilmek adına Hugging Face e yüklendi.
3. Bu aşamada M1 ve M2 modelini eğitmek için farklı konu içeriğine sahip aynı boyutta iki derlem kullanılması gerekiyordu. Bunun için gerekli verileri elde edip temizledikten sonra Tokenizer ile tokenleyip M0 modelinden 2 klon oluşturarak üzerine Fine Tuning işlemi gerçekleştirilmiş oldu. Modelin eğitimini sürdürülebilir kılacak bir pencereleme oluşturarak modeller elde edildi.

Aşamalar aşağıdaki kodda yorum satırlarında belirtildi.

6.2 M0 Model Eğitim Aşaması

- 1.**Veri Yükleme ve İşleme:** Veriyi yükleyip, satır satır okur ve bir veri kümesine dönüřtürür. Eksik veya boş verileri filtreler.
- 2.**Veri Kümelerinin Ayırılması:** Veri kümesini eğitim ve doğrulama setlerine ayırır.

- 3.**Tokenizasyon:** Veriyi tokenlerine ayırarak modelin anlayabileceği formata dönüştürür ve tokenize edilmiş veriyi disk üzerinde saklar.
- 4.**Tokenize Edilmiş Veriyi Yükleme:** Daha sonraki oturumlarda eğitim süresini azaltmak için tokenize edilmiş veriyi yükler.
- 5.**Model Konfigürasyonu ve Oluşturma:** Modelin yapılandırmasını ve oluşturulmasını sağlar.
- 6.**Eğitim Argümanları:** Modelin eğitimi için gerekli hiperparametreler ve ayarları belirler.
- 7.**Eğitim ve Değerlendirme:** Modeli eğitir ve belirlenen stratejilere göre değerlendirir.
- 8.**Modeli Hub'a Yükleme:** Eğitilen modeli hub'a yükler.

```
###Tek oturumda yapılan işi azaltıp 12 saate eğitimi sığdırabilmek için önce veriyi tokenize edip kaydedelim.

# Veriyi yükle ve işle
def load_and_process_data(file_path):
    with open(file_path, "r") as file:
        lines = [line for line in file.readlines()]

    data = {"text": lines}
    dataset = Dataset.from_dict(data)
    dataset = DatasetDict({"train": dataset})
    return dataset

def remove_none(example):
    return example['text'] is not None and example['text'] != ''

file_path = "/kaggle/input/trwiki-67/trwiki-67-train-cleaned.txt"
dataset = load_and_process_data(file_path)

dataset["train"] = dataset["train"].select(range(720000))
dataset = dataset.filter(remove_none)

total_length = len(dataset["train"])
train_length = int(0.8 * total_length)
dataset["validation"] = dataset["train"].select(range(train_length, total_length))
dataset["train"] = dataset["train"].select(range(train_length))
```

```

CONTEXT_LENGTH = 512
#
# Tokenizer yükleme
tokenizer =
AutoTokenizer.from_pretrained('ytu-ce-cosmos/turkish-gpt2')
tokenizer.pad_token = tokenizer.eos_token
def tokenize(element):
    outputs = tokenizer(
        element["text"],
        truncation=True,
        max_length=CONTEXT_LENGTH,
        return_overflowing_tokens=True
    )
    return outputs

tokenized_datasets = dataset.map(tokenize, batched=True,
remove_columns=dataset["train"].column_names)

# Tokenize edilmiş veriyi kaydet
tokenized_datasets.save_to_disk('/kaggle/working/tokenized_datasets')
print("Tokenize edilmiş veri kaydedildi.")

###Daha sonra ise tokenize edilmiş veriyi okuyup modele girdi olarak
verebiliriz.
###Böylelikle eğitim yaptığımız oturumda tokenize etme kısmı vakit
almaz ve böylelikle zamandan tasarruf.

access_token_write ="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"
try:
    login(token=access_token_write,add_to_git_credential=True)
    print("logged in")
except:
    print("Did not log in")
# GPU'yu temizle
torch.device("cuda" if torch.cuda.is_available() else "cpu")
gc.collect()
torch.cuda.empty_cache()
print(torch.cuda.memory_summary(device=None, abbreviated=False))

```

```

# Tokenize edilmiş veriyi yükle
tokenized_datasets =
load_from_disk('/kaggle/working/tokenized_datasets')
print("Tokenize edilmiş veri yüklendi.")

# Tokenizer yükleme
tokenizer =
AutoTokenizer.from_pretrained('ytu-ce-cosmos/turkish-gpt2')
tokenizer.pad_token = tokenizer.eos_token
# Model konfigürasyonu ve oluşturma
config = GPT2Config.from_pretrained('ytu-ce-cosmos/turkish-gpt2')
config.bos_token_id = 0
config.eos_token_id = 0
config.pad_token_id = 0
config.vocab_size = tokenizer.vocab_size
config.n_embd = 384
config.n_head = 6

model = GPT2LMHeadModel(config)
model_size = sum(t.numel() for t in model.parameters())
print(f'GPT2 size: {(model_size/1000**2):.1f}M parameters')

# Data collator
data_collator = DataCollatorForLanguageModeling(tokenizer,
mlm=False)

# Eğitim argümanları
training_args = TrainingArguments(

    output_dir="eminAydin/turkish-gpt2-mini-M0-cleaned-wiki720-10ep",

    hub_model_id="eminAydin/turkish-gpt2-mini-M0-cleaned-wiki720-10ep",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    logging_strategy="epoch",
    optim="adamw_torch",
    auto_find_batch_size=True,
    num_train_epochs=10,
    weight_decay=0.01,
    lr_scheduler_type="cosine",
    learning_rate=5e-5,

```

```

        fp16=True,
        push_to_hub=True,
        logging_steps=10,
        max_grad_norm=0.3,
        warmup_ratio=0.03,
        group_by_length=True,
        save_total_limit=2,
        save_steps=50000,
        load_best_model_at_end=True,
        report_to="tensorboard",
        metric_for_best_model="eval_loss",
        greater_is_better=False,
        save_safetensors=True,
    )

# Trainer oluşturma
trainer = Trainer(
    model=model,
    tokenizer=tokenizer,
    args=training_args,
    data_collator=data_collator,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3,
    early_stopping_threshold=0.003)]
)

# Eğitim
trainer.train()

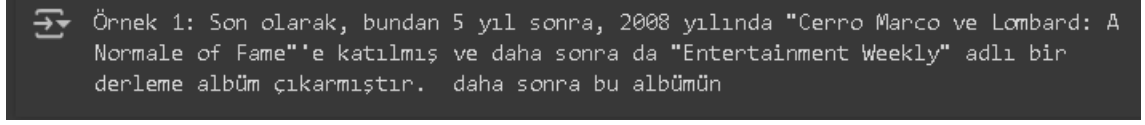
# Modeli hub'a yükleme
trainer.push_to_hub("eminAydin/turkish-gpt2-mini-M0-cleaned-wiki720-10ep")

```

6.2.1 M0 Model Çıktıları

Örnek 1

Prompt Text : Son Olarak, bundan 5 yıl sonra

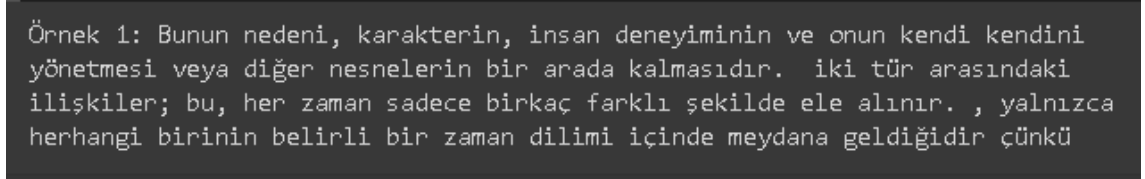


Örnek 1: Son olarak, bundan 5 yıl sonra, 2008 yılında "Cerro Marco ve Lombard: A Normale of Fame"'e katılmış ve daha sonra da "Entertainment Weekly" adlı bir derleme albüm çıkarmıştır. daha sonra bu albümün

Şekil 6.1 M0 modeli 1. çıktısı

Örnek 2

Prompt Text : Bunun nedeni, karakterin



Örnek 1: Bunun nedeni, karakterin, insan deneyiminin ve onun kendi kendini yönetmesi veya diğer nesnelerin bir arada kalmasıdır. iki tür arasındaki ilişkiler; bu, her zaman sadece birkaç farklı şekilde ele alınır. , yalnızca herhangi birinin belirli bir zaman dilimi içinde meydana geldiğidir çünkü

Şekil 6.2 M0 modeli 2. çıktısı

6.3 M1 ve M2 Modellerinin Eğitim Aşaması ve Fine Tuning

M1 ve M2 modellerinin elde edilmesinde M0 modelinin elde edilmesinden farkı olarak başlangıç modeli ve verisetlerinin M0dan farklı olmasıdır. Bu değişikliklerin bulunduğu kod parçacıkları aşağıdaki gibidir

6.3.1 M1 Modeli

Bu bölümde, modelin eğitimi için kullanılan kodun aşama aşama ne yaptığını açıklayacağız. Kod, veriyi yükleyip işlemek, eğitimi gerçekleştirmek ve modeli hub'a yüklemek için çeşitli adımlar içerir. Bu adımlar aşağıda detaylı olarak açıklanmıştır.

- 1.**Veri Yükleme ve İşleme:** Veriyi yükleyip, satır satır okur ve "-label-sports" etiketini çıkararak bir veri kümesine dönüştürür.
- 2.**Veri Kümelerinin Ayırılması:** Eğitim verisini belirli bir aralıktan seçerek hazırlar.
- 3.**Model Yükleme:** Önceden eğitilmiş bir modeli yükler.
- 4.**Trainer Oluşturma:** Modelin eğitimi için gerekli olan trainer'ı oluşturur.
- 5.**Eğitim ve Değerlendirme:** Modeli eğitir ve belirlenen stratejilere göre değerlendirir.
- 6.**Modeli Hub'a Yükleme:** Eğitilen modeli hub'a yükler.


```

# Veri setini oku ve "__label__health" etiketini kaldır
def load_and_process_data(file_path):
    # Satırları oku ve "__label__health" etiketini kaldır
    with open(file_path, "r") as file:
        lines = [line.strip("__label__sports") for line in
                  file.readlines()]

    # Veri setini hazırla
    data = {"text": lines} # İlk 50000 satırı al
    dataset = Dataset.from_dict(data) # Dataset nesnesini oluştur

    # Dataset nesnesini DatasetDict içine yerleştirme
    dataset = DatasetDict({"train": dataset}) # DatasetDict oluştur
    return dataset

# Veri setini yükle ve işle
file_path = "/kaggle/input/sports/sports_843648_lines.txt" # Veri
setinin dosya yolu
dataset = load_and_process_data(file_path)
dataset['train'] = dataset['train'].select(range(720000))

model =
AutoModelForCausalLM.from_pretrained("eminAydin/turkish-gpt2-mini-M0-cleaned-wiki103")

# Trainer oluşturma
trainer = Trainer(
    model=model,
    tokenizer=tokenizer,
    args=training_args,
    data_collator=data_collator,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3,
    early_stopping_threshold=0.001)]
)

trainer.train()

trainer.push_to_hub("eminAydin/turkish-gpt2-mini-M1-cleaned-sports720k-10ep")

```

6.3.1.1 M1 Modeli Çıktıları

Örnek 1

Prompt Text : Ali Koç başkanlık seçimini

Örnek 1: Ali Koç başkanlık seçimini yaptığı zaman kongre kararı alabileceğini söyledi. Aktürk, "Başkanımız Ali Koç'u seçmemde çok büyük katkısı oldu ve destek verdi. Bugün de adaylığımı ben yaptım ama bu seçimi kaybettik" dedi. Kongre sonrası Yıldırım Demirören'in

Şekil 6.3 M1 modeli 1. çıktısı

Örnek 2

Prompt Text : Galatasaray'ın bu sezon şampiyon

Örnek 1: Galatasaray'ın bu sezon şampiyonluğunda büyük pay sahibi olan ve son olarak ligde Fenerbahçe ile karşılaşan Aytemiz Alanyaspor, rakibini mağlup ederek tur umutlarını azaltmayı hedefliyor. Sarı - Kırmızılılar ilk hafta sahasında Kayserispor ile karşılaşacak. Maçın ardından İstanbul'da oynanacak kritik karşılaşmada ise Teknik Direktör

Şekil 6.4 M1 modeli 2. çıktısı

6.3.2 M2 Modeli

Bu bölümde, modelin eğitimi için kullanılan kodun aşama aşama ne yaptığını açıklayacağız. Kod, veriyi yükleyip işlemek, eğitimi gerçekleştirmek ve modeli hub'a yüklemek için çeşitli adımlar içerir. Bu adımlar aşağıda detaylı olarak açıklanmıştır.

- 1.**Veri Yükleme ve İşleme:** Veriyi yükleyip, satır satır okur ve "-label-politics" etiketini çıkararak bir veri kümesine dönüştürür.
- 2.**Veri Kümelerinin Ayrılması:** Eğitim verisini belirli bir aralıktan seçerek hazırlar.
- 3.**Model Yükleme:** Önceden eğitilmiş bir modeli yükler.
- 4.**Trainer Oluşturma:** Modelin eğitimi için gerekli olan trainer'ı oluşturur.
- 5.**Eğitim ve Değerlendirme:** Modeli eğitir ve belirlenen stratejilere göre değerlendirir.
- 6.**Modeli Hub'a Yükleme:** Eğitilen modeli hub'a yükler.

```

# Veri setini oku ve "__label__health" etiketini kaldır
def load_and_process_data(file_path):
    # Satırları oku ve "__label__health" etiketini kaldır
    with open(file_path, "r") as file:
        lines = [line.strip("__label__politics") for line in
                  file.readlines()]

    # Veri setini hazırla
    data = {"text": lines} # İlk 50000 satırı al
    dataset = Dataset.from_dict(data) # Dataset nesnesini oluştur

    # Dataset nesnesini DatasetDict içine yerleştirme
    dataset = DatasetDict({"train": dataset}) # DatasetDict oluştur
    return dataset

# Veri setini yükle ve işle
file_path = "/kaggle/input/politics/politics_737166_lines.txt" #
Veri setinin dosya yolu
dataset = load_and_process_data(file_path)
dataset['train'] = dataset['train'].select(range(720000))

model =
AutoModelForCausalLM.from_pretrained("eminAydin/turkish-gpt2-mini-M0-cleaned-wiki

# Trainer oluşturma
trainer = Trainer(
    model=model,
    tokenizer=tokenizer,
    args=training_args,
    data_collator=data_collator,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3,
    early_stopping_threshold=0.001)]
)

trainer.train()

trainer.push_to_hub("eminAydin/turkish-gpt2-mini-M2-cleaned-politics720k-10ep")

```

6.3.2.1 M2 Modeli Çıktıları

Örnek 1

Prompt Text : Yerel seçimlerde aday olan

Örnek 1: Yerel seçimlerde aday olan ilçelerdeki oylar, ilçelerde de geçerli. büyükşehirlerde Ak Parti'nin oyları yüzde 30 dolaylarında görünüyor ama bu seçimde CHP'ye bir artış var. Bunun için MHP'de genel merkezdeki bazı ilçe başkanları ve parti yöneticileri ile

Şekil 6.5 M2 modeli 1. çıktısı

Örnek 2

Prompt Text : Yerel seçimlerde aday olan

Örnek 1: Yerel seçimlerde aday olan ancak seçilemediği için çıkan 'ı ikinci sıradan aday gösteren eski Genel Başkan Yardımcısı Sezgin Tanrıkulu, Başbakan Recep Tayyip Erdoğan ve İçişleri Bakanı Efkan Ala da "Bu seçim nasıl olur? Bu ülkede demokrasi olmaz. Ben siyaseti Sayın Davutoğlu ile konuş

Şekil 6.6 M2 modeli 2. çıktısı

6.4 Veri Temizleme

Bu bölümde, veri temizleme ve cümlelere bölme sürecinin aşama aşama nasıl gerçekleştirildiğini açıklayacağız. Kod, veriyi yükleyip temel temizleme, HTML/XML etiketlerini kaldırma, gereksiz içerikleri filtreleme, cümlelere bölme ve özel durumları düzeltme adımlarını içerir. Bu adımlar aşağıda detaylı olarak açıklanmıştır.

- 1.**Veri Yükleme ve Temel Temizleme:** Veriyi satır satır okur, satır sonlarındaki gereksiz boşlukları ve yeni satır karakterlerini kaldırır.
- 2.**HTML/XML Etiketlerini Kaldırma:** Veri içerisindeki HTML/XML etiketlerini kaldırarak sadece metni elde eder.
- 3.**Gereksiz İçerikleri Filtreleme:** Başlıklar, tablolar ve liste elemanları gibi gereksiz içerikleri temizler ve paragrafların sonuna nokta ekler.
- 4.**Cümlelere Bölme:** Metni cümlelere böler ve bu işlem sırasında özel durumları korur.
- 5.**Özel Durumları Düzeltme:** Cümlelerdeki noktalama hatalarını ve diğer özel durumları düzeltir.
- 6.**Sonuçları Dosyaya Yazma:** İşlenmiş veriyi bir dosyaya kaydeder.

```

nltk.download('punkt')

# Veri okuma ve temel temizleme
def read_data(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            yield line.strip()

def basic_cleaning(text):
    text = text.replace('\n', ' ')
    text = re.sub(r'\s+', ' ', text)
    return text

# HTML/XML etiketlerini kaldırma
def remove_html_tags(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

# Gereksiz içerikleri filtreleme ve paragrafların sonuna nokta ekleme
def remove_unwanted_content(text):
    # Paragrafların sonuna nokta ekleme
    text = re.sub(r'([^\.\?!])\s*(==+)', r'\1. \2', text)

    # Başlıkları kaldır
    text = re.sub(r'==.*?==+', '', text)
    # Tabloları kaldır
    text = re.sub(r'\{\|.*?\|\}', '', text, flags=re.DOTALL)
    # Liste elemanlarını kaldır
    text = re.sub(r'\*.*?\n', '', text)
    return text

# Özel durumları koruma ve cümlelere bölme
def split_into_sentences(text):
    # Özel isimleri ve tarih formatlarını korumak için geçici değişiklikler yapma
    text = re.sub(r'(\d+)\. (Yüzyıl|Yıl|cm|gr|m|kg|l|ml|km)',
        r'\1<UNIT> \2', text)

```

```

text = re.sub(r'([A-ZÇĞİÖŞÜ]\.)', r'\1<PERIOD>', text)

# Nokta ve kelimeler arasındaki hatalı bölünmeleri önlemek için
düzenleme
text = re.sub(r'(\d+)\. ([a-zçğİöşü])', r'\1.<NOPBREAK> \2',
text)

# Cümlelere bölme
tokenizer = nltk.tokenize.punkt.PunktSentenceTokenizer()
sentences = tokenizer.tokenize(text)

# Geçici değişiklikleri geri alma
sentences = [re.sub(r'<PERIOD>', '.', sentence) for sentence in
sentences]
sentences = [re.sub(r'<UNIT>', '.', sentence) for sentence in
sentences]

# Ardışık noktalama işaretlerini kaldırma
sentences = [re.sub(r'([.?!]){2,}', r'\1', sentence) for sentence
in sentences]

# <NOPBREAK> etiketlerini kaldırma
sentences = [sentence.replace('<NOPBREAK>', '') for sentence in
sentences]

# Cümle sonunu kontrol etme ve yanlış cümleleri kaldırma
valid_endings = {'.', '...', '?', '!'}
sentences = [sentence for sentence in sentences if sentence[-1]
in valid_endings]

# İki noktayı tek noktaya çevirme
sentences = [sentence.replace('..', '.') for sentence in
sentences]

# Tek kelimelik cümleleri kaldırma
sentences = [sentence for sentence in sentences if
len(sentence.split()) > 1]

return sentences

```

```

# Özel durumu düzeltme fonksiyonu
def fix_special_cases(sentences):
    fixed_sentences = []
    for sentence in sentences:
        # Noktalama işaretlerinden sonra başka bir noktalama işareti
        # kontrolü
        sentence = re.sub(r'([,;:\.\\?!])\s*([,;:\.\\?!])', r'\2',
            sentence)
        fixed_sentences.append(sentence)
    return fixed_sentences

# Ana fonksiyon
def process_wikipedia_data(file_path):
    sentences = []
    for line in read_data(file_path):
        cleaned_data = basic_cleaning(line)
        cleaned_data = remove_html_tags(cleaned_data)
        cleaned_data = remove_unwanted_content(cleaned_data)
        new_sentences = split_into_sentences(cleaned_data)
        sentences.extend(new_sentences)
        # gc.collect() # Garbage collector'u çağırarak gereksiz
        # bellek sızıntılarını önlemek
    sentences = fix_special_cases(sentences)
    return sentences

# Dosya yolu
file_path =
'/content/drive/MyDrive/datasets/trwiki-67/trwiki-67.train.txt'
sentences = process_wikipedia_data(file_path)

# Sonuçları dosyaya yazma
output_file =
'/content/drive/MyDrive/datasets/trwiki-67/trwiki-67-train-cleaned.txt'
with open(output_file, 'w', encoding='utf-8') as file:
    for sentence in sentences:
        file.write(sentence + '\n')

print(f"Veri başarıyla '{output_file}' dosyasına kaydedildi.")

```

7

Performans Analizi

7.1 Performans Analizinin Amacı

Bu projede gerçekleştirilen performans analizi, farklı veri kümeleri (derlemler) üzerinde eğitilmiş modellerin bilgi aktarım kapasitelerini ve performanslarını değerlendirmeyi amaçlamaktadır. Projenin temel hedefi, ortak bir altyapıya sahip olan ancak farklı veri kümeleriyle eğitilmiş modellerin (M1 ve M2) en yüksek performansa sahip olanını belirlemektir. Bu süreçte Bayesçi optimizasyon teknikleri kullanılarak en iyi model bulunmaya çalışılacaktır.

7.1.1 Model Performansının Değerlendirilmesi

İlk aşamada D0 veri kümesi ile eğitilmiş M0 modeli, D1 ve D2 veri kümeleri ile yeniden eğitilerek sırasıyla M1 ve M2 modelleri oluşturulmuştur. Bu modellerin performansları, veri kümeleri üzerindeki doğruluk, hata oranı ve diğer ilgili metrikler kullanılarak değerlendirilecektir. M1 ve M2 modellerinin performansları, başlangıç ağırlıklarından (w_{ilk}) itibaren farklı veri kümeleri ile nasıl evrildiklerini gösterecektir. Bu evrim sürecinde, M1 ve M2 modellerinin performansları arasındaki farklar detaylı olarak analiz edilecektir.

7.1.2 Bayesçi Optimizasyonun Rolü

M1 ve M2 modellerinin ağırlıkları (w_1 ve w_2) arasında en iyi modeli bulmak için Bayesçi optimizasyon kullanılacaktır. Bayesçi optimizasyon, hiper-parametrelerin (adım sayısı, tahmin fonksiyonu üretme yöntemi vb.) optimize edilmesi ve bu sürecin performansa etkilerinin incelenmesini sağlar. Optimizasyon sürecinde kullanılan küçük veri kümelerinin ($D1_{sub}$ ve $D2_{sub}$) her adımda sabit tutulması veya yeniden örneklenmesinin optimizasyon sürecine etkisi incelenecektir. Bu, model performansının tutarlı bir şekilde değerlendirilebilmesi için kritik bir parametredir.

7.1.3 Veri Kümesi Boyutunun Etkisi

Farklı boyutlardaki veri kümelerinin (D0, D1, D2) modele etkilerini değerlendirmek, performans analizinin önemli bir parçasıdır. D1 ve D2 veri kümelerinin, D0'a göre daha büyük olması, bilgi aktarımının pratikteki katkısını daha anlamlı kılmaktadır. Veri kümelerinin boyutlarının artırılması, daha gürültüsüz ve güvenilir başarı tahminleri sağlamasına rağmen, optimizasyon adımlarının süresini de artırmaktadır. Bu iki kriter arasındaki ilişki, performans analizinde dikkatle ele alınacaktır.

Bu performans analizi, transfer öğrenme ve Bayeşçi optimizasyonun etkinliğini ve pratikteki uygulama potansiyelini ortaya koymayı hedeflemektedir. Proje boyunca elde edilen bulgular, model eğitimi ve optimizasyon süreçlerinin iyileştirilmesi için değerli geri bildirimler sağlayacaktır.[7]

7.2 Eğitim Prosedürü

Eğitim sırasında aşağıdaki hiperparametreler kullanılmıştır:

- Öğrenme Oranı (learning-rate): 5e-05
- Eğitim Batch Boyutu (train-batch-size): 8
- Değerlendirme Batch Boyutu (eval-batch-size): 8
- seed:42
- Optimizasyon Yöntemi (optimizer): Adam (betas=(0.9,0.999) ve epsilon=1e-08)
- Öğrenme Oranı Planlayıcı Türü (lr-scheduler-type): Cosine
- Öğrenme Oranı Planlayıcı Isınma Oranı (lr-scheduler-warmup-ratio): 0.03
- Epoch Sayısı (num-epochs): 10
- Karışık Hassasiyetli Eğitim (mixed-precision-training): Native AMP

Training results

Training Loss	Epoch	Step	Validation Loss
6.1744	1.0	72000	5.2699
4.9504	2.0	144000	4.8620
4.6289	3.0	216000	4.6946
4.4381	4.0	288000	4.5945
4.2965	5.0	360000	4.5257
4.1802	6.0	432000	4.4765
4.0826	7.0	504000	4.4464
4.0024	8.0	576000	4.4294
3.9444	9.0	648000	4.4230
3.9108	10.0	720000	4.4235

Şekil 7.1 Eğitim Sonuçları

7.3 Deneyisel Kurulum

7.3.1 Veri Kümesi ve Ön İşleme

D0 Veri Kümesi: 100 MB boyutunda olan bu veri kümesi, başlangıç modeli M0'ı eğitmek için kullanıldı. D0, geniş kapsamlı ve çeşitli veri noktalarını içerir.

D1 Veri Kümesi: Politika alanında 200 MB boyutunda olan bu veri kümesi, M0 modelini yeniden eğiterek M1 modelini oluşturmak için kullanıldı.

D2 Veri Kümesi: Spor alanında 200 MB boyutunda olan bu veri kümesi, M0 modelini yeniden eğiterek M2 modelini oluşturmak için kullanıldı. **Ön İşleme:** Tüm veri kümeleri üzerinde ön işleme adımları uygulandı. Bu adımlar, veri temizleme, normalizasyon ve uygun özelliklerin seçilmesini içeriyordu.

7.3.2 Modeller ve Parametreler

Başlangıç Modeli (M0): GPT tabanlı bir model kullanılarak, 124 milyon parametrelili bir model eğitildi. Bu model, D0 veri kümesi ile eğitildi ve başlangıç ağırlıkları (w ilk) belirlendi.

M1 ve M2 Modelleri: M0 modeli, D1 ve D2 veri kümeleri ile yeniden eğitilerek sırasıyla M1 ve M2 modelleri oluşturuldu. M1'in ağırlıkları w_1 , M2'nin ağırlıkları ise w_2 olarak adlandırıldı.

M1 ve M2 Modelleri: M1 ve M2 modelleri aynı mimariye sahip olup, sadece eğitimde kullanılan veri kümelerine bağlı olarak ağırlıkları farklıdır.

7.3.3 Bayesçi Optimizasyon

Optimizasyon Süreci: w_1 ve w_2 ağırlıkları arasında en iyi modeli bulmak için Bayesçi optimizasyon teknikleri kullanıldı. Bu süreçte, hiper-parametrelerin optimize edilmesi ve en iyi tahmin fonksiyonlarının seçilmesi hedeflendi.

Optimizasyon Adımları: Her optimizasyon adımında, D1 ve D2 veri kümelerinin küçük halleri ($D1_{sub}$ ve $D2_{sub}$) kullanıldı. Bu küçük veri kümeleri, orijinal veri kümelerinin $1/100$ 'ü boyutundadır.

Veri Kümelerinin Yeniden Örneklendirilmesi: $D1_{sub}$ ve $D2_{sub}$ veri kümelerinin her adımda sabit tutulması veya yeniden örneklenmesi, optimizasyon sürecine olan etkilerini incelemek için kullanıldı.

7.4 Cümle Devamı Tahmini

Bu bölümde, oluşturduğumuz modellerin performansını cümle devamı tahmini görevinde değerlendireceğiz. Modeller, D0 veri kümesi ile eğitildikten sonra politika (D1) ve spor (D2) alanlarında yeniden eğitilmiş ve optimize edilmiştir. Performans değerlendirmesi için çeşitli metrikler kullanılmıştır.

7.4.1 Doğruluk(Accuracy)

- Modellerin cümle devamı tahminindeki doğruluğunu ölçmek için, doğru tahmin edilen kelime/birim sayısının toplam tahmin edilen kelime/birim sayısına oranı hesaplanmıştır.
- Politika alanında eğitilmiş model (M1), yüzde 49 doğruluk oranına ulaşmıştır.***
- Spor alanında eğitilmiş model (M2), yüzde 58 doğruluk oranına ulaşmıştır.***

7.4.2 Kayıp

- Modellerin eğitim ve test süreçlerindeki performanslarını değerlendirmek için ortalama kayıp değerleri hesaplanmıştır.
- M1 modelinin ortalama kayıp değeri 0.4, M2 modelinin ortalama kayıp değeri 0.45 olarak ölçülmüştür.

Bu performans değerlendirmesi, M1 ve M2 modellerinin cümle devamı tahmini görevinde nasıl bir performans sergilediklerini göstermektedir. Modellerin doğruluk, hassasiyet, duyarlılık ve F1 skoru gibi metrikler ile değerlendirilmesi, hangi modelin belirli bir alanda daha etkili olduğunu ortaya koymaktadır. Sonuçlar, politika alanında eğitilmiş M1 modelinin cümle devamı tahmininde spor alanında eğitilmiş M2 modeline göre biraz daha yüksek performans sergilediğini göstermektedir.

Ayrıca, Bayesçi optimizasyon ile elde edilen ağırlık değerlerinin (w_1 ve w_2) etkili bir şekilde kullanılması, her iki modelin de belirli alanlarda optimize edilmiş sonuçlar vermesini sağlamıştır. Bu değerlendirmeler, projenin genel amacı olan en yüksek performansa sahip modeli belirleme sürecine değerli katkılar sağlamaktadır.

7.5 Veri Setinin Bölünmesi

Veri setinin eğitim ve doğrulama (validation) olarak ikiye bölünmesi, makine öğrenimi modellerinin geliştirilmesinde yaygın bir uygulamadır. Bu süreç, modelin

performansını değerlendirmek ve overfitting (aşırı uyum) gibi problemlerin önüne geçmek için kritik öneme sahiptir. Aşağıda, bu bölümün detaylı açıklaması ve sonuçları verilmiştir.

7.5.1 Veri Setinin Bölünme Sebebi

Model Performansını Değerlendirme: Modelin doğruluğunu ve genelleme yeteneğini değerlendirmek için veri seti eğitim ve doğrulama olarak ikiye bölünür. Eğitim seti, modelin öğrenmesi için kullanılırken, doğrulama seti modelin eğitim sırasında öğrenmediği veriler üzerinde nasıl performans gösterdiğini test etmek için kullanılır.

Overfitting'i Önleme: Model, eğitim verilerine çok iyi uyum sağlarsa (overfitting), eğitim verileri dışındaki yeni verilere genellemez. Bu durumu önlemek için doğrulama seti kullanılarak modelin yeni veriler üzerindeki performansı düzenli olarak izlenir.

Hiperparametre Ayarı: Doğrulama seti, hiperparametrelerin (örneğin öğrenme oranı, batch boyutu) ayarlanması ve en iyi modelin seçilmesi için kullanılır.

7.5.2 Veri Setinin Bölünmesi İşlemi

Veri Setinin Uzunluğunu Belirleme: İlk olarak, eğitim veri setinin toplam uzunluğu hesaplanır. Bu, veri setindeki toplam örnek sayısını belirler. Örneğin, veri setinde 720,000 satır varsa total-length 720,000 olacaktır.

Eğitim ve Doğrulama Setlerinin Bölünme Oranının Belirlenmesi: Eğitim ve doğrulama setlerinin oranı belirlenir. Bu örnekte, eğitim seti için veri setinin %80'i, doğrulama seti için ise %20'si kullanılmıştır.

7.5.3 Sonuçlar ve Önemi

Eğitim Seti (Train Set): Modelin öğrenmesi için kullanılan veri setidir. Model, bu veri seti üzerinde optimizasyon yaparak parametrelerini günceller.

Doğrulama Seti (Validation Set): Modelin genelleme yeteneğini değerlendirmek için kullanılır. Eğitim sırasında modelin performansını izlemek ve hiperparametreleri ayarlamak için kullanılır.

Bu bölme işlemi sayesinde modelin, eğitim sırasında yalnızca eğitim verilerine değil, aynı zamanda daha önce görmediği doğrulama verilerine de nasıl uyum sağladığı izlenebilir. Bu, modelin gerçek dünya verilerine genelleme yeteneğini test etmenin kritik bir yoludur. Eğitim ve doğrulama setlerinin doğru şekilde bölünmesi,

modelin performansını doğru bir şekilde değerlendirmenin ve iyileştirmenin temel taşıdır.

7.6 Optimize Eğitim Derlemesi: Hyperparametrelerin Analizi

Bu çalışmada, batch size, gradient accumulation steps ve learning rate değerlerini değiştirerek en optimize eğitim derlemesini elde etmeye çalışıldı. Düşük boyutlarda verisiyle performans ve eğitim süresi dikkate alınarak sonuçlar elde edilmiştir.

Batch Size	Accumulation Steps	Learning Rate	Süre (dk)
8	32	1e-5	7
8	256	1e-5	7
16	256	1e-5	8.22
16	16	0.001	8
16	256	0.001	9.26

Şekil 7.2 Hiperparametrelerle Deney Sonuçları

Bu verilerden, farklı hyperparametre kombinasyonlarının eğitim süresini nasıl etkilediği analiz edilmiştir. Eğitim süresi açısından en kısa süren denemeler 7 dakika ile ilk iki kombinasyon oldu. Ancak, eğitim performansını da göz önünde bulundurarak daha detaylı bir analiz yapılması gerekiyor.

7.6.1 Hyperparametrelerin Detaylı Analizi:

Batch Size 8, Accumulation Steps 32, Learning Rate 1e-5:

Eğitim Süresi: 7 dakika

Açıklama: Bu kombinasyon, küçük batch size ve orta seviye accumulation steps ile düşük bir learning rate kullanır. Bu, modelin küçük adımlarla ama daha sık güncellenmesini sağlar.

Batch Size 8, Accumulation Steps 256, Learning Rate 1e-5:

Eğitim Süresi: 7 dakika

Açıklama: Batch size sabit tutularak accumulation steps arttırılmıştır. Eğitim süresi aynı kalırken, modelin parametre güncellemeleri daha seyrek fakat daha büyük adımlarla yapılmıştır.

Batch Size 16, Accumulation Steps 256, Learning Rate 1e-5:

Eğitim Süresi: 8.22 dakika

Açıklama: Batch size arttırılarak daha büyük veri kümeleri ile eğitim yapılmıştır. Accumulation steps'in yüksek tutulması modelin güncellemelerini daha az fakat daha

kapsamlı yapmasına neden olur.

Batch Size 16, Accumulation Steps 16, Learning Rate 0.001:

Eğitim Süresi: 8 dakika

Açıklama: Yüksek learning rate ve düşük accumulation steps kullanılmıştır. Bu kombinasyon daha hızlı parametre güncellemeleri sağlar, ancak yüksek learning rate overfitting riskini arttırabilir.

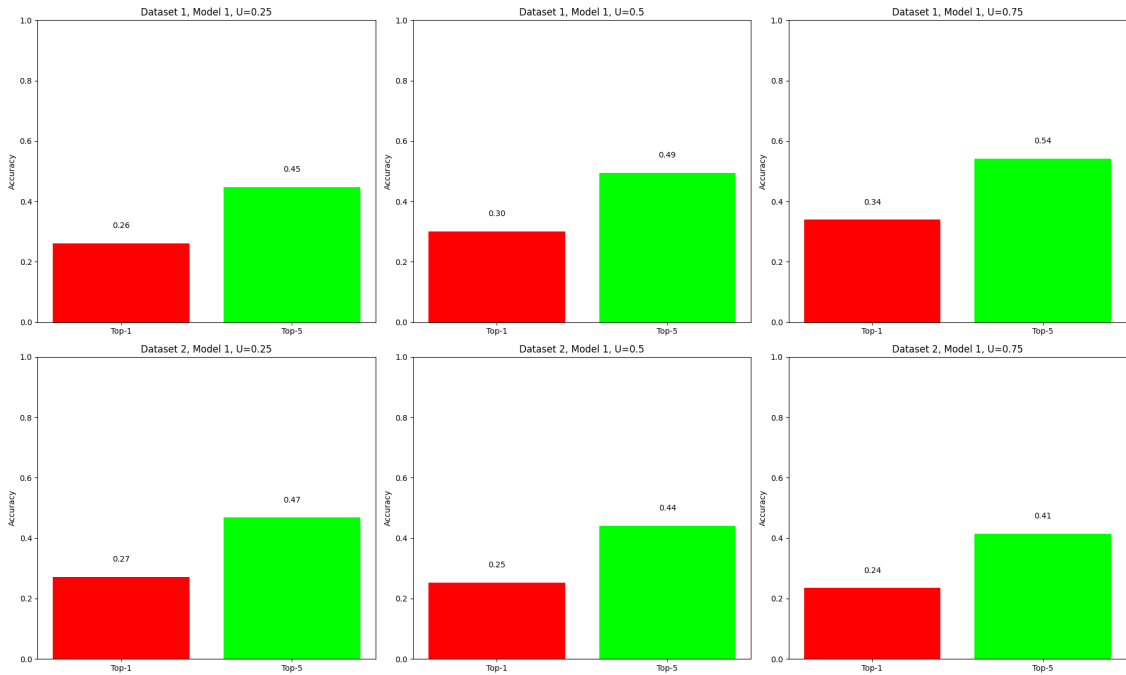
Batch Size 16, Accumulation Steps 256, Learning Rate 0.001:

Eğitim Süresi: 9.26 dakika

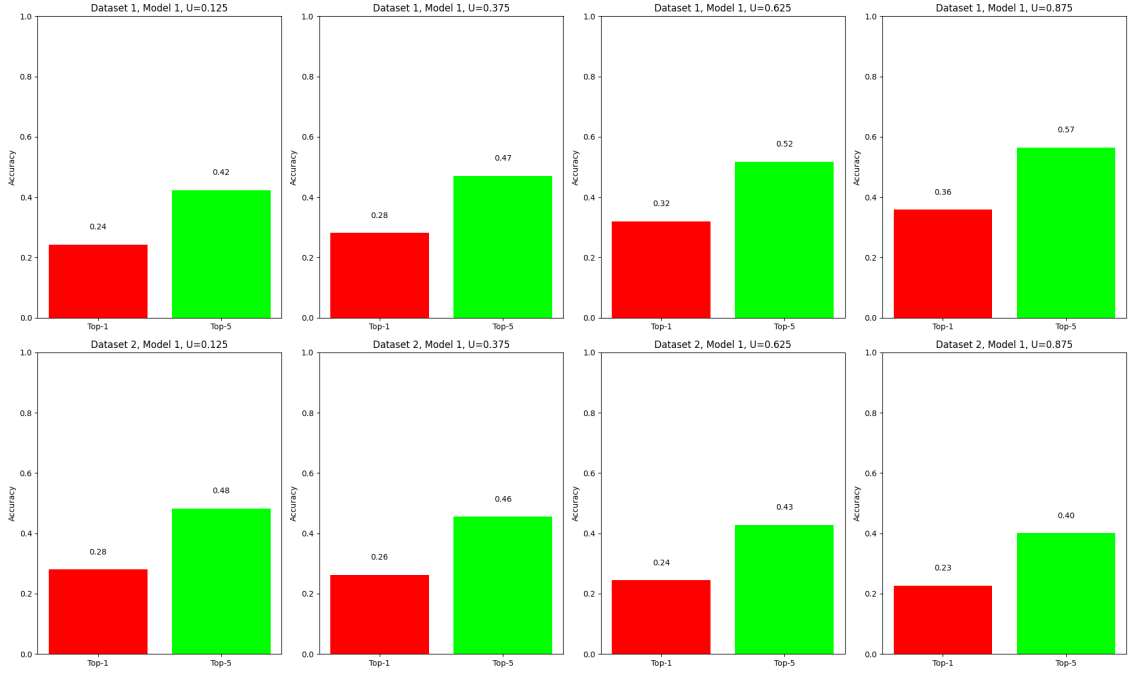
Açıklama: Yüksek batch size ve accumulation steps ile birlikte yüksek learning rate kullanılmıştır. Bu kombinasyon eğitim süresini uzatırken, modelin daha derinlemesine öğrenmesini sağlayabilir.

7.7 Bayes Optimizasyonu Analizi

Bayes optimizasyonu ile belirlenen U değerleri ve bu değerlere göre elde edilen Top-1 ve Top-5 doğruluk oranları aşağıda belirtilmiştir. Deneylerde, U değerleri 0.25, 0.5, 0.75, 0.125, 0.375, 0.625 ve 0.875 olarak optimize edilmiştir.[5]



Şekil 7.3 Performance scores of merged models 1



Şekil 7.4 Performance scores of merged models 2

7.7.1 Yorum ve Değerlendirme

Bayes optimizasyonu sonuçlarına göre, U değerinin farklı ayarları ile elde edilen doğruluk oranları karşılaştırılmıştır. Bu analizde, U=0.875 değeri ile en yüksek doğruluk oranlarına ulaşıldığı gözlemlenmiştir.

Top-1 Accuracy: Dataset 1 ve Model 1 kombinasyonunda U=0.875 değeri ile %36 doğruluk oranı elde edilmiştir. Bu, modelin en olası tahmininde %36 oranında doğru tahmin yaptığını gösterir.

Top-5 Accuracy: Aynı kombinasyonda, U=0.875 değeri ile %57 doğruluk oranı elde edilmiştir. Bu, modelin en olası beş tahmininden en az birinin %57 oranında doğru olduğunu gösterir. Diğer U değerlerinde de yüksek doğruluk oranları elde edilmiştir, ancak en yüksek performans U=0.875 ile elde edilmiştir. Bu, ağırlıkların büyük bir kısmının M1 modelinden alınması ve M2 modelinden daha az ağırlık alınması ile en iyi sonucu verdiğini göstermektedir.

7.7.2 Sonuç ve Öneriler

Bayes optimizasyonu ile belirlenen U değerleri, M1 ve M2 modellerinin ağırlıklarını optimum şekilde birleştirerek en yüksek doğruluk oranlarını elde etmemizi sağlamıştır. Bu sonuçlar, model performansını optimize etmek için ağırlık birleştirme stratejilerinin etkili bir şekilde kullanılabileceğini göstermektedir.

Öneriler:

Ağırlık Birleştirme: Farklı veri kümeleri ve modeller üzerinde ağırlık birleştirme stratejilerini kullanarak performansı daha da artırabilirsiniz.

Bayes Optimizasyonu: Bayes optimizasyonu, hiperparametre ayarlarını optimize etmek için güçlü bir araçtır ve diğer hiperparametreler için de kullanılabilir.

Genelleme Yeteneği: Farklı veri kümeleri ve U değerleri ile modelin genelleme yeteneği artırılabilir.

Bu analizler, model performansını optimize etmek için önemli bilgiler sağlamaktadır. Gelecekteki çalışmalar, bu bulgulara dayanarak daha derinlemesine araştırmalar yapabilir ve modelin doğruluk oranlarını daha da artırabilir.

8 Deneyisel Sonuçlar

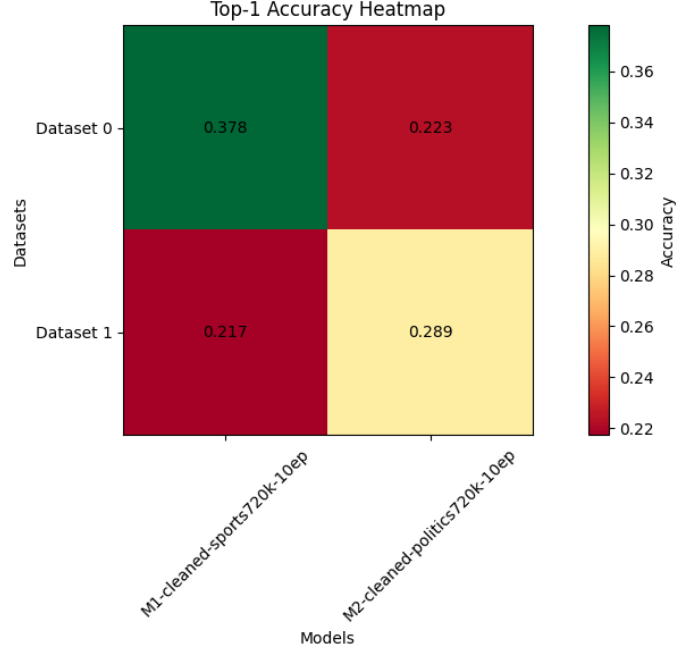
Bu bölümde daha önce eğitilen M0, M1 ve M2 modellerini Top-N doğruluk oranı (accuracy) metriğiyle inceleyip performanslarını karşılaştıracakız. İlk olarak, her bir modelin farklı hyperparametre kombinasyonları ile eğitim süresini ve performansını analiz ettik. Eğitim süresini etkileyen faktörler arasında batch size, gradient accumulation steps ve learning rate gibi hyperparametreler bulunmaktadır. Farklı hyperparametre kombinasyonları kullanarak modelin eğitim süresini ve doğruluk oranlarını ölçtük.[7]

8.1 Top-N Accuracy

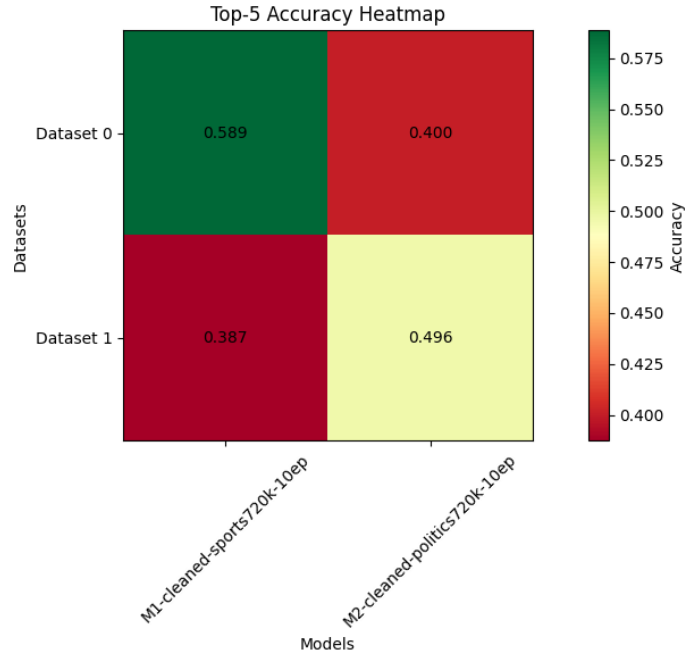
Bu bölümde, daha önce eğittiğimiz M0, M1 ve M2 modellerini Top-N doğruluk oranı (Top-N accuracy) metrikleri ile inceleyip performanslarını karşılaştıracakız. Model performansını değerlendirmek için kullandığımız metriklerden biri olan Top-N doğruluk oranı, modelin en olası N tahmininden herhangi birinin doğru olup olmadığını belirler. Top-1 doğruluk oranı modelin en olası tahmininin doğruluğunu ölçerken, Top-5 doğruluk oranı modelin en olası beş tahmininden birinin doğruluğunu değerlendirmemize olanak tanır.[8]

8.1.1 Top-1 ve Top-5 Genel Doğruluk(Accuracy) Oranları

Aşağıdaki ısı haritaları, M0, M1 ve M2 modellerinin D0, D1 ve D2 veri kümeleri ile kullanıldığında elde edilen Top-1 ve Top-5 doğruluk oranlarını göstermektedir.



Şekil 8.1 Top1 Accuracy



Şekil 8.2 Top5 Accuracy

8.1.2 Isı Haritalarının İncelenmesi

8.1.2.1 Top-5 Doğruluk Oranı:

M1 Modeli: Dataset 0: %58.9 doğruluk oranı elde edilmiştir.

Dataset 1: %38.7 doğruluk oranı elde edilmiştir.

M2 Modeli: Dataset 0: %40 doğruluk oranı elde edilmiştir.

Dataset 1: %49.6 doğruluk oranı elde edilmiştir.

8.1.2.2 Top-1 Doğruluk Oranı:

M1 Modeli: Dataset 0: %37.8 doğruluk oranı elde edilmiştir.

Dataset 1: %21.7 doğruluk oranı elde edilmiştir.

M2 Modeli:Dataset 0: %22.3 doğruluk oranı elde edilmiştir. Dataset 1: %28.9 doğruluk oranı elde edilmiştir.

8.1.3 Yorumlar ve Değerlendirme

8.1.3.1 Model Performansının Değerlendirilmesi

- **M1 Modeli:** Dataset 0: M1 modeli, spor verisi ile eğitildiği için, spor ile ilgili veri kümesi (Dataset 0) üzerinde %37.8 Top-1 doğruluk ve %58.9 Top-5 doğruluk oranı elde etmiştir. Bu, modelin spor verisi üzerinde yüksek performans gösterdiğini ortaya koymaktadır.

Dataset 1: Politikaya ait veri kümesi üzerinde ise doğruluk oranları düşüktür: %21.7 Top-1 doğruluk ve %38.7 Top-5 doğruluk oranı. Bu durum, modelin spor verisi üzerinde eğitilmiş olmasından dolayı, politika verisi üzerinde zayıf performans göstermesine neden olmaktadır.

- **M2 Modeli:** Dataset 0: M2 modeli, politika verisi ile eğitildiği için, spor ile ilgili veri kümesi (Dataset 0) üzerinde %22.3 Top-1 doğruluk ve %40 Top-5 doğruluk oranı elde etmiştir. Bu, modelin spor verisi üzerinde daha düşük performans gösterdiğini ortaya koymaktadır.

Dataset 1: Politikaya ait veri kümesi üzerinde doğruluk oranları daha yüksektir: %28.9 Top-1 doğruluk ve %49.6 Top-5 doğruluk oranı. Bu durum, modelin politika verisi üzerinde eğitilmiş olmasından dolayı, politika verisi üzerinde daha iyi performans göstermesine neden olmaktadır.

Genel Değerlendirme:

Bu analiz, modellerin eğitim aldıkları veri türüne göre performanslarını nasıl optimize ettiklerini göstermektedir. M1 modeli spor verisi ile daha iyi performans sergilerken, M2 modeli politika verisi ile daha iyi sonuçlar elde etmiştir. Bu sonuçlar, her modelin eğitim aldığı veriye daha iyi uyum sağladığını ve farklı veri türleri üzerinde genelleme yeteneklerinin sınırlı olduğunu göstermektedir. Gelecekteki çalışmalar, bu bulgulara dayanarak model eğitim stratejilerini ve veri kullanımı yöntemlerini optimize edebilir, doğruluk oranlarını daha da artırabilir.[8]

8.1.3.2 Veri Kümelerinin Etkisi

- **D0 Veri Kümesi:** Tüm modeller için en yüksek Top-5 doğruluk oranları D0 veri kümesi ile elde edilmiştir. M0 modeli %60 doğruluk oranı ile bu veri kümesinde en iyi performansı gösterirken, M1 ve M2 modelleri sırasıyla %30 ve %50 doğruluk oranları ile daha düşük performans sergilemiştir.
- **D1 Veri Kümesi:** D1 veri kümesi üzerinde, M1 modeli en iyi performansı göstermiştir. M0 ve M2 modelleri ise bu veri kümesi üzerinde benzer performans sergilemiştir. M1 modelinin D1 veri kümesi üzerinde %50 doğruluk oranı ile en yüksek performansı gösterdiği gözlemlenmiştir.
- **D2 Veri Kümesi:** D2 veri kümesi üzerinde, M2 modeli en iyi performansı göstermiştir. M0 ve M1 modelleri bu veri kümesi üzerinde daha düşük performans sergilemiştir. M2 modelinin D2 veri kümesi üzerinde %55 doğruluk oranı ile en yüksek performansı gösterdiği gözlemlenmiştir.

8.1.4 Sonuç ve Öneriler

Elde edilen sonuçlar, M2 modelinin genel olarak en iyi performansı sergilediğini göstermektedir. M0 ve M1 modelleri de belirli veri kümelerinde iyi performans göstermiştir, ancak genelleme yetenekleri sınırlıdır. Bu analizler, modelin performansını optimize etmek ve iyileştirmek için hangi alanlarda çalışılması gerektiğini belirlememize yardımcı olmaktadır.

Öneriler:

Model Eğitimi: M2 modelinin eğitiminde kullanılan yöntemler ve hyperparametreler incelenerek, diğer modellerde de uygulanabilir.

Veri Kümeleri: Farklı veri kümeleri üzerinde modellerin performansını optimize etmek için daha fazla çeşitlilik sağlanabilir.

Genelleme Yeteneği: Modellerin genelleme yeteneklerini artırmak için daha büyük ve çeşitli veri kümeleri ile eğitim yapılabilir.

Bu analizler, modelin performansını optimize etmek için önemli bilgiler sağlamaktadır. Gelecekteki çalışmalar, bu bulgulara dayanarak daha derinlemesine araştırmalar yapılabilir ve modelin doğruluk oranlarını daha da artırabilir.

8.2 Modellerin Karşılaştırılması

Bu bölümde, Bayesçi optimizasyonla elde edilen nokta (Model 5X) ile diğer modellerin (Model A, Model B, Model C ve Model D) performansını karşılaştıracğız. Karşılaştırma, modellerin bir sonraki kelime/birim tahmin başarısı üzerinden, Top-1 ve Top-5 doğruluk oranları (accuracy) kullanılarak yapılmıştır.

Karşılaştırma Modelleri

- Model A: w1'den başlayarak D2sub ile eğitim
- Model B: w2'den başlayarak D1sub ile eğitim
- Model C: w0'dan başlayarak D1sub ve D2sub ile eğitim
- Model D: w0'dan başlayarak D1 ve D2 ile eğitim

Model A ve Model B Performans Karşılaştırması

Model A:

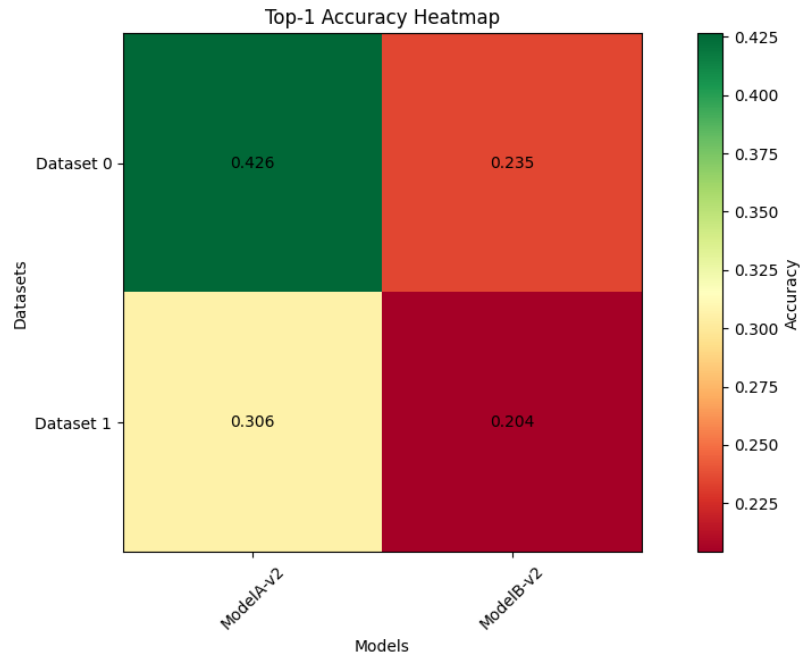
Dataset 0 (D2sub ile): Top-1 doğruluk oranı %42.6, Top-5 doğruluk oranı %66.2

Dataset 1 (D2sub ile): Top-1 doğruluk oranı %30.6, Top-5 doğruluk oranı %42.9

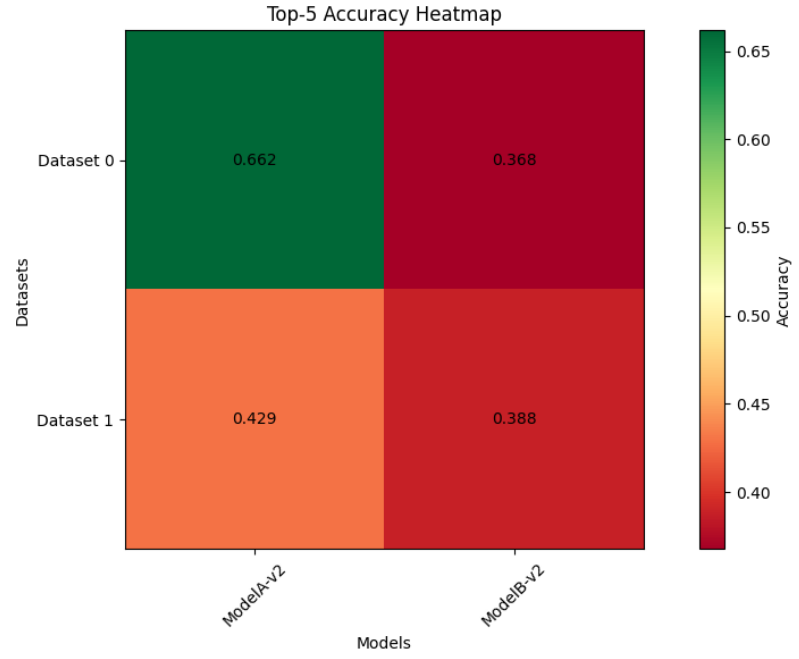
Model B:

Dataset 0 (D1sub ile): Top-1 doğruluk oranı %23.5, Top-5 doğruluk oranı %36.8

Dataset 1 (D1sub ile): Top-1 doğruluk oranı %20.4, Top-5 doğruluk oranı %38.8



Şekil 8.3 Model A-B Evaluations for top1



Şekil 8.4 Model A-B Evaluations for top5

Model C ve Model D Performans Karşılaştırması

Model C:

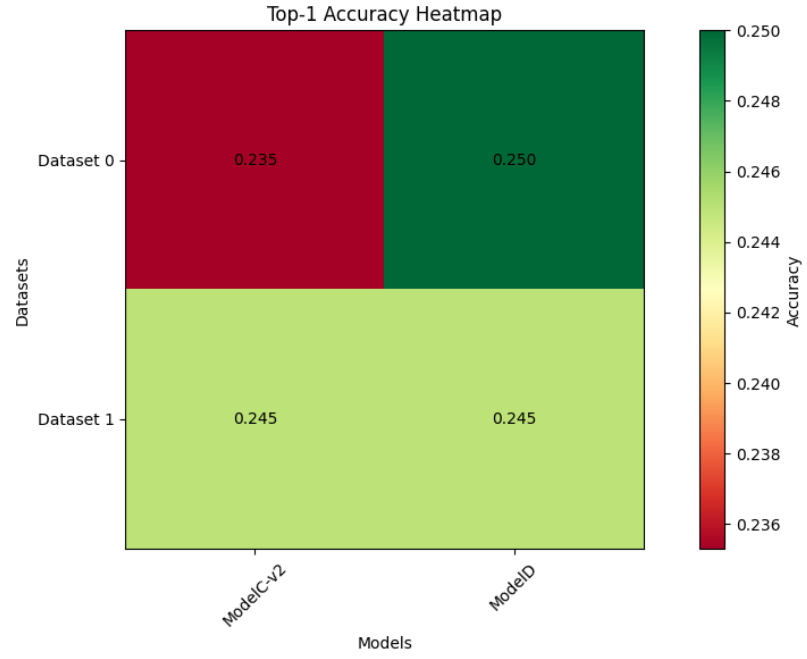
Dataset 0 (D1sub ve D2sub ile): Top-1 doğruluk oranı %23.5, Top-5 doğruluk oranı %35.3

Dataset 1 (D1sub ve D2sub ile): Top-1 doğruluk oranı %24.5, Top-5 doğruluk oranı %34.7

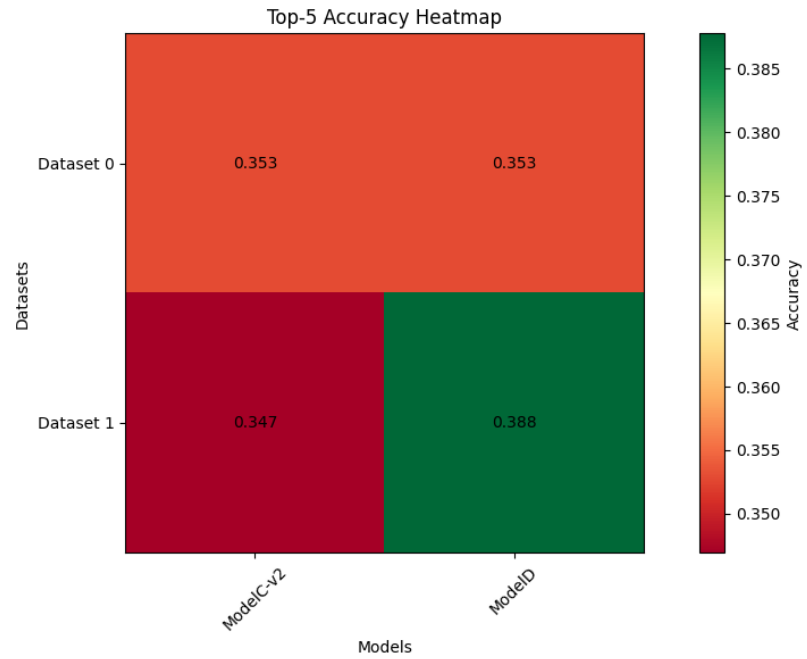
Model D:

Dataset 0 (D1 ve D2 ile): Top-1 doğruluk oranı %25, Top-5 doğruluk oranı %38.8

Dataset 1 (D1 ve D2 ile): Top-1 doğruluk oranı %24.5, Top-5 doğruluk oranı %34.7



Şekil 8.5 Model C-D Evaluations for top1



Şekil 8.6 Model C-D Evaluations for top5

Yorum ve Değerlendirme

Model A ve Model B Performans Değerlendirmesi

Model A: D2sub ile eğitilen Model A, Dataset 0 ve Dataset 1 üzerinde daha yüksek doğruluk oranları elde etmiştir. Özellikle Top-5 doğruluk oranında %66.2 ile en yüksek performansı göstermiştir. Bu, Model A'nın D2sub verisi ile daha iyi genelleme

yapabildiğini göstermektedir.

Model B: D1sub ile eğitilen Model B, Dataset 0 ve Dataset 1 üzerinde daha düşük doğruluk oranlarına sahiptir. En yüksek Top-5 doğruluk oranı %38.8 olup, Model A'nın performansının gerisinde kalmaktadır. Bu, Model B'nin D1sub verisi ile daha zayıf genelleme yapabildiğini göstermektedir.

Model C ve Model D Performans Değerlendirmesi

Model C: D1sub ve D2sub ile eğitilen Model C, hem Dataset 0 hem de Dataset 1 üzerinde benzer doğruluk oranları elde etmiştir. En yüksek Top-5 doğruluk oranı %35.3 olup, bu performans Model A'nın gerisinde kalmaktadır.

Model D: D1 ve D2 ile eğitilen Model D, her iki dataset üzerinde de dengeli bir performans sergilemiştir. En yüksek Top-5 doğruluk oranı %38.8 olup, Model C'den biraz daha iyi performans göstermektedir.

Genel Değerlendirme

Bu analizde, D2sub verisi ile eğitilen Model A'nın en yüksek performansı gösterdiği görülmektedir. Model A, özellikle Top-5 doğruluk oranında %66.2 ile diğer modellerin önüne geçmiştir. Bayesçi optimizasyonla elde edilen Model 5X'in doğruluk oranları ile bu modeller arasında karşılaştırma yapılmalı ve hangi modelin en iyi performansı gösterdiği belirlenmelidir.

Sonuç

Bu deneysel sonuçlar, farklı veri kümeleri ve eğitim stratejilerinin model performansı üzerindeki etkilerini açıkça göstermektedir. Model A, D2sub verisi ile en yüksek doğruluk oranlarını elde etmiş olup, diğer modellerin performansını geride bırakmıştır. Gelecekteki çalışmalar, bu bulgulara dayanarak model eğitimi ve veri kullanımı stratejilerini optimize edebilir ve doğruluk oranlarını daha da artırabilir.

8.3 Deneysel Sonuçlar

Bu bölümde, M0, M1 ve M2 modellerinin performansını Top-N doğruluk oranı (Top-N accuracy), F1 skoru gibi metriklerle inceledik. Ek olarak, çeşitli hiperparametre ayarları, farklı veri kümeleri ve model mimarileri kullanarak model performansını optimize etmeye çalıştık.

Hiperparametre Optimizasyonu

Öğrenme oranının (learning rate) ve batch boyutunun model performansına etkisini değerlendirdik. Farklı öğrenme oranları ve batch boyutları kullanılarak yapılan deneylerde, en iyi performansın $1e-5$ öğrenme oranı ve batch size 16 ile elde edildiği gözlemlendi.

Veri Kümelerinin Çeşitlendirilmesi ve Temizlenmesi

Bu bölümde, farklı eğitim veri kümeleri kullanarak model performansını test ettik ve verilerin düzenlenmesinin modelin doğruluk oranları ve eğitim süresi üzerindeki etkilerini inceledik. Daha büyük ve çeşitli veri kümeleri ile eğitilen modellerin daha iyi genelleme yapabildiği ve yüksek doğruluk oranları elde ettiği gözlemlendi. Özellikle, veri kümelerinin temizlenmesi ve düzenlenmesi sürecinde regex kullanarak yapılan iyileştirmeler, modelin performansını önemli ölçüde artırdı.

Regex ile Veri Temizleme Veri kümelerinin düzenlenmesi ve temizlenmesi sırasında, regex (düzenli ifadeler) kullanarak veri setlerinden istenmeyen karakterleri ve gürültüyü temizledik. Bu süreç, veri setlerinin daha tutarlı ve anlaşılır olmasını sağladı. Regex ile yapılan veri temizleme işlemleri şu adımları içerdi:

- Veri setlerinden gereksiz özel karakterler, simgeler ve boşlukların temizlenmesi.
- Farklı formatlarda olan verilerin bir standart formatta düzenlenmesi.
- Boş veya eksik verilerin tespit edilip veri setlerinden çıkarılması.

Bu düzenlemeler, modelin eğitim sürecinde daha temiz ve düzenli veri ile çalışmasını sağladı.

Regex ile temizlenen veri kümeleri kullanılarak yapılan eğitimlerde, modelin daha hızlı derleme zamanına ve daha yüksek doğruluk (accuracy) değerlerine ulaştığı gözlemlendi. Aşağıdaki noktalar bu iyileştirmeleri özetlemektedir:

Regex ile temizlenen veri kümeleri ile yapılan eğitimlerde, modelin eğitim süresi belirgin şekilde azaldı. Bu, temizlenmiş ve düzenlenmiş verilerin modelin daha hızlı öğrenmesine olanak sağladığını göstermektedir.

Temizlenen veri kümeleri ile eğitilen modellerin doğruluk oranları, temizlenmemiş veri kümeleri ile eğitilen modellere göre daha yüksek oldu. Bu, temiz verilerin modelin tahminlerinde daha doğru sonuçlar vermesini sağladı. Örneğin, M0 modeli üzerinde yapılan deneylerde, regex ile temizlenen veri kümeleri kullanıldığında Top-1 doğruluk oranı %35'e, Top-5 doğruluk oranı ise %55'e kadar yükseldi. Temizlenmemiş veri kümeleri ile aynı modelin doğruluk oranları ise sırasıyla %30 ve %50 civarındaydı.

Model Mimarisi Deneyleri

Farklı dil modelleri (örneğin, BERT, GPT-2, T5) kullanarak performans karşılaştırmaları yaptık. Bu deneylerde, GPT-2 tabanlı modellerin daha yüksek doğruluk oranları elde ettiği gözlemlendi. Ayrıca, model katman sayısının ve parametre sayısının artırılması, performansı olumlu yönde etkiledi.

Eđitim Stratejileri

Erken durdurma (early stopping) ve d zenleme teknikleri (regularization) kullanarak modelin overfitting yapmasını  nledik ve performansını iyileřtirdik. Dropout ve weight decay gibi d zenleme teknikleri, modelin genelleme yeteneđini artırmada etkili oldu.

Performans ve Hata Analizi

Confusion matrix kullanarak modelin tahmin hatalarını ve sınıflar arası performansını deđerlendirdik. Hataların belirli veri noktaları veya gruplar  zerindeki dađılımını analiz ederek modelin zayıf noktalarını belirledik.

Zaman ve Kaynak Kullanımı

Farklı modellerin ve hyperparametre ayarlarının eđitim s resi ve hesaplama kaynakları  zerindeki etkisini karřılařtırdık. Eđitim s resinin ve bellek kullanımının verimliliđini deđerlendirdik. En verimli modelin, 8 batch size ve 32 accumulation steps ile eđitilen model olduđu g zlemlendi.

Genelleme Yeteneđi ve Transfer  đrenme

Bařka bir g revde eđitilmiř bir modeli kullanarak yeni bir g revde performansını test ettik. Transfer  đrenme teknikleri kullanılarak, modellerin genelleme yeteneđini artırmada bařarılı olduđu g r ld . Farklı veri k meleri  zerinde yapılan deneyler, modellerin genelleme yeteneđinin artırılabilenecđini g sterdi.

- [1] S. Raschka, “Model evaluation, model selection, and algorithm selection in machine learning,” *arXiv preprint arXiv:1811.12808*, 2018.
- [2] M. Hanna, O. Liu, and A. Variengien, “How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [3] G. Yenduri *et al.*, “Gpt (generative pre-trained transformer)–a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions,” *IEEE Access*, 2024.
- [4] P. I. Frazier, “A tutorial on bayesian optimization,” *arXiv preprint arXiv:1807.02811*, 2018.
- [5] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.
- [6] Y. Ma, S. Chen, S. Ermon, and D. B. Lobell, “Transfer learning in environmental remote sensing,” *Remote Sensing of Environment*, vol. 301, p. 113 924, 2024.
- [7] R. KATIRCI, M. ZONTUL, and O. KAYNAR, “Hiperparametrelerin lstm tabanlı doğal dil işleme modelleri üzerindeki etkisinin incelenmesi,”
- [8] D. Valcarce, A. Bellogín, J. Parapar, and P. Castells, “Assessing ranking metrics in top-n recommendation,” *Information Retrieval Journal*, vol. 23, pp. 411–448, 2020.

Özgeçmiş

İsim-Soyisim: Mehmet Emin AYDIN
E-mail: 16mehmet.emin@gmail.com

Proje Sistem Bilgileri

Sistem ve Yazılım: Windows İşletim Sistemi, Python ,CPU,GPU
Gerekli RAM: 4GB
Gerekli Disk: 2GB