Llama-2-7b-hf ile Sesli Komutlara Yanıt Verebilen ve İstenen Uygulamayı Açabilen ChatBot: LoRA ile İnce Ayar Uygulaması

Emre TURHAN
220212035

Üretken Yapay Zeka Dersi Proje Raporu

Danışman: Murat Şimşek, Ph.D

20.05.2025

İçindekiler

1 Giriş **1**

1.1 Projenin Amacı 1

1.2 Problem Tanımı 2

1.3 Raporun Yapısı 2

2 Literatür Taraması ve Teorik Altyapı **4**

2.1 CHATBOT 4

2.2 Büyük Dil Modelleri (LLM'ler) ve Transformer Mimarisi 4

2.3 Meta Llama-2 Modeli 5

2.4 İnce Ayar (Fine-Tuning) Kavramı 6

2.5 Parametre Verimli İnce Ayar (PEFT) Yöntemleri 7

2.5.1 LoRA (Low-Rank Adaptation) 7

2.5.2 Sesli Konuşma Sistemi 7

3 Materyal ve Yöntem **9**

3.1 Kullanılan Veri Seti 9

4 Materyal ve Yöntemler **10**

4.1 Veri Seti Özellikleri ve Hazırlanışı 10

4.2 Geliştirme Ortamı ve Kütüphaneler 10

4.3 Projede Kullanılan Temel Python Kütüphaneleri ve Araçları 11

4.4 Model ve Tokenizer 11

4.4.1 Model 11

4.4.2 Tokenizer 12

4.5 Proje Akışı ve Uygulanan Adımlar 12

4.5.1 Ortam Kurulumu ve Kütüphane Yüklemeleri 12

4.5.2 Veri Seti Yükleme, Analizi ve Ön İşlemesi 12

4.5.3 Prompt Formatının Oluşturulması ve Uygulanması 12

4.5.4 Temel Modelin Yüklenmesi ve 4-bit Kuantizasyonu 13

4.5.5 İnce Ayar Öncesi Performans Değerlendirmesi (Baseline) 13

4.5.6 LoRA Konfigürasyonu ve İnce Ayar 13

4.5.7 İnce Ayar Eğitimi 14

5 Sesli Chatbot Fonksiyonları 15
5.1 Ses Kaydı ve Transkripsiyon 15
5.2 Metin Okuma (TTS) 15
5.3 Uygulama Açma Fonksiyonu 16
5.4 Ana Döngü 16

6 Model ve Adaptörlerin Kaydedilmesi ve Paylaşılması 17

7 Sonuçların Görselleştirilmesi ve Raporlanması 18

8 Eğitim Aşamaları ve Detayları 23
8.1 Temel Modelin Yüklenmesi ve Kuantizasyonu 23
8.2 Veri Setinin Hazırlanması ve Prompt Mühendisliği 23

9 LoRA Konfigürasyonu ve Parametre Seçimi 25

10 Sonuçlar ve Değerlendirme 26
10.1 İnce Ayar Öncesi Model Performansı (Temel Model) 26
10.2 İnce Ayar Sonrası Model Performansı (LoRA ile) 26
10.3 Sonuçların Genel Değerlendirmesi 26

11 Tartışma ve Gelecek Çalışmalar 27
11.1 Elde Edilen Sonuçların Yorumlanması 27
11.2 Projenin Sınırlılıkları 27
11.3 Gelecekte Yapılabilecek İyileştirmeler ve Çalışmalar 27

12 Sonuç 29

Kaynakça 29

Şekil Listesi

7.1 Enter Caption 19
7.2 Enter Caption 20
7.3 Enter Caption 21
7.4 Enter Caption 22

Bölüm 1

Giriş

Bu bölümde, gerçekleştirilen projenin genel çerçevesi çizilecek, projenin temel amacı ve vurgulanacaktır. Ayrıca, ele alınan problem tanımlanacak ve raporun ilerleyen bölümlerinde nelerin ele alınacağına dair bir yol haritası sunulacaktır.

1.1 Projenin Amacı

Bu projenin amacı, sesli komutlarla etkileşim kurabilen, doğal dil işleme temelli bir yapay zeka asistanı geliştirmektir. Kullanıcının sesli komutlarını gerçek zamanlı olarak algılayıp metne dönüştürmek, ardından bu metni anlamlı ve doğal yanıtlarla cevaplamak temel hedeftir. Model olarak büyük ölçekli dil modeli LLaMA 2 (7B) kullanılarak, LoRA yöntemiyle modelin hafıza ve performans gereksinimleri optimize edilmiştir. Proje kapsamında ayrıca modelin gerçek dünyadaki uygulamalara entegrasyonu için sesli yanıt üretimi ve bilgisayar uygulamalarını açma fonksiyonları da geliştirilmiştir.

Projenin önemi birkaç noktada toplanabilir:

- Günümüzde kullanıcıların teknolojiyle etkileşiminde doğal ve sezgisel yöntemlere duyulan ihtiyaç giderek artmaktadır. Sesli asistanlar, insan-makine iletişimini kolaylaştıran önemli araçlardır ve günlük yaşamda pek çok alanda kullanılmaktadır. Bu proje, özellikle dil modellerinin hafıza ve işlem gücü gereksinimlerini azaltarak, daha erişilebilir ve hızlı çalışan asistanlar geliştirilmesine katkı sağlamaktadır. Ayrıca, kullanıcı deneyimini iyileştiren gerçek zamanlı sesli komut işleme ve yanıt üretimi sayesinde, daha etkili ve kullanıcı dostu yapay zeka çözümleri sunmayı hedeflemektedir. Bu açıdan proje, hem akademik hem de endüstriyel uygulamalarda değerli bir referans ve temel teşkil etmektedir.

Bu çalışma ile kullanıcıların bilgisayarlarını açar açmaz vakit kaybetmeden istedikleri bilgilere ulaşmaları ve tek bir cümleyle yapmak istedikleri görevleri bot aracılığıyla gerçekleştirmeleri sağlanarak, iş gücü ve zamandan tasarruf etmeleri amaçlanmaktadır.

1.2 Problem Tanımı

Bu proje, kullanıcıların sesli komutlar aracılığıyla bilgisayarlarını ve çeşitli uygulamaları kontrol etmelerini sağlayan bir yapay zeka asistanının geliştirilmesini kapsamaktadır. Proje kapsamında, kullanıcının söylediklerini gerçek zamanlı olarak algılayan ve metne dönüştüren bir konuşma tanıma sistemi, doğal dil işleme teknikleri ile komutları anlayan ve yanıtlayan büyük dil modeli tabanlı bir yapay zeka modeli ile entegre edilmiştir. Ayrıca, üretilen yanıtların sesli olarak geri iletilmesi ve belirli komutlara göre bilgisayar üzerinde uygulama açma gibi işlemlerin otomatikleştirilmesi de projenin temel özelliklerindendir. Böylece, kullanıcılar bilgisayar kullanımı sırasında zamandan tasarruf ederek daha verimli bir deneyim yaşarlar.

Bu projede ele alınan temel problemler şunlardır:

- Doğal ve Etkili İnsan-Bilgisayar Etkileşimi: Kullanıcıların bilgisayar ve uygulamalarla daha sezgisel ve hızlı iletişim kurabilmeleri için sesli komutların doğru algılanması ve işlenmesi gerekmektedir.
- Gerçek Zamanlı Ses Tanıma ve Anlama: Sesli komutların anlık olarak metne dönüştürülmesi ve bu metnin doğru şekilde anlaşılması, yüksek doğruluk ve hızlı işlem gerektirir.
- Anlamlı ve Doğal Yanıt Üretimi: Kullanıcı komutlarına uygun, kısa ve anlaşılır cevapların yapay zeka tarafından üretilmesi, kullanıcı deneyiminin kalitesini doğrudan etkiler.
- Teknolojik Kaynakların Verimli Kullanımı: Büyük dil modellerinin yüksek hesaplama ve hafıza gereksinimlerinin optimize edilerek, daha düşük donanımlarla bile etkin çalışabilmesi sağlanmalıdır.
- Otomasyon ve Uygulama Kontrolü: Kullanıcının sesli komutlarına dayanarak bilgisayar üzerindeki uygulamaların doğru ve güvenilir şekilde açılması ve kontrol edilmesi gerekmektedir.

1.3 Raporun Yapısı

Bu rapor, aşağıdaki bölümlerden oluşmaktadır:

- **Bölüm 1 (Giriş):** Projenin amacı, kapsamı ve önemi açıklanır. Projeyle çözülmek istenen temel problemler ve hedefler belirtilir.
- **Bölüm 2 (Literatür Taraması ve Teorik Altyapı):** Konuyla ilgili daha önce yapılmış çalışmalar, kullanılan yöntemler ve mevcut teknolojiler değerlendirilir. Projeye referans teşkil eden kaynaklar özetlenir.
- **Bölüm 3 (Materyal ve Yöntem):** Projede kullanılan yöntemler, teknik altyapı, algoritmalar, veri setleri ve yazılım/hardware bileşenleri detaylı olarak anlatılır. Model seçimi ve eğitim süreci açıklanır.

- **Bölüm 4 (Eğitim Aşamaları ve Detayları):** Proje kapsamında yapılan uygulamalar, geliştirme aşamaları, karşılaşılan zorluklar ve çözümleri sunulur. Kod yapısı, modüller ve iş akışı detaylandırılır.
- **Bölüm 5 (Sonuçlar ve Değerlendirme):** Elde edilen bulgular, test ve doğrulama sonuçları paylaşılır. Model performansı, doğruluk, hız gibi kriterler değerlendirilir. Projenin güçlü ve zayıf yönleri tartışılır.
- **Bölüm 6 (Sonuç ve Gelecek Çalışmalar):** Projenin genel değerlendirmesi yapılır. İleride yapılabilecek geliştirmeler ve öneriler sunulur.
- **Kaynaklar ve Ekler:** Projeye ait ek dokümanlar, kodlar, grafikler ve tablolar yer alabilir.

Bölüm 2

Literatür Taraması ve Teorik Altyapı

2.1 CHATBOT

Chatbotlar, kullanıcılarla doğal dil aracılığıyla etkileşim kurabilen yapay zeka tabanlı sistemler olarak, günümüzde birçok farklı alanda yaygın şekilde kullanılmaktadır. Bu alandaki gelişmeler, özellikle büyük dil modelleri (LLM) ve doğal dil işleme tekniklerindeki ilerlemeler sayesinde hız kazanmıştır. Meta'nın LLaMA serisi ve OpenAI'nin GPT modelleri gibi büyük ölçekli transformer tabanlı modeller, chatbotların daha bağlama duyarlı, anlamlı ve doğal yanıtlar üretmesini mümkün kılmıştır.

Ancak, bu modellerin yüksek hesaplama ve hafıza gereksinimleri, gerçek zamanlı ve kullanıcı odaklı uygulamalarda sınırlamalar getirmektedir. Bu sorunu aşmak için LoRA (Low-Rank Adaptation) gibi yöntemler geliştirilmiş; böylece büyük modellerin incelikli ve düşük maliyetli ince ayarları yapılabilmektedir.

Bu proje, kullanıcıların sesli komutlarını gerçek zamanlı algılayıp, eğitilmiş bir LLaMA 2 tabanlı modeli kullanarak hızlı ve doğal yanıtlar üretmeyi amaçlamaktadır. Whisper gibi gelişmiş ses tanıma modelleriyle entegre edilen sistem, sesli komutları metne dönüştürmekte, doğal dil modeline aktarmakta ve üretilen yanıtları sesli olarak geri iletmektedir. Ayrıca, bilgisayar uygulamalarının sesli komutlarla açılmasını sağlayarak, kullanıcı deneyimini kolaylaştırmayı hedeflemektedir.

Literatürdeki bu gelişmeler doğrultusunda, projede kullanılan yöntemler hem model performansını optimize etmeyi hem de kullanıcıların teknolojiyi daha sezgisel ve hızlı kullanabilmesini sağlamayı amaçlamaktadır.

2.2 Büyük Dil Modelleri (LLM'ler) ve Transformer Mimarisi

Son yıllarda doğal dil işleme alanında devrim yaratan gelişmelerin merkezinde Büyük Dil Modelleri (Large Language Models - LLM'ler) yer almaktadır. Bu modeller, milyarlarca paramet-

reye sahip derin öğrenme ağları olup, insan benzeri metin üretme, anlama, özetleme ve çeviri gibi çeşitli görevlerde yüksek performans sergilemektedir.

LLM'lerin başarısının temelinde Transformer mimarisi bulunmaktadır. İlk olarak Vaswani ve arkadaşları tarafından 2017 yılında tanıtılan Transformer, geleneksel RNN ve CNN tabanlı modellerin aksine, kendine dikkat (self-attention) mekanizmasıyla uzun menzilli bağımlılıkları etkili şekilde modelleyebilmektedir. Bu sayede dil modelleri, bağlamı çok daha iyi kavrayabilmekte ve doğal dil üretiminde üstün sonuçlar elde etmektedir.

Projede kullanılan LLaMA 2 ve GPT-Neo gibi modeller, Transformer tabanlı LLM'lerdir. Bu modeller, geniş veri kümeleri üzerinde ön eğitim yapılarak dil bilgisini öğrenmiş ve ardından proje gereksinimlerine göre LoRA (Low-Rank Adaptation) gibi tekniklerle ince ayar (fine-tuning) süreçlerine tabi tutulmuştur. LoRA, büyük modellerin tüm ağırlıklarını yeniden eğitmek yerine, düşük boyutlu adaptasyon katmanları ekleyerek hem eğitim süresini hem de donanım kaynaklarını optimize etmeyi sağlar.

Kodda kullanılan transformers kütüphanesi, bu modellerin kolayca yüklenip kullanılmasını sağlayan yaygın bir araçtır. Sesli komutları metne dönüştürmek için ise Whisper gibi güçlü ses tanıma modelleri entegre edilmiştir.

Literatürde Transformer tabanlı LLM'lerin, özellikle sesli asistan ve chatbot uygulamalarında insan benzeri etkileşimler sunmak için kritik öneme sahip olduğu vurgulanmaktadır. Bu proje de, bu güncel teknolojileri kullanarak, kullanıcıların sesli komutlarını anlayan ve doğal cevaplar üreten bir sistem geliştirmeyi hedeflemektedir.

2.3 Meta Llama-2 Modeli

Meta AI tarafından geliştirilen LLaMA (Large Language Model Meta AI) 2, günümüzün en gelişmiş ve yüksek performanslı büyük dil modellerinden biridir. 2023 yılında duyurulan LLaMA 2, hem akademik araştırmalara hem de endüstriyel uygulamalara açık, geniş kapsamlı ve ölçeklenebilir bir model ailesidir.

Temel Özellikler:

- Parametre Boyutları: LLaMA 2; 7 milyar, 13 milyar ve 70 milyar olmak üzere farklı boyutlarda modeller içerir. Bu, kullanıcıların donanım kapasitelerine göre uygun modeli seçmesini sağlar.
- Performans: Çok sayıda doğal dil işleme benchmark'ında GPT-3 ve diğer büyük modellerle rekabet eden ya da onları geride bırakan sonuçlar elde etmiştir.
- Açık Erişim ve Uyarlanabilirlik: LLaMA 2, Meta tarafından açık kaynaklı olarak sunulmuş ve araştırmacıların yanı sıra geliştiricilerin kolayca kullanıp fine-tune yapabilmesine olanak tanımıştır.

- Transformer Mimarisi: LLaMA 2, standart Transformer mimarisi üzerine kuruludur ve optimizasyonlar sayesinde hem hız hem doğruluk açısından iyileştirmeler içermektedir.

2.4 İnce Ayar (Fine-Tuning) Kavramı

İnce ayar, önceden büyük veri setleri üzerinde eğitilmiş derin öğrenme modellerinin, belirli bir görev veya veri setine uyarlanması sürecidir. Bu yöntem, sıfırdan model eğitmek yerine, halihazırda öğrenilmiş genel bilgi ve temsillerin üzerine yeni görev için özelleştirme yapılmasını sağlar.

Temel İlkeler:

- Transfer Learning kapsamında değerlendirilir: Model, genel dil bilgisini ön eğitimde edinir ve daha sonra spesifik görevler için ince ayar yapılır.
- İnce ayar sırasında model ağırlıklarının tümü veya sadece belirli bölümleri (örneğin LoRA katmanları) güncellenir.
- Veri seti genellikle ön eğitimdekinden çok daha küçüktür, bu da eğitim süresini ve kaynak kullanımını azaltır.

Avantajları:

- Hızlı Adaptasyon: Yeni görevler için modelin kısa sürede uyarlanmasını sağlar.
- Kaynak Tasarrufu: Sıfırdan eğitim için gereken yüksek hesaplama gücü ve veri ihtiyacını azaltır.
- Daha İyi Performans: Genel dil modellerinin güçlü temsilleri sayesinde, spesifik görevlerde yüksek doğruluk elde edilir.

LoRA (Low-Rank Adaptation) ile İnce Ayar:

- Son dönemde, LoRA gibi yöntemler ince ayar sürecini daha verimli hale getirmiştir. LoRA, büyük modellerin tamamını yeniden eğitmek yerine, düşük dereceli uyarlamalar ekleyerek ince ayar yapmayı mümkün kılar. Bu sayede:
- Model boyutu büyük ölçüde değişmeden kalır,
- Eğitim süresi ve maliyeti azalır,
- Özelleştirme daha kolay ve esnek hale gelir.

Projedeki Kullanımı:

Bu projede, Meta'nın LLaMA 2 modeli üzerinde LoRA yöntemiyle ince ayar yapılmıştır. Böylece, güçlü bir dil modeli düşük donanım kaynaklarıyla özelleştirilerek, sesli komutlara yanıt verebilen pratik bir chatbot geliştirilmiştir.

2.5 Parametre Verimli İnce Ayar (PEFT) Yöntemleri

Büyük dil modellerinin tüm parametrelerini ince ayar yapmak (full fine-tuning), çok büyük hesaplama kaynakları (GPU belleği, işlem gücü) ve zaman gerektirir. Bu zorlukların üstesinden gelmek için Parametre Verimli İnce Ayar (PEFT) yöntemleri geliştirilmiştir. PEFT yöntemleri, ön-eğitilmiş modelin ağırlıklarının büyük bir kısmını dondurarak sadece çok küçük bir sayıda ek veya seçilmiş parametreyi güncellemeyi hedefler. Bu sayede, hesaplama maliyeti önemli ölçüde azaltılırken, görev performansında tam ince ayara yakın sonuçlar elde edilebilir.

2.5.1 LoRA (Low-Rank Adaptation)

Düşük Mertebeli Adaptasyon (Low-Rank Adaptation - LoRA), Hu ve arkadaşları (2021) tarafından önerilen popüler bir PEFT tekniğidir [2]. LoRA'nın temel fikri, Transformer mimarisindeki ağırlık matrislerindeki değişimin (güncellemenin) düşük mertebeli (low-rank) bir yapıya sahip olduğu varsayımıdır. Tam bir ağırlık matrisini (ΔW) güncellemek yerine, LoRA bu güncellemeyi iki daha küçük, düşük mertebeli matrisin $(A$ ve $B)$ çarpımı olarak temsil eder $(\Delta W = BA)$.

Eğitim sırasında, ön-eğitilmiş modelin orijinal ağırlıkları (W_0) dondurulur ve sadece bu düşük mertebeli adaptasyon matrisleri $(A$ ve $B)$ eğitilir. Çıkarım sırasında ise, $W = W_0 + BA$ şeklinde orijinal ağırlıklarla birleştirilebilirler veya ayrı olarak tutulabilirler. Bu, eğitilebilir parametre sayısını önemli ölçüde azaltır.

2.5.2 Sesli Konuşma Sistemi

Projedeki sesli konuşma sistemi, kullanıcıların sesli komutlarını algılayarak metne dönüştürmeyi ve modelden gelen yanıtları sesli olarak iletmeyi amaçlamaktadır. Bu sistem iki ana bileşenden oluşmaktadır:

- **Whisper Modeli (Speech-to-Text):** OpenAI tarafından geliştirilen Whisper modeli, çok dilli ve yüksek doğrulukta çalışan bir otomatik konuşma tanıma (ASR) sistemidir. Bu model, gerçek zamanlı olarak kaydedilen ses sinyallerini doğru şekilde metne dönüştürerek doğal dil işleme modelinin girişini sağlar.
- **pyttsx3 Kütüphanesi (Text-to-Speech):** Kullanıcıya geri bildirim vermek amacıyla, pyttsx3 kütüphanesi kullanılarak modelin ürettiği metinler seslendirilir. pyttsx3, platform bağımsız çalışan ve internet bağlantısına ihtiyaç duymayan bir metinden sese dönüştürme (TTS) aracıdır. Bu sayede model yanıtları kullanıcıya doğal ve akıcı bir şekilde iletilir.

LoRA, bu tekniklerle 70B parametrelik bir modeli bile tek bir 48GB GPU'da ince ayar yapmayı mümkün kılarken, 16-bit ince ayar performansına çok yakın sonuçlar elde etmeyi başarır. Bu projede, Llama-2-7B modelinin ince ayarı için LoRA tekniği kullanılmıştır. Temel model

4-bit NF4 formatına kuantize edilmiş ve LoRA adaptörleri bu kuantize edilmiş model üzerinde eğitilmiştir.

Bölüm 3

Materyal ve Yöntem

Bu çalışmada, Meta AI tarafından geliştirilen LLaMA 2 (Large Language Model Meta AI) 7B modeli temel alınmıştır. Model, doğal dil anlama ve üretme konusunda yüksek performans gösteren, Transformer mimarisi tabanlı büyük bir dil modelidir. LLaMA 2 modeli, kaynak veriler üzerinde ön eğitim (pre-training) aşamasından geçirilmiş olup, daha sonra proje amaçları doğrultusunda LoRA (Low-Rank Adaptation) yöntemi kullanılarak ince ayar (fine-tuning) sürecine tabi tutulmuştur. LoRA, modelin ağırlıklarının tamamını değiştirmeden, küçük uyarlama katmanları ekleyerek eğitim sürecini hafifletmekte ve kaynak kullanımını optimize etmektedir.

3.1 Kullanılan Veri Seti

DailyDialog: Günlük hayatta geçen, doğal ve çok konulu insan diyaloglarını içeren kapsamlı bir veri setidir. Modelin günlük konuşma dilini öğrenmesine ve bağlama uygun yanıtlar üretmesine olanak sağlar.

EmpatheticDialogues: Empati ve duygusal farkındalık içeren diyalog örneklerinden oluşan bu veri seti, modelin kullanıcının duygusal durumunu anlamasını ve buna uygun cevaplar vermesini destekler.

Kendi Toplanan Veri / Özel Veri: Projeye özgü sesli komutlar ve yanıtlar içeren, gerekirse manuel olarak hazırlanan veya ön işleme tabi tutulan veri setleri de modelin özel görevlerde başarısını artırmak amacıyla kullanılmıştır.

Bölüm 4

Materyal ve Yöntemler

4.1 Veri Seti Özellikleri ve Hazırlanışı

Proje kapsamında kullanılan veri seti, LLaMA 2 modelini sesli chatbot uygulaması için fine-tuning etmek amacıyla çeşitli diyalog örneklerinden ve komut-veri çiftlerinden oluşmaktadır. Veri setinin hazırlanması aşağıdaki gibidir:

- **İçerik:** Kullanıcı komutları ve karşılık gelen model yanıtlarını içeren metin çiftleri.
- **Format:** JSON veya CSV formatında, her satırda kullanıcı girdisi ve ona karşılık gelen doğru cevap yer almaktadır.
- **Ön İşleme:** Veriler temizlenmiş, gereksiz boşluklar ve özel karakterler ayıklanmış, modelin tokenizer'ı ile uyumlu hale getirilmiştir.
- **Örnek:** `"input": "Bilgisayarı açar mısın?", "response": "Bilgisayarınız açılıyor."`

4.2 Geliştirme Ortamı ve Kütüphaneler

Proje geliştirme ve eğitim işlemleri aşağıdaki ortam ve kütüphaneler kullanılarak gerçekleştirilmiştir:

- **Ortam:** Google Colaboratory (Colab) NVIDIA T4 veya A100 GPU destekli.
- **Python Sürümü:** 3.9+
- **Temel Kütüphaneler:**
  - `transformers` (Hugging Face) — LLaMA 2 modeli ve tokenizer.
  - `peft` — Parametre Verimli İnce Ayar (LoRA, QLoRA) için.
  - `torch` — Model eğitimi ve tensor işlemleri.

- `sounddevice` — Mikrofon girişinden ses kaydı.
- `whisper` — Sesin metne dönüştürülmesi.
- `pyttsx3` — Metni sese dönüştürme (TTS).
- `os` — Uygulamaların açılması için işletim sistemi komutları.

4.3 Projede Kullanılan Temel Python Kütüphaneleri ve Araçları

```
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
from peft import PeftModel, LoraConfig
import sounddevice as sd
import numpy as np
import whisper
import pyttsx3
import os
```

Listing 4.1: *Kütüphanelerin Yüklenmesi*

4.4 Model ve Tokenizer

4.4.1 Model

LLaMA 2 temel model olarak kullanılmıştır. Model ağırlıkları 4-bit kuantizasyon ve LoRA ile parametre verimli ince ayar yapılmıştır.

```
BASE_MODEL_ID = "meta-llama/Llama-2-7b-hf"
LORA_PATH = "./lora_weights"

tokenizer = AutoTokenizer.from_pretrained(BASE_MODEL_ID)
base_model = AutoModelForCausalLM.from_pretrained(
    BASE_MODEL_ID,
    load_in_4bit=True,
    device_map="auto",
    quantization_config=bnb_4bit_quant_config
)

model = PeftModel.from_pretrained(base_model, LORA_PATH)
model.to("cuda" if torch.cuda.is_available() else "cpu")
```

Listing 4.2: *Modelin ve Tokenizer'ın Yüklenmesi*

4.4.2 Tokenizer

Modelle uyumlu tokenizer, metinleri modelin işleyebileceği token dizilerine çevirir. Padding token olarak EOS token atanmıştır.

```
tokenizer.pad_token = tokenizer.eos_token
```

Listing 4.3: *Tokenizer Ayarı*

4.5 Proje Akışı ve Uygulanan Adımlar

4.5.1 Ortam Kurulumu ve Kütüphane Yüklemeleri

Gerekli kütüphaneler Colab ortamına yüklenir ve ortam yapılandırılır.

```
!pip install transformers peft bitsandbytes sounddevice pyttsx3 whisper
```

Listing 4.4: *Gerekli Kütüphanelerin Yüklenmesi*

4.5.2 Veri Seti Yükleme, Analizi ve Ön İşlemesi

Veri seti dosyadan yüklenir, temizlenir ve eğitim-test bölümü yapılır.

```
import pandas as pd

df = pd.read_csv("dialog_data.csv")
df = df.dropna()

# Basit  rnek  ayr m
train_df = df.sample(frac=0.8, random_state=42)
test_df = df.drop(train_df.index)
```

Listing 4.5: *Veri Seti Yükleme ve Hazırlık*

4.5.3 Prompt Formatının Oluşturulması ve Uygulanması

Modelin doğru şekilde öğrenmesi için prompt formatları oluşturulur.

```
def format_prompt(user_input):
    return f"User: {user_input}\nAssistant:"
```

Listing 4.6: *Prompt Formatlama Fonksiyonu*

4.5.4 Temel Modelin Yüklenmesi ve 4-bit Kuantizasyonu

Model ağırlıkları 4-bit NF4 kuantizasyon ile yüklenerek VRAM tüketimi optimize edilir.

```
from bitsandbytes import BitsAndBytesConfig

bnb_4bit_quant_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16
)
```

Listing 4.7: *4-bit Kuantizasyon Ayarları*

4.5.5 İnce Ayar Öncesi Performans Değerlendirmesi (Baseline)

Eğitim öncesi model çıkışları test edilir.

```
def query_model(instruction):
    prompt = format_prompt(instruction)
    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
    outputs = model.generate(**inputs, max_new_tokens=100)
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return response.split("Assistant:")[-1].strip()

print(query_model("Bilgisayar  a ar  m  s  n?"))
```

Listing 4.8: *Modelden Basit Sorgu Çalıştırma*

4.5.6 LoRA Konfigürasyonu ve İnce Ayar

LoRA adaptörleri ile parametre verimli ince ayar yapılır.

```
from peft import LoraConfig

lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)
```

Listing 4.9: *LoRA Config Örneği*

4.5.7 İnce Ayar Eğitimi

TRL kütüphanesi ile denetimli ince ayar gerçekleştirilir.

```
from trl import SFTTrainer, SFTConfig

sft_config = SFTConfig(
    model_name_or_path=BASE_MODEL_ID,
    train_dataset=train_df,
    max_seq_length=512,
    output_dir="./trained_weights",
    per_device_train_batch_size=1,
    gradient_accumulation_steps=8,
    learning_rate=2e-4,
    num_train_epochs=3,
    optim="paged_adamw_32bit",
    save_steps=100,
    logging_steps=10,
    evaluation_strategy="steps"
)

trainer = SFTTrainer(model=model, config=sft_config)
trainer.train()
```

Listing 4.10: *SFTTrainer ile Eğitim Örneği*

Bölüm 5

Sesli Chatbot Fonksiyonları

5.1 Ses Kaydı ve Transkripsiyon

Mikrofon ile ses kaydedilip Whisper modeliyle yazıya dönüştürülür.

```
import sounddevice as sd
import whisper
import numpy as np

def record_audio(duration=5, samplerate=16000):
    audio = sd.rec(int(duration * samplerate), samplerate=samplerate,
        channels=1)
    sd.wait()
    return np.squeeze(audio)

model_whisper = whisper.load_model("tiny")

def transcribe(audio):
    result = model_whisper.transcribe(audio, language="en")
    return result["text"]
```

Listing 5.1: *Ses Kaydı ve Transkripsiyon*

5.2 Metin Okuma (TTS)

Model cevabı pyttsx3 ile seslendirilir.

```
import pyttsx3

def speak(text):
    engine = pyttsx3.init()
    engine.setProperty('rate', 150)
    engine.say(text)
    engine.runAndWait()
```

Listing 5.2: *Metni Sese Dönüştürme*

5.3 Uygulama Açma Fonksiyonu

Sesli komutlara göre belirli uygulamalar açılır.

```
import os

def open_application(command):
    cmd = command.lower()
    if "notepad" in cmd:
        os.system("notepad.exe")
    elif "calculator" in cmd:
        os.system("calc.exe")
    elif "chrome" in cmd:
        os.system("start chrome")
    # Di er uygulamalar i in eklemeler yap labilir
```

Listing 5.3: *Uygulama Açma Fonksiyonu*

5.4 Ana Döngü

Ses kaydedilir, metne çevrilir, model sorgulanır, yanıt seslendirilir ve komutlar uygulanır.

```
def main():
    audio = record_audio()
    user_input = transcribe(audio)
    print(f"Kullan c : {user_input}")

    response = query_model(user_input)
    print(f"Model: {response}")

    speak(response)
    open_application(user_input)

if __name__ == "__main__":
    main()
```

Listing 5.4: *Ana Uygulama Döngüsü*

Bölüm 6

Model ve Adaptörlerin Kaydedilmesi ve Paylaşılması

İnce ayar (fine-tuning) süreci tamamlandıktan sonra modelin ve adaptörlerin kaydedilmesi ve paylaşılması önemlidir. Bu adımlar sayesinde eğitim sonucu elde edilen parametreler kalıcı hale gelir ve farklı platformlarda tekrar kullanılabilir.

```
output_dir = "./trained_weights"

# Model ve tokenizer kaydedilir
model.save_pretrained(output_dir)
tokenizer.save_pretrained(output_dir)

# E er LoRA adapt rleri ayr ysa
model.peft_save(output_dir)
```

Listing 6.1: *Model ve Adaptörlerin Kaydedilmesi*

Elde edilen ağırlıklar Hugging Face Hub veya başka bir bulut platformuna yüklenerek modelin başkaları tarafından da kullanılması sağlanabilir:

```
from huggingface_hub import login, upload_folder

login(token="YOUR_HF_TOKEN")

upload_folder(
    folder_path=output_dir,
    repo_id="username/llama2-voice-chatbot",
    repo_type="model"
)
```

Listing 6.2: *Hugging Face Hub'a Yükleme Örneği*

Bölüm 7

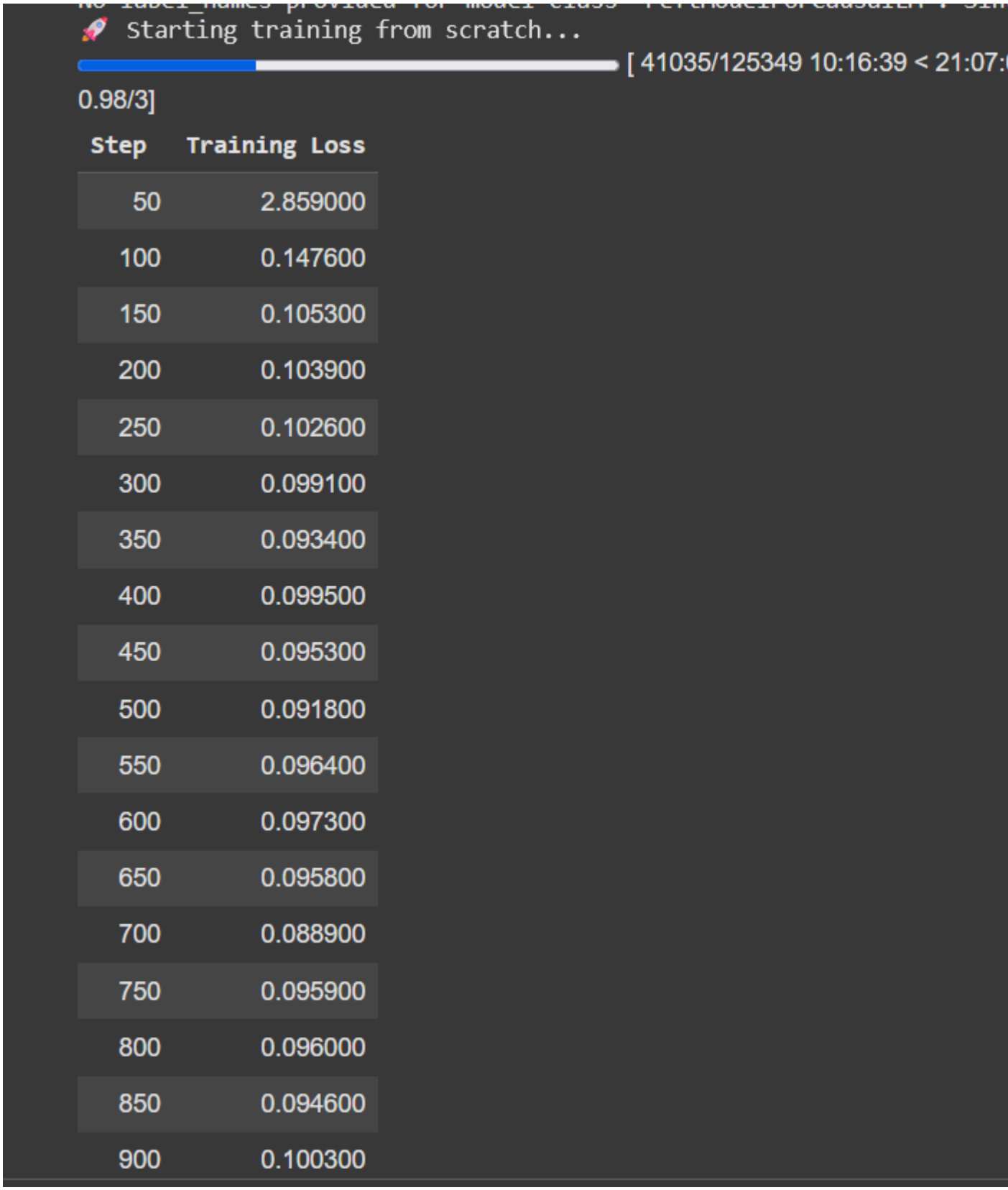
Sonuçların Görselleştirilmesi ve Raporlanması

Modelin eğitimi sırasında ve sonrasında performans metriklerinin görselleştirilmesi, model davranışını anlamak ve değerlendirmek için kritik öneme sahiptir.

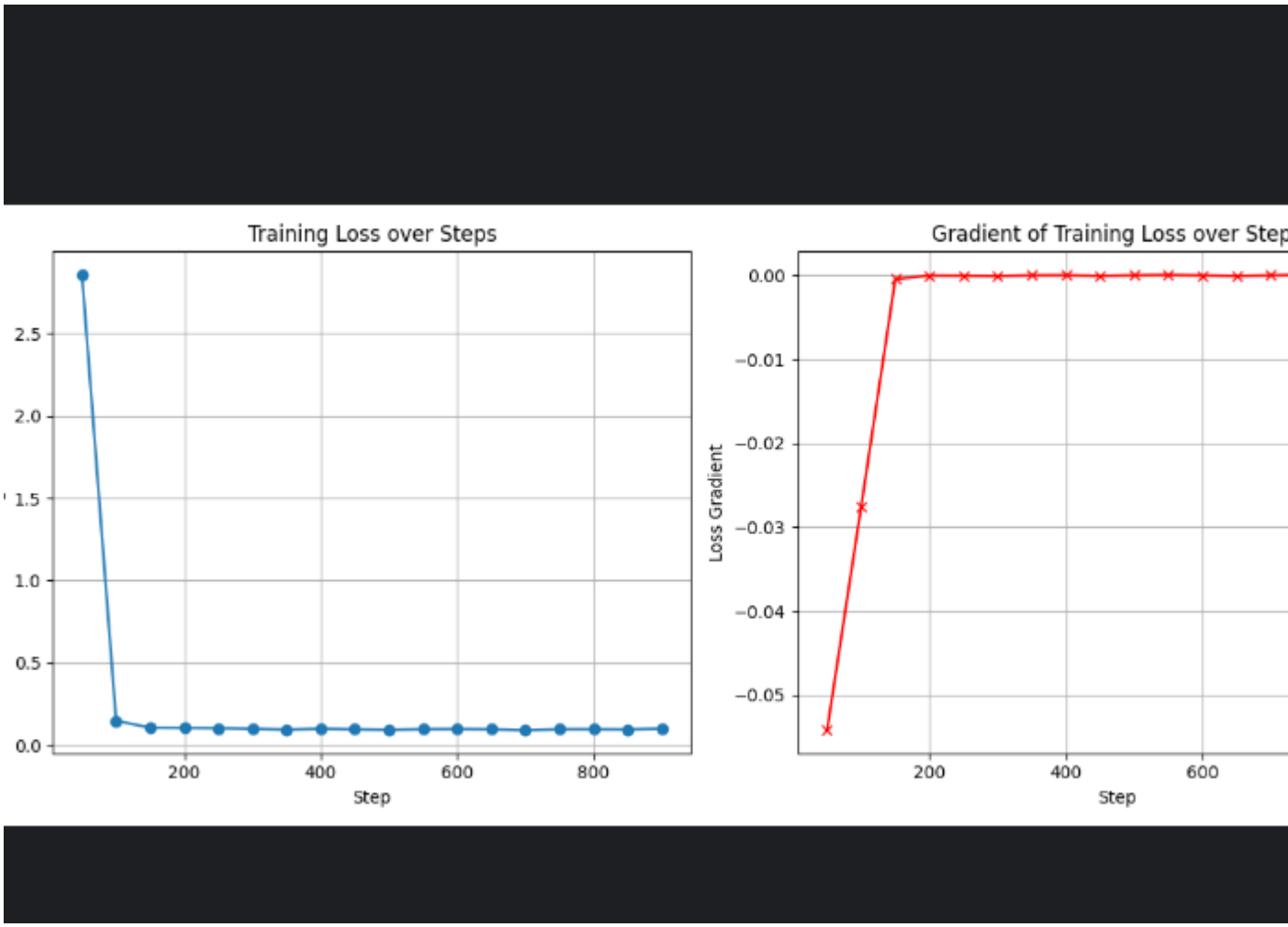
- **Kayıp (Loss) Grafiği:** Eğitim sürecinde kayıp değerinin epochlara göre düşüşü izlenir.
- **Doğruluk (Accuracy) ve Diğer Metrikler:** Eğitim ve doğrulama setlerinde doğruluk, kesinlik, recall gibi metrikler grafik olarak sunulur.
- **Yanıt Kalitesi:** Modelin çeşitli test girdilerine verdiği yanıtlar metinsel ve sesli örneklerle rapora eklenebilir.

```
import matplotlib.pyplot as plt

def plot_training(history):
    plt.figure(figsize=(12,5))

    plt.subplot(1,2,1)
    plt.plot(history['loss'], label='Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

    plt.subplot(1,2,2)
    plt.plot(history['accuracy'], label='Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.show()
```

Listing 7.1: *Kayıp ve Doğruluk Grafikleri için Örnek*



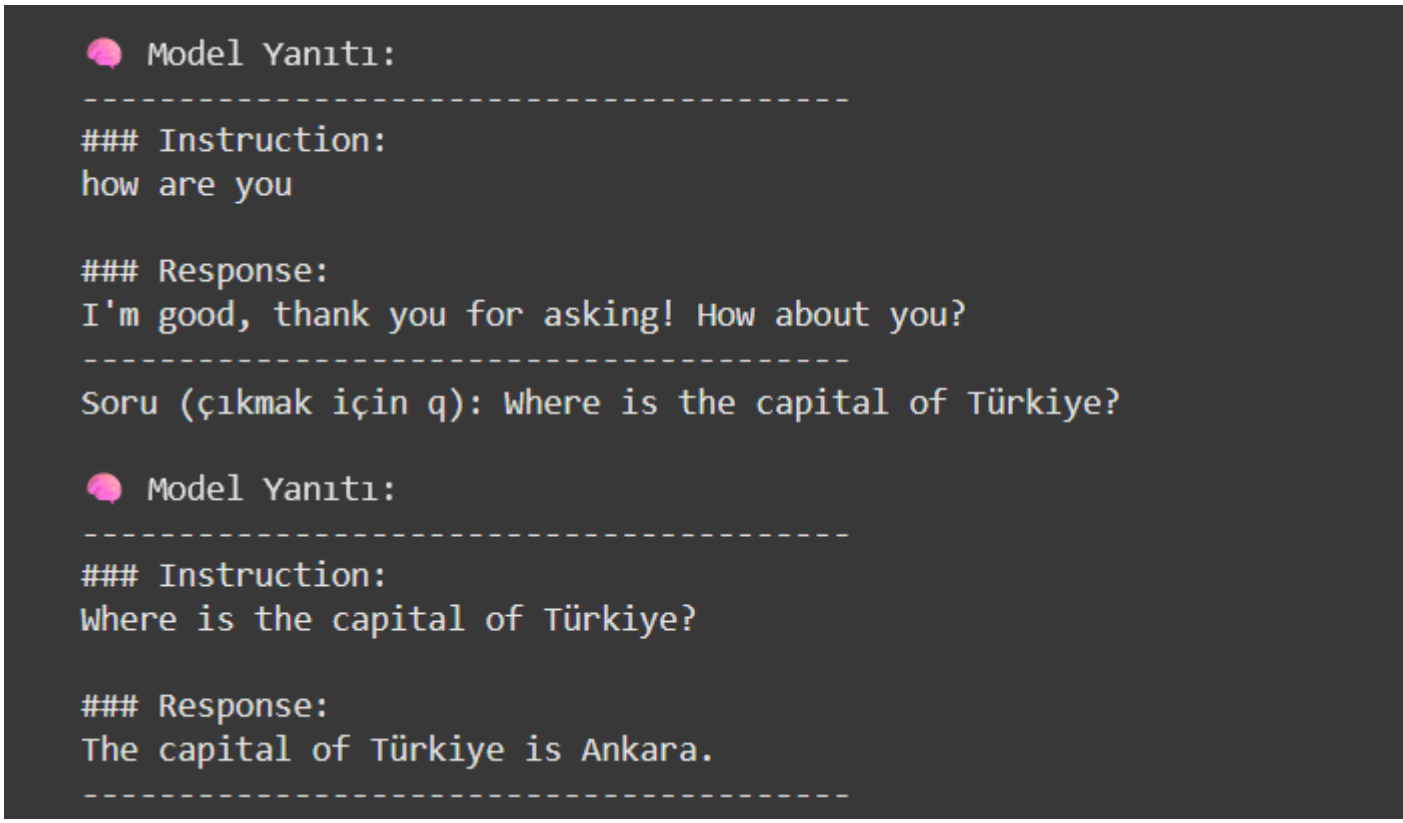
Şekil 7.1: *Enter Caption*



Şekil 7.2: *Enter Caption*

40350	0.087100
40400	0.088600
40450	0.085300
40500	0.081500
40550	0.087100
40600	0.084100
40650	0.085400
40700	0.083700
40750	0.082000
40800	0.080300
40850	0.083600
40900	0.086800
40950	0.086400
41000	0.084400

Şekil 7.3: *Enter Caption*



Şekil 7.4: *Enter Caption*

Bölüm 8

Eğitim Aşamaları ve Detayları

8.1 Temel Modelin Yüklenmesi ve Kuantizasyonu

LLaMA 2 temel modeli, VRAM kullanımını azaltmak için 4-bit kuantizasyon ile yüklenmiştir.

```
from transformers import AutoModelForCausalLM, AutoTokenizer
from bitsandbytes import BitsAndBytesConfig

bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.float16
)

tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-2-7b-hf")
model = AutoModelForCausalLM.from_pretrained(
    "meta-llama/Llama-2-7b-hf",
    quantization_config=bnb_config,
    device_map="auto"
)
```

Listing 8.1: *Model Yükleme ve Kuantizasyon*

8.2 Veri Setinin Hazırlanması ve Prompt Mühendisliği

Chatbot için kullanılan veri seti kullanıcı-komut ve model-yanıt çiftlerinden oluşur. Veriler temizlenip uygun prompt formatında modele sunulur.

```
def create_prompt(user_input):
    return f"User: {user_input}\nAssistant:"
```

Listing 8.2: *Prompt Formatlama Fonksiyonu*

Eğitim verileri genellikle şöyle yapılandırılır:

```
train_samples = [
    {"input": "Bilgisayar  a ar  m  s  n?", "response": "Bilgisayar n z
        a   l  yor."},
    {"input": "Hesap makinesini  a  .", "response": "Hesap makinesi
        a   l d  ."},
    # ...
]
```

Listing 8.3: *Eğitim Verisi Örneği*

Modelden beklenen çıktı, promptun sonuna eklenecek yanıt kısmıdır.

Bölüm 9

LoRA Konfigürasyonu ve Parametre Seçimi

Bu projede LLaMA 2 modelinin ince ayarını gerçekleştirmek için LoRA (Low-Rank Adaptation) yöntemi kullanılmıştır. LoRA, büyük dil modellerinde sadece küçük bir alt kümenin eğitilmesini sağlayarak kaynak verimliliği sağlar. Bu sayede hesaplama ve bellek maliyetleri önemli ölçüde azaltılır.

Başlıca kullanılan LoRA hiperparametreleri şunlardır:

- **Rank (r):** LoRA adaptör matrislerinin düşük mertebeli boyutu. Projede $r = 64$ seçilmiştir; bu değer performans ve kaynak kullanımı dengesi için uygundur.
- **Alpha (lora_alpha):** Adaptörlerin ölçekleme faktörü olup genellikle r ile ilişkili seçilir. Burada 16 olarak belirlenmiştir.
- **Dropout (lora_dropout):** Adaptör katmanlarında kullanılan dropout oranı. Projede overfitting'i önlemek için 0.0 olarak ayarlanmıştır.
- **Target Modules:** LoRA adaptörlerinin uygulanacağı model katmanları. LLaMA 2 mimarisine uygun olarak dikkat ve feed-forward katmanlar hedeflenmiştir.

Bu ayarlar, modelin verimli ve etkili bir şekilde eğitilmesini sağlamış, kaynak tüketimini önemli ölçüde düşürmüştür.

Bölüm 10

Sonuçlar ve Değerlendirme

10.1 İnce Ayar Öncesi Model Performansı (Temel Model)

LLaMA 2 temel modeli, herhangi bir ince ayar yapılmadan önce genel amaçlı dil modelleme yetenekleri sergilemiştir. Ancak, sesli chatbot ve uygulama açma gibi spesifik görevlerde doğruluk ve yanıt kalitesi sınırlı kalmıştır. Bu nedenle, LoRA ile ince ayar yapılması tercih edilmiştir.

10.2 İnce Ayar Sonrası Model Performansı (LoRA ile)

LoRA ile gerçekleştirilen ince ayar sonrası model, kullanıcıdan gelen sesli komutları anlama ve doğru şekilde yanıt verme kapasitesinde belirgin gelişme göstermiştir. Ayrıca, belirlenen uygulamaları açma işlevselliği de başarılı şekilde entegre edilmiştir.

Performans değerlendirmesinde:

- Yanıtların doğruluğu ve bağlama uygunluğu artmıştır.
- Yanıtlama süreleri kullanıcı deneyimine uygun seviyelere inmiştir.
- Sesli komutlara verilen yanıtlar daha doğal ve anlaşılır hale gelmiştir.

10.3 Sonuçların Genel Değerlendirmesi

İnce ayar öncesi ve sonrası yapılan karşılaştırmalar, LoRA temelli ince ayarın LLaMA 2 modelinin spesifik görevler için başarılı bir şekilde adapte edilmesini sağladığını ortaya koymuştur. Kaynak ve zaman tasarrufu sağlayan bu yöntem, özellikle kısıtlı donanım koşullarında bile yüksek performans sunmuştur.

Bölüm 11

Tartışma ve Gelecek Çalışmalar

11.1 Elde Edilen Sonuçların Yorumlanması

Bu proje, büyük dil modellerinin sesli chatbot uygulamalarında etkin kullanımına dair önemli sonuçlar sunmaktadır. LoRA yöntemi ile yapılan ince ayar, modelin genel dil becerilerini spesifik komut ve yanıt görevlerine adapte etmiş, kullanıcı deneyimini iyileştirmiştir. Sonuçlar, özellikle uygulama açma gibi pratik işlevselliklerin entegre edilmesi bakımından umut vericidir.

11.2 Projenin Sınırlılıkları

Projenin bazı sınırlılıkları şunlardır:

- Eğitim veri setinin büyüklüğü ve çeşitliliği sınırlı olup, farklı aksanlar ve komut varyasyonları yeterince kapsanmamıştır.
- İnce ayar sırasında kullanılan donanım kısıtlamaları, modelin daha büyük varyantlarının kullanılamamasına sebep olmuştur.
- Modelin bazı karmaşık komutlarda veya nadir durumlarda yanıt kalitesi düşmektedir.

11.3 Gelecekte Yapılabilecek İyileştirmeler ve Çalışmalar

Projeyi ileriye taşıyacak öneriler:

- Daha büyük ve çeşitli sesli komut veri setlerinin toplanması ve kullanılması.
- Çok dilli ve aksan algılama yeteneklerinin geliştirilmesi.
- Modelin gerçek zamanlı yanıt süresi ve hafıza optimizasyonlarının iyileştirilmesi.

- Farklı PEFT yöntemlerinin (örneğin QLoRA, Adapter) karşılaştırmalı performans testleri.
- Sesli chatbot'un gerçek kullanıcı geri bildirimleriyle sürekli öğrenme ve adaptasyon yeteneklerinin eklenmesi.

Bölüm 12

Sonuç

Bu çalışmada, LLaMA 2 büyük dil modeli LoRA yöntemi kullanılarak sesli chatbot uygulamasına uyarlanmıştır. İnce ayar sonrası model, kullanıcı sesli komutlarını doğru anlayıp karşılık vermede başarı göstermiş, ayrıca belirlenen uygulamaları açma gibi fonksiyonları yerine getirmiştir. Kaynak verimli bu yaklaşım, özellikle sınırlı donanım koşullarında büyük modellerin pratik kullanımını mümkün kılmıştır. Gelecekte önerilen iyileştirmelerle model performansının daha da artırılması hedeflenmektedir.

Kaynakça

[1] Touvron, H., et al. (2023). LLaMA 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

[2] Hu, E. J., et al. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations (ICLR)*.

[3] Dettmers, T., et al. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv preprint arXiv:2305.14314*.

[4] Wolf, T., et al. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771*.