05.08.2025

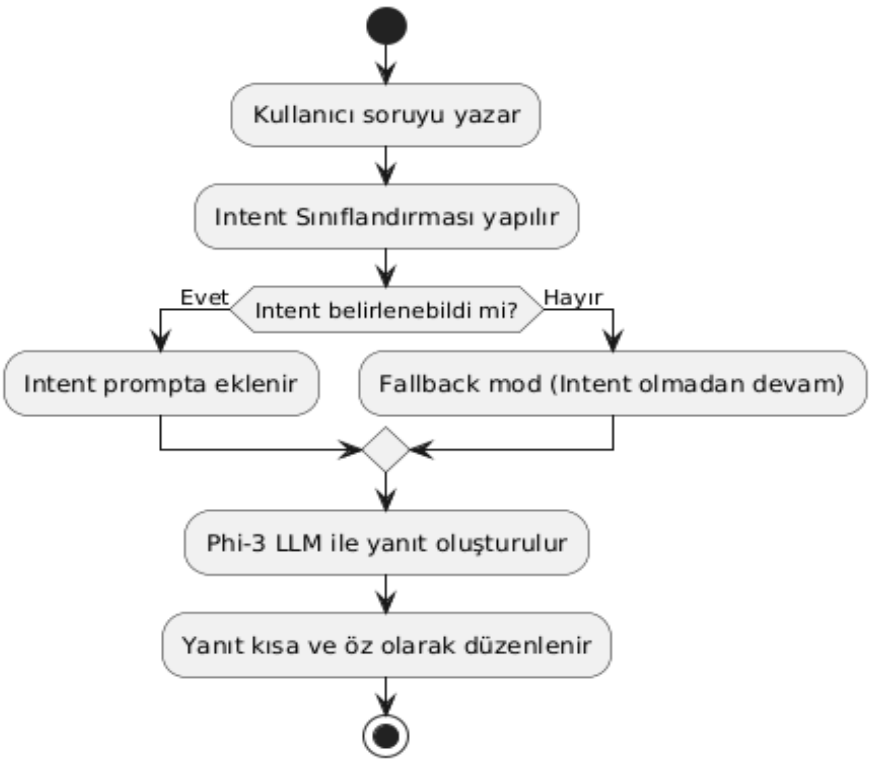
Hibrit Intent + LLM Tabanlı Chatbot Sistemi

Salih Barkın AKKAYA

1. Projenin Amacı

Bu projenin amacı, kullanıcıların sorularına hızlı, doğru ve anlamlı yanıtlar verebilen hibrit bir yapay zeka sistemi geliştirmektir. Sistem, iki temel bileşeni bir araya getirir: niyet (intent) sınıflandırması ve büyük dil modeli (LLM) tabanlı cevap üretimi. Intent sınıflandırması, kullanıcının amacını belirleyerek doğru bağlamı oluşturur ve yanıt üretim sürecinde LLM'e rehberlik eder. Bu yaklaşım, özellikle bilgi tabanlı soru-cevap sistemlerinde doğruluğu artırmak ve tutarlı yanıtlar sağlamak için gereklidir.

LLM bileşeni, kullanıcı sorusuna kısa, öz ve güvenilir yanıtlar üreterek kullanıcı deneyimini iyileştirir. Bu hibrit yapı, sadece genel sorulara cevap vermekle kalmaz, aynı zamanda yanlış veya alakasız cevapların önüne geçmek için niyet odaklı bir filtreleme katmanı sunar. Proje, müşteri destek sistemlerinden akıllı asistanlara, kurumsal bilgi tabanlarından eğitim amaçlı çözümlere kadar geniş bir kullanım alanına hitap eder. Nihai hedef, kullanıcıların bilgiye hızlı ve doğru şekilde ulaşmasını sağlarken, yapay zekâ destekli etkileşimlerin güvenilirliğini artırmaktır.



2. Kısaltmalar

Kısaltma	Açıklama
LLM	Large Language Model
CLI	Command Line Interface
NLP	Natural Language Processing
HF	Hugging Face
OOS	Out of Scope
GPU	Graphics Processing Unit
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
DeBERTa	Decoding-enhanced BERT with Disentangled Attention

3. Projenin Kapsamı

Bu proje, yalnızca **CLI (Command Line Interface)** tabanlı bir yapıda çalışacak şekilde tasarlanmıştır. Kullanıcı, komut satırı üzerinden doğal dilde bir soru yazar ve sistem hibrit yaklaşımı kullanarak yanıt üretir. Hibrit yapı; öncelikle intent sınıflandırması (DeBERTa tabanlı model) ile kullanıcının niyetini belirler, ardından LLM (Large Language Model) bileşeni olan **Phi-3-mini-4k-instruct** ile kısa ve öz bir yanıt oluşturur.

Projenin temel fonksiyonları arasında **model eğitimi**, **hibrit tahmin süreci** ve **cevap üretimi** bulunur. Tüm yapı, **Docker ile containerize edilerek** farklı platformlarda taşınabilirlik sağlanmıştır. Veri gizliliği öncelikli olarak dikkate alınmıştır. **Kullanıcı sorguları sistem içinde işlenir, herhangi bir şekilde kaydedilmez veya loglanmaz.** Bu sayede, kullanıcının girdiği veriler tamamen lokal ortamda kalır.

4. DeBERTa ile Phi-3 Mini

DeBERTa (Decoding-enhanced BERT with Disentangled Attention)

DeBERTa, Microsoft tarafından geliştirilmiş ve BERT (Bidirectional Encoder Representations from Transformers) mimarisine dayalı, daha gelişmiş bir dil modelidir. BERT ile aynı Transformer tabanlı yapıyı kullanır ancak iki önemli yenilikle ayrılır:

1**. Disentangled Attention Mekanizması**

- Geleneksel BERT, kelimelerin hem pozisyon bilgisini hem de içerik bilgisini tek bir embedding vektöründe birleştirerek işler.
- DeBERTa, bu iki bilgiyi ayırır (disentangle) ve attention mekanizması içerisinde ayrı ayrı değerlendirir.
- Bu sayede model, hem kelimenin anlamını hem de cümledeki konumunu daha iyi kavrayabilir. Özellikle karmaşık cümle yapılarında bağlamsal anlayışı artırır.

2**. Enhanced Mask Decoder**

- Maskeli dil modellemesinde (MLM), DeBERTa, decoder aşamasında pozisyon bilgisini daha verimli kullanır.
- Bu geliştirme, özellikle intent classification gibi bağlam duyarlı görevlerde doğruluk oranını yükseltir.

Performans Avantajı:

- BERT ve RoBERTa'ya kıyasla daha yüksek doğruluk ve F1 skorları.
- NLP benchmark'larında (GLUE, SQuAD) lider modellerden biri.

Projedeki Rolü:

Kullanıcının sorusunu analiz ederek hangi niyet (intent) kategorisine ait olduğunu belirler (ör. "Bugün hava nasıl?" → intent: weather). Ayrıca bir güven skoru üretir. Bu skor, LLM'e intent bilgisinin eklenip eklenmeyeceğini belirlemede kullanılır.

Phi-3 Mini (LLM)

Phi-3 Mini, Microsoft tarafından geliştirilen **Large Language Model (LLM)** ailesinin küçük ama güçlü bir üyesidir. Özellikleri:

1. Küçük Boyut, Yüksek Verimlilik

- Yaklaşık 3.8 milyar parametre (4B civarı) ile GPT gibi dev modellere göre çok daha hafif.
- Düşük kaynak tüketimi sayesinde hem CPU hem de GPU üzerinde çalıştırılabilir.
- 4-bit quantization (BitsAndBytes) ile bellek kullanımı büyük ölçüde azaltılır, bu da Docker konteyner içinde çalıştırılmasını mümkün kılar.

2. Geniş Kullanım Alanı

- Metin tamamlama, özetleme, çeviri, kod yazma, soru-cevap sistemleri için optimize edilmiştir.
- 4K context window ile uzun soruları anlayabilir, kısa ve anlamlı yanıtlar üretebilir.

3. Güvenilir Yanıt Üretimi

- Modelin eğitimi, temiz ve seçilmiş veri üzerinde yapılmıştır, bu da yanıt kalitesini artırır.
- Yanıtlar kısa, öz ve kurallara uygun şekilde oluşturulur. Prompt ile yanıt sınırları (örn. max 2 cümle) kontrol edilir.

Projedeki Rolü:

Intent modeli tarafından belirlenen bağlam bilgisini (varsa) prompt'a ekleyerek daha doğru yanıt verir. Kullanıcı sorusuna maksimum iki kısa cümle içinde cevap üretir. Eğer yanıt verilemeyecek durumdaysa "I don't know." döner, böylece yanlış bilgi riski azalır.

Hibrit Yapıda Avantajlar

- DeBERTa + Phi-3 Mini kombinasyonu, hem niyetin anlaşılmasını hem de dil modelinin güçlü üretim kapasitesini kullanarak yüksek doğruluk sağlar.
- Bu hibrit yapı sayesinde:
    - Alakasız cevap oranı azalır (çünkü intent filtreleme yapar).
    - Cevaplar daha kısa ve öz (LLM prompt kuralları sayesinde).
    - Performans ve maliyet dengesi sağlanır (küçük boyutlu LLM + verimli quantization).

5. Kurulum Çalıştırma ve Gereksinimler (Docker ile)

Gereksinimler:

Bu proje, Hugging Face ekosistemi ve PyTorch tabanlıdır. Model eğitimi, değerlendirme ve hibrit sistemin çalışabilmesi için gerekli olan kütüphaneler ve indirilmesi gereken modeller aşağıda belirtilmiştir.

Proje için yüklenmesi gereken kütüphaneler requirements.txt dosyasında belirtilmiştir. Bu dosya şu paketleri içerir:

accelerate==1.9.0 # Dağıtık eğitim ve model hızlandırma

aiohttp>=3.8.5,<3.13 # Asenkron HTTP istekleri için

datasets==2.19.1 # Hugging Face dataset yönetimi

evaluate==0.4.2 # Model değerlendirme metrikleri

transformers==4.38.2 # Hugging Face modelleri ve tokenizer

tokenizers==0.15.2 # Hızlı tokenizasyon

safetensors>=0.4.3 # Model ağırlıkları için güvenli format

huggingface-hub>=0.24.0 # HF Hub entegrasyonu

bitsandbytes==0.46.1 # 4-bit ve 8-bit quantization

scikit-learn==1.3.2 # LabelEncoder, metrikler

scipy==1.11.4 # Bilimsel hesaplamalar

pandas==2.2.2 # Veri işleme

matplotlib==3.8.4 # Grafik çizimleri

seaborn==0.13.2 # Görselleştirme

tiktoken==0.5.2 # Token sayımı (LLM için)

sentencepiece==0.1.99 # Tokenizer desteği

xxhash==3.4.1 # Hızlı hashing (HF datasets)

pyarrow==15.0.2 # Dataset I/O

Tüm kütüphaneleri yüklemek için:

pip install -r requirements.txt

Kurulum ve Çalıştırma:

Bu projede Docker kullanımı, ortam bağımlılıklarını ortadan kaldırmak ve yazılımın taşınabilirliğini sağlamak amacıyla tercih edilmiştir. Docker imajı PyTorch ve CUDA desteğiyle hazırlanmıştır ve hem model eğitimi hem de hibrit sistemin çalıştırılması için gerekli tüm bileşenleri içerir. Kurulum ve çalıştırma adımları Linux ve Windows işletim sistemleri için aşağıda açıklanmıştır.

Linux Üzerinde Kurulum ve Çalıştırma

İlk olarak sistemde Docker kurulumu yapılmalıdır. Bunun için terminalden aşağıdaki komut çalıştırılır:

sudo apt-get update

sudo apt-get install -y docker.io

sudo systemctl enable --now docker

Kurulum sonrası docker --version komutu ile doğrulama yapılabilir. GPU desteği için NVIDIA Container Toolkit kurulması gerekir. Bunun için aşağıdaki komutlar kullanılabilir:

sudo apt-get install -y nvidia-container-toolkit

sudo systemctl restart docker

Kurulumun başarılı olup olmadığını doğrulamak için şu komut çalıştırılır:

docker run --rm --gpus all nvidia/cuda:11.8.0-base nvidia-smi

Bu işlemlerin ardından proje dizinine giderek Docker imajı oluşturulur:

docker build -t hybrid-system .

İmaj oluşturulduktan sonra, container GPU ile çalıştırılmak istenirse aşağıdaki komut kullanılır:

docker run --gpus all -it hybrid-system

GPU desteği yoksa CPU modunda çalıştırmak için:

docker run -it hybrid-system

Container çalıştırıldığında otomatik olarak start.sh scripti devreye girer. Bu script, eğer deberta_intent_model dizini mevcut değilse modeli eğitir ve ardından hibrit sistemi başlatır. Hugging Face model indirme işlemlerinde tekrar indirmeleri önlemek için opsiyonel olarak cache dizini belirlenebilir:

export TRANSFORMERS_CACHE=/app/cache

Windows Üzerinde Kurulum ve Çalıştırma

Windows ortamında öncelikle Docker Desktop kurulmalıdır. Kurulum sırasında WSL 2 backend ve NVIDIA GPU desteğinin etkinleştirilmesi gereklidir. Kurulum tamamlandıktan sonra PowerShell üzerinden Docker'ın kurulumunu şu komut ile doğrulayabilirsiniz:

docker –version

GPU desteği için gerekli NVIDIA Container Toolkit entegrasyonu Docker Desktop ile birlikte gelir. Bu nedenle yalnızca doğrulama yapmak için aşağıdaki komut kullanılabilir**:**

docker run --rm --gpus all nvidia/cuda:11.8.0-base nvidia-smi

İmaj oluşturma aşamasında PowerShell veya CMD üzerinden proje dizinine giderek şu komut çalıştırılır:

docker build -t hybrid-system .

Container'ı GPU ile çalıştırmak için:

docker run --gpus all -it hybrid-system

GPU olmadan CPU modunda çalıştırmak için:

docker run -it hybrid-system

İhtiyaç halinde Hugging Face cache klasörü ayarlanabilir. Windows için bu işlem şu şekilde yapılır:

setx TRANSFORMERS_CACHE "C:\app\cache"

Container çalıştırıldığında Linux'ta olduğu gibi start.sh scripti otomatik olarak modeli eğitir (model yoksa) ve hibrit sistemi CLI modunda çalıştırır.

Ek Bağımlılıklar

CUDA 11.8 ve NVIDIA Driver: GPU desteği için gerekli.

NVIDIA Container Toolkit: Docker GPU entegrasyonu için gerekli.

Git: Kod versiyon kontrolü ve model indirme için.

6. Yazılımla İlgili Fonksiyonel Gereksinimler

Bu projenin yazılımsal gereksinimleri, hibrit yapının iki temel bileşeni olan intent sınıflandırma modeli ve büyük dil modeli (LLM) etrafında şekillenmektedir. İlk olarak, intent sınıflandırma modeli Hugging Face'in **Trainer API** yapısı kullanılarak eğitilecektir. Bu API, eğitim sürecinde **optimizer, learning rate scheduler ve gerektiğinde early stopping** gibi mekanizmaları otomatik olarak yönetir. Bu sayede hem eğitim süreci basitleşir hem de performans optimize edilir. Modelin eğitimi için kullanılacak veri seti **data_full.json** dosyasında tutulacak, model olarak **microsoft/deberta-v3-base** seçilecektir. Değerlendirme sırasında **Accuracy ve Weighted F1** metrikleri dikkate alınacak, en iyi model çıktı olarak **./deberta_intent_model** klasöründe saklanacaktır. Bu yapı, sistemin soruların niyetini doğru şekilde anlayabilmesini sağlayan kritik bileşendir.

Sistemin ikinci ana bileşeni olan büyük dil modeli ise Hugging Face API'si üzerinden yüklenecektir. Bu noktada **microsoft/Phi-3-mini-4k-instruct** modeli kullanılacak ve bellekte verimli bir şekilde çalıştırılabilmesi için **BitsAndBytes kütüphanesiyle 4-bit quantization** uygulanacaktır. Quantization sayesinde **GPU bellek kullanımı** ciddi oranda azalır ve sistem daha düşük donanımlarda bile çalışabilir hale gelir. Model yükleme aşamasında Hugging Face'in sağladığı **AutoModelForCausalLM.from_pretrained(...)** fonksiyonu kullanılacaktır. Bu yöntem, yüksek performanslı metin üretimi için optimize edilmiştir.

Yanıt üretim süreci için önemli bir gereksinim de kullanıcıya verilen cevabın kısa ve öz olmasıdır. LLM'den gelen yanıtlar en fazla iki kısa cümle ile sınırlandırılacaktır. Bunun sağlanması için LLM'e gönderilen **prompt** içerisine özel talimatlar yerleştirilecektir. Bu talimatlar arasında **"ONLY output the final answer, max 2 short sentences."** gibi açık bir yönerge bulunacaktır. Ek olarak, modelin yanıtı gereksiz tekrar veya sorunun yeniden yazılması gibi durumları engellemek için **"Do NOT repeat the question."** kuralı da eklenir. Yanıtın belirsiz olduğu durumlarda modelin güvenilirlik açısından **"I don't know."** şeklinde cevap vermesi istenir. Ayrıca token sınırı **MAX_NEW_TOKENS = 80** olarak belirlenmiştir, böylece üretilen yanıtın uzunluğu kontrol altına alınır. Tüm bu kurallar, sistemin hızlı, anlaşılır ve güvenilir cevaplar üretmesini garanti altına alır.

7. Yazılımla İlgili Fonksiyonel Olmayan Teknik Gereksinimler

Sistem, öncelikli olarak **CLI (Command Line Interface)** tabanlı kullanım için tasarlanmıştır ve bu nedenle başlangıç aşamasında yalnızca tek kullanıcı desteği sunmaktadır. Kullanıcı, komut satırından sisteme soru sorabilir ve yanıt alabilir. Gelecekte yapılacak genişletmelerle, bu yapı web tabanlı bir arayüz veya API ile çok kullanıcı desteğine uygun hale getirilebilir.

Erişim şekli açısından sistem tamamen lokal çalışacak şekilde planlanmıştır. Kullanıcı, Docker desteği sayesinde platform bağımsız bir şekilde uygulamayı kurabilir ve çalıştırabilir.

Güvenlik gereksinimleri kapsamında, modeller Hugging Face API'si üzerinden indirilecektir. Bu nedenle, **gerekli durumlarda HF API token yönetimi** için uygun mekanizmalar kullanılacaktır. Ancak proje kapsamında kullanılan modellerin çoğu public olduğu için temel seviyede güvenlik yeterli olacaktır.

Gizlilik açısından, kullanıcı tarafından girilen sorular sistem içinde işlenir ancak herhangi bir loglama yapılmaz. İstenirse opsiyonel olarak loglama eklenebilir, fakat mevcut tasarımda veri gizliliği sağlamak için bu özellik devre dışıdır.

Son olarak, sistemi geliştirmek ve bakımını yapmak için geliştiricilerin temel düzeyde **Hugging Face ekosistemi** ve **PyTorch** bilgisine sahip olması gereklidir. Bu bilgi, model eğitimi, optimizasyonu ve inference sürecini sorunsuz yönetmek için kritik öneme sahiptir.

8. SWOT Analizi

Güçlü Yönler (Strengths)

- Hibrit yapı (LLM + Intent): Kullanıcı sorularını hem bağlam hem de anlam açısından analiz ederek daha güvenilir ve doğru yanıtlar üretilir.
- Açık kaynaklı kütüphaneler kullanımı: Hugging Face, PyTorch ve BitsAndBytes gibi güçlü ekosistemler sayesinde hızlı geliştirme ve optimizasyon imkânı.
- Docker entegrasyonu ile platform bağımsız çalışma.

Zayıf Yönler (Weaknesses)

- LLM'in hızlı çalışması için güçlü bir GPU gerekir; bu durum maliyeti artırır ve düşük donanımlarda performansı düşürür.
- Kullanıcı sorularının geçmişi kaydedilmediği için bağlam devamlılığı sağlanamaz.
- Sistem yalnızca CLI üzerinden kullanılabiliyor, kullanıcı deneyimi sınırlı.

Fırsatlar (Opportunities)

- Web entegrasyonu: Gelecekte web tabanlı arayüz ve API eklenerek daha geniş bir kullanıcı kitlesine ulaşılabilir.
- Yeni ve spesifik veri setleriyle fine-tuning yapılarak sektörel veya görev odaklı asistanlara dönüşebilir.

Tehditler (Threats)

- Daha karmaşık veya spesifik görevler için ek eğitim (fine-tuning) gerektirir, bu da veri toplama ve etiketleme maliyetini artırır.
- Hugging Face veya Microsoft modellerinde gelecekte lisanslama değişiklikleri entegrasyon zorlukları yaratabilir..

9. Kaynakça

1. **Hugging Face Transformers** – Model ve tokenizer yükleme, Trainer API
   https://huggingface.co/docs/transformers
2. **Hugging Face Datasets** – Veri seti yükleme ve işleme
   https://huggingface.co/docs/datasets
3. **microsoft/deberta-v3-base Model Kartı**
   https://huggingface.co/microsoft/deberta-v3-base
4. **microsoft/Phi-3-mini-4k-instruct Model Kartı**
   https://huggingface.co/microsoft/Phi-3-mini-4k-instruct
5. **BitsAndBytes (Quantization) – Hugging Face Entegrasyonu**
   https://huggingface.co/docs/transformers/main_classes/quantization
6. **PyTorch Resmi Dokümantasyonu** – Model eğitimi ve GPU kullanımı
   https://pytorch.org/docs/stable/index.html
7. **Docker Resmi Dokümantasyonu** – Dockerfile ve Container yönetimi
   https://docs.docker.com/
8. **NVIDIA Container Toolkit** – Docker GPU desteği
   https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html