

LLM Booster SDK Developer Edition

Kurulum, komut satırı ve Python API kullanımı

Developer Edition lisansı: Bu sürüm yalnızca değerlendirme ve geliştirme içindir. Üretim veya ticari kullanım ve yeniden dağıtım lisans kapsamında değildir. Özellikler teknik olarak kısıtlanmaz. Dağıtımdan önce LICENSE-DEVELOPER.md belgesini inceleyin.

Bu kılavuz ne anlatır? LLM Booster, ajanlara ait metin bağlamını yerel olarak düzenleyen, yönlendiren ve deneysel kodlama yardımcıları sunan bir Python paketidir. Varsayılan SDK akışı herhangi bir LLM servisini çağırır.

1. Kurulum

Paket Python 3.10 veya üzerini gerektirir. Developer Edition ZIP arşivini dağıtım konumundan indirin ve aşağıdaki PowerShell komutlarını arşivin bulunduğu klasörde çalıştırın. Komutlar arşivi açar, sanal ortam kurar ve arşivdeki wheel'i yükler:

Developer Edition ZIP'inden kurulum (Windows PowerShell)

```
Expand-Archive .\llm_booster_developer_edition-0.1.0.zip .\llm-booster-dev
Set-Location .\llm-booster-dev
py -0p
py -3.10 -m venv .venv
$wheel = Get-ChildItem -Filter 'llm_booster-*.whl' | Select-Object -First 1
if (-not $wheel) { throw "Developer Edition wheel not found." }
.\.venv\Scripts\python.exe -m pip install $wheel.FullName
$cli = ".\venv\Scripts\llm-booster.exe"
& $cli --help
```

ZIP arşivi içinde wheel, README, lisans ve Türkçe/İngilizce PDF kılavuzları bulunur. Arşivdeki örnek JSON bağlamı examples/context.json yolundadır. Kaynak depodan geliştirme kurulumu yapıyorsanız depo kökünde py -3.10 -m venv .venv ve .\venv\Scripts\python.exe -m pip install -e . kullanın.

Kurulumu doğrulama

```
& $cli run --help
```

2. Komut satırından çalıştırma

CLI girdisi, ajan adlarını bağlam metinlerine eşleyen UTF-8 JSON dosyasıdır. Örnek dosya: examples/context.json.

```
{
  "billing_agent": "Invoice 426 TL is not paid.",
  "device_agent": "SIM 123 is active.",
  "support_agent": "Customer requested escalation."
}
```

Dört çalıştırma modu

Mod	Ne yapar? / Ne zaman kullanılır?
baseline	Tüm ajanları ve özgün metni değiştirmeden kullanır; karşılaştırma tabanıdır.
cleanup	Varsayılan. Satır kenarlarını temizler; boş ve yinelenen satırları çıkarır.
semantic	İstekle en çok eşleşen Top-K ajanı seçer ve seçilen bağlamı temizler.
linear_network_coding	Özgün metni korur; GF(2) kaynak/parite paketleri ekler. Tek paket kaybını tolere eder.

Mod örnekleri

```
$cli = ".\venv\Scripts\llm-booster.exe"
& $cli run --mode baseline --request "Check the invoice" `
  --context examples/context.json
& $cli run --mode cleanup --request "Check the invoice" `
  --context examples/context.json
& $cli run --mode semantic --top-k 2 --request "Check the invoice" `
  --context examples/context.json
& $cli run --mode linear_network_coding `
  --request "Check the invoice" --context examples/context.json
```

Komutlar PowerShell içindir; satır devamı karakteri olan ters tırnak (`) satırın en son karakteri olmalıdır. Her komut tek satırda da çalıştırılabilir.

CLI seçenekleri ve özel ajanlar

Seçenek	Gerekli?	Açıklama
--mode	Hayır	baseline, cleanup, semantic veya linear_network_coding. Varsayılan: cleanup.
--request	Evet	Yönlendirme için kullanılacak boş olmayan görev/istek metni.
--context	Evet	Ajan adından bağlam metnine eşleme yapan UTF-8 JSON dosyası.
--agents	Hayır	Ajan tanımlarını içeren UTF-8 JSON listesi; varsayılan ajanlar kullanılır.
--top-k	Hayır	semantic modunda seçilecek ajan sayısı; pozitif tam sayı, varsayılan 2.

Özel ajan tanımı

Bağlamdaki her anahtar, yapılandırılmış ajanlardan birinin benzersiz name değeriyle eşleşmelidir. Tanımlarda açıklama ve beceri anahtar sözcükleri isteğe bağlıdır.

```
[
  {"name": "billing_agent", "description": "Billing support",
   "skills": ["invoice", "payment"]},
  {"name": "device_agent", "skills": ["SIM", "activation"]}
]
```

```
$cli = ".\venv\Scripts\llm-booster.exe"
& $cli run --mode semantic --top-k 1 `
  --request "Find the unpaid invoice" `
  --context examples/context.json --agents agents.json
```

Çıktı ve hata davranışı

- Başarılı çalıştırma sonucu stdout'a girintili JSON olarak yazılır.
- Sonuçta mode, request, selected_agents, context ve metrics alanları bulunur.
- linear_network_coding modunda coded_context alanı da eklenir.
- Geçersiz JSON, bilinmeyen ajan anahtarları veya geçersiz seçenek hata üretir ve CLI sıfırdan farklı kodla çıkar.

3. Python API

Booster, aynı dört modu Python kodundan kullanır. Aşağıdaki örnekte temizlenmiş bağlam ve metrikler alınır:

```
from llm_booster import Booster

booster = Booster()
context = {
  "billing_agent": "Invoice 426 TL is not paid.\n"
                  "Invoice 426 TL is not paid.",
  "device_agent": "SIM 123 is active.",
}

result = booster.run(
  "Check the invoice",
  context,
  mode="cleanup",
)

print(result["selected_agents"])
print(result["context"])
print(result["metrics"])
```

Özel ajanlar, varsayılan ajan listesinin yerine verilir. Mod adı string veya RunMode enum değeri olabilir:

```
from llm_booster import Booster, RunMode
```

```
booster = Booster([
    {"name": "billing_agent", "skills": ["invoice", "payment"]}
])
result = booster.run(
    "Check invoice",
    {"billing_agent": "Invoice is unpaid."},
    mode=RunMode.SEMANTIC,
    top_k=1,
)
```

Modları doğru seçme

Mod	Ajan seçimi	Metin etkisi ve dikkat noktası
baseline	Tüm yapılandırılmış ajanlar	Metin olduğu gibi korunur; eksik ajan bağlamı boş metin olur.
cleanup	Tüm yapılandırılmış ajanlar	Satır başı/sonu boşlukları ve boş satırlar temizlenir; yinelenen satırlar kaldırılır. Tekrarlar anlam taşıyorsa baseline seçin.
semantic	İstekle en çok eşleşen Top-K	Anahtar sözcük tabanlı sıralama yapar; seçilen ajanların bağlamına cleanup uygulanır. Eşleşme sıfır olsa da Top-K seçilir.
linear_network_coding	Tüm yapılandırılmış ajanlar	Özgün metin korunur; JSON paketleri ve parite yükü eklenir. Herhangi bir tek paket kaybı onarılabilir.

Dikkat edilmesi gerekenler

- cleanup tekrarları ve boşlukları değiştirir; sayılar, tarihler, noktalama, büyük/küçük harf ve olumsuzluk korunur.
- semantic modu anahtar sözcük eşleşmesine dayanır; model tabanlı anlamsal benzerlik veya LLM çağırısı yapmaz.
- top_k pozitif bir tam sayı olmalıdır. Top-K bağlamı, tüm ajanların toplamından küçük olabilir.
- metrics içindeki sözcük sayımı boşlukla ayrılan sözcüklerdir; tokenizer tokenı, maliyet, gecikme veya tasarruf garantisi değildir.

4. Paket kaybı kodlaması ve GF(256) deneyi

linear_network_coding modu, encode_context ve decode_context ile çalışan GF(2) sistematik XOR codec'idir. Her bağlam girdisi kaynak paketi olur; ayrıca tek bir XOR parite paketi eklenir. Tek bir paketin silinmesi onarılabilir.

```
from llm_booster import Booster, decode_context

result = Booster().run(
    "Check the invoice",
    {"billing_agent": "Invoice 426 TL is unpaid."},
    mode="linear_network_coding",
)
bundle = result["coded_context"]
bundle["packets"].pop(0) # Tek paket kaybını simüle et
recovered = decode_context(bundle)
```

rlnc_roundtrip ise ayrı bir GF(256) benchmark deneyidir. Bayt verisine terslenebilir rastgele doğrusal dönüşüm uygular ve aynı fonksiyon içinde geri çözer; çalışması için tüm kodlu bloklar gereklidir, paket kaybı kurtarma testi değildir.

```
from llm_booster import rlnc_roundtrip

decoded, metrics = rlnc_roundtrip(b"sample bytes")
assert decoded == b"sample bytes"
print(metrics["exact_recovery"])
print(metrics["wire_bytes"])
```

Her iki deney de bağlamı sıkıştırdığını veya LLM tokenlarını azalttığını iddia etmez. GF(2) modu fazladan parity ve Base64 yükü ekler; şifreleme ve bozulma tespiti sağlamaz.

5. Doğrulama ve sorun giderme

SDK testlerini çalıştırma

```
.\.venv\Scripts\python.exe -m unittest discover -s tests -v
```

Sık karşılaşılan durumlar

Durum	Kontrol / çözüm
llm-booster tanınmıyor	Kurulum yaptığınız sanal ortamı etkinleştirin veya .venv\Scripts\python.exe -m llm_booster biçimini kullanın.
Context contains unknown agents	context.json anahtarlarının varsayılan ya da --agents ile verilen adlarla birebir eşleştiğini kontrol edin.
Invalid JSON / dosya okunamıyor	Dosya yolunu ve UTF-8 JSON biçimini doğrulayın; context kökü ajan adı/metin eşleme nesnesi olmalıdır.
Insufficient independent packets	Paket kaybı sayısı mevcut parity kapasitesini aşıyor veya paketler yineleniyor olabilir; tam paketi ve metadata'yı koruyun.

Özet

Kurulumdan sonra hızlı deneme için llm-booster run kullanın; uygulama kodunda Booster.run çağırın. Metni değiştirmeyen referans için baseline, tekrar temizliği için cleanup, ajan seçimi için semantic ve tek paket silinmesine toleranslı GF(2) taşıma deneyi için linear_network_coding uygundur.

Paket sürümü: 0.1.0 • Kılavuz, bu depodaki README ve SDK davranışına göre hazırlanmıştır.