

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

Öğrenci bilgileri:

Adı Soyadı :Emine ŞENER
Numarası :21360859058
Bölümü :Bilgisayar Mühendisliği
E-posta adresi :eminesener063@gmail.com
BTÜ-İMEP akademik danışmanı :Doç. Dr. Ergün GÜMÜŞ

BTÜ-İMEP Kurum Bilgileri:

Adı : Huawei
Adresi : Saray Mah. Ahmet Tevfik ileri Cad. Onur Ofis Park İş Merkezi Sit. A1Blok
Apt. No.10 B/1 Ümraniye İstanbul
İnternet sitesi : <https://www.huawei.com/tr/>
Görev alınan birim : Artificial Intelligence Research And Development

Görev alınan birim yöneticisinin

Adı soyadı : Halit ÖRENBAŞ
e-posta adresi : halit.orenbas@huawei.com

BTÜ İMEP sektör danışmanının

Adı soyadı : Halit ÖRENBAŞ
e-posta adresi : halit.orenbas@huawei.com
Telefon numarası: : +90 533 960 40 67

Çalışma dönemi bilgileri:

Çalışma dönemi : 30.09.2024-10.01.2025

A. Çalışma Dönemi Faaliyetleri:

1. Görevlerinizi tanımlayınız.

Öğrencinin görev alanı:Araştırma-geliştirme

- 1.1 TravelPlanner: A Benchmark for Real-World Planning with Language Agents Makale Analizi
- 1.2 Travel Planner Sorgularının Agent Kullanmadan Büyük Dil Modeline Verilmesi
- 1.3 TravelPlanner Implementasyonu
- 1.4 LangGraph ve Groq ile TravelPlanner Geliştirilmesi
- 1.5 LangGraph ve HuggingFace ile React Agent Temelli Travel Planner Geliştirilmesi
- 1.6 LangGraph ve HuggingFace ile Sequential Düzendeki Travel Planner Geliştirilmesi
- 1.7 Büyük Dil Modeli Uygulamaları için UI Geliştirilmesi
- 1.8 Travel Planner: Ara Çıktıların UI'da Gösterilmesi

1.1 TravelPlanner: A Benchmark for Real-World Planning with Language Agents Makale Analizi

1.1.1.Görev Tanımı ve Hedefler

Travel Planner projesi için gerekli olan veri seti ve kaynak kod Github[1] üzerinden açık kaynak olarak yayımlanmıştır. Kullanılan teknolojiler, uygulanan yöntemler ve alınan sonuçlar makalede[2] ayrıntılı olarak yer almaktadır. Söz konusu olan görev, makalenin incelenmesi, analiz edilmesi, makale içerisinde yer alan ve öğrenilmesi gereken konu başlıklarının belirlenmesi ve benzer bir Seyahat Planlayıcısının oluşturulmasını hedeflemektedir.

Görevin Alt Başlıkları:

- Makalenin analiz edilmesi
- Makalede bahsedilen projenin anlaşılması
- Makalede yer alan öğrenilmesi gereken konu başlıklarının belirlenmesi
- Makale kaynak kodunun anlaşılması ve kodun çalıştırılması için gerekli olan teknoloji ve yöntemlerin belirlenmesi

1.1.2.Teknolojik Altyapı ve Uygulanan Yöntemler

Travel Planner makalesi analiz edildi. Travel Planner, kullanıcıdan string formatında sorgular alır ve bunları değerlendirip uygun seyahat planı çıktıları oluşturur. Projenin değerlendirilmesi için kullanılan sorguların birkaç örneği Şekil 1’de gösterilmiştir.

```
Please plan a trip for me starting from Sarasota to Chicago for 3 days, from March 22nd to March 24th, 2022. The budget for this trip is set at $1,900.
-----
Seeking assistance to develop a travel itinerary for a 3-day trip for one person. The trip will begin in Washington, with Tampa as the destination from March 25th through March 27th, 2022. The budget for this journey is $1,800.
-----
Please create a travel plan that starts in Charleston and leads to St. Louis over a span of three days, from March 16th to March 18th, 2022. This trip is designed for one individual with a budget of $900.
-----
Please design a travel plan that departs from Jacksonville, heading for Norfolk for 3 days spanning from March 3rd to March 5th, 2022. The journey is designed for one person, with a budget of $1,900.
-----
Please create a travel itinerary for a solo traveler departing from Jacksonville and heading to Los Angeles for a period of 3 days, specifically from March 25th to March 27th, 2022. The budget for this trip is now set at $2,400.
```

Şekil 1

Makaleye Şekil 1’de yer alan tipte sorguları büyük dil modeline verdiğinde Şekil 2’de gösterildiği gibi bir plan oluşturulmasını hedeflemektedir.

```

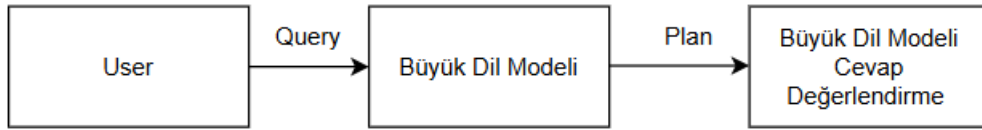
Query: Please create a travel plan for a 3-day trip from Missoula to Dallas scheduled from March 23rd to March 25th, 2022. The budget for this trip is set at $1,900.

Plan:
[
  {
    "day": 1,
    "current_city": "from Missoula to Dallas",
    "transportation": "Flight Number: F3604254, from Missoula to Dallas, Departure Time: 14:27, Arrival Time: 18:26",
    "breakfast": "-",
    "attraction": "-",
    "lunch": "-",
    "dinner": "Coconuts Fish Cafe, Dallas",
    "accommodation": "1BR, elevator, kitchen, doorman1, Dallas"
  },
  {
    "day": 2,
    "current_city": "Dallas",
    "transportation": "-",
    "breakfast": "Cafe Gatherings, Dallas",
    "attraction": "The Dallas World Aquarium, Dallas; The Sixth Floor Museum at Dealey Plaza, Dallas;",
    "lunch": "1918 Bistro & Grill, Dallas",
    "dinner": "Yanki Sizzlers, Dallas",
    "accommodation": "1BR, elevator, kitchen, doorman1, Dallas"
  },
  {
    "day": 3,
    "current_city": "from Dallas to Missoula",
    "transportation": "Flight Number: F3604227, from Dallas to Missoula, Departure Time: 11:28, Arrival Time: 13:48",
    "breakfast": "MONKS, Dallas",
    "attraction": "Reunion Tower, Dallas",
    "lunch": "-",
    "dinner": "-",
    "accommodation": "-"
  }
]

```

Şekil 2

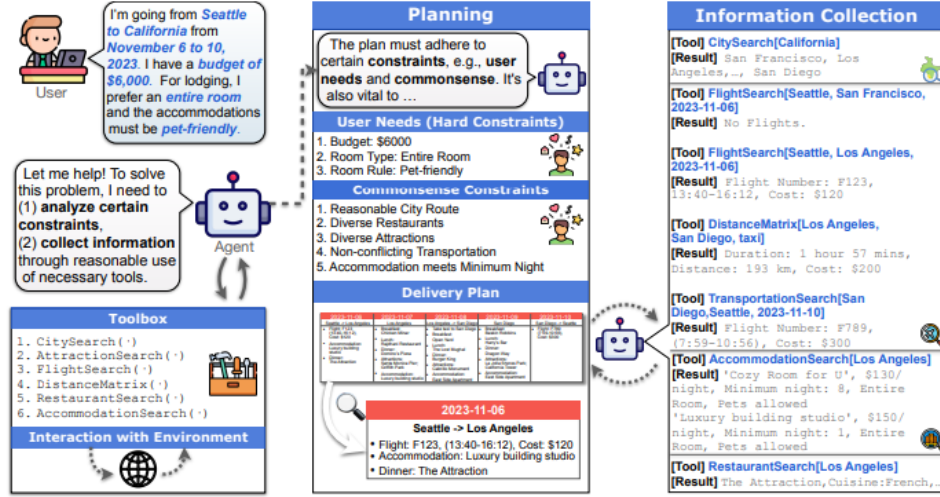
Makalede gerçekleştirilen ve gerçekleştirilmesi hedeflenen projenin yüzeysel akış şeması Şekil 3’te gösterilmektedir.



Şekil 3

Makaleye göre Şekil 1’de yer alan sorgular, Şekil 3’te gösterildiği gibi direkt olarak bir büyük dil modeline verildiğinde büyük dil modelinin Şekil 4’te yer alan mavi punto ile belirtilen seyahat kısıtlamalarını doğru algılayamadığı gösterilmiştir. Makalede büyük dil modelinin bu kısıtlamaları tespit edip buna uygun olarak plan üretmesi için büyük dil modelinin farklı prompt içerikleri ve farklı değerlendirme adımlarıyla özelleştirilmesi gerektiği belirtilmiştir.

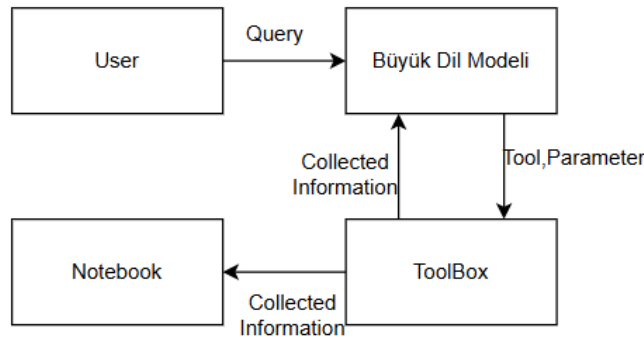
BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3



Şekil 4[2]

Büyük dil modelinin özelleştirilmesinin Agent kavramıyla mümkün olduğu tespit edildi. Böylelikle kodun uygulanması için öncelikle Agent bilgi birikiminin edinilmesi gerekmektedir.[3]

Makalede projenin real-world planlama yapması istenmektedir. Bundan dolayı büyük dil modelinin cevap üretirken gerçek verilerden hatta yalnızca gerçek verilerden yararlanması hedeflenmektedir. Bunu gerçekleştirmek için makalede ek bir büyük dil modeli kullanılmıştır. Kullanılan bu büyük dil modeli kullanıcı sorgusuna göre Şekil 4'te yer alan ToolBox içerisinde bulunan CitySearch, AttractionSearch, AccommodationSearch ve FlightSearch gibi tool seçenekleri arasından tool ve buna uygun parametre seçimi yapacaktır. Makalede[2] kullanılan tool'lar, herhangi bir büyük dil modeli kullanmadan verilen parametrelere göre var olan veri setinden veri çekme işlemi yapar. Veri toplama işlemi şekil 5'teki şemada özetlenmiştir.

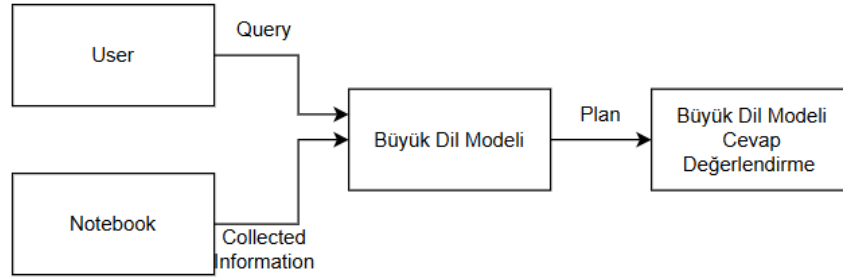


Şekil 5

Şekil 5’te görüldüğü gibi büyük dil modeli tool seçimi ve parametre seçimi yapar ve seçilen tool ve parametreye göre uygun fonksiyon çalıştırılır ve gerekli bilgiler veri setlerinden toplanır. Toplanan bu bilgiler dil modelinin çağrılan tool’u tekrar çağırması için büyük dil modeline geri döndürülür. Ayrıca projede kullanılan bir diğer büyük dil modeli plan üretimi yaparken toplanan bu verileri kullanacağı için veriler Notebook olarak adlandırılan bir dosya yapısına kaydedilir.

Projede[1] birden fazla büyük dil modeli için birden fazla özelleştirilmiş büyük dil modeli kullanılıyor. Bu teknik, multi-agent olarak adlandırılmaktadır. Multi-agent kavramı da kaynak kodun çalıştırılması aşamasına geçilmeden önce öğrenilmesi gereken kavramlar arasına eklenilmiştir.

Projede tool çağırısı için kullanılan büyük dil modeli gerekli tool çağrılarını tamamladığında, sorgu için gerekli bilgiler toplanmış olur. Toplanan bilgiler sorgu ile birlikte büyük dil modeline verilir ve sorguda yer alan kısıtlamalara uyarak bir plan oluşturması istenir. Projenin nihai aşaması olan plan üretimi Şekil 6’da gösterilmiştir.



Şekil 6

Kullanıcı sorgusunda yer alabilecek olan kısıtlamalar Şekil 7’de gösterildiği üzere temel olarak 3 ayrı grupta incelenebilir:

- **Environment Constraint:** Büyük dil modelinin plan üretimine karar vermeden önce kontrol etmesi gereken kısıtlama tipidir. Büyük dil modeli öncelikle seyahat etmek istenilen şehirdeki etkinlik durumunu, güzergaha uygun ulaşımın olup olmadığını kontrol etmelidir.
- **Hard Constraint:** Büyük dil modelinin plan üretimine başladığında dikkat etmesi gereken kurallardır. Sorguda yer alan bütçe, oda kuralları, oda tipi, restaurant için mutfak tipi, konaklama yeri için minimum kalınabilecek gece sayısı gibi unsurların büyük dil modeli tarafından anlaşılıp buna uygun olarak plan üretmesi gerekir.
- **Commonsense Constraint:** Büyük dil modelinin daha kullanışlı planlar üretmesini sağlamak için var olan ek kısıtlamalardır. Diğer kısıtlamalar gibi öncelik taşımaz. Önerilerin çeşitli olmasını sağlar.



BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

Constraint	Description
Environment Constraint	
Unavailable Transportation	There is no available flight or driving information between the two cities.
Unavailable Attractions	There is no available attraction information in the queried city.
Commonsense Constraint	
Within Sandbox	All information in the plan must be within the closed sandbox; otherwise, it will be considered a hallucination.
Complete Information	No key information should be left out of the plan, such as the lack of accommodation during travel.
Within Current City	All scheduled activities for the day must be located within that day's city(s).
Reasonable City Route	Changes in cities during the trip must be reasonable.
Diverse Restaurants	Restaurant choices should not be repeated throughout the trip.
Diverse Attractions	Attraction choices should not be repeated throughout the trip.
Non-conf. Transportation	Transportation choices within the trip must be reasonable. For example, having both "self-driving" and "flight" would be considered a conflict.
Minimum Nights Stay	The number of consecutive days spent in a specific accommodation during the trip must meet the corresponding required minimum number of nights' stay.
Hard Constraint	
Budget	The total budget of the trip.
Room Rule	Room rules include "No parties", "No smoking", "No children under 10", "No pets", and "No visitors".
Room Type	Room types include "Entire Room", "Private Room", "Shared Room", and "No Shared Room".
Cuisine	Cuisines include "Chinese", "American", "Italian", "Mexican", "Indian", "Mediterranean", and "French".
Transportation	Transportation options include "No flight" and "No self-driving".

Şekil 7 [2]

Makalede yer alan proje two-stage ve sole-planning olmak üzere iki farklı şekilde çalışabilir. Şekil 8'de yer alan makale sonuçları, açık kaynak büyük dil modellerinin seyahat planı oluşturma görevinde oldukça başarısız olduğunu göstermektedir.

	Validation (#180)					Test (#1,000)						
	Delivery Rate	Commonsense		Hard Constraint		Final Pass Rate	Delivery Rate	Commonsense		Hard Constraint		Final Pass Rate
		Pass Rate		Pass Rate				Pass Rate		Pass Rate		
		Micro	Macro	Micro	Macro			Micro	Macro	Micro	Macro	
Greedy Search	100	74.4	0	60.8	37.8	0	100	72.0	0	52.4	31.8	0
Two-stage												
Mistral-7B-32K (Jiang et al., 2023)	8.9	5.9	0	0	0	0	7.0	4.8	0	0	0	0
Mixtral-8x7B-MoE (Jiang et al., 2024)	49.4	30.0	0	1.2	0.6	0	51.2	32.2	0.2	0.7	0.4	0
Gemini Pro (G Team et al., 2023)	28.9	18.9	0	0.5	0.6	0	39.1	24.9	0	0.6	0.1	0
GPT-3.5-Turbo (OpenAI, 2022)	86.7	54.0	0	0	0	0	91.8	57.9	0	0.5	0.6	0
GPT-4-Turbo (OpenAI, 2023)	89.4	61.1	2.8	15.2	10.6	0.6	93.1	63.3	2.0	10.5	5.5	0.6
Sole-planning												
DirectGPT-3.5-Turbo	100	60.2	4.4	11.0	2.8	0	100	59.5	2.7	9.5	4.4	0.6
CoTGPT-3.5-Turbo	100	66.3	3.3	11.9	5.0	0	100	64.4	2.3	9.8	3.8	0.4
ReActGPT-3.5-Turbo	82.2	47.6	3.9	11.4	6.7	0.6	81.6	45.9	2.5	10.7	3.1	0.7
ReflexionGPT-3.5-Turbo	93.9	53.8	2.8	11.0	2.8	0	92.1	52.1	2.2	9.9	3.8	0.6
DirectMixtral-8x7B-MoE	100	68.1	5.0	3.3	1.1	0	99.3	67.0	3.7	3.9	1.6	0.7
DirectGemini Pro	93.9	65.0	8.3	9.3	4.4	0.6	93.7	64.7	7.9	10.6	4.7	2.1
DirectGPT-4-Turbo	100	80.4	17.2	47.1	22.2	4.4	100	80.6	15.2	44.3	23.1	4.4

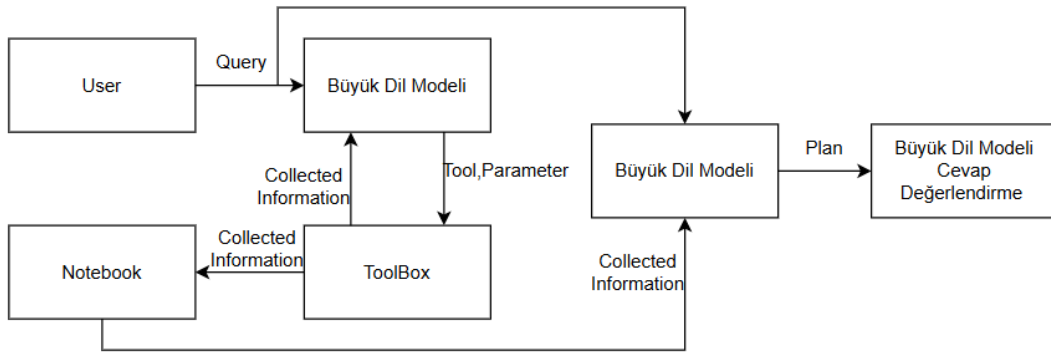
Şekil 8

Makalede iki farklı veri setiyle çalışmaktadır:

- Tool Veri Seti : Tool’ların büyük dil modelinin plan üretimini gerçek dünya verileriyle yapması için veri topladığı veri seti. İçerisinde uçuş verileri, restaurant, etkinlik ve konaklama gibi Kaggle gibi farklı ortamlard yer alan gerçek dünya verileri yer almaktadır. Bu verilerin işleniş biçimleri de makalede yer almaktadır.
- Validation-Test Veri Seti: Şekil 1’de gösterilen sorgulardan binlercesinin ve üretilmesi gereken planların yer aldığı veri setidir.

1.1.3. Elde Edilen Sonuçlar

Makale incelemesi tamamlandı ve projenin temel gereksinimler belirlendi. Proje şeması Şekil 9’da gösterildiği gibidir.



Şekil 9

- Kullanıcı sorgusu string formatında alınır
- Büyük dil modeli sorguda yer alan anahtar kelimeleri ve çağırılması gereken tool’ları belirler.
- Tool’lar sorguya göre gerekli bilgileri toplar.
- Sorgudaki kısıtlamalar ve toplanan bilgilere göre bir plan üretilir.

Kullanılacak teknolojiler belirlendi:

- Proje büyük dil modeli ile çalışır ve bunun için GPU desteği gerekmektedir. Şirket GPU’larına bağlanmak için gerekli hesaplar oluşturuldu ve bağlanma ve çalıştırma yöntemleri öğrenildi. SSH bağlantısı sağlandı.
- Şirket server’ındaki Proxy büyük dil modellerine HuggingFace token ile bağlanmayı[4] engellediği için server’da halihazırda yüklü olan büyük dil modelleri kullanılacak. Makalede yer alan Mistral modeli [5] server’da yüklüdür.

Edinilmesi gereken bilgi birikimi belirlendi:

- Agent Kavramı
- Multi-Agent Çalışma Prensibi
- Makalede Yer Alan Anahtar Kavramlar

1.2. Travel Planner Sorgularının Agent Kullanmadan Büyük Dil Modeline Verilmesi

1.2.1. Görev Tanımı ve Hedefler

Agent tekniği kullanılmadan Şekil 1’de yer alan tipte sorguların bir büyük dil modeline direkt olarak verilip bir plan üretmesine odaklanılmaktadır. Temel hedefler aşağıda listelenmiştir:

- GPU’ya bağlanmak
- Soru cevap için uygun modeli seçmek
- Büyük dil modelini yüklemek
- Kullanıcı sorgusunu büyük dil modelinin anlayabileceği formata dönüştürmek
- Büyük dil modelinin döndürdüğü çıktıyı metne dönüştürmek
- Çıktıyı değerlendirmek

1.2.2. Uygulanan Yöntemler ve Teknolojik Altyapı

Büyük dil modeli olarak server’da yüklü olan ve makalede de kullanılan Mistral modeli kullanıldı. Mistral üzerinde araştırma yapıldı. Araştırma sonuçları aşağıda listelenmiştir:

- Mistral açık kaynak bir büyük dil modelidir ve Hugging Face ile erişilebildiği gibi model local bilgisayarda da kullanılabilir.
- Verimlilik odaklı geliştirilmiş bir modeldir ve daha az hesaplama ile doğru sonuçlar üretmeyi hedefler.

- Server’da yüklü olan model Şekil 10’da görüldüğü üzere farklı dosyaları içermektedir.
 - config.json: Modelin doğru bir şekilde yüklenmesi için rehber görevi üstlenen bu dosya, model türü, katman sayısı ve daha birçok model özelliği hakkında bilgiler içerir.[7]
 - pytorch_model-00001-of-00002.bin: Modelin önceden eğitilmiş ağırlıklarını içermektedir. Model üretim yaparken bu ağırlıkları kullanır.[7]
 - pytorch_model_bin.index.json: Modelin parçalara ayrılmış olan ağırlık dosyalarının sırasını belirten bir index dosyasıdır.[7]
 - tokenizer.json: Model tokenization işlemi için gerekli olan tüm ayarları ve kelime dağarcığını içerir.[6]
 - tokenizer.model: Bir modelin tokenizer’ının eğitilmiş parametrelerini içeren dosyadır.[5]
 - tokenizer_config.json, tokenizer’ın çalışma şeklini özelleştiren ayarları içeren bir dosyadır.[5]

Name	Size (KB)
..	
config.json	1
generation_config.json	1
pytorch_model-00001-of-00002.bin	9 709 988
pytorch_model-00002-of-00002.bin	4 946 116
pytorch_model_bin.index.json	23
tokenizer.json	1 753
tokenizer.model	481
tokenizer_config.json	1

Şekil 10

Önceden eğitilmiş ve local bilgisayarda yüklü olan büyük dil modelinin kullanılması için yapılan araştırmalar yapıldı. Başta Transformer dokümantasyonu[9] AutoClasses bölümü[8] olmak üzere çeşitli dokümanlar okundu. Araştırmalar sonucunda ulaşılan çıktılar aşağıda listelenmiştir:

- Hugging Face Transformers kütüphanesinde yer alan AutoClass’lar kullanılacak.
- AutoTokenizer, stringlerin tokenize işlemi için tokenizer.json, tokenizer.model,tokenizer_config.json dosyalarını kullanarak tokenizer oluşturur. Yüklenen tokenizer Şekil 11’de gösterilmiştir.

```
LlamaTokenizerFast(name_or_path='/media/storage/llm_resources/models/Mistral-7B/', vocab_size=32000, model_max_length=100000000,
0: AddedToken("<unk>", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
1: AddedToken("<s>", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
2: AddedToken("</s>", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
}
```

Şekil 11

- AutoModelForCausalLM, config.json rehberliğinde model ağırlıklarını yükleyerek büyük dil modelini kullanıma hazır hale getirir. Yüklenen büyük dil modeli Şekil 12’de gösterilmektedir.

```
MistralForCausalLM(
  (model): MistralModel(
    (embed_tokens): Embedding(32000, 4096)
    (layers): ModuleList(
      (0-31): 32 x MistralDecoderLayer(
        (self_attn): MistralSdpaAttention(
          (q_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (v_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (o_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (rotary_emb): MistralRotaryEmbedding()
        )
        (mlp): MistralMLP(
          (gate_proj): Linear(in_features=4096, out_features=14336, bias=False)
          (up_proj): Linear(in_features=4096, out_features=14336, bias=False)
          (down_proj): Linear(in_features=14336, out_features=4096, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): MistralRMSNorm((4096,,), eps=1e-05)
        (post_attention_layernorm): MistralRMSNorm((4096,,), eps=1e-05)
      )
    )
    (norm): MistralRMSNorm((4096,,), eps=1e-05)
  )
  (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)
```

Şekil 12

Bir büyük dil modelinin sorguyu değerlendirip cevap üretmesi Şekil 13'te gösterilen 3 temel aşamadan oluşur[10]:



Şekil 13

- **Tokenizer – Encode** : Model sorguyu Şekil 14’teki gibi vektör temsillerine dönüştürür.

[illegible]

Şekil 14

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

- Model – Generate: Model vektör temsillerini alır ve üretim yapar. Model çıktısı Şekil 15'te gösterilmiştir.

```
tensor([[ 1, 5919, 1316, 528, 2623, 264, 6596, 477, 662, 28723,
17658, 5467, 298, 8107, 4401, 668, 4800, 28705, 28770, 2202,
477, 4117, 28705, 28740, 28784, 362, 298, 4117, 28705, 28740,
28783, 362, 28725, 28705, 28750, 28734, 28750, 28750, 28723, 415,
4530, 1023, 347, 10233, 354, 264, 2692, 1338, 395, 264,
8326, 302, 429, 28740, 28725, 28787, 28734, 28734, 28723, 13,
13, 3260, 6596, 349, 5035, 298, 11418, 304, 3555, 4735,
438, 272, 1348, 727, 28725, 304, 369, 5532, 17203, 438,
1665, 24682, 28725, 4282, 24682, 28725, 295, 16040, 28725, 304,
287, 16040, 28280, 28723, 2993, 28725, 272, 6596, 1023, 459,
347, 579, 341, 951, 10741, 369, 315, 3573, 6305, 272,
28705, 28770, 5240, 2125, 297, 264, 7689, 28723, 13, 13,
1014, 6596, 1023, 347, 390, 28411, 390, 2572, 1312, 1309,
5239, 264, 746, 15382, 2659, 28723, 13, 13, 3260, 349,
767, 315, 506, 14775, 575, 579, 2082, 28747, 13, 13,
718, 28723, 17658, 5467, 298, 9955, 5485, 486, 2423, 13209,
13, 13, 26803, 1265, 5485, 298, 451, 1391, 28708, 3610,
15777, 4213, 1253, 13, 13, 28762, 1391, 28708, 298, 8703,
14831, 304, 295, 2474, 1059, 272, 8703, 14831, 25440, 24494,
13, 13, 28769, 16040, 298, 415, 18864, 6396, 10668, 10387,
6064, 304, 868, 298, 8107, 4401, 354, 272, 7689, 304,
7854, 28723, 13, 13, 3381, 368, 541, 1316, 528, 395,
2994, 1378, 575, 741, 5563, 304, 1722, 298, 511, 2373,
272, 451, 1391, 28708, 3610, 15777, 298, 8703, 14831, 304,
272, 18864, 6396, 10668, 10387, 6064, 28725, 369, 682, 347,
6821, 28723, 13, 13, 15896, 368, 297, 8670, 354, 787
```

Şekil 15

- Tokenizer – Decode : Model çıktısı tokenizer ile string karşılıklarına dönüştürülür. Nihai çıktı Şekil 16'da gösterildi.

```
Please help me plan a trip from St. Petersburg to Rockford spanning 3 days from March 16th to March 18th, 2022. The travel sho:
For transportation, you need to plan for transportation between St. Petersburg and Rockford using public or shared transportat:
For accommodation, you need to plan for a hotel or motel for 2 nights with an average price of no more than $50 per night for :
Please plan for sightseeing activities, food, and any other expenses that you think are necessary for a successful trip within
Please plan for transportation and accommodation for up to 3 days, with a budget of $1,700. For example, you can plan for a bur
You need to create a detailed itinerary with all travel details and timings, including transportation modes, departure and arr:
To plan the trip, use the following steps:
1. Research transportation options between St. Petersburg and Rockford: Find out the available transportation options such as :
2. Research accommodation options: Find hotels or motels in Rockford that fit your budget and meet your needs.
3. Plan sightseeing activities and food: Research activities and restaurants in Rockford that you would like to visit and estir
4. Create an itinerary: Plan the details of your trip, including the transportation modes, timings, and accommodation for each
5. Create a detailed budget: Create a detailed budget breakdown
```

Şekil 16

1.2.3. Elde Edilen Sonuçlar

Yapılan çalışmayla Şekil 16'da gösterildiği gibi bir büyük dil modelinin sorguda istenen şekilde kapsamlı bir plan üretilmediği gözlemlenmiştir. Yapılan kapsamlı araştırmaların sonucunda bu tip problemlerin özelleştirilmiş büyük dil modeli olan Agent mimarisıyla[11] çözülebileceği belirlendi.

1.3. TravelPlanner Makalesinin Uygulanması

1.3.1 Görev Tanımı ve Hedefler

Bu görevde Görev 1.1’de ele alınan makalenin uygulanması ve yine Görev 1.1’de belirlenen kavramların öğrenilmesi amaçlanmıştır. Temel hedefler aşağıda listelenmiştir:

- Agent Kavramının Öğrenilmesi
- Multi- Agent Kavramının Öğrenilmesi
- Makaledeki Agent Kavramlarının Değerlendirilmesi
- Makalenin Uygulanması ve Sonuçların Karşılaştırılması

1.3.2 Uygulanan Yöntemler ve Teknolojik Altyapı

DeepLearning.AI, Coursera gibi platformlarda yer alan kurslar tamamlandı. Çeşitli kaynaklardan araştırmalar yapıldı. Araştırma sonuçları şu şekildedir:

- Agent bir büyük dil modelinin spesifik bir göreve odaklanmasıdır. [11]
- Multi-agent sistemlerde, ele alınan projenin gerektirdiği her bir görev için farklı bir büyük dil modeli kullanılır ve bu modellerin çıktıları birbirleriyle etkileşim halindedir. [14]
- Birden fazla dil modelinin birbiriyle etkileşimi döngüler, fonksiyonlar, dosyalar gibi basit yapılarla gerçekleştirebileceği gibi; CrewAI[13], LangGraph[12] gibi platformlar ile de gerçekleştirilebilir.

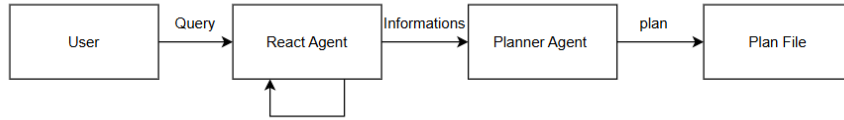
Agent ve Multi-agent kavramları hakkında yapılan kapsamlı araştırma sonucu makalede yer alan bazı önemli kavramlar hakkında da araştırmalar yapıldı:

- Çeşitli prompt teknikleri hakkında incelemeler yapıldı [2][15]:
 - ZEROSHOT_REACT_INSTRUCTION: Büyük Dil Modelinin cevap üretimini tek seferde yapmak yerine Through, Action, Observation adımlarını takip ederek yapmasını sağlar.
 - COT_PLANNER_INSTRUCTION: Daha detaylı örneklerin verildiği ve prompt’a “think step by step” gibi eklemeler yapılarak büyük dil modelini direkt cevap üretmek yerine düşünerek ve düşündüklerini cevaba ekleyerek cevap üretmeye yönlendirir.
 - REACT_PLANNER_INSTRUCTION: ZEROSHOT_REACT gibi gibi plan üretimine odaklanır. Yalnızca React mantığıyla çalışır ve büyük dil modelini Through, Action ve Observation süreçlerine yönlendirir.
 - REFLECT_INSTRUCTION: Büyük dil modelinin, cevap ürettikten sonra cevabını değerlendirmesini sağlayan prompt şeklindedir.
 - PLANNER_INSTRUCTION: Sorgu-plan örnekleri ve büyük dil modelini plan üretmeye yönlendiren talimatlar içerir.

- Farklı prompt'lar aracılığıyla özelleştirilmiş farklı agent'lar incelendi:
 - ReactAgent: Planner ZEROSHOT_REACT_INSTRUCTION prompt'una sorgu eklenir. Agent kendi içerisinde bir döngü içerir; through aşamasında llm sorguyu alır, action aşamasında prompt'u değerlendirerek bir tool çağrısı üretir, observation aşamasında üretilen tool çağrısı koşul bloklarıyla tespit edilir ve gerekli fonksiyon çalışır, toplanan bilgileri kaydeder. Yeterli bilgi toplandıysa planner tool'u çağırılır. Planner Tool ise Planner Agent'a bağlıdır.
 - Planner Agent: PLANNER_INSTRUCTION ile çalışır. Bir tool olarak çalışmaktadır. Tool'lar aracılığıyla toplanmış bilgiler ve prompt'u alır ve nihai planı üretir.
 - React Planner: Planlama tool'u React tekniğiyle geliştirilerek, plan üretimi yaparken tool çağrısı da yapılabilir.

Makalenin ele aldığı kaynak kod baz[2] alınarak proje aşağıdaki adımlar takip edilerek uygulandı.

Makalede içeriğinden yola çıkılarak React Agent ve Planner Agent tanımlanarak Şekil 17'de gösterilen şekilde bir multi-agent sistemi oluşturulmasına karar verildi.



Şekil 17

- Büyük dil modelinin belirli bir göreve odaklanması için farklı yapıda prompt'lar tanımlandı:
 - ZEROSHOT_REACT_INSTRUCTION: Hazırlanan prompt içeriği Şekil 18'de gösterilmiştir. Bu prompt büyük dil modelinin doğru tool'u doğru parametre ile seçmesini sağlar.

```

ZEROSHOT_REACT_INSTRUCTION = """Collect information for a query plan using interleaving 'Thought',
(1) FlightSearch[Departure City, Destination City, Date]:
Description: A flight information retrieval tool.
Parameters:
Departure City: The city you'll be flying out from.
Destination City: The city you aim to reach.
Date: The date of your travel in YYYY-MM-DD format.
Example: FlightSearch[New York, London, 2022-10-01] would fetch flights from New York to London on

(2) GoogleDistanceMatrix[Origin, Destination, Mode]:
Description: Estimate the distance, time and cost between two cities.
Parameters:
Origin: The departure city of your journey.
Destination: The destination city of your journey.
Mode: The method of transportation. Choices include 'self-driving' and 'taxi'.
Example: GoogleDistanceMatrix[Paris, Lyon, self-driving] would provide driving distance, time and

(3) AccommodationSearch[City]:
Description: Discover accommodations in your desired city.
Parameter: City - The name of the city where you're seeking accommodation.
Example: AccommodationSearch[Rome] would present a list of hotel rooms in Rome.

(4) RestaurantSearch[City]:
Description: Explore dining options in a city of your choice
  
```

Şekil 18

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

- PLANNER_INSTRUCTION: Şekil 19’da gösterilen prompt, projedeki nihai planı oluşturmayı sağlar. Prompt içeriğinde sorgu-plan örnekleri yer almaktadır.

```
PLANNER_INSTRUCTION = ""You are a proficient planner. Based on the provided information and query, please

**** Example ****
Query: Could you create a travel plan for 7 people from Ithaca to Charlotte spanning 3 days, from March 8th
Travel Plan:
Day 1:
Current City: from Ithaca to Charlotte
Transportation: Flight Number: F3633413, from Ithaca to Charlotte, Departure Time: 05:38, Arrival Time: 07:
Breakfast: Nagaland's Kitchen, Charlotte
Attraction: The Charlotte Museum of History, Charlotte
Lunch: Cafe Maple Street, Charlotte
Dinner: Bombay Vada Pav, Charlotte
Accommodation: Affordable Spacious Refurbished Room in Bushwick!, Charlotte

Day 2:
Current City: Charlotte
Transportation: -
Breakfast: Olive Tree Cafe, Charlotte
Attraction: The Mint Museum, Charlotte;Romare Bearden Park, Charlotte.
Lunch: Birbal Ji Dhaba, Charlotte
Dinner: Pind Balluchi, Charlotte
Accommodation: Affordable Spacious Refurbished Room in Bushwick!, Charlotte
```

Şekil 19

- Özelleştirilmiş büyük dil modelleri olan farklı agent’lar tanımlandı:
 - ReactAgent: Planner ZEROSHOT_REACT_INSTRUCTION prompt’una sorgu eklenir. Agent kendi içerisinde bir döngü içerir; through aşamasında llm sorguyu alır, action aşamasında prompt’u değerlendirerek bir tool çağrısı üretir, observation aşamasında üretilen tool çağrısı koşul bloklarıyla tespit edilir ve gerekli fonksiyon çalışır, toplanan bilgileri kaydeder. Yeterli bilgi toplandıysa planner tool’u çağrılır. Planner Tool ise Planner Agent’a bağlıdır.
 - Planner Agent: PLANNER_INSTRUCTION ile çalışır. Bir tool olarak çalışmaktadır. Tool’lar aracılığıyla toplanmış bilgiler ve prompt’u alır ve nihai planı üretir.

Her iki agent için de Mistral, LLama, Gemma, Gpt2 gibi farklı büyük dil modelleriyle denemeler yapıldı.

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

- Tool çağrılarının bilgi toplaması için yeni veri setleri oluşturuldu. Oluşturulan veri setleri, kullanılan kaynaklar ve oluşturulma biçimleri Şekil 20’de gösterilmiştir.

Tool	Source	Extract and Produce
Flight Search	Kaggle	from 2022-03-01 to 2022-04-01
Cities	Flight Search	Unique Origin and Destination Cities
Cities and Coordinates	Cities	Coordinate API
Distance Matrix	Cities and Coordinates	Haversine
Restaurant	Kaggle	Some value is random
Attraction	Cities and Coordinates	Faker
Accommodations	Kaggle	All rows and random

Şekil 20

- Şekil 21’de tool’lar ve parametreleri yer almaktadır. Bu tool’lar oluşturulan veri setlerinden veri çekme işlemi yapar. Topladıkları veriyi Notebook ismi verilen bir değişkene aktarırlar. Bu değişken düzenli aralıklarla verileri dosyaya kaydeder.

1	Planner (Notebook)
2	Accommodation Search (City)
3	Flight Search (Origin, Destination, Departure Data)
4	Attraction Search (City)
5	Restaurant Search (City)

Şekil 21

1.3.3 Elde Edilen Çıktılar

Proje 4 farklı büyük dil modeli ile denendi ve farklı çıktılar elde edildi:

- Mistral: Plan üretimi yaptı, yalnız çok yavaş çalıştığı için ürettiği planları ancak ekrana yazdırabildi. Tüm sorguları tamamlayamadı.
- Llama: Mistral ile benzer çıktılar verdi, ama ondan daha yavaş çalıştı.
- Gemma: Server'daki model çok büyük olduğu için yükleme esnasında hata verdi.
- GPT: Encode-decode kısımlarında tokenizer config hatası verdi, kod çalışmadı.

Bu aşamanın sonunda şirket AI takımına bir sunum yapıldı. Yapılan sunum sonucu oluşturulan kod yapısının, kod içerisindeki bilgi akışının, kullanılan büyük dil modelleri arasındaki iletişimin efektif bir şekilde yönetilmediği tespit edildi.

Yapılan tartışmalar ve araştırmalar sonucu LangGraph teknolojisiyle projeye devam edilmesine karar verildi. LangGraph birden fazla büyük dil modeli kullanılan projelerin yönetimini kolaylaştıran bir platformdur. LangGraph ile geliştirilmiş örnek projeler incelendi ve bu alanda başarılı olduğu onaylandı.

1.4. LangGraph ve Groq ile TravelPlanner Geliştirilmesi

1.4.1 Görev Tanımı ve Hedefler

Travel Planner uygulaması LangGraph kullanılarak yeniden geliştirilmiştir. Temel hedefler:

- LangGraph bilgi birikiminin edinilmesi
- Langchain Academy tarafından yayımlanan LangGraph kursunun tamamlanması ve tüm kodların uygulanması
- Travel Planner Projesinin LangGraph ile gerçekleştirilmesi

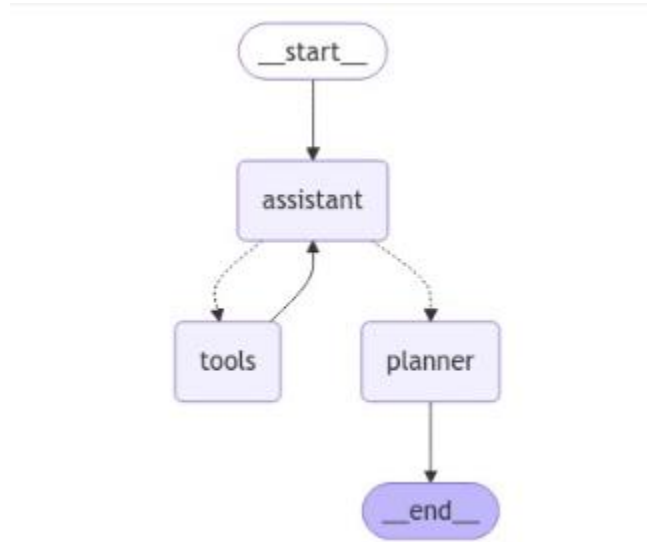
1.4.2 Teknolojik Altyapı ve Uygulanan Yöntemler

LangGraph kursları ve dokümantasyonu, büyük dil modeli olarak OpenAI kullanmaktadır. OpenAI, ChatOpenAI class'ı ile kolayca yüklenebilir. ChatOpenAI class'ı invoke, bind_tools, ToolNode gibi langGraph projelerinde kullanılan önceden tanımlı bir çok kod ve class içerir [16]. Transformer modelleri AutoClass ile yüklendiği için önceden tanımlı bu metodları içerebilmesi için pipeline kurulması gerekir. Pipeline kurabilmek için çeşitli denemeler yapıldı, sonucunda pipeline kurulmadığı için Groq ile bir deneme yapılmasına karar verildi. [17]

Groq, çeşitli büyük dil modellerine API aracılığıyla erişilmesini sağlayan bir platformdur. Kullanıcılara ücretsiz GPU desteği de sunmaktadır. Açık kaynak büyük dil modellerini C++ koduna çevirerek kendi server'larında çalıştırır ve kullanıcıya cevabı gönderir. [18]

Groq bağlantısı ChatGroq class'ı aracılığıyla gerçekleşir ve bind_tools, invoke ve ToolNode gibi önceden tanımlı yapılar kullanılabilir.

Groq ve LangGraph kullanılarak gerçekleştirilen bu model Şekil 22'de gösterildiği gibi React yaklaşımıyla geliştirilmiştir.



Şekil 22

Şekil 22'de gösterilen graph farklı görevleri gerçekleştiren 3 node ile oluşturulmuştur:

- Assistant: Hangi tool'un kullanılacağına karar verme işlemi yapar.
- Tools: Belirlenen tool'un çalışmasını sağlar.
- Planner: Toplanan bilgiler ile bir plan üretir.

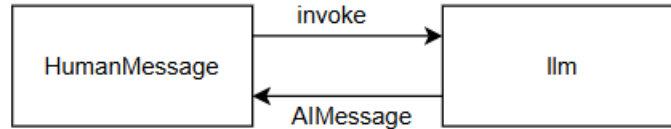
Graph yapısı oluşturulan edge'ler ile yönetilir.

- Assistant -> Tool : Assistant node bir tool çağrısı oluşturur ve Tool Node oluşturulan tool çağrısını ayrıştırır ve çalıştırır.
- Tool -> Assistant: Toplanan bilgiler assistant'a teslim edilir ve assistant'ın yeni bir tool çağrısı oluşturması sağlanır.
- Assistant -> Planner: Assistant node tarafından üretilen AIMessage eğer tool çağrısı içermemesi gerekli tüm bilgiler toplanmış ve graph'ın plan üretimi yapmaya hazır olduğu anlamına gelir.
- Start -> Assistant: Graph ilk çalıştırıldığında sorguyu assistant node'una aktarır.
- Planner -> end : Planner plan ürettiğinde graph sonlandırılır.
- **Groq Bağlantısı**
- ChatGroq class'ı ile llama3-8b modeli yüklendi. Groq ile bir dil modeline bağlanma süreci Şekil 24'te gösterilmiştir.



Şekil 23

ChatGroq ile oluşturulan büyük dil modeli invoke isimli önceden tanımlı bir fonksiyona sahiptir ve tokenize ,encode, decode, generate gibi doğal dil işleme süreçleri, Şekil 24'te gösterildiği gibi sadece invoke methodunu çağırarak tamamlanabilir. Invoke methodu çıktı olarak AIMessage tipinde mesajlar döndürmektedir

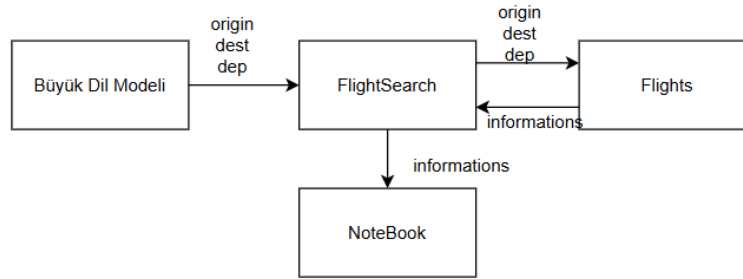


Şekil 24

Tool Tanımlamaları

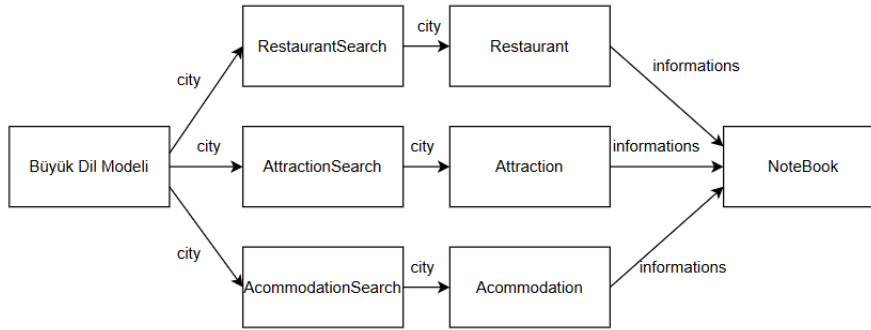
Travel Planner projesinin Groq ile geliştirilen bu versiyonu için 4 farklı tool tanımlandı:

- Flight Search: Şekil 26’da gösterilmiş olan bu tool, gerçek verilerden belirlenen parametrelerle veri toplama işlemi yapar.



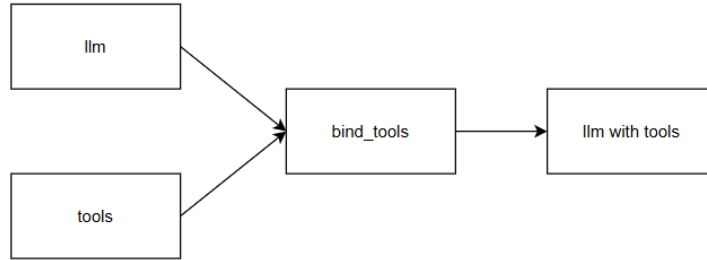
Şekil 25

- Restaurant Search: Şekil 27’de gösterilen bu fonksiyon, city parametresi aracılığıyla veri setinden veri seçme işlemi yapar.
- Attraction Search: Restaurant araması yapan fonksiyona oldukça benzer şekilde çalışır ve belirlenen şehirdeki etkinlikleri toplar.
- Accommodation Search: Diğer fonksiyonlara oldukça benzer şekilde çalışır ve belirlenen şehirdeki konaklama olanaklarını toplar. Planner agent’ının ‘Hard Constraint’ özelliklere uyması için konaklama olanaklarının house_rules, room_type gibi özellikleri de toplanır.



Şekil 26

Projede multi agent yapısında karşılıklı 2 büyük dil modeli çalışmaktadır: ilki tool seçimi yaparken, ikincisinin görevi plan üretmektir. Tool çağrısı yapması gereken büyük dil modeline seçebileceği tool'lar Şekil 28'de gösterildiği gibi bind_tools methoduyla gösterilir. bind_tools, tool'larla çalışan bir büyük dil modeli döndürür.



Şekil 27

Bir büyük dil modelinin doğru tool çağrısı yapabilmesi için sadece tool listesini ona göstermek yeterli olmaz: uygun prompt ile büyük dil modeli spesifikleştirilir. Şekil 29'da gösterilen prompt büyük dil modeline iki kritik yönlendirme yapar:

- Prompt'un başlangıcında, büyük dil modeline görevinden, yapması gerekenlerden bahseder.
- Prompt'un sonunda, seçebileceği tool'lar ve parametreleri hakkında bilgilendirme yapılır.

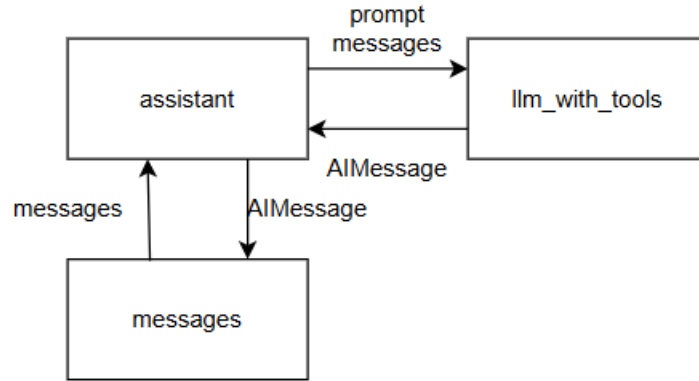
```
ZEROSHOT_REACT_INSTRUCTION = """Collect information for a query plan. Ensure you gather valid information related to  
Note that the nested use of tools is prohibited.You can use 4 different types:  
(1) flight_search(origin:str,destination:str,departure_date:str):  
Description: A flight information retrieval tool.  
Parameters:  
Departure City: The city you'll be flying out from.  
Destination City: The city you aim to reach.  
Date: The date of your travel in YYYY-MM-DD format.  
Example: FlightSearch[New York, London, 2022-10-01] would fetch flights from New York to London on October 1, 2022.  
  
(2) accommodation_search(City):  
Description: Discover accommodations in your desired city.  
Parameter: City - The name of the city where you're seeking accommodation.  
Example: AccommodationSearch[Rome] would present a list of hotel rooms in Rome.  
  
(3) restaurant_search(city:str):  
Description: Explore dining options in a city of your choice.  
Parameter: City [ ] The name of the city where you're seeking restaurants.  
Example: RestaurantSearch[Tokyo] would show a curated list of restaurants in Tokyo.  
  
(5) attraction_search(city):  
Description: Find attractions in a city of your choice.  
Parameter: City [ ] The name of the city where you're seeking attractions.  
Example: AttractionSearch[London] would return attractions in London.
```

Şekil 28

Node Tanımlamaları

LangGraph ile bir graph oluşturulurken işlemler node'larda gerçekleşirken akış edge'ler ile yönetilir. Travel Planner'ın içerdiği 3 farklı görev için 3 farklı node tanımlanır: assistant, tools, planner.

İlk node olan assistant, Şekil 30'da gösterildiği gibi llm_with_tools olarak isimlendirilen ve tool çağrısı oluşturmak üzere özelleştirilmiş olan büyük dil modeline, tool seçimi için oluşturulan prompt ve kullanıcı sorgusu gönderir.



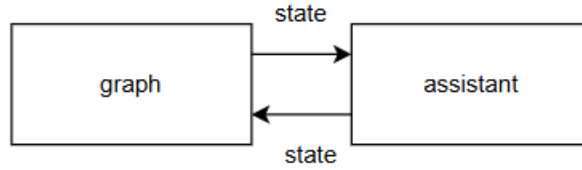
Şekil 29

Kullanıcı sorgusu state class'ının bir özelliği olarak gönderilir. LangGraph'ta graph içerisindeki node'lar arasındaki bilgi ve durum akışı genellikle state olarak isimlendirilen class'lar aracılığıyla gerçekleştirilir. Proje için gerekli bilgi ve durum akışı için yalnızca büyük dil modellerinin ürettiği cevaplar yeterli olduğu için, class içerisinde yalnızca bu mesajları kaydeden bir listenin olması yeterlidir. LangGraph, mesajları kaydeden ve yeni mesajları ekleyen, içerisinde messages isminde bir liste bulunduran önceden tanımlanmış bir MessagesState class'ı içerir. MessagesState Şekil 31'de gösterildiği üzere var olan mesaj listesine yeni mesaj ekler ve add_messages isminde bir reducer barındırmaktadır.

```
class MessagesState(TypedDict):  
    messages: Annotated[list[AnyMessage], add_messages]
```

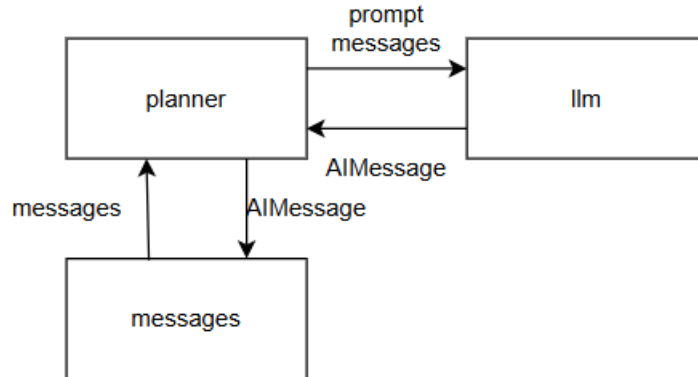
Şekil 30 [19]

Graph içerisindeki bir node, çalıştıktan sonra akışı bir başka node'a teslim edeceği için veri döndürmek zorundadır. Şekil 32'de gösterildiği gibi değiştirilen state class'ı döndürülür.



Şekil 31

Planner Node, önceki açıklamalarda bahsedildiği gibi toplanan bilgiler ile plan oluşturması gereken node. Bu node llm_with_tools büyük dil modelinden farklı olarak, Groq ile yüklenen büyük dil modelini kullanır. Şekil 32'de gösterildiği üzere yine prompt ve state büyük dil modeline verilir ve üretilen plan yine state'e mesaj olarak kaydedilir.



Şekil 32

Plan üreten büyük dil modelinin yine bir prompt aracılığıyla özelleştirilmesi gerekir. Şekil 33'te gösterilen prompt, öncelikle büyük dil modeline görevini tanımlar, ardından sorgu ve plan örneği büyük dil modeline gösterilir. Örnek gösterilmesi, büyük dil modelinin çıktısını yapılandırmayı sağlar.

```
PLANNER_INSTRUCTION = ""You are a proficient planner. Based on the provided information and query, please give me a c

**** Example ****
Query: Could you create a travel plan for 7 people from Ithaca to Charlotte spanning 3 days, from March 8th to March 9th?
Travel Plan:
Day 1:
Current City: from Ithaca to Charlotte
Transportation: Flight Number: F3633413, from Ithaca to Charlotte, Departure Time: 05:38, Arrival Time: 07:46
Breakfast: Nagaland's Kitchen, Charlotte
Attraction: The Charlotte Museum of History, Charlotte
Lunch: Cafe Maple Street, Charlotte
Dinner: Bombay Vada Pav, Charlotte
Accommodation: Affordable Spacious Refurbished Room in Bushwick!, Charlotte

Day 2:
Current City: Charlotte
Transportation: -
Breakfast: Olive Tree Cafe, Charlotte
Attraction: The Mint Museum, Charlotte;Romare Bearden Park, Charlotte.
Lunch: Birbal Ji Dhaba, Charlotte
Dinner: Pind Balluchi, Charlotte
Accommodation: Affordable Spacious Refurbished Room in Bushwick!, Charlotte

Day 3:
Current City: from Charlotte to Ithaca
Transportation: Flight Number: F3786167, from Charlotte to Ithaca, Departure Time: 21:42, Arrival Time: 23:26
Breakfast: Subway, Charlotte
Attraction: Books Monument, Charlotte
```

Şekil 33

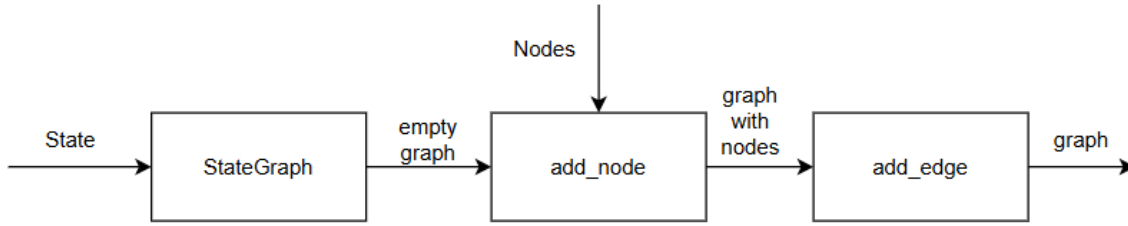
Graph Akışının Yönetilmesi

Oluşturulan graph, edge'ler ile yönetilir. İki node'un birbirine bağlanması oldukça basittir. Yalnız bir node, iki farklı durum altında, birden fazla node ile devam edebilme ihtimaline sahip ise, bir koşul yönetimi gereklidir. Bunun için LangGraph içerisinde React mimarisıyla tanımlanmış graph'ları yönetmeyi sağlayan bir tools_condition methodu bulunur. Bu method, üretilen son AIMessage içerisinde tool çağrısı bulunması halinde tool node'una, bulunmaması halinde ise __end__ node'una yönlendirme yapar. Bu yaklaşımda, tool çağrısı olmadığı durumda graph'ı sonlandırmak yerine planlama sürecine geçilmesi planlandığı için tools_condition [20] methodu güncellendi.

Graph Oluşturulması

LangGraph ile bir graph oluşturmak için Şekil 35'te gösterilen 3 aşama gereklidir.

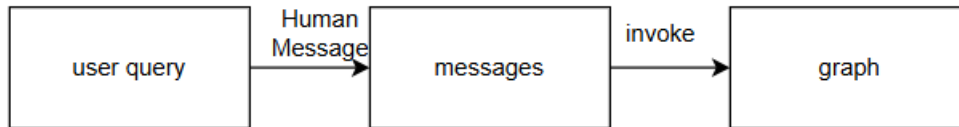
- Graph iskeleti oluşturma: LangGraph'ta tanımlı olan StateGraph class'ı ile bir graph nesnesi oluşturulur. Bu nesne oluşturulurken, MessagesState gibi akışı kaydedecek bir class almak zorundadır.
- Node Ekleme: Oluşturulan her bir node, add_node methoduyla graph'a eklenir. Tools node'u önceden tanımlı ToolNode ile kolayca oluşturulur ve graph'a eklenir.
- Edge ile Node'ların birbirine bağlanması: add_edge ile iki node birbirine bağlanır. Bağlanan node'lar, birinin çalışması bitince diğerine geçecek şekilde çalışır. add_conditional_edges ise, bir node'u iki farklı node'a bir koşul ile bağlar.
- Graph'ın derlenmesi: Graph derlenir ve sorgu kabul edebilir hale getirilir.



Şekil 34

Graph'ın Çalıştırılması

Derlenen graph'ın çalıştırılması Şekil 36'da gösterildiği gibi invoke methoduyla kolayca gerçekleştirilebilir. Graph, çalıştırılırken girdi olarak, GraphState iskeletine verilen state class'ı verilir. Bunun için sorgu direkt gönderilmez, HumanMessage tipine dönüştürülür ve messages listesine kaydedilerek gönderilir.



Şekil 35

1.4.3 Elde Edilen Çıktılar

Graph çalıştırıldığında elde edilen çıktılar Şekil 37’de gösterilmiştir. Graph çalışmasını tamamladığında mesaj listesini döndürmektedir. Mesaj listesinin ilk üyesi, HumanMessage tipindeki kullanıcı sorgusudur. HumanMessage’ı assistant node tarafından üretilen AIMessage takip eder ve flight_search çağrısı üretir. Assistant node’unu Tool Node takip eder ve ToolMessage ile toplanan bilgileri döndürür. AIMessage ve ToolMessage arasında 4 kez tekrarlayan bir döngü sonucu tüm tool’lar kullanıldı ve gerekli bilgileri topladı. 5. AIMessage ise, assistant tarafından üretilen ‘I have gathered all the necessary information.’ mesajıdır ve bunun bir tool call olmadığı belirlenince, planner node’una geçilir ve son mesaj olan plan üretilir.

```
messages

Python

{'messages': [HumanMessage(content='Please plan a trip for me starting from Sarasota to Chicago for 3 days, from March 22nd to March 24th, 2022. The budget is $1,900.', additional_kwargs={'tool_calls': [{'id': 'call_yxqh', 'function': {'arguments': '{"origin": "Sarasota", "destination": "Chicago", "departure": "2022-03-22", "arrival": "2022-03-24", "budget": 1900, "days": 3}}'}]}), AIMessage(content='', additional_kwargs={'tool_calls': [{'id': 'call_d3an', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_d8e2', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_a163', 'function': {'arguments': '{"city": "Chicago"}'}]}]), ToolMessage(content='Flight Number Price DepTime ArrTime ActualElapsedTime FlightDate OriginCityName DestCityName Distance\n F3600033 497 15:30 16:00 2022-03-22 2022-03-24 1900\n', additional_kwargs={'tool_calls': [{'id': 'call_d3an', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_d8e2', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_a163', 'function': {'arguments': '{"city": "Chicago"}'}]}]), AIMessage(content='', additional_kwargs={'tool_calls': [{'id': 'call_d3an', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_d8e2', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_a163', 'function': {'arguments': '{"city": "Chicago"}'}]}]), ToolMessage(content='Name ... City\n736 The Black Pearl ... Chicago\n807 Navy Pier ... Chicago\n1', additional_kwargs={'tool_calls': [{'id': 'call_d3an', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_d8e2', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_a163', 'function': {'arguments': '{"city": "Chicago"}'}]}]), AIMessage(content='', additional_kwargs={'tool_calls': [{'id': 'call_d3an', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_d8e2', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_a163', 'function': {'arguments': '{"city": "Chicago"}'}]}]), ToolMessage(content='Unnamed: 0 ... city\n554 554 ... Chicago\n580 580 ... Chicago\n913 913 ... Chicago\n941', additional_kwargs={'tool_calls': [{'id': 'call_d3an', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_d8e2', 'function': {'arguments': '{"city": "Chicago"}'}}, {'id': 'call_a163', 'function': {'arguments': '{"city": "Chicago"}'}]}]), AIMessage(content='I have gathered all the necessary information.', additional_kwargs={}, response_metadata={'token_usage': {'completion_tokens': 9, 'prompt_tokens': 10, 'total_tokens': 19}}, tool_calls=[{}]), AIMessage(content='Here is the detailed plan for your trip from Sarasota to Chicago for 3 days, from March 22nd to March 24th, 2022, with a budget of $1,900:')]

Şekil 36
```

Şekil 38’de hedeflenen çıktı olan seyahat planının kullanıcı sorgusuna uyumlu bir şekilde oluşturulduğu gözlemlenmiştir.

```
import textwrap
print(textwrap.dedent(messages["messages"][-1].content))

Here is the detailed plan for your trip from Sarasota to Chicago for 3 days, from March 22nd to March 24th, 2022, with a budget of $1,900:

**Day 1:**

* Current City: from Sarasota to Chicago
* Transportation: Flight Number: F3984576, from Sarasota to Chicago, Departure Time: 05:14, Arrival Time: 06:50
* Breakfast: Subway, Chicago
* Attraction: Navy Pier, Chicago
* Lunch: FIO Cookhouse and Bar, Chicago
* Dinner: Shahi Muradabadi, Chicago
* Accommodation: Affordable options such as 554, Chicago

**Day 2:**

* Current City: Chicago
* Transportation: -
* Breakfast: The Black Pearl, Chicago
* Attraction: Willis Tower, Chicago; Millennium Park, Chicago
* Lunch: Pantry d'or, Chicago
* Dinner: Bro's Kitchenette, Chicago
* Accommodation: Affordable options such as 580, Chicago

**Day 3:**

* Current City: from Chicago to Sarasota
...
```

Şekil 37

1.5.LangGraph ve HuggingFace ile React Agent Temelli Travel Planner Geliştirilmesi

1.5.1 Görev Tanımı ve Hedefler

Groq aracılığıyla bağlanılan llama büyük dil modeli ile oluşturulan büyük dil modeli oldukça doğru sonuçlar üretti. Bu görevde benzer mantıkla çalışan bir graph oluşturulması hedeflenmektedir. Yalnız bu aşamada bir büyük dil modeline Groq ile bağlanmak yerine, Huawei server'larına yüklü olan büyük dil modellerinin yüklenilmesi gerekmektedir. Görev 1.2'de uygulandığı gibi modelin Transformer ile yüklenmesi ve çalıştırılması hedeflenir. Başlıca hedefler şu şekildedir:

- Modelin Hugging Face Transformer ile yüklenmesi
- Groq modeli ChatGroq class'ı ile yüklendi, böylelikle bind_tools, ToolNode, invoke gibi doğal dil işleme süreçlerini oldukça kolaylaştıran fonksiyonlar kullanıldı. LangGraph eğitimleri de OpenAI modelleriyle uygulandığı için, ChatOpenAI class'ı içerisindeki önceden oluşturulmuş fonksiyonlarla işlemleri kolayca tamamlar. Transformer ile yüklenen modeller ile bir LangGraph uygulaması oluşturmak için büyük dil modeline tool bağlama, tool çağrısını yapılandırma ve işleme süreçleri ele alınmalıdır.

1.5.2 Uygulanan Yöntemler ve Teknolojik Altyapı

Oluşturulması planlanan graph Şekil 39'da gösterilmektedir. React yaklaşımı yine korunmuş olup, graph'taki fark süreçlerin manuel olarak ele alınmasından dolayıdır.



Şekil 38

Transformer Modelleri

Görev boyunca server’da yer alan farklı modellerle denemeler yapıldı: Mistral, Llama ve Gemma gibi modeller kullanıldı. Alınan sonuçlar “elde edilen sonuçlar” bölümünde raporlanmıştır.

LangGraph State

Görev 1.4’te Groq ile geliştirilen uygulamada MessagesState’den bahsedilmişti. State LangGraph’ın iskeleti oluşturulurken eklenmesi gereken, graph içerisinde bilgi akışını sağlayan class’tır. Bu uygulamada yine MessagesState ile çalışmalar yapıldı.[20] Yalnız proje istenildiği gibi çalışmayınca Şekil 40’da gösterildiği gibi yeni bir class tanımlandı. Bu class MessagesState class’ından miras alınarak oluşturuldu. State’e yeni değişkenlerin eklenmesinin sebebi, graph’ta ilk tool çağrısının doğru olması ama sürekli aynı tool çağrısının oluşturulmasıdır. Sürekli flight_search tool’unu çağıran büyük dil modelinin farklı tool’ları da çağırması için eklenen değişkenler şu şekildedir:

Flights: Başlangıçta null olarak tanımlanan bu değişkene, flight_search tool’u tarafından toplanan Flights bilgileri kaydedilir. Toplanan bilgiler büyük dil modeline gösterilir ve aynı büyük dil modelini tekrar çağdırmaması amaçlanır.

Restaurant, attractions, accommodations değişkenleri, flights değişkeniyle aynı amaca hizmet eder.

- query: büyük dil modeli mesajlar içerinden kullanıcı sorgusunu ayırt etmekte yetersiz kalınca query özellikle belirtildi.
- Tool_calls: Sürekli aynı tool çağrılıp kod sonsuz döngüye girince, büyük dil modeline daha önce oluşturduğu tool çağrıları gösterildi.
- Available_tools: tool_calls listesi eklenince de çözülmeyen sonsuz döngü problemi için eklendi. Çağrılan tool’un ismi listeden silindi. Büyük dil modeline kullanabileceği tool’lar gösterildi.

State
messages (inherited)
flights
restaurants
attractions
acommodations
available_tools
tools
is_continue
query
plan

Şekil 39

- Plan, oluşturulan nihai planı kaydetmek için eklendi.
- is_continue: graph akışını yöneten Router fonksiyonunda kullanılan bir değişken.

Tool Tanımları

- Tool tanımlamaları görev 1.4'te gösterildiği gibi gerçekleşti. Yalnızca büyük dil modeline tool bağlama işlemi bind_tool gibi bir fonksiyonla gerçekleşmediği için, yapılan araştırmalar gereği modele verilen tool'ların json formatlarına çevrilmesi gerekti. Bunun için ise fonksiyonlara Şekil 41'de örneği gösterildiği gibi ek açıklamalar eklendi.

```
def get_current_temperature(location: str, unit: str) -> float:
    """
    Get the current temperature at a location.

    Args:
        location: The location to get the temperature for, in the format "City, Country"
        unit: The unit to return the temperature in. (choices: ["celsius", "fahrenheit"])
    Returns:
        The current temperature at the specified location in the specified units, as a float.
    """
    return 22. # A real function should probably actually get the temperature!
```

Şekil 40

Prompt Tanımlamaları

Kod yapılan tüm güncellemelere rağmen sonsuz döngüye girdiği için ek prompt'lar tanımlandı:

- System Prompt: Büyük dil modelinin tool çağrısı yapabilmesi için geliştirilen prompt. Groq modelinde kullanılan prompt'lar ile sürekli aynı tool çağrısı oluşturulduğu için, daha düzenli ve büyük dil modelini farklı tool çağrıları oluşturmaya yönelten bir prompt olarak hazırlandı. Şekil 42 ve 43'te gösterildiği gibi prompt iki kısımdan oluşmaktadır:
 - İlk kısım yazılı olarak büyük dil modeline görevinden bahseder. Farklı tool'ları çağırması gerektiğini talimatlar. Daha önceden çağırdığı tool'ları ve çağırabileceği tool'ları gösterir.
 - Diğer kısım ise, daha önceden toplanan bilgileri içerir. Ayrıca büyük dil modeline bağlanan tool'ların json formatlarını içerir.
 - Büyük dil modeli bu prompt ile de sürekli aynı tool'u çağırınca, toplanan bilgiler kısmına, "eğer burada bilgi varsa farklı tool çağır" şeklinde uyarılar eklendi. Buna rağmen kodda bir iyileşme olmadı.

```

system_prompt = """
You are Helper assistant.
You task to collect information for a query plan.
You should choose necessary tool to collect information.
Ensure you gather valid information related to transportation, dining, attraction
Note that the nested use of tools is prohibited.
If you collect the all necessary information for query, don't choose any tool.
Instead , respond with 'We can create a plan.'
You have 4 different tools:
You should choose one of them each step.
You can see previously messages that include choosen tools previously. don't cho
()
And you can see previously collected information in here. [1] flights, [2] resta
If there is no information about any constraint, the tool has not been used yet.
If the information is there, DON'T use that tool again.
DON'T USE A TOOL MORE THAN ONE times.
One resons can include ONLY ONE Tool Call.
[1]flights:

```

```
[1]Flights:
  {}
  (If in here, there are Flight data, then you don't choose flight_sea)
[2]Restaurants:
  {}
  (If in here, there are Restaurants data, then you don't choose resta)
[3]Accommodations:
  {}
  (If in here, there are Accommodations data, then you don't choose ac)
[4]Attractions:
  {}
  (If in here, there are Attractions data, then you don't choose attra)
You have 4 different tools:
[1]Flights:
  {}
[2]Restaurants:
  {}
[3]Accommodations:
  {}
[4]Attractions:
  {}
```

- **New_prompt:** İlk prompt ile başarıya ulaşamayınca araştırmalar yapıldı ve yeni bir prompt tanımlandı. Şekil 44'te gösterilen bu prompt, diğer prompt'a göre daha az ve öz bilgi içermektedir. Bu prompt ile de istenilen sonuca ulaşılamadı.

```
new_prompt = """
You are a helper assistant.
Your task is to collect information for query plan

PREVIOUSLY USED TOOLS:
{}

COLLECTED INFORMATION:
Flights:{}
Restaurants:{}
AccommodationS:{}
Attractions:{}

You have 4 different tools:

[2]Restaurants:
    {}
[3]Accommodations:
    {}
[4]Attractions:
    {}

CRITICAL RULES:
```

29

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

- Örnek planlar gösterilerek, çıktı yapılandırıldı.
- Makalede yer alan uyulması gereken kısıtlamalar eklendi. Şekil 46'da gösterilmektedir.

```
planner_prompt = """You are a proficient planner.
Based on the provided information and query, please give me a travel plan
including specifics such as flight numbers (e.g., F0123456).
Note that all the information in your plan should be derived from the provided data.
You must adhere to the format given in the example.
Additionally, all details should align with commonsense.
The symbol '-' indicates that information is unnecessary.
For example, in the provided sample, you do not need to plan a route between cities.
When you travel to two cities in one day, you should note that.

You should create a travel plan according user query.

You can use ONLY provided data to create a travel plan.
In here, these provided data are [1]flights,[2]accommodations,[3]restaurants,[4]attractions.

[1]flights:
{}
[2]restaurants:
{}
[3]accommodations:
{}
[4]attractions:
{}

```

Şekil 44

```
Also, pay attention to the query constraint when generating plan:
[1]Environment Constraint:
  Unavailable Transportation: There is no available flight or driving information between the two city.
  Unavailable Attractions: There is no available attraction information in the queried city.
[2]Commonsense Constraint:
  Within Current City: All scheduled activities for the day must be located within the day's city.
  Reasonable City Route: Changes in cities during the trip must be reasonable.
  Diverse Restaurants: Restaurant choices should not be repeated throughout the trip.
  Diverse Attractions: Attraction choices should not be repeated throughout the trip.
  Non-conf. Transportation: Transportation choices within the trip must be reasonable. For example, having both "self-driving" and "No self-driving" be considered a conflict.
  Minimum Nights Stay: The number of consecutive days spent in a specific accommodation during the trip must meet the required minimum number of nights stay.
[3]Hard Constraint
  Budget: The total budget of the trip.
  Room Rule: Room rules include "No parties", "No smoking", "No children under 10", "No pets", and "No visitors".
  Room Type: Room types include "Entire Room", "Private Room", "Shared Room", and "No Shared Room".
  Cuisine: Cuisines include "Chinese", "American", "Italian", "Mexican", "Indian", "Mediterranean", and "French".
  Transportation: Transportation options include "No flight" and "No self-driving"

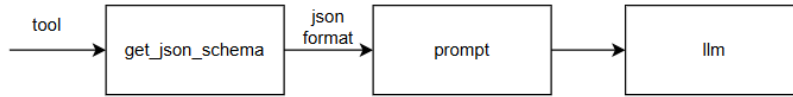
```

Şekil 45

Büyük Dil Modeline Tool Bağlamak

Görev 1.4'te geliştirilen uygulamada, bind_tools fonksiyonu ile llm_with_tools isminde bir büyük dil modeli oluşturulmuştu. Bu uygulamada böyle bir hazır fonksiyon olmadığı için çeşitli denemeler yapıldı.

Öncelikle llm_with_tools isminde bir fonksiyon oluşturuldu. Bu fonksiyonda tool'lar Şekil 47'de gösterildiği üzere get_json_schema ile json formatları prompt aracılığıyla büyük dil modeline bağlandı. Ayrıca new_prompt tanımlanırken {} ile boş bırakılan kısımlar state objesindeki gerekli değişkenler ile doldurulur.



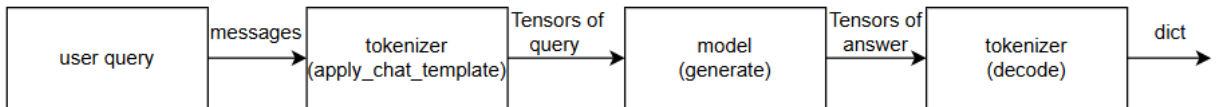
Şekil 46

Başlangıçta Görev 1.2'de gerçekleştirildiği gibi tokenizer encode-decode yöntemi kullanıldı. Elde edilen sonuçlara göre:

- Büyük dil modelinin 2 kritik noktayı ele alması gerekmektedir:
 - Tool ile çalışıp tool çağırısı oluşturması
 - Yapılandırılmış formda çıktı sunması

Encode-decode yöntemiyle bu iki hedefe ulaşılamadı. Tool bağlantısı yalnızca prompt aracılığıyla gerçekleştirildiği için büyük dil modelinin yetersiz kaldığı gözlemlendi. Ayrıca istenen yapılandırılmış çıktıya da ulaşılamadı.

- Yapılan kapsamlı araştırmalar sonucunda Transformer kütüphanesinde apply_chat_template[21] isimli bir fonksiyon keşfedildi ve gerekli denemeler yapıldı. Fonksiyonun tool çağırma konusunda başarılı olduğu, hesaplama sorguları gibi basit projeler ile test edildi.



Şekil 47

Assistant Node

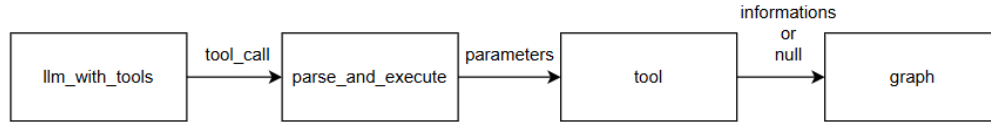
Assistant node, aldığı state objesini yukarda tanımlanan `llm_with_tools` fonksiyonu gönderir. Çıktı olarak döndürülen `tool_call` state kaydedilir. Ayrıca `AIMessage` tipine dönüştürülerek mesaj listesine de kaydedilir.

Tool Çağrısının İşlenmesi

`llm_with_tools` tarafından üretilen çıktılar gözlemlendi. Şekil 49'da gösterilen fonksiyon tool çağrısını, fonksiyon ve parametreler olarak ayrıştırır. Ardından tool çalıştırma işlemi gerçekleştirilir. Büyük dil modelinin tool çağrısını bazen `<|python_tag|>` arasında bazen ise direct dict verisi olarak ürettiği gözlemlendi. Bunun için iki farklı ayrıştırma biçimi oluşturuldu.

Fonksiyon, tool çalışması durumunda toplanan bilgileri, toplanan bilgilerin state'e kaydedilmesi gereken ismini ve `available_tools` listesinden silinmesi için tool ismini döndürür.

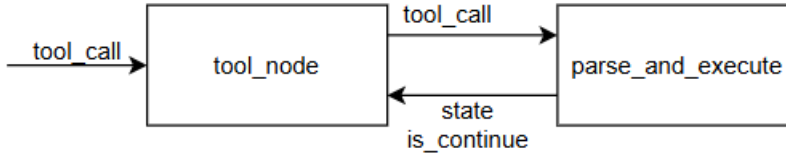
Eğer büyük dil modeli tool çağrısı üretmediyse her 3 değişken null olarak döndürülür.



Şekil 48

Tool Node

Görev 1.4'te gerçekleştirilen uygulamada `LangGraph`'ta önceden tanımlı olan `ToolNode` ile tool'lar birleştirilmiş, tool çağrıları ayrıştırılmış ve çalıştırılmıştı. Burada ise assistant node tarafından oluşturulan tool çağrısı Şekil 50'de gösterildiği gibi tools node tarafından `parse_and_execute_tool` fonksiyonu aracılığıyla ayrıştırılır ve çalıştırılır. Tool işlemleri tamamlandıysa, üretilen bilgiler state'e kaydedilir, eğer büyük dil modeli tool çağrısı oluşturmadıysa `is_continue` değişkeni'ne `false` değeri atanır.



Şekil 49

Router

Router LangGraph içerisinde, conditional_edge'e eklenerek, graph akışını yönetir. should_continue fonksiyonu, is_continue değişkenini kontrol ederek, assistant ve planner node'ları arasında bir seçim yapılmasını sağlar.

Planner Node

Planner Node, tanımlanan llm fonksiyonunu kullanır. Üretilen planı state'e kaydeder. Bu fonksiyon, planner prompt'u tool'lar tarafından toplanılan ve state'e kaydedilen bilgileri ekleyerek formatlar. Yine apply_chat_template fonksiyonu kullanılır, sadece max_new_tokens değişkeninin değeri arttırılarak uzun planlar üretilmesi sağlanır.

Graph Oluşturulması ve Çalıştırılması

Önceki anlatımlarda gösterildiği gibi tanımlanan tüm bileşenler ile bir graph oluşturulur ve derlenir.

Graph state objesi ile çalıştırılacağı için State class'ı ile bir nesne tanımlanır. Nesne yalnızca query değişkeni ve başlangıç tool'larını içermektedir.

1.5.3 Elde Edilen Sonuçlar

- Projede bahsedildiği üzere yapılan tüm güncellemelere rağmen, assistant node sürekli flight_search tool'unu çağırırdı.
- Flight_search çağırısı doğru parametreler ile gerçekleştirildi.
- 25 defa arka arkaya aynı çağrı üretilince graph error vererek çalışmasını sonlandırmaktadır.
- Gemma ile deneme yapıldı ve apply_chat_template fonksiyonu Gemma modeli ile çalışmadı.
- Llama ve Mistral ile de bahsedilen hata alındı.

Bu aşamaların sonucunda Huawei Head of AI, IMEP danışmanına bir sunum yapıldı. Sunum sonucunda yapılan saptamalar:

- Bu projede kullanılan büyük dil modeli 3B parametre içermektedir.
- Yapılan kapsamlı araştırmalar, hangi dil modeli olursa olsun 3B büyüklüğünde bir dil modelinin tool çağırma konusunda yeterince iyi olmadığını göstermektedir.
- 3B üzerindeki modeller ise cihaza fazla gelebilir, bu modellerin quantize edilmesi gerekmektedir.

Projenin geliştirilmesi için yeni bilgi birikimleri edinildi:

- Farklı büyük dil modelleri hakkında araştırmalar yapıldı, dokümantasyonları okundu.
- Büyük dil modeli quantize etme konusu araştırıldı ve uygulandı.
- Büyük dil modelleri için parametre kavramı öğrenildi.
- Farklı büyük dil modelleri için farklı prompt teknikleri öğrenildi.
- Tool çağırısı yapmak üzere geliştirilmiş, fine tune edilmiş modeller hakkında araştırma yapıldı.
- Tool çağırısı yapan yeni bir model server'a yüklenildi.

1.5. LangGraph ve HuggingFace ile Sequential Düzende Travel Planner Geliştirilmesi

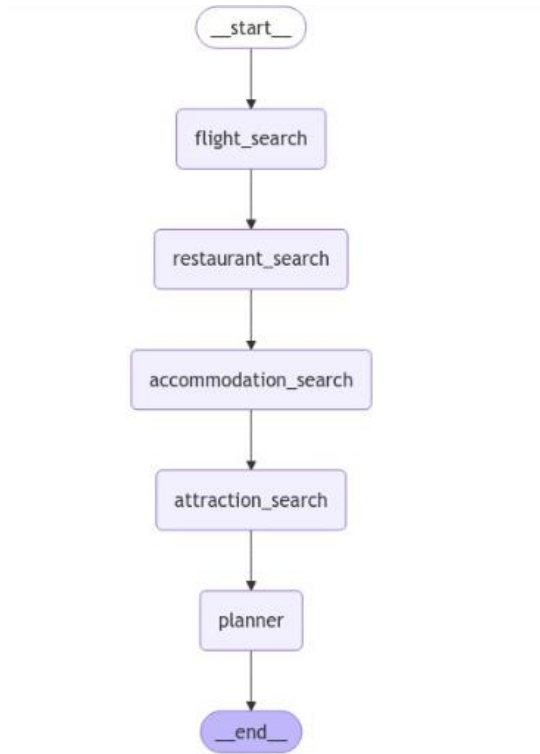
1.6.1 Görev Tanımı ve Hedefler

Bu görevde, 1.5 görevindeki hatanın giderilmesi için tüm tool'ların kullanıldığı bir uygulama geliştirilmeye karar verildi. Hatanın giderilmesi için hedeflenen adımlar:

- Her tool ayrı bir node'a aktarıldı
- Tüm tool'lar sıralı bir yapıda birbirine bağlandı. Her tool'un çalışması sağlandı.

1.6.2 Uygulanan Yöntemler ve Teknolojik Altyapı

Geliştirilen bu graph, 5 node'dan oluşmaktadır ve herhangi bir koşul durumu içermez. Graph Şekil 58'de gösterilmiştir.



Şekil 50

State

State 1.5 görevindeki uygulamaya göre Şekil 52’de gösterildiği gibi sadeleştirildi.

State
messages (inherited)
flights
restaurants
atractions
acommodations
query
plan

Şekil 51

Tool tanımlamaları ve parse_and_execute fonksiyonu görev 1.5’ten kopyalandı.

Prompt Tanımlamaları

Tanımlanan graph’ta iki tipte node bulunmaktadır:

- Tool Node’ları: Bu node’lar flight_search, attraction_search, accommodation_search, restaurant search tool’larından bir tanesini çağırmakla görevlidir.
- Planner Node: Önceki anlatımlarda bahsedilen planlama görevini gerçekleştirir.

Her bir node için ayrı bir prompt tanımlamak yerine tool node içerisinde farklı şekilde formatlanacak, iki farklı prompt template oluşturuldu: tool_prompt ve planner_prompt. Planner prompt olarak görev 1.5’te tanımlanan prompt kullanıldı.

Şekil 53’te tanımlanan Tool prompt, diğer prompt özelliklerine ek olarak, sadece bir tane fonksiyonun json haline almak için {} içerir. Her node bu prompt’u kendine özgü şekilde formatlar. Ayrıca prompt içerisinde ikinci bir {} bulunur ve bu boşluğa da, toplanan bilgilerin tutarlı olması için akış öncesinde toplanan bilgi varsa eklenir.

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

```
tool_prompt = """
You are a helper assistant.
Your task is to collect information for query plan.

Determine the appropriate parameters for the function given to you.
The function given to you:
{}
Some information may have been collected before you. Here is the information.
{}
Highlights.
1. Focus on the query given to you.
2. Look at the function given to you.
3. Examine any previously collected information.
Determine parameters for the given function based on the query and previously collected information.

Your responses must be either:
- One or many parameters

Query:
"""
```

Şekil 52

Görev 1.5'te tanımlanan llm ve llm_with_tools fonksiyonları, prompt format kısımları değiştirilerek yeniden kullanılır.

Graph Oluşturulması

Tüm bileşenler sıralı bir şekilde eklenir ve graph çalıştırılır.

1.6.3 Elde Edilen Sonuçlar

Oluşturulan graph, doğru bir şekilde çalıştı ve Şekil 54'te gösterilen seyahat planını sorunsuz bir şekilde üretti. Bu aşamanın sonunda sunum yapıldı ve bazı değerlendirmeler yapıldı:

```
**Travel Plan**

**Day 1: March 16th, 2022**

* **Current City:** St. Petersburg to Rockford
* **Transportation:** Flight Number: F0123456, from St. Petersburg to Rockford, Depa
* **Breakfast:** Coco Bambu, Rockford (Average Cost: $72)
* **Attraction:** The Anderson Japanese Gardens, Rockford
* **Lunch:** Dunkin' Donuts, Rockford (Average Cost: $24)
* **Dinner:** Gajalee Sea Food, Rockford (Average Cost: $49)
* **Accommodation:** Spacious 3BDR Prime Location!, Rockford (Average Cost: $1030.0,

**Day 2: March 17th, 2022**

* **Current City:** Rockford
* **Breakfast:** Moets Arabica, Rockford (Average Cost: $43)
* **Attraction:** The Burpee Museum of Natural History, Rockford
* **Lunch:** Eggspectation - Jaypee Vasant Continental, Rockford (Average Cost: $77)
* **Dinner:** Nutri Punch, Rockford (Average Cost: $34)
* **Accommodation:** Spacious 3BDR Prime Location!, Rockford (Average Cost: $1030.0,
```

Şekil 53

- Görev 1.6 her ne kadar doğru çalışsa da, oluşturulan sıralı yapı büyük dil modelini kısıtlamaktadır.
- Groq ile geliştirilen uygulama 8B parametre içeren bir büyük dil modeli ile çalışmaktadır. Görev 1.5'te ise 3B parametrelili büyük dil modeli kullanılmıştır. 3B parametre içeren büyük dil modelleri tool çağırma konusunda yetersiz kalmaktadır. Çözüm için büyük dil modelleri üzerine öğrenilmesi gereken alanlar belirlendi.
 - Büyük dil modellerinde parametre kavramı
 - Büyük dil modellerinin quantize edilmesi
 - Farklı büyük dil modelleri için özel prompt'lar

1.7. Büyük Dil Modeli Uygulamaları için UI Geliştirilmesi

1.7.1 Görev Tanımı ve Hedefler

Geliştirilen Travel Planner uygulamasını için bir UI geliştirilmesi amaçlanmıştır. Bunun için Streamlit, Gradio gibi seçeneklerin değerlendirildi. Görevin hedeflenen alt başlıkları şu şekildedir:

- Streamlit bilgi birikimi edinilecek
- Gradio, Chanlit gibi UI teknolojileri hakkında araştırma yapılması
- LangGraph projesi için UI geliştirilmesi

1.7.2 Uygulanan Yöntemler ve Teknolojik Altyapı

Gradio, Chanlit ve Streamlit gibi teknolojiler hakkında araştırma yapıldı ve basit uygulamalar oluşturuldu. Sonucunda Streamlit'in diğer seçeneklere göre daha gelişmiş ve esnek olması nedeniyle devam etmeye karar verildi.[21][22]

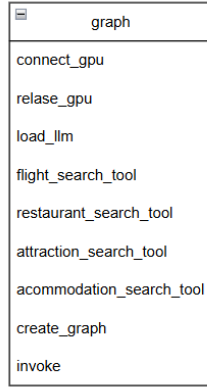
Streamlit ve LangGraph, Streamlit ve Agent entegrasyonuna sahip projeler incelendi. Streamlit dokümantasyonu incelenip uygulandı. [24]

Önceki görevlerde oluşturulan projeler .ipynb uzantılı script'lere kaydedildi. Bu görevde streamlit uygulaması geliştirildiği için .py uzantılı scriptler ile çalışıldı. 5 yeni script oluşturuldu:

- app.py :streamlit arayüz etkileşimi için gerekli kodlar bulunur.
- graph_components.py: Önceki görevlerde tanımlanmış olan, State class'ı, parse_and_execute_tool, llm_with_tools, llm fonksiyonları bu script'e kopyalandı.
- prompts: Önceki görevlerde tanımlanmış olan tool_prompt, planner_prompt isimindeki prompt template'leri kaydedildi.
- tools.py: Tanımlanan 4 farklı tool bu script'e kopyalandı.
- graph.py: Bir graph class'ı oluşturuldu ve graph oluşturma ve çalıştırma işlemleri bu script'te tamamlandı.

Graph Oluşturma ve Çalıştırma

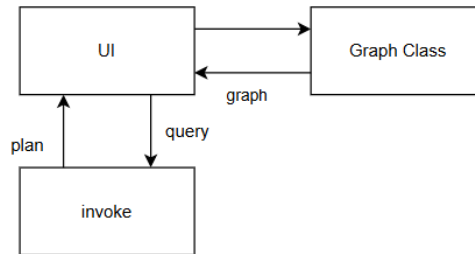
Streamlit arayüzünde graph oluşturmak ve ardından oluşturulan bu graph'ın defalarca çalıştırılmasını sağlamak için bir şekil 55'te görüldüğü gibi bir graph class'ı oluşturuldu. Graph nesnesi ilk oluşturulduğunda bir graph oluşturulur.



Şekil 54

Oluşturulan bu class Şekil 55'te gösterildiği gibi connect_gpu, release_gpu ve load_model gibi temel işlemleri tamamlayan fonksiyonlar içerir. Önceki görevlerde tanımlanmış olan node'lar da class içerisinde tanımlandı.

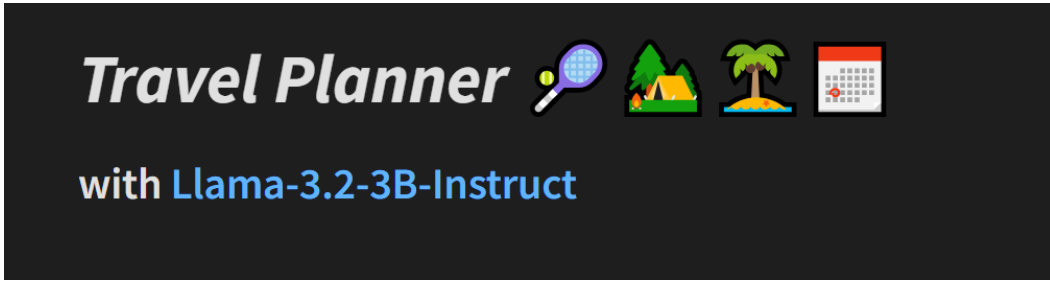
Graph oluşturma işlemi constructor içerisinde çağırılan create_graph fonksiyonu içerisinde gerçekleştirildi. Graph'ın çalıştırılması ise, Şekil 56'da gösterildiği gibi, kullanıcıdan alınan query ile bir nesne oluşturan ve graph.invoke ile graph'ı çalıştıran, üretilen plan'ı döndüren bir fonksiyon ile gerçekleştirildi.



Şekil 55

Streamlit Arayüzü

Streamlit arayüzü için başlık ve altbaşlık gibi bileşenler Şekil 57’de gösterildiği gibi eklendi. Streamlit uygulamalarında kullanıcı UI ile etkileşime geçtiğinde kod yukarıdan aşağıya yeniden çalışır. Yeniden çalışma esnasında bazı değişkenlerin sabit kalması istenebilir. Bu değişkenler `st.session_state` ile kaydedilir.

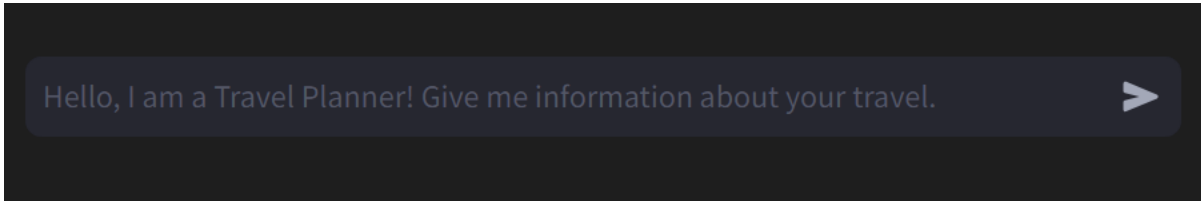


Şekil 56

Streamlit UI başlatılıp query gönderildiğinde graph oluşturulur. Oluşturulan graph’ın yeniden çalışma sonrası korunması için, `@st.cache_resources` dekaratörü kullanıldı. Ayrıca `@st.cachedata`’da bu işlemi yapabilir. `cache_data` verileri kopyalama işlemi yaparken, resource kopyalamadan saklar. Graph içerisinde Transformer modeli bulunduğu için, kopyalayarak saklayan `cache_data` kullanıldığında kod hata vermektedir. [23]

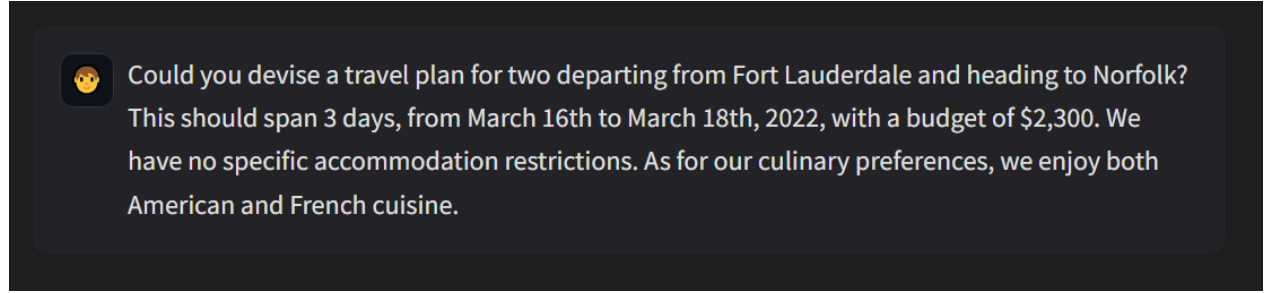
Travel Planner uygulaması bir chat-bot arayüzü ile çalıştırılacaktır. Bunun için mesajlaşma bileşenleri arayüze eklendi. Elde edilen arayüz çıktısı, elde edilen çıktılar bölümünde gösterilecektir.

Arayüze Şekil 58’de gösterildiği gibi query alan bir bileşen eklenir. Bu bileşen hem arayüzde çalışır hem de backend için pointer görevi görür. Kullanıcının sorgu girip girmediğinin control edilmesini sağlar.



Şekil 57

Chat_input bileşenine bir sorgu girilmesi durumunda Şekil 59’de gösterilen chat_message bileşeni oluşturulur ve sorgu gösterilir.

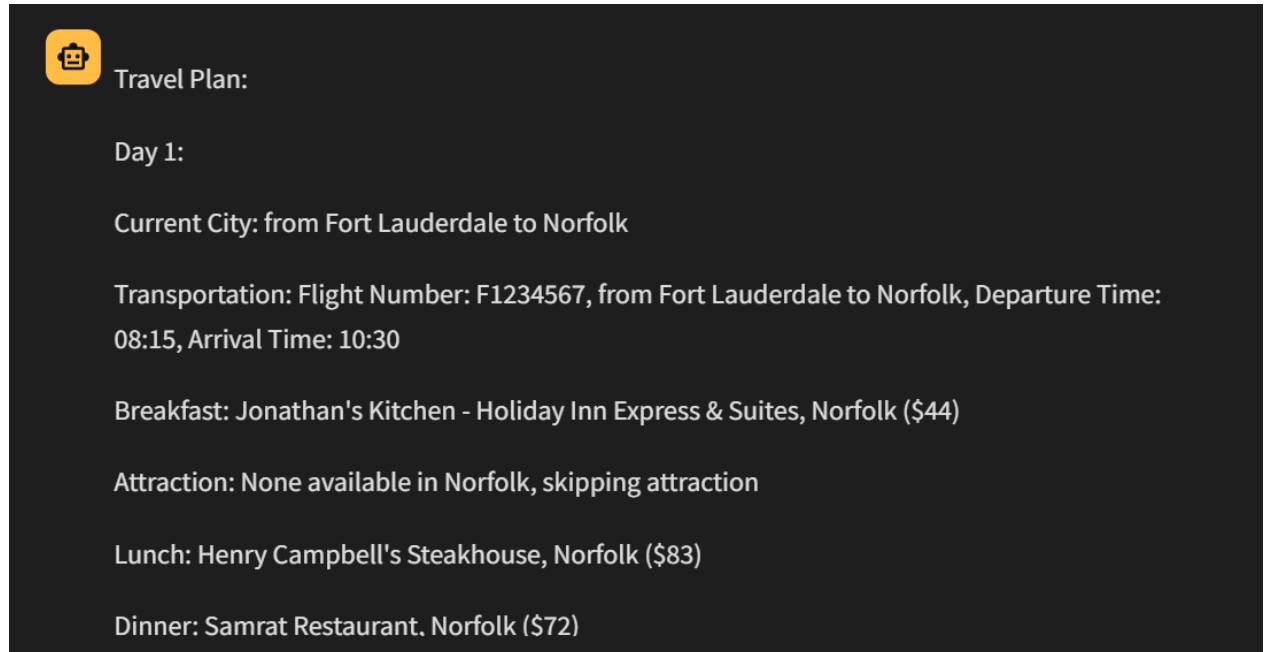


Şekil 58

Graph oluşturulup sorgu gönderilir, ilk sorgu gönderilmesi biraz uzun sürebilir çünkü bu aşamada graph oluşturuluyor. Sonraki sorgularda, oluşturulan graph yeniden kullanılacağı için cevap süresi kısalmaktadır.

1.7.3 Elde Edilen Sonuçlar

Streamlit UI aracılığıyla, LangGraph uygulamasına sorgu gönderildiğinde aşağıda gösterilen çıktılar alındı. Sonuçlar analiz edildiğinde uygulamanın başarıyla çalıştığı gözlemlendi.



Şekil 59

1.8. Travel Planner Ara Çıktıların UI'da Gösterilmesi

1.8.1 Görev Tanımı ve Hedefler

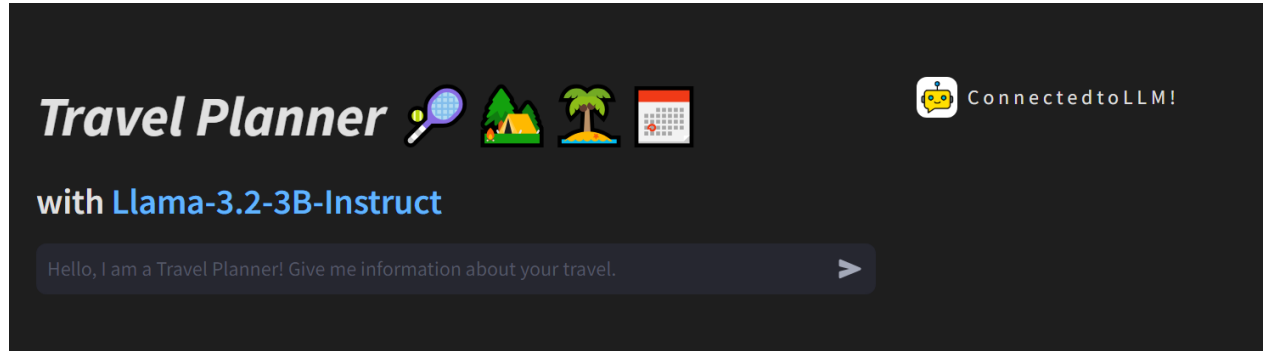
Bu görevde, Görev 1.7'de geliştirilen projeye benzer yalnız graph çalışırken elde edilen ara çıktıların da ekrana bastırıldığı bir uygulama geliştirilmesi hedeflendi. Başlıca hedefler şu şekildedir:

- Streamlit arayüzünde graph çıktılarının gösterilmesi
- Arayüzün görsel açıdan zenginleştirilmesi

1.8.2 Görev Tanımı ve Hedefler

Bu aşamada hem nihai çıktı olan planın hem de ara çıktılar olan tool çağrıları, toplanan bilgiler, büyük dil modeli cevapları gibi graph süreçlerinin ekrana yansıtılması gerekmektedir. Öncelikle arayüz Şekil 75 ile iki bölüme ayrıldı.

Böylelikle, Şekil 61'de gösterildiği üzere ekranın sağ tarafına her bir aşamayı göstermek için ek bir bölüm eklendi.



Şekil 60

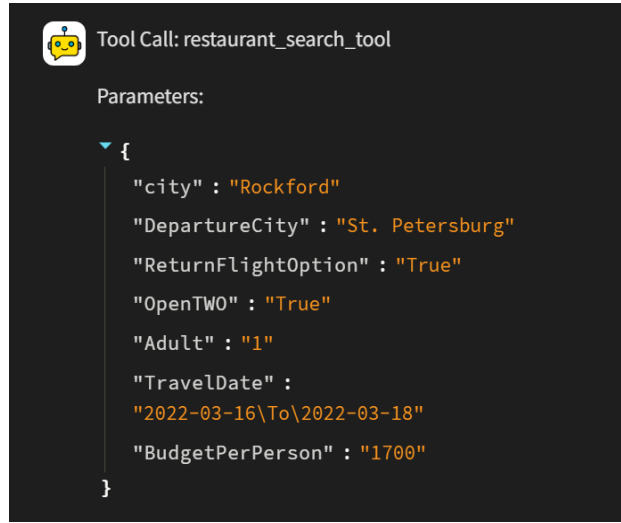
Bir önceki bölümde oluşturulan tools.py, graph_components.py ve prompts.py dosyaları kopyalandı. Ara çıktıların da gösterilmesi için graph.py ve app.py dosyalarındaki kod blokları birleştirildi ve işlendi. Yapılan başlıca değişiklikler aşağıda özetlenmiştir:

- Parse_and_execute_tool fonksiyonu yine aynı şekilde tool_call ayrıştırma ve çalıştırma işlemi yapar.

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

- Önceki uygulamalara ek olarak ek streamlit kodları eklendi. Böylelikle ayrıştırılan tool_call, parametreler ve toplanan bilgiler kendilerine uygun bileşenler ile ekranda gösterildi.

Tool call ve parametreler Şekil 62’de gösterilmiştir. Toplanan bilgiler ise Şekil 63’de gösterildi.



```

Tool Call: restaurant_search_tool

Parameters:
{
  "city": "Rockford"
  "DepartureCity": "St. Petersburg"
  "ReturnFlightOption": "True"
  "OpenTWO": "True"
  "Adult": "1"
  "TravelDate": 
    "2022-03-16\To\2022-03-18"
  "BudgetPerPerson": "1700"
}

```

Şekil 61




	Name	Average Cost	Cuis
2,861	Shree Balaji Chaat Bhandar	97	Frer
3,052	Moets Arabica	43	Tea,
3,163	Cafe Coffee Day	28	Chir
3,700	Nutri Punch	34	Tea,
4,319	Cafe Southall	56	Sea
4,348	Eggspectation - Jaypee Vasar	77	Tea,
4,542	Aroma Rest O Bar	58	Bak
4,735	Advance Bakery	100	Des
4,789	Dial A Cake	29	Café
5,309	U Like	32	Tea,

Şekil 62

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

Tool node'larına da streamlit kodları eklendi ve büyük dil modelleri tarafından oluşturulan cevabın Şekil 64'deki gibi gösterilmesi sağlandı.



```
<|python_tag|>{"name": "attraction_search_tool",  
"parameters": {"city": "Rockford", "date": "["2022-03-16",  
"2022-03-17", "2022-03-18"]", "budget": "1700"}}  
<|eom_id|>
```

Şekil 63

Graph çalışmasını tamamladığında planner node'unda state ekrana yazdırılır, Şekil 65'de gösterilmektedir.



Completed Plan!

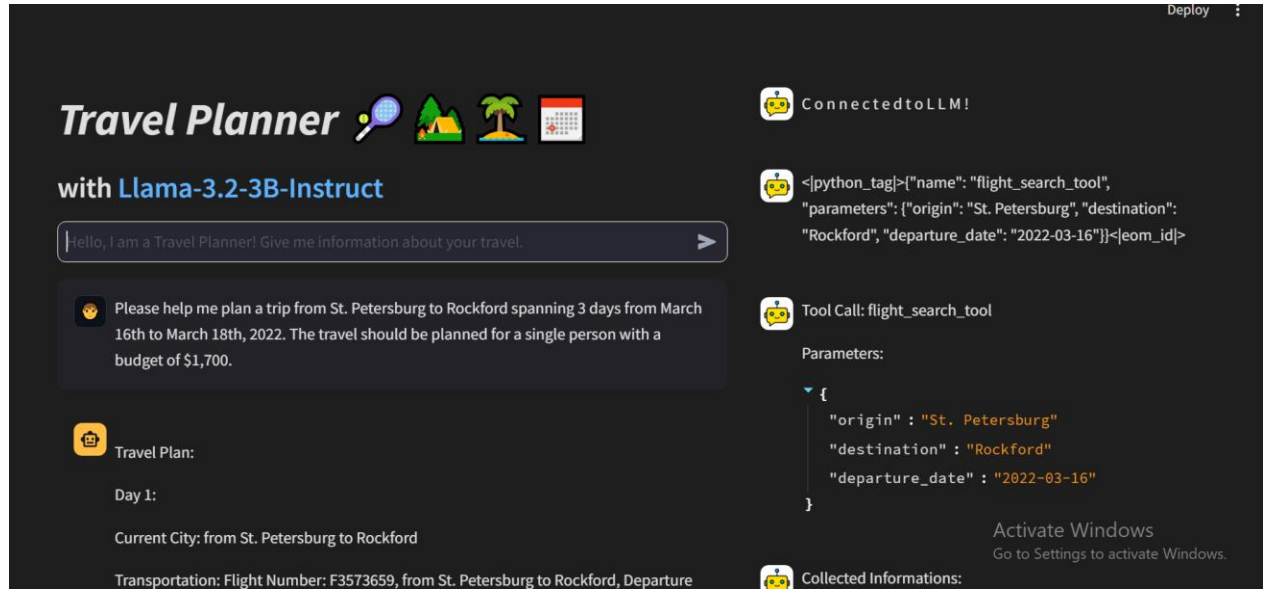
```
{  
  "messages" : []  
  "flights" :  
    "



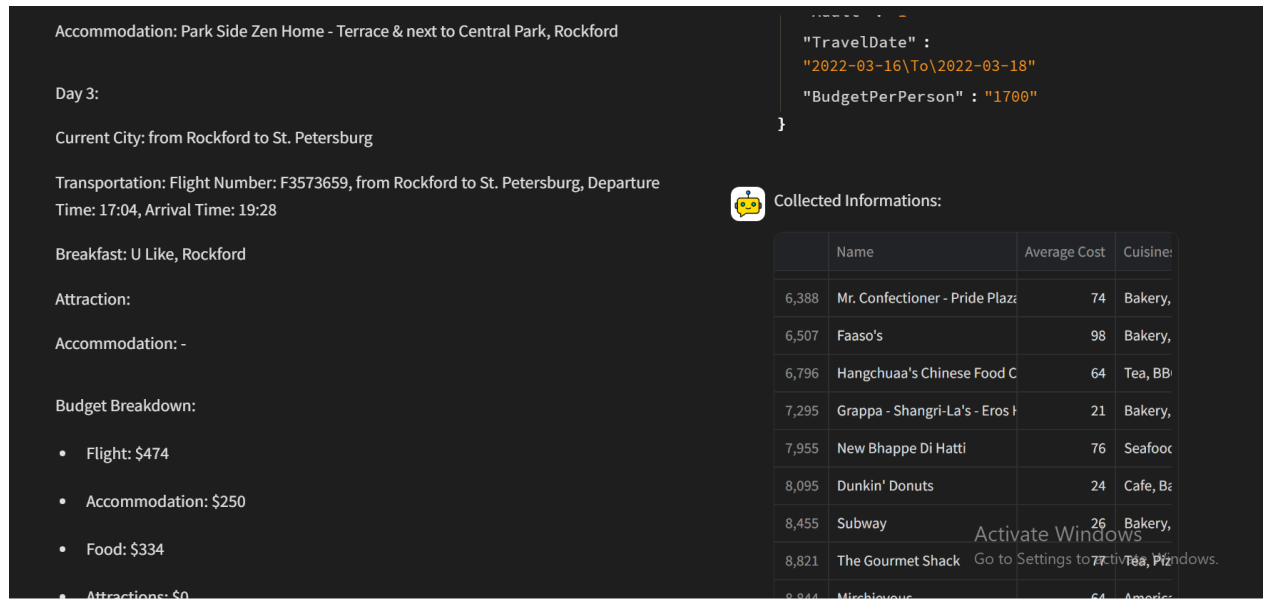

```

1.8.3 Elde Edilen Çıktılar

Ara çıktılar ve nihai sonuç şekil 66 ve 67’da gösterildiği gibi sorunsuz bir şekilde ekranda gösterildi.



Şekil 65



Şekil 66

B. BTÜ-İMEP süreci hakkındaki değerlendirmeleriniz:

Değerli öğrencimiz,

- BTÜ-İMEP hakkındaki olumlu ya da olumsuz görüşlerinizi ifade ediniz.
- BTÜ-İMEP'in kişisel/mesleki gelişiminize katkısını değerlendiriniz.

Bilgisayar Mühendisliği müfredatı gereği 6 dönem boyunca oldukça verimli dersler aldım. Ardından yaz döneminde, okulumun desteğiyle iki adet yaz stajı yaptım. Bunun devamında da Huawei gibi bir teknoloji şirketinde staj yapıyor olmak beni akademik ve sektörel kariyerimde oldukça ileriye taşımaktadır.

Bulduğum ekip Araştırma ve Geliştirme ekibi olduğu için, hem sektörel hem de akademik olarak kendimi geliştirmekteyim. Her bir mesai günüm yeni kavramlar öğrenerek geçiyor.

Huawei'de staj yaparken aynı zamanda sınırsız GPU erişimine sahibim. Böylelikle sürekli büyük dil modelleri üzerinde çalışmalar yapabiliyorum.

Bulduğum ekipteki birçok kişi doktora yapmış, şu an ise sektörde ilerlemekteler. Ben de kariyer planımda hem akademi hem de sektör düşündüğüm için bu durum benim için oldukça avantajlı.

Huawei'de yapıyor olduğum stajımda aynı zamanda her hafta sunum yapmaktayım. Bu sunumlar sayesinde özgüvenim geliyor, kendimi ifade etme konusunda iyileşiyorum.

İmep hakkında yalnızca bir tane olumsuz görüşüm var: dördüncü sınıfın ikinci döneminde yapmamız daha efektif olabilir. Çünkü İmep yaptığım şirket beni bir projeye almak istiyor ama 10 Ocak'ta okula dönecek olmam bu ihtiamli çürütüyor.

Kaynakça

- [1] Jian Xie, K. Z. (2024, 02 02). TravelPlanner: A Benchmark for Real-World Planning with Language Agents. *Computation and Language (cs.CL)*. doi:<https://arxiv.org/abs/2402.01622>
- [2] *OSU-NLP-Group/TravelPlanner*. (tarih yok). OSU-NLP-Group/TravelPlanner: <https://github.com/OSU-NLP-Group/TravelPlanner> adresinden alındı
- [3] Zane Durante, Q. H.-F. (2024, 01 07). Agent AI: Surveying the Horizons of Multimodal Interaction. *Artificial Intelligence (cs.AI)*. doi:Agent AI: Surveying the Horizons of Multimodal Interaction
- [4] *huggingface/llm-vscode*. (tarih yok). huggingface/llm-vscode: <https://github.com/huggingface/llm-vscode> adresinden alındı
- [5] *mistral.ai*. (tarih yok). mistral.ai: <https://docs.mistral.ai/> adresinden alındı
- [6] *tokenizer*. (tarih yok). transformers/main_classes/tokenizer: https://huggingface.co/docs/transformers/main_classes/tokenizer adresinden alındı
- [7] *using_safetensors*. (tarih yok). diffusers/main/using-diffusers/using_safetensors: https://huggingface.co/docs/diffusers/main/using-diffusers/using_safetensors adresinden alındı
- [8] *load-pretrained-instances-with-an-autoclass*. (tarih yok). transformers/v4.46.3/en/autoclass_tutorial#load-pretrained-instances-with-an-autoclass: https://huggingface.co/docs/transformers/v4.46.3/en/autoclass_tutorial#load-pretrained-instances-with-an-autoclass adresinden alındı
- [9] *transformers*. (tarih yok). docs/transformers/index: <https://huggingface.co/docs/transformers/index> adresinden alındı
- [10] *transformers/llm_tutorial*. (tarih yok). llm_tutorial: https://huggingface.co/docs/transformers/llm_tutorial adresinden alındı
- [11] *introduction-to-ai-agents-autogpt-agentgpt*. (tarih yok). datacamp.com/tutorial/introduction-to-ai-agents-autogpt-agentgpt: https://www.datacamp.com/tutorial/introduction-to-ai-agents-autogpt-agentgpt-babyagi?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720821&utm_adgroupid=157098104375&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adposition=&utm_c adresinden alındı

-
- [12] *ai-agents-in-langgraph/*. (tarih yok). [deeplearning.ai/short-courses/ai-agents-in-langgraph/](https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/):
<https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/> adresinden alındı
- [13] *multi-ai-agent-systems-with-crewai/*. (tarih yok). [deeplearning.ai/short-courses/multi-ai-agent-systems-with-crewai/](https://www.deeplearning.ai/short-courses/multi-ai-agent-systems-with-crewai/):
<https://www.deeplearning.ai/short-courses/multi-ai-agent-systems-with-crewai/> adresinden alındı
- [14] *functions-tools-agents-langchain/*. (tarih yok). [deeplearning.ai/short-courses/functions-tools-agents-langchain/](https://www.deeplearning.ai/short-courses/functions-tools-agents-langchain/):
<https://www.deeplearning.ai/short-courses/functions-tools-agents-langchain/> adresinden alındı
- [15] *prompt-engineering-with-llama-2/*. (tarih yok). [deeplearning.ai/short-courses/prompt-engineering-with-llama-2/](https://www.deeplearning.ai/short-courses/prompt-engineering-with-llama-2/):
<https://www.deeplearning.ai/short-courses/prompt-engineering-with-llama-2/> adresinden alındı
- [16] *intro-to-langgraph*. (tarih yok). [langchain.com/courses/intro-to-langgraph](https://academy.langchain.com/courses/intro-to-langgraph):
<https://academy.langchain.com/courses/intro-to-langgraph> adresinden alındı
- [17] *pipelines*. (tarih yok). huggingface.co/docs/transformers/main_classes/pipelines:
https://huggingface.co/docs/transformers/main_classes/pipelines adresinden alındı
- [18] *groq.com/docs/overview*. (tarih yok). console.groq.com/docs/overview:
<https://console.groq.com/docs/overview> adresinden alındı
- [19] *github.com/langchain-ai/langgraph/blob/main/libs/langgraph/langgraph/graph/message.py*. (tarih yok).
<https://github.com/langchain-ai/langgraph/blob/main/libs/langgraph/langgraph/graph/message.py>
- [20] *langchain-ai.github.io/langgraph/reference/prebuilt/*. (tarih yok). <https://langchain-ai.github.io/langgraph/reference/prebuilt/>:
<https://langchain-ai.github.io/langgraph/reference/prebuilt/> adresinden alındı
- [21] *chat_templating*. (tarih yok). https://huggingface.co/docs/transformers/main/en/chat_templating:
https://huggingface.co/docs/transformers/main/en/chat_templating adresinden alındı
- [22] *building-generative-ai-applications-with-gradio/*. (tarih yok). <https://www.deeplearning.ai/short-courses/building-generative-ai-applications-with-gradio/>:
<https://www.deeplearning.ai/short-courses/building-generative-ai-applications-with-gradio/> adresinden alındı

BURSA TEKNİK ÜNİVERSİTESİ
PROJE TABANLI SEKTÖREL EĞİTİM PROGRAMI
BTÜ-İMEP ÖĞRENCİ ARA RAPOR FORMU F3

[23] *streamlit.io*. (tarih yok). <https://docs.streamlit.io/>: <https://docs.streamlit.io/> adresinden alındı

[24] *langchain-ai/streamlit-agent*. (tarih yok). <https://github.com/langchain-ai/streamlit-agent>:
<https://github.com/langchain-ai/streamlit-agent> adresinden alındı