1 Bağlam Mühendisliği Stratejileri

Bağlam mühendisliği stratejilerini daha yakından incelemek için önce bağlam mühendisliği ya da kontekst mühendisliği kavramına bir bakalım. Bağlam Mühendisliği (Context Engineering), yapay zeka uygulamaları, özellikle de dil modelleri (LLM'ler) ve ajanlar ile etkileşim kurma biçimimizde önemli bir evrimi temsil eden yeni bir disiplindir. Bu, bir dil modeline bir görevi tamamlaması için doğru bilgiyi, doğru formatta ve doğru zamanda sağlamak üzere **dinamik sistemler tasarlama ve inşa etme** pratiğidir.

Bu makaleyi Spotify'da sesli olarak dinlemek için podcast'ine bu linkten ulaşabilirsiniz.

1.1 Bağlam Mühendisliğinin Prompt Mühendisliğinden Farkları Nelerdir?

Bağlam Mühendisliği ve Prompt Mühendisliği arasındaki farklar, yapay zeka topluluğunda son zamanlarda sıkça tartışılan ve Prompt Mühendisliğinin bir evrimi veya daha kapsamlı bir disiplini olarak görülen kavramlardır.

İşte bu iki kavram arasındaki temel farklar:

- **Prompt Mühendisliği**
- **Odak Noktası:** Prompt Mühendisliği, bir dil modeline (LLM) en uygun cevabı alabilmek için **kullanıcının doğrudan yazdığı komutun (prompt) ideal formatta yazılması çabalarına odaklanır**. Bu genellikle tek bir etkileşim veya kısa süreli sohbetler için geçerlidir, örneğin ChatGPT ile koşu ayakkabıları hakkında sohbet etmek gibi.
- **Tek Yönlü/Statik:** Geleneksel Prompt Mühendisliği, belirli bir girdi veri setiyle iyi çalışacak bir prompt tasarlamayı amaçlar, genellikle daha statik bir yaklaşıma sahiptir.
- **Sınırlamalar:** Prompt'ın uzunluğu tek başına bir çözüm oluşturmaz, aksine uzadıkça kafa karıştırıcı durumlara yol açabilir. Modelin projenin bütününü, dosya yapısını veya kodlama standartları gibi detayları bilmediği durumlarda, kullanıcı açıkça belirtmezse model en çok karşılaştığı genel ve tahmine dayalı çıktılar üretmeye çalışır.
- **Bağlam Mühendisliği**
- **Kapsamlı ve Dinamik:** Bağlam Mühendisliği, bir görevi tamamlaması için bir LLM'e **doğru bilgiyi, doğru formatta ve doğru zamanda** vermek amacıyla dinamik sistemler tasarlama ve oluşturma pratiğidir. Bu, modelin "bağlam penceresi" adı verilen giriş alanını doğru şekilde "paketlemek" anlamına gelir.
- **Sistem Tasarımı:** Bağlam Mühendisliği, yalnızca kullanıcı talimatlarıyla sınırlı değildir; **sistemlerin kendisiyle ilgilenir** ve ajanın ihtiyaçlarına göre bağlamı dinamik olarak sağlamayı ve değiştirmeyi hedefler. LLM uygulamaları tekli prompt'lardan daha karmaşık, dinamik ajantik sistemlere doğru evrildikçe Bağlam Mühendisliği en önemli AI mühendisi becerisi haline gelmektedir.

- **AI Ajanları için Hayati:** Bağlam Mühendisliği, özellikle AI ajanları gibi yapay zeka uygulamaları geliştirirken önem kazanır. Çünkü bir sohbet botuyla karşılıklı konuşmanın aksine, bir AI ajanı inşa ederken, ajanın tüm olası senaryoları ve alması gereken eylemleri kapsayan bir dizi talimatla donatılması gerekir.
- **Çok Katmanlı Bilgi:** Bir LLM'e verilen "Bağlam" sadece kullanıcı prompt'unu değil, aynı zamanda çok çeşitli bilgi türlerini içerir. Bu bilgiler şunları kapsar:
  - **Sistem Talimatları/Promptları:** Modelin rolünü, davranışını ve kısıtlamalarını baştan tanımlayan yönergeler.
  - **Hafıza ve Geçmiş Etkileşimler:** Önceki konuşmalardan veya oturumlardan hatırlanması gereken bilgiler. Örneğin, bir terapi ajanı için geçmiş konuşmaları hatırlama veya bir kullanıcının geçmiş araç bilgilerini anlama gibi.
  - **Harici Bilgi Tabanı ve Belgeler (RAG):** LLM'in bilgi tabanını genişletmek için harici kaynaklardan (veritabanları, belgeler, dosyalar) çekilen ilgili veriler.
  - **Araç Açıklamaları ve Çıktıları:** Ajanın kullanabileceği harici sistemlerin (API'ler, arama motorları, takvimler vb.) açıklamaları ve bu araçlardan dönen sonuçlar. Örneğin, bir kişisel asistanın Google Takvim'e erişim aracı.
  - **Birkaç Atışlık Örnekler (Few-shot examples):** Modelin istenen davranışı veya çıktı formatını anlamasına yardımcı olan belirli örnekler.
  - **Yapılandırılmış Veri:** XML etiketleri veya Markdown gibi belirli formatlarda sağlanan veriler.
  - **Olasılıksal Bağlam:** Web erişimi veya Model Bağlam Protokolü (MCP) sunucuları aracılığıyla ajanın kendisinin keşfettiği ve kontrol etmesi daha zor olan geniş kapsamlı veriler.
- **Karmaşıklık ve Ölçeklenebilirlik:** Bağlam Mühendisliği, AI ajanlarının daha uzun süreli ve daha karmaşık görevleri ele almasından kaynaklanan artan bağlam kullanımını yönetir. Prompt'lar giderek büyüdükçe ve kod gibi karmaşık hale geldikçe ortaya çıkan sorunları çözmeyi hedefler. Bu, ölçeklenebilir, tutarlı ve güvenilir bir yapay zeka geliştirme süreci kurmak için kaçınılmazdır.
- **Bağlam Hatalarının Yönetimi:** Bağlam Mühendisliği, **Bağlam zehirlenmesi (hallüsinasyonların tekrarlanması), Bağlam dağılması (aşırı odaklanma), Bağlam karışıklığı (alakasız bilginin kullanılması) ve Bağlam çakışması (çelişkili bilgi)** gibi bağlam penceresi hatalarını azaltmayı amaçlar.
- **Uygulanan Teknikler:** Bağlam Mühendisliği, Prompt Mühendisliğinin ötesine geçen çeşitli teknikleri içerir:
  - **Bağlam Yazma (Writing Context):** Bilgiyi bağlam penceresi dışına kaydederek ajanın daha sonra kullanmasını sağlamak (örn. not alma, hafıza kullanımı).
  - **Bağlam Seçimi (Selecting Context):** Harici kaynaklardan veya hafızadan yalnızca ilgili bilgiyi çekerek bağlam penceresine dahil etmek (örn. RAG, araç seçimi).
  - **Bağlam Sıkıştırma (Compressing Context):** Token kullanımını azaltmak için gereksiz veya yedekli bilgileri özetleme veya budama (pruning) yoluyla küçültmek.
  - **Bağlam İzolasyonu (Isolating Context):** Bağlamı farklı ortamlara veya ajanlara bölerek her birinin kendi bağlam penceresine sahip olmasını sağlamak

(örn. çoklu ajan sistemleri, sanal ortamlar).
    - **Bağlam Boşaltma (Context Offloading):** Bilgiyi LLM'in bağlam penceresi dışında depolamak ve yönetmek için araçlar kullanmak.
    - **Araç Yükleme (Tool Loadout):** Bir görev için yalnızca ilgili araçları seçici olarak bağlamak.

Özetle, Prompt Mühendisliği **tek bir komutun etkinliğini optimize etmeye** odaklanırken, Bağlam Mühendisliği, **LLM'lere karmaşık görevleri, özellikle de AI ajanları bağlamında, verimli ve güvenilir bir şekilde tamamlamaları için dinamik olarak doğru bilgiyi sağlama sanatı ve bilimidir**. Bağlam Mühendisliği, modelin davranışı üzerinde dinamik bir "programlama" şekli olarak düşünülebilir.

Bu makaleyi Youtube'da görüntülü olarak izlemek için videosuna bu linkten ulaşabilirsiniz.

1.2 Bağlam Mühendisliği Nedir ve Neden Ortaya Çıktı?

Bağlam Mühendisliği, yapay zeka (YZ) topluluğunda yakın zamanda ortaya çıkan ve **prompt mühendisliğinin bir evrimi veya daha kapsamlı bir disiplini** olarak görülen önemli bir alandır.

- **Bağlam Mühendisliği Nedir?**

Bağlam Mühendisliği, bir Büyük Dil Modeline (LLM) bir görevi başarması için **doğru bilgiyi, doğru formatta ve doğru zamanda** vermek amacıyla **dinamik sistemler tasarlama ve oluşturma pratiğidir**. Başka bir deyişle, bu, bir dil modelinin girdi alanı olan "Bağlam penceresini" doğru şekilde "paketlemek" anlamına gelir.

Bağlam Mühendisliği, modelin davranışını **dinamik olarak programlamak** gibidir ve yalnızca doğrudan kullanıcı komutlarıyla (promptlar) sınırlı değildir. Modelin yanıt üretmeden önce gördüğü her şeyi, yani **tüm girdi bağlamını tasarlama ve bir araya getirme** pratiğini içerir.

Bu bağlam, aşağıdakiler gibi çeşitli bilgi türlerini kapsar:

- **Sistem Talimatları/Promptları:** Modelin rolünü, davranışını ve kısıtlamalarını baştan tanımlayan yönergeler.
- **Alınan Belgeler (Retrieval Augmented Generation – RAG):** Dış bilgi kaynaklarından (veritabanları, belgeler, dosyalar) çekilen ilgili veriler.
- **Hafıza ve Geçmiş Etkileşimler:** Önceki konuşmalardan veya oturumlardan hatırlanması gereken bilgiler.
- **Araç Açıklamaları ve Çıktıları:** Ajanın kullanabileceği harici sistemlerin (API'ler, arama motorları vb.) açıklamaları ve bu araçlardan dönen sonuçlar.
- **Birkaç Atışlık Örnekler (Few-shot examples):** Modelin istenen davranışı veya çıktı

formatını anlamasına yardımcı olan belirli örnekler.
- **Yapılandırılmış Veri:** XML etiketleri veya Markdown gibi belirli formatlarda sağlanan veriler.
- **Olasılıksal Bağlam:** Web erişimi veya Model Bağlam Protokolü (MCP) sunucuları aracılığıyla ajanın kendisinin keşfettiği ve kontrol etmesi daha zor olan geniş kapsamlı veriler.
- **Neden Ortaya Çıktı?**

Bağlam Mühendisliği'nin ortaya çıkışının ve önem kazanmasının temel nedenleri şunlardır:

1. **Prompt Mühendisliğinin Evrimi:**
    - Geleneksel Prompt Mühendisliği, genellikle tek bir etkileşimde veya kısa süreli sohbetlerde bir dil modelinden en uygun cevabı alabilmek için kullanıcının doğrudan yazdığı komutun (prompt) ideal formatta yazılmasına odaklanır.
    - Bağlam Mühendisliği ise, **dinamik olarak değişen verilere ve araçlara göre promptları ve genel bağlamı biçimlendirme** yeteneği ile prompt mühendisliğinin bir uzantısıdır.
2. **YZ Ajanlarının ve Karmaşık YZ Uygulamalarının Yükselişi:**
    - Bağlam Mühendisliği, özellikle **yapay zeka ajanları** gibi daha karmaşık ve uzun soluklu görevleri ele alan uygulamalar geliştirirken önem kazanır.
    - Bir sohbet robotuyla karşılıklı konuşmanın aksine, bir YZ ajanı inşa ederken, ajanın tüm olası senaryoları ve alması gereken eylemleri kapsayan bir dizi talimatla donatılması gerekir.
    - Bu durum, promptların giderek **daha büyük ve daha karmaşık** hale gelmesine, hatta kod gibi XML etiketleri ve Markdown içermesine neden olur.
3. **Dil Modellerinin Sınırlı Bağlam Penceresi ve Belleği:**
    - Andre Karpathy'nin benzetmesiyle, LLM işlemci (CPU) ve Bağlam penceresi de RAM (çalışma belleği) gibidir. Bu RAM'in sınırlı bir kapasitesi vardır.
    - Uzun süreli görevler veya araç çağrılarından gelen geri bildirimler, Bağlam penceresinde birikerek **token kullanımını artırır** ve modeli "boğabilir".
4. **Bağlam Hatalarının Yönetimi:** Bağlam penceresinin uzunluğu arttıkça, modellerde çeşitli "Bağlam hataları" ortaya çıkabilir:
    - **Bağlam Zehirlenmesi (Context Poisoning):** Halüsinasyonlar veya hatalar bağlama sızdığında ve tekrar tekrar referans alındığında modelin yanlış hedeflere odaklanmasına yol açar.
    - **Bağlam Dağılması/Dikkatsizliği (Context Distraction):** Bağlam çok uzadığında modelin aşırı odaklanmasına veya eğitim sırasında öğrendiklerini ihmal etmesine neden olur. Özellikle küçük modellerde bu sorun daha düşük token limitlerinde görülür.
    - **Bağlam Karışıklığı (Context Confusion):** Bağlamdaki gereksiz içeriğin düşük kaliteli yanıtlar üretmesine yol açar. Örneğin, bir ajana çok fazla araç tanımlaması verildiğinde, alakasız araçları çağırmaya çalışabilir.
    - **Bağlam Çakışması (Context Clash):** Yeni bilgilerin veya araçların bağlamdaki diğer bilgilerle çelişmesi durumunda ortaya çıkar, modelin tutarsız davranmasına neden olabilir.

5. **Ölçeklenebilirlik, Tutarlılık ve Güvenilirlik İhtiyacı:**
    - Geliştirilen YZ uygulamalarının sadece konsept kanıtı veya demo olmaktan öteye geçmesi için, **ölçeklenebilir, tutarlı ve güvenilir** bir geliştirme sürecine ihtiyaç duyulur.
    - Bağlam Mühendisliği, yapay zekaya sadece ne yapacağını değil, **neyi neden ve nasıl yapması gerektiğini** de anlatarak, modelin projenin genel yapısına uygun, tutarlı ve bütünsel cevaplar vermesini sağlar.

Özetle, Bağlam Mühendisliği, YZ uygulamalarının artan karmaşıklığına, dil modellerinin sınırlamalarına ve YZ ajanlarının dinamik ihtiyaçlarına bir yanıt olarak ortaya çıkmıştır. Yapay zeka mühendisleri için **en önemli becerilerden biri** haline gelmektedir.

1.3 AI Ajanlarının Temel Bileşenleri ve Bağlam Mühendisliğinin Rolü

Yapay Zeka (YZ) ajanlarının temel bileşenleri ve Bağlam Mühendisliği'nin bu bileşenler içindeki rolü, özellikle yapay zeka uygulamalarının gelişiminde merkezi bir öneme sahiptir.

- **Yapay Zeka Ajanlarının Tanımı ve Temel Bileşenleri**

Bir **Yapay Zeka ajanı**, kullanıcılar adına hedeflere ulaşmak ve görevleri tamamlamak için yapay zekayı kullanan bir yazılım sistemidir. Geleneksel bir sohbet robotuyla tek seferlik karşılıklı konuşmanın aksine, bir YZ ajanı inşa ederken, ajanın gerçekleştirmesi gereken tüm eylemleri ve karşılaşabileceği farklı senaryoları kapsayan **kapsamlı bir talimat setine** sahip olması gerekir. Bu ajanlar genellikle daha uzun süreli veya daha karmaşık görevleri üstlenir ve araç kullanma (tool calling) yeteneğine sahiptirler.

Bir YZ ajanı oluşturan altı temel bileşen bulunmaktadır:

- **Model:** Her YZ ajanı bir yapay zeka modeline ihtiyaç duyar. Bu model açık kaynaklı (örneğin GPT modelleri, Claude, Gemini gibi büyük veya daha küçük modeller) veya özel modeller olabilir.
- **Araçlar (Tools):** Araçlar, ajanların harici sistemlerle etkileşim kurmasını sağlar. Örneğin, kişisel bir asistan ajanı, randevuları ayarlamak için Google Takviminize erişim sağlayan bir araca ihtiyaç duyabilir.
- **Bilgi ve Bellek (Knowledge and Memory):** Çoğu ajanın bilgiyi depolaması ve geri alması gerekir. Örneğin, bir terapi YZ ajanı, önceki konuşmalarda söylenenleri hatırlayabilen bir belleğe sahip olmalıdır. Bir yasal YZ ajanı ise belirli vakalar hakkında bir bilgi tabanına ihtiyaç duyabilir. Bellek, kısa süreli (oturum içi) ve uzun süreli (oturumlara yayılan) olabilir.
- **Ses ve Konuşma (Audio and Speech):** YZ ajanını ses ve konuşma yeteneğiyle donatmak, etkileşimi daha doğal ve kolay hale getirir.
- **Güvenlik Bariyerleri (Guardrails):** Bunlar, ajanın uygunsuz veya istenmeyen

davranışlarını önlemek için tasarlanmış güvenlik mekanizmalarıdır. Örneğin, bir müşteri hizmetleri YZ ajanı kullanıcılara kötü söz söylememelidir.

- **Orkestrasyon (Orchestration):** Bu sistemler, YZ ajanlarını dağıtmanıza, izlemenize ve zamanla iyileştirmenize olanak tanır. Ajanın davranışlarını sürekli olarak takip etmek önemlidir.
- **Bağlam Mühendisliğinin Rolü**

Bağlam Mühendisliği, YZ ajanları gibi karmaşık LLM uygulamaları oluştururken ortaya çıkan **en önemli yapay zeka mühendisi becerisi** olarak kabul edilmektedir. Andre Karpathy'nin benzetmesiyle, LLM'ler bir CPU gibiyken, **Bağlam penceresi de RAM (çalışma belleği)** gibidir ve sınırlı bir kapasiteye sahiptir.

Bağlam Mühendisliği, bir LLM'ye bir görevi tamamlaması için **doğru bilgiyi, doğru formatta ve doğru zamanda** vermek amacıyla **dinamik sistemler tasarlama ve oluşturma** pratiğidir. Bu, dil modelinin girdi alanı olan "Bağlam penceresini" doğru şekilde "paketlemek" anlamına gelir. Bağlam Mühendisliği, bir dil modelinin davranışını **dinamik olarak programlamak** gibi düşünülebilir.

Bağlam Mühendisliğinin YZ ajanlarının bileşenleri ve genel işleyişindeki rolleri şunlardır:

1. **Ajanın "Talimat Kılavuzunu" Oluşturma:**
   - YZ ajanlarının tüm bileşenlerinin (model, araçlar, bellek vb.) nasıl bir araya getirileceğini ve nasıl kullanılacağını detaylandıran "talimat kılavuzu"nu **Bağlam mühendisi hazırlar**. Bu talimatlar, zamanla daha karmaşık hale gelerek XML etiketleri ve Markdown gibi kod benzeri yapılar içerebilir.
   - Bu, modelin rolünü, davranışını ve kısıtlamalarını baştan tanımlayan **sistem talimatlarını/promptlarını** içerir.
2. **Dinamik Bilgi Sağlama ve Yönetimi:**
   - Prompt mühendisliği genellikle tek bir komutun etkinliğini optimize etmeye odaklanırken, Bağlam Mühendisliği, **dinamik olarak değişen verilere ve araçlara göre promptları ve genel bağlamı biçimlendirme** yeteneğiyle prompt mühendisliğinin bir uzantısıdır.
   - Bu, sadece kullanıcı prompt'unu değil, aynı zamanda dış bilgi kaynaklarından (RAG yoluyla çekilen belgeler), geçmiş etkileşimlerden (hafıza), araç açıklamalarından ve çıktılarından (API yanıtları, arama sonuçları) ve yapılandırılmış verilerden (JSON, XML) gelen çeşitli bilgi türlerini de içerir.
   - Bağlam mühendisliği, ajanın ihtiyaçlarına göre bağlamı dinamik olarak sağlamayı ve değiştirmeyi hedefler.
3. **Bağlam Penceresi Sınırlamalarını ve Hatalarını Yönetme:**
   - LLM'lerin sınırlı bağlam penceresi vardır. Ajanlar uzun süreli görevler yürüttükçe veya araç çağrılarından gelen geri bildirimleri kullandıkça, bağlam penceresinde bilgi birikimi ve "token şişmesi" (token bloat) yaşanabilir.
   - Bu durum, **Bağlam zehirlenmesi (hallüsinasyonların tekrarlanması), Bağlam dağılması (aşırı odaklanma), Bağlam karışıklığı (alakasız bilginin kullanılması) ve Bağlam çakışması (çelişkili bilgi)** gibi çeşitli Bağlam

hatalarına yol açabilir. Bağlam Mühendisliği, bu hataları azaltmayı amaçlar.

4. **Temel Bağlam Mühendisliği Teknikleri:** YZ ajanlarının performansını optimize etmek ve Bağlam hatalarını gidermek için çeşitli teknikler kullanılır:
    - **Bağlam Yazma (Writing Context) / Boşaltma (Offloading) / Kalıcılık (Persisting):** Bilgiyi LLM'in bağlam penceresi dışına kaydederek (örneğin bir dosya, durum nesnesi veya uzun süreli belleğe) ajanın daha sonra kullanmasını sağlamak.
    - **Bağlam Seçimi (Selecting Context) / Bilgi Geri Çekmeli Üretim (RAG):** Harici kaynaklardan veya bellekten yalnızca göreve en uygun bilgiyi seçerek bağlam penceresine dahil etmek. Bu, ajanın kullanabileceği çok sayıda araç arasından sadece ilgili olanları seçmek için de kullanılabilir.
    - **Bağlam Sıkıştırma (Compressing Context):** Token kullanımını azaltmak için gereksiz veya yedekli bilgileri özetleme (summarization) veya budama (pruning) yoluyla küçültmek. Ancak bu, bilgi kaybı riski taşıdığı için dikkatli yapılmalıdır.
    - **Bağlam İzolasyonu (Isolating Context) / Karantina (Quarantine) / Çoklu Ajan Sistemleri (Multi-Agent Systems):** Bağlamı farklı ortamlara veya ajanlara bölerek her birinin kendi bağlam penceresine sahip olmasını sağlamak. Bu, sistemin genel olarak işleyebileceği token miktarını artırır.

Özetle, Bağlam Mühendisliği, YZ ajanlarının karmaşık ve dinamik görevleri verimli, tutarlı ve güvenilir bir şekilde yerine getirebilmesi için **modelin doğru "işletim sistemine" ve "RAM" yönetimine sahip olmasını** sağlar. Bu disiplin, LLM uygulamalarının tekli komutlardan çok daha karmaşık ve dinamik sistemlere evrildiği günümüz yapay zeka mühendisliğinde kritik bir rol oynamaktadır.

1.4 Bağlam Mühendisliği Stratejileri ve Teknikleri

Bağlam Mühendisliği, Yapay Zeka (YZ) ajanları ve Büyük Dil Modeli (BBD – Large Language Model – LLM) uygulamaları geliştirilirken ortaya çıkan ve **en önemli yapay zeka mühendisi becerisi** olarak kabul edilen bir alandır. Bu, bir dil modeline bir görevi tamamlaması için **doğru bilgiyi, doğru formatta ve doğru zamanda** vermek amacıyla **dinamik sistemler tasarlama ve oluşturma** pratiğidir. Andre Karpathy'nin benzetmesiyle, BBD'ler bir CPU gibiyken, **Bağlam penceresi de RAM (çalışma belleği)** gibidir ve sınırlı bir kapasiteye sahiptir.

- **Neden Bağlam Mühendisliği?**

Geleneksel prompt mühendisliğinden farklı olarak, Bağlam mühendisliği **karmaşık ve uzun soluklu YZ uygulamaları** ile ilgilenir. YZ ajanları, sohbet robotlarının tek seferlik etkileşimlerinin aksine, genellikle **daha uzun süreli veya daha karmaşık görevleri üstlenir** ve **araç kullanma (tool calling)** yeteneğine sahiptirler. Bu durum, modelin Bağlam penceresinde **bilgi birikimine ve "token şişmesi"ne (token bloat)** neden olabilir.

Bu birikim, çeşitli Bağlam hatalarına yol açabilir:

- **Bağlam Zehirlenmesi (Context Poisoning):** Halüsinasyon veya diğer hataların Bağlame girerek tekrar tekrar referans alınması, modelin hatalı veya sanrısal hedeflere odaklanmasına neden olur.
- **Bağlam Dağılması / Odak Kaybı (Context Distraction):** Bağlam çok uzadığında, modelin aşırı odaklanması ve eğitim sırasında öğrendiklerini ihmal etmesi sonucu tekrarlayan veya yaratıcı olmayan eylemler gerçekleştirmesidir. Bu durum, özellikle daha küçük modellerde daha erken token limitlerinde görülür.
- **Bağlam Karışıklığı (Context Confusion):** Bağlam içinde gereksiz bilginin kullanılmasıyla düşük kaliteli yanıtlar üretilmesidir. Özellikle çok sayıda araç tanımlandığında, model alakasız bir aracı kullanma eğilimi gösterebilir.
- **Bağlam Çakışması (Context Clash):** Bağlame eklenen yeni bilgilerin mevcut bilgilerle çelişmesidir. Bu durum, özellikle çok turlu ve parçalı (sharded) talimatlarda modelin performansını düşürebilir.

Bu sorunları gidermek ve ajanın etkinliğini artırmak için Bağlam mühendisliği stratejileri devreye girer.

- **Bağlam Mühendisliği Stratejileri ve Teknikleri**

Bağlam mühendisliği, temel olarak **dört ana kategoriye** ayrılan teknikleri içerir:

- **1. Bağlam Yazma (Writing Context)**

Bu strateji, bilginin doğrudan BBD'nin **Bağlam penceresi dışına kaydedilmesini** içerir, böylece ajan bu bilgileri daha sonra kullanabilir. İnsanların not almasına veya şeyleri hatırlamasına benzer bir işlev görür.

- **Not Defterleri (Scratchpads):** Bir ajan, bir görevi yerine getirirken geçici bilgileri kaydetmek için bir not defteri (scratchpad) kullanabilir. Bu, genellikle tek bir oturum içinde geçerlidir ve ajanın bir planı veya ara düşünceleri kaydetmesine olanak tanır.
- **Bellek (Memory):** Bilgiyi birden fazla oturumda saklamak için kullanılır. Örneğin, bir terapi ajanı önceki konuşmaları hatırlamak için uzun süreli belleğe ihtiyaç duyabilir. Bellek, kullanıcı-ajan etkileşimlerine dayalı olarak dinamik olarak güncellenebilir. Bu bilgiler dosyalara, durum nesnelerine veya uzun süreli bellek depolarına kaydedilebilir.

- **2. Bağlam Seçimi (Selecting Context)**

Bu teknik, harici kaynaklardan veya bellekten yalnızca **göreve en uygun bilginin seçilerek** bağlam penceresine dahil edilmesini ifade eder.

- **Bilgi Geri Çekmeli Üretim (RAG – Retrieval Augmented Generation):** BBD'nin bilgi tabanını harici, özelleştirilmiş verilerle (belgeler, veritabanları, web aramaları) genişletmek için kullanılır. Bu, özellikle çok büyük bilgi koleksiyonlarından ilgili bilgiyi çekmek için gömme tabanlı benzerlik aramasını (embedding-based similarity search)

veya bilgi grafiklerini (knowledge graphs) kullanır.
- **Araç Yüklemesi (Tool Loadout):** Ajanların çok sayıda araca sahip olması durumunda, yalnızca göreve **semantik olarak en alakalı araçların** açıklamalarını bağlam penceresine dahil etmek için kullanılır. Bu, özellikle araç sayısının 30'u aştığında performans düşüşünü önlemeye yardımcı olabilir.
- **Hafıza Türlerinin Seçimi:** Öğrenilmiş davranışlar (few-shot examples), olgular (facts) veya prosedürel bilgiler (instructions) gibi farklı bellek türlerinin görevin gerektirdiğine göre seçici olarak Bağlame çekilmesi.

- **3. Bağlam Sıkıştırma (Compressing Context)**

Bu yöntem, **token kullanımını azaltmak** ve modeli aşırı yüklenmekten korumak için gereksiz veya yedekli bilgileri küçültmeyi amaçlar.

- **Özetleme (Summarization):** Büyük miktardaki bilgiyi (örneğin sohbet geçmişi veya tamamlanmış iş bölümleri) daha kısa, yoğun bir özete dönüştürmektir. Bu, Bağlam penceresi dolduğunda veya ajanlar arasında bilgi aktarımı yapılırken faydalıdır. Ancak özetleme, **bilgi kaybı riski** taşıdığı için dikkatli yapılmalıdır.
- **Budama / Kırpma (Pruning / Trimming):** Bağlamteki alakasız veya istenmeyen bilgileri seçici olarak kaldırmaktır. Örneğin, sadece en son mesajları tutmak veya BBD tabanlı yöntemlerle alakasız kısımları ayıklamak gibi. Bu da özetleme gibi bilgi kaybı riski taşıyabilir.

- **4. Bağlam İzolasyonu (Isolating Context)**

Bu strateji, modeli aşırı yüklemekten veya çelişkili bilgilerle karıştırmaktan kaçınmak için **bağlamı farklı ortamlara veya ajanlara bölmeyi** içerir.

- **Çoklu Ajan Sistemleri (Multi-Agent Systems):** Görevleri paralel olarak yürütmek üzere tasarlanmış birden fazla uzmanlaşmış ajanın kullanılmasıdır. Her bir alt ajan kendi Bağlam penceresine sahip olur, böylece sistemin genel olarak işleyebileceği token miktarı artar. Ancak, görevler çok sıkı bağlıysa veya kararlar bağımsız olarak alındığında çelişkiler ortaya çıkma riski vardır. Bu tür sistemlerde, genellikle bilgi toplama işleri paralelleştirilirken, raporlama veya karar verme tek bir ajanda birleştirilir.
- **Karantina (Quarantine) / Sanal Alanlar (Sandboxes):** Kod yürütme veya token yoğun nesnelerin (resim, ses dosyaları) BBD'nin ana Bağlam penceresinden ayrı, izole bir ortamda tutulmasıdır. Yalnızca BBD'nin bir sonraki karar için ihtiyaç duyduğu seçici bilgiler ana bağlama geri iletilir.
- **Durum Nesneleri (State Objects):** Ajanın durumunu ve çeşitli bilgi bölümlerini yapılandırılmış bir veri modelinde (örneğin Python'da bir sözlük veya Pydantic modeli) saklamaktır. Bu, belirli bilgilerin yalnızca ihtiyaç duyulduğunda BBD'ye gönderilmesine olanak tanır, böylece bağlam penceresi yönetilebilir kalır.

- **Pratik Uygulama**

Bağlam mühendisliği, modelin rolünü, davranışını ve kısıtlamalarını baştan tanımlayan **sistem talimatlarını/promptlarını** içerir. Bu talimatlar, zamanla daha karmaşık hale gelerek XML etiketleri ve Markdown gibi kod benzeri yapılar içerebilir. Projeye özgü klasör yapısı, teknoloji ve sürüm bilgileri, stil kuralları, kodlama standartları ve veri saklama stratejileri gibi yapısal kararlar, **kalıcı ve düzenli dosyalarda** tanımlanarak modelin sürekli olarak uygun bağlamı kullanması sağlanır. Bağlam dosyaları birbiriyle çelişmemeli, gereksiz tekrar içermemeli ve modüler bir yapıda düzenlenmelidir.

Sonuç olarak, Bağlam Mühendisliği, YZ ajanlarının karmaşık ve dinamik görevleri verimli, tutarlı ve güvenilir bir şekilde yerine getirebilmesi için modelin doğru "işletim sistemine" ve "RAM" yönetimine sahip olmasını sağlar. Bu disiplin, BBD uygulamalarının tekli komutlardan çok daha karmaşık ve dinamik sistemlere evrildiği günümüz yapay zeka mühendisliğinde kritik bir rol oynamaktadır.

1.5 Örnek Bir Bağlam Mühendisliği İstemi

Bağlam Mühendisliği istemi, bir Yapay Zeka (YZ) ajanına veya Büyük Dil Modeli (BBD) uygulamasına bir görevi başarıyla tamamlaması için **doğru bilgiyi, doğru formatta ve doğru zamanda** vermek üzere tasarlanmış detaylı ve yapılandırılmış talimatlar setidir. Bu, geleneksel prompt mühendisliğinden daha ileri bir adımdır, zira sadece tek seferlik sohbetler için değil, **uzun soluklu ve karmaşık görevleri** yerine getiren YZ ajanları için hayati öneme sahiptir. Bağlam mühendisliği istemleri zamanla XML etiketleri ve Markdown gibi kod benzeri yapılar içerecek şekilde daha büyük ve karmaşık hale gelebilir.

Aşağıda, bir yapay zeka araştırma asistanı için hazırlanmış örnek bir Bağlam mühendisliği isteminin genel yapısı ve içeriği detaylandırılmıştır:

1. **Yapay Zeka Araştırma Asistanı İçin Bağlam Mühendisliği İstemi Örneği**

Bu örnek istem, bir YZ araştırma asistanının görevini, çıktı formatını, kısıtlamalarını ve yeteneklerini açıkça belirterek modelin karmaşık bir araştırma sürecini yönetmesini sağlar. İstem genellikle altı ana bileşenden oluşur:

1. **Rol (Role)**:
    - **Tanım:** "Yapay zeka alanındaki son trendleri birden fazla kaynak türünden belirlemeye ve özetlemeye odaklanmış bir YZ araştırma asistanısın."
    - **Amaç:** "Kullanıcının sorgusunu uygulanabilir alt görevlere ayırmak ve etkileşime ve kaynağın yetkisine (otorite) göre en alakalı bilgileri döndürmek."
    - Bu bölüm, ajanın genel amacını ve kimliğini tanımlar.
2. **Görev (Task)**:
    - **Adım 1:** "XML etiketleri <user_query> ve </user_query> ile sınırlandırılmış bir araştırma sorgusu verildiğinde, her biri farklı bir açı veya kaynak türünü hedefleyen 10 adede kadar çeşitli yüksek öncelikli alt görev çıkar."

- **Adım 2:** “Görünümler, beğeniler, yeniden paylaşımlar ve alıntılar gibi etkileşime ve yayın itibarı ve alan uzmanlığı gibi kaynağın yetkisine göre önceliklendir.”
- **Adım 3:** “Her alt görev için aşağıdaki formatta bir JSON çıktısı oluştur.”
- **Adım 4:** “Belirtilen zaman dilimine göre doğru başlangıç ve bitiş tarihlerini UTC ISO formatında hesapla.”
- **Adım 5:** “Tüm bulguları tek, özlü bir trend özetinde (yaklaşık 300 kelime maksimum) özetle.”
- Bu bölüm, ajanın takip etmesi gereken sıralı ve detaylı adımları tanımlar. Her adımın spesifik bir çıktısı veya kriteri vardır.

3. **Girdi (Input)**:
   - Kullanıcının araştırma sorgusu, XML etiketleri (<user_query> ve </user_query>) arasına yerleştirilir. Örneğin:
   - <user_query>
   - [Arama sorgusunu buraya ekleyin]
   - </user_query>
   - Bu, modelin kullanıcı girdisini net bir şekilde tanımasını sağlar.
4. **Çıktı (Output)**:
   - **Alt Görevler İçin JSON Formatı:** Her alt görev için tam olarak uyulması gereken bir JSON formatı belirtilir. Bu format aşağıdaki değişkenleri içerir:
     - id: Kimlik numarası.
     - query: Ana konunun bir yönüyle ilgili alt sorgu.
     - source_type: Kaynak türü (örn. news, X, Reddit, LinkedIn, newsletter, academic, specialized).
     - time_period: Kaynağın yayınlandığı zaman dilimi (örn. 1 day ila 10 days arası).
     - domain_focus: Alan odağı (örn. technology, science, health).
     - priority: Kaynağın önemi (1’den 10’a kadar, 1 en yüksek).
     - start_date: Başlangıç tarihi (spesifik ISO formatında).
     - end_date: Bitiş tarihi (spesifik ISO formatında).
   - **Nihai Özet Formatı:** Tüm alt görevler tamamlandıktan sonra, son çıktı 300 kelimeyle sınırlı, madde işaretleri veya kısa paragraflar halinde yazılmış bir özet olmalıdır. “Yeni, ilgili, yüksek sinyalli gelişmeleri” içermeli ve “gereksiz ayrıntılardan, arka plan bilgisinden veya kişisel yorumdan” kaçınmalıdır.
   - Çıktı formatının bu kadar detaylı belirtilmesi, modelin tahmin edilebilir ve işlenebilir sonuçlar üretmesini sağlar.
5. **Kısıtlamalar (Constraints)**:
   - **Odak:** “Ana noktayı özlü bir şekilde yakalamaya odaklan.”
   - **Dilbilgisi:** “Tam cümleler ve mükemmel dilbilgisi gereksizdir.”
   - **Kaçınılması Gerekenler:** “Gereksiz ayrıntıları, arka plan bilgilerini ve yorumları göz ardı et.” “Kendi analizlerini veya görüşlerini dahil etme.”
   - Kısıtlamalar, modelin istenmeyen davranışlardan kaçınmasını ve belirli kurallara uymasını sağlar.
6. **Yetenekler ve Hatırlatıcılar (Capabilities and Reminders)**:
   - **Araç Erişimi:** “Arama terimiyle ilgili son haber makalelerini bulmak ve almak

için web arama aracına erişimin var.”
    - **Tarih Bilinci:** “Haberlerin uygunluğunu sağlamak için güncel tarihten derinlemesine haberdar olmalısın.”
    - **Zaman Kısıtı:** “Yalnızca son 10 gün içinde yayınlanan bilgileri özetle.”
    - Bu bölüm, ajanın kullanabileceği araçları ve görevini yerine getirirken dikkat etmesi gereken önemli noktaları belirtir.

Bu istem örneği, Bağlam mühendisliğinin **yalnızca kullanıcıya yönelik bir prompt yazmakla kalmayıp**, aynı zamanda **ajanın sistemik davranışını ve etkileşimlerini mimarileştirme** sanatını nasıl içerdiğini gösterir. İstemdeki detay ve yapı, ajanın otonom bir şekilde karmaşık bir görevi verimli ve tutarlı bir şekilde tamamlaması için gereklidir. Bu yaklaşım, YZ ajanlarının belirsizliği azaltarak daha doğru ve faydalı yanıtlar üretmesine olanak tanır.

1.6 Bağlam Mühendisliği Geliştiriciler ve Son Kullanıcılar için Ne Anlama Geliyor?

Bağlam mühendisliği, geniş dil modelleri (BBD) ve yapay zeka (YZ) ajanlarıyla etkileşim kurma yönteminde önemli bir evrimi temsil etmektedir. Bu kavram, **doğru bilgiyi, doğru formatta ve doğru zamanda** bir BBD’ye sağlayarak belirli bir görevi başarıyla tamamlamasını sağlamak sanatı ve bilimidir. Bu yaklaşım, sadece tek bir istemle sınırlı kalmayıp, **dinamik ve karmaşık sistemlerin** tasarlanması ve inşa edilmesi üzerine odaklanır.

- **Geliştiriciler ve Yapay Zeka Mühendisleri İçin Bağlam Mühendisliği**

Geliştiriciler ve YZ mühendisleri için bağlam mühendisliği, özellikle YZ uygulamaları ve ajanları inşa ederken **kritik bir beceri** haline gelmiştir. Bunun nedenleri ve anlamı şunlardır:

- **Karmaşık YZ Uygulamaları ve Ajanları İnşa Etme:** Bağlam mühendisliği, özellikle müşteri hizmetleri asistanları, satış asistanları veya kodlama asistanları gibi **uzun soluklu ve yüksek karmaşıklıkta görevleri** yerine getiren YZ ajanları inşa etmek için gereklidir. Bu ajanlar, faturalandırma sorunları, iade sorunları, giriş sorunları gibi çeşitli sorguları ele almak ve gerektiğinde insanlara aktarım yapmak gibi birçok senaryoyu yönetme yeteneğine sahip olmalıdır.
- **Sistem Tasarımı ve Mimari:** Geliştiriciler, BBD’yi “işletim sistemi” olarak, bağlam penceresini ise “RAM” veya “çalışma belleği” olarak düşünebilir. Bağlam mühendisliği, bu BBD’nin belleğine neyin sığdırılması gerektiğine karar verme disiplinidir. Bu, yalnızca kullanıcı istemlerini yazmaktan çok, **ajanın sistemik davranışını ve etkileşimlerini mimarileştirme** sanatını içerir.
- **Talimat Kılavuzu Oluşturma:** YZ ajanları; model, araçlar, bilgi ve hafıza, ses ve

konuşma, koruma kalkanları (guardrails) ve orkestrasyon gibi altı temel bileşene sahiptir. Bağlam mühendisleri, tüm bu bileşenlerin bir araya nasıl geleceğini, araçların nasıl kullanılacağını, hafızaya nasıl erişileceğini, bilgi tabanının ne içerdiğini ve konuşma işlevlerinin ne zaman kullanılacağını **detaylandıran bir talimat kılavuzu** (istem) oluşturur.

- **Artan İstem Karmaşıklığı:** YZ ajanları için verilen talimatlar ve kaynaklar büyüdükçe, istemler de giderek daha karmaşık hale gelir ve hatta XML etiketleri ve Markdown gibi **kod benzeri yapılar** içerebilir.
- **Bağlam Yönetimi Stratejileri:** Ajanlar daha uzun görevleri ele aldıkça ve araç çağrıları kullandıkça, bağlam penceresinde **token birikimi** (token bloat) yaşanabilir. Bu durum, modelin **bağlam zehirlenmesi (context poisoning), dikkat dağınıklığı (distraction), karışıklık (confusion)** veya **çatışma (clash)** gibi hatalar yapmasına yol açabilir. Geliştiriciler, bu sorunları çözmek için çeşitli bağlam mühendisliği stratejileri kullanır:
  - **Bağlam Yazma (Writing Context):** Bilgiyi bağlam penceresinin dışında kaydetme (örneğin not defterleri veya hafıza sistemleri aracılığıyla), böylece ajan daha sonra bu bilgilere başvurabilir.
  - **Bağlam Seçme (Selecting Context):** İlgili bilgileri harici kaynaklardan veya farklı bellek türlerinden (örneğin, az sayıda örnek, olgular, araç açıklamaları) bağlam penceresine çekme. Bu, özellikle Bilgi Geri Çekmeyle Zenginleştirme (RAG) uygulamalarında önemlidir.
  - **Bağlam Sıkıştırma (Compressing Context):** Görevi yerine getirmek için gerekli en alakalı token'ları koruma, gereksiz veya yedekli bilgiyi özetleme veya budama (trimming) yoluyla azaltma.
  - **Bağlam İzole Etme (Isolating Context):** Bağlamı farklı ortamlar veya ajanlar arasında bölme. Çoklu ajan sistemleri, her alt ajanın kendi bağlam penceresine sahip olmasıyla sistemin genel token işleme kapasitesini artırır.
- **Olasılıksal Bağlamı Şekillendirme:** YZ ajanlarının web erişimi veya diğer harici veri kaynaklarına bağlanması durumunda, deterministik (doğrudan kontrol edilebilen) bağlamın yanı sıra **olasılıksal (non-deterministic) bağlam** da devreye girer. Bu, YZ'nin web'de veya büyük iç veri yapılarında bilgi ararken karşılaştığı çok daha geniş bilgi hacmini ifade eder. Geliştiricilerin görevi, bu olasılıksal bağlamı doğru istemlerle şekillendirmek ve çıkan bilgi kaynaklarının kalitesini izlemektir. Güvenlik (LLM enjeksiyon saldırıları) ve veri kaynaklarının kalitesini ölçme de bu kapsamda önemlidir.
- **Son Kullanıcılar İçin Bağlam Mühendisliği**

Son kullanıcılar için bağlam mühendisliği terimi, YZ mühendisleri kadar teknik derinliğe sahip olmasa da, günlük YZ etkileşimlerinde giderek daha fazla önem kazanmaktadır.

- **Model Seçimi:** Kullanıcılar, farklı bilgi türlerini daha iyi işleyebilen modelleri seçerek (örneğin, çok fazla bağlamı işlemek için optimize edilmiş GPT-4 Pro gibi) bağlam mühendisliği yapabilirler.
- **Doğru Bilgiyi Sağlama:** YZ sohbet botları veya uygulamalarıyla etkileşim kurarken, kullanıcıların doğru, yeterli ve ilgili bilgiyi sağlaması, modelin daha akıllı ve doğru yanıtlar üretmesi için hayati önem taşır. Bu, bir isteme ek dosyalar eklemek veya bir

konuyu ayrıntılı bir şekilde açıklamak anlamına gelebilir.

- **Büyük Resmi Anlama:** Bir YZ'den yardım istendiğinde, modelin "büyük resmi" anlamasını sağlamak, ondan maksimum verim almayı sağlar. Tıpkı bir insandan yardım istenirken ona işin arka planının iyi anlatılması gibi, YZ'ye de bağlam sağlamak önemlidir. Bu, yalnızca tek bir basit soru sormak yerine, konuya dair tüm ilgili arka plan bilgilerini sağlamayı içerir.
- **İstemin Ötesi:** Başlangıçta sadece "istem mühendisliği" (prompt engineering) olarak adlandırılan ve kullanıcıların doğru soruyu sormaya odaklandığı bu alan, YZ'ler daha akıllı hale geldikçe ve daha geniş bağlamlara ihtiyaç duydukça, "bağlam mühendisliği" olarak adlandırılan daha geniş bir disipline dönüşmüştür. Kullanıcılar, istemlerinin yanı sıra sisteme sağladıkları diğer bilgilerle (kurallar, yüklenen belgeler, geçmiş konuşmalar, araç çıktıları gibi) modelin davranışını ve çıktısını etkilerler.

Özetle, bağlam mühendisliği, geliştiriciler için **YZ ajanlarının ve sistemlerinin performansı, ölçeklenebilirliği ve güvenilirliğini sağlayan temel bir sistem mimarisi ve yönetim becerisi** iken, son kullanıcılar için **YZ araçlarından en iyi sonuçları elde etmek adına modele sunulan bilgi kalitesini ve miktarını bilinçli olarak yönetme** anlamına gelmektedir.

1.7 Yapay Zeka Ajanları Nelerdir ve Temel Bileşenleri Nelerdir?

Yapay Zeka Ajanları, kullanıcılar adına belirli **hedefleri takip etmek ve görevleri tamamlamak için yapay zekayı kullanan yazılım sistemleridir**. Bu ajanlar, insanlarla doğal bir şekilde etkileşim kurabilir ve çeşitli karmaşık görevleri yerine getirebilir.

Örneğin, bir **müşteri hizmetleri yapay zeka ajanı** müşteri sorgularını yanıtlayabilir, **satış asistanı yapay zeka ajanı** potansiyel müşterileri belirleyip takip edebilir veya bir **kodlama ajanı** kod yazımında yardımcı olabilir. Bu ajanlar, uzun soluklu ve daha yüksek karmaşıklıkta görevleri yerine getirebilir ve araç çağırma yeteneğini kullanabilirler. Faturalandırma sorunları, iade sorunları, giriş sorunları gibi çeşitli sorguları ele alabilir ve gerektiğinde bir insana aktarım yapabilirler.

Yapay Zeka Ajanlarının Temel Bileşenleri:

Bir yapay zeka ajanını oluşturan altı temel yapı taşı bulunmaktadır. Bu bileşenler, bir burgerin farklı kısımları gibi düşünülebilir; tıpkı bir burgerin ekmek, köfte ve sebzelerden oluşması gerektiği gibi, bir YZ ajanının da belirli bileşenlere sahip olması gerekir.

1. **Model (Model):** Her yapay zeka ajanı bir yapay zeka modeline ihtiyaç duyar. Bu model; GPT modelleri, Claude, Gemini gibi büyük modeller olabileceği gibi daha küçük veya açık kaynaklı modeller de olabilir.
2. **Araçlar (Tools):** Araçlar, ajanların harici sistemlerle etkileşim kurmasını sağlar.

Örneğin, kişisel bir asistan ajanı, randevu ayarlamak için Google Takviminize erişim sağlayan bir araca sahip olmalıdır. Bir YZ ajanı web üzerinde tarama yapabilir veya belirli bir veri toplamak için kod yazabilir. Araç çağrılarından gelen geri bildirimler bağlam penceresinde birikebilir, bu da daha fazla token kullanımına yol açar. Ajanların çok sayıda aracı aynı anda yönetmekte zorlandığı, hatta 30'dan fazla araçla performansın düştüğü ve 100 araçla tamamen başarısız olduğu gözlemlenmiştir. Bu durum, araç açıklamaları üzerinde bilgi geri çekmeyle zenginleştirme (RAG) kullanarak yalnızca ilgili araçları seçme gibi yöntemlerle iyileştirilebilir.

3. **Bilgi ve Hafıza (Knowledge and Memory):** Çoğu ajan, bilgiyi depolamak ve gerektiğinde geri çağırmak için bir yönteme ihtiyaç duyar. Örneğin, bir terapi ajanı, önceki konuşmalarda söylenenleri hatırlamak için hafızaya sahip olmalıdır. Bir hukuk ajanı ise belirli vakaları incelemek için ilgili bilgi tabanına erişebilmelidir. Hafıza sistemleri, ajanların not almasını ve gelecekteki ilgili görevler için hatırlamasını sağlar. Kısa vadeli bellek (scratchpad) tek bir oturum içinde bilgi saklarken, uzun vadeli bellek (long-term memory) ajanların farklı oturumlar arasında bilgiyi korumasına olanak tanır.
4. **Ses ve Konuşma (Audio and Speech):** Bir yapay zeka ajanını ses ve konuşma yeteneğiyle donatmak, onunla etkileşimini daha doğal ve kolay hale getirir.
5. **Koruma Kalkanları (Guardrails):** Bunlar, yapay zeka ajanının uygun şekilde davranmasını sağlayan güvenlik mekanizmalarıdır. Örneğin, bir müşteri hizmetleri ajanının kullanıcılara küfretmesini istemezsiniz.
6. **Orkestrasyon (Orchestration):** Orkestrasyon, yapay zeka ajanlarını dağıtmanıza, izlemenize ve zamanla iyileştirmenize olanak tanıyan sistemlerdir.

Bu altı bileşen bir araya getirilirken, bir **bağlam mühendisi** tüm bunların nasıl birlikte çalışacağını ayrıntılandıran bir "talimat kılavuzu" (istem) oluşturur. Bu talimat kılavuzu, araçların nasıl kullanılacağını, hafızaya nasıl erişileceğini, bilgi tabanının ne içerdiğini ve konuşma işlevlerinin ne zaman kullanılacağını belirtir. Ajanlar daha karmaşık hale geldikçe, bu talimatlar XML etiketleri ve Markdown gibi kod benzeri yapılar içerecek şekilde büyüyebilir.

Andre Karpathy'nin benzetmesine göre, büyük dil modeli (BBD) bir **CPU** gibidir ve bağlam penceresi (context window) onun **RAM'i veya çalışma belleğidir**. Tıpkı bir işletim sisteminin RAM'e neyin sığacağını düzenlemesi gibi, bağlam mühendisliği de BBD'nin bağlam penceresine neyin sığdırılması gerektiğine karar verme disiplinidir. Bu "RAM"in sınırlı bir kapasitesi olduğundan, doğru bilginin doğru zamanda ve doğru formatta sağlanması kritiktir. Ajanların uzun süreli görevleri ve araç çağrılarını kullanmaları nedeniyle bağlam pencerelerinde token birikimi (token bloat) yaşanabilir, bu da modelin hatalar yapmasına yol açabilir. Bu durumları yönetmek için bağlam yazma, bağlam seçme, bağlam sıkıştırma ve bağlam izole etme gibi çeşitli bağlam mühendisliği stratejileri kullanılır.

1.8 Pratik Uygulamalar ve Gelecek

Bağlam mühendisliği, yapay zeka (YZ) uygulamaları ve özellikle YZ ajanlarının geliştirilmesinde temel bir beceri haline gelmiştir ve hem mevcut pratik uygulamaları hem de gelecekteki rolü açısından büyük önem taşımaktadır.

- **Pratik Uygulamalar**

Bağlam mühendisliği, çeşitli YZ uygulamalarında ve ajanlarında halihazırda kullanılmaktadır:

- **Yapay Zeka Ajanları Geliştirme:** Bağlam mühendisliği, **karmaşık ve uzun soluklu görevleri** yerine getiren YZ ajanlarının inşası için hayati öneme sahiptir. Bu ajanlar şunları içerebilir:
  - **Müşteri Hizmetleri Ajanları:** Faturalandırma sorunları, iade sorunları, giriş sorunları gibi çeşitli müşteri sorgularını ele alabilir ve gerektiğinde insanlara aktarım yapabilir.
  - **Satış Asistanları:** Potansiyel müşterileri belirleyip takip edebilir.
  - **Kodlama Asistanları/Ajanları:** Kod yazma, hata ayıklama, test yazma, yeniden düzenleme ve karmaşık sistemlerde gezinme konularında yardımcı olur. Projenin amacı, kod ve dosya yapısı, kodlama standartları ve proje mimarisi gibi bilgileri anlayarak projenin yapısına uyumlu geliştirme yapar.
  - **Araştırma Asistanları:** Kullanıcı sorgularını eyleme dönüştürülebilir alt görevlere ayırır, çeşitli kaynak türlerinden (web, haber bültenleri, Reddit gibi) bilgi toplar, bu bilgileri önceliklendirir, belirli bir çıktı formatında (örneğin JSON) sunar ve bulguları özetler.
  - **Terapi veya Hukuk Ajanları:** Önceki konuşmaları hatırlamak için hafızayı veya belirli vakaları incelemek için bilgi tabanlarını kullanır.
  - **Genel Amaçlı Otonom Ajanlar:** Muhakeme, planlama ve eylem için bağlamı kullanır.
- **Bellek Yönetimi ve Bilgi Geri Çekimi (RAG):**
  - Ajanların not almasını (scratchpad) ve gelecekteki ilgili görevler için bu bilgileri hatırlamasını sağlar. Bu, tek bir oturum içindeki kısa vadeli bellek veya farklı oturumlar arası bilgiyi koruyan uzun vadeli bellek aracılığıyla yapılabilir.
  - Harici kaynaklardan (vektör veri tabanları, bilgi grafikleri) ilgili bilginin geri çekilmesi (Retrieval Augmented Generation – RAG) için kullanılır. Bu, BBD'nin bilgi tabanını genişletmeye ve modellerin mevcut bilgiyle sınırlı kalmamasını sağlamaya yardımcı olur.
- **Araç Entegrasyonu:** Ajanların harici sistemlerle (Google Takvim, web tarayıcıları, API'ler) etkileşim kurmasını sağlar. Bu, ajanların çok sayıda aracı verimli bir şekilde yönetebilmesi için araç açıklamaları üzerinde RAG kullanarak yalnızca ilgili araçları seçmeyi de içerir.
- **Eğitim Uygulamaları:** OpenAI'nin "Study Modu" gibi özellikler, kullanıcıların bilgi düzeyine göre dinamik öğrenme planları oluşturur, sorularla yönlendirir, karmaşık problemleri yönetilebilir parçalara böler ve kişiselleştirilmiş, destekleyici bir öğrenme

ortamı sunar.
- **Sistem Tasarımı:** Mühendisler için, YZ ajanlarının performansı, ölçeklenebilirliği ve güvenilirliği için temel bir sistem mimarisi ve yönetim becerisi olarak önem taşır.
- **Gelecek**

Bağlam mühendisliğinin geleceği, YZ'nin evrimiyle yakından ilişkilidir ve birkaç önemli yönde gelişmesi beklenmektedir:

- **Prompt Mühendisliğinin Evrimi ve Yeni Bir Disiplin:** Bağlam mühendisliği, prompt mühendisliğinin **daha karmaşık YZ uygulamaları ve ajanları** geliştirme bağlamındaki doğal bir ilerlemesi olarak ortaya çıkmıştır. Artık YZ topluluğunda "yapay zeka mühendislerinin bir numaralı işi" olarak görülmektedir. Silikon Vadisi'ndeki iş sektöründe "prompt mühendisliği" yerine "bağlam mühendisliği" aramalarının arttığı belirtilmektedir.
- **Dinamik ve Probabilistik Bağlam Yönetimi:** Gelecekte, büyük dil modelleri (BBD) web erişimi ve geniş veri kaynaklarıyla daha fazla etkileşime girdikçe, **olasılıksal (probabilistic) bağlamın** yönetimi giderek daha kritik hale gelecektir. Bu, sadece deterministik (doğrudan kontrol edilebilir) bağlamı değil, aynı zamanda ajanın dış dünyadan çektiği bilgi miktarını ve kalitesini de şekillendirmeyi gerektirecektir. Bu durum, YZ modellerinin web'den veya şirket içi büyük veri yapılarından bilgi ararken ortaya çıkabilecek güvenlik tehditleri (LLM enjeksiyon saldırıları) ve bilgi kaynaklarının kalitesinin izlenmesi gibi konuları da beraberinde getirecektir.
- **Ölçeklenebilirlik ve Tutarlılık:** Bağlam mühendisliği, BBD'yi daha verimli kullanarak **ölçeklenebilir, tutarlı ve güvenilir** yapay zeka geliştirme süreçleri kurmak için kaçınılmaz bir yaklaşım haline gelmektedir.
- **"Süper Uygulama" Trendi:** OpenAI gibi platformların "Study Modu" gibi özelliklerle eğitim sektörüne girmesi, ChatGPT'nin çeşitli dijital endüstrileri (sağlık, haber analizi vb.) kendi içine çeken bir "süper uygulama" olma potansiyelini göstermektedir. Bu durum, küçük girişimciler için ilgili alanlarda rekabet etmeyi zorlaştırabilir.
- **Çoklu Ajan Sistemleri:** Bağlam mühendisliği, çoklu ajan sistemlerinin (farklı alt ajanların paralel çalışarak her birinin kendi bağlam penceresine sahip olması) geliştirilmesini destekleyerek sistemin genel token işleme kapasitesini artırır ve daha zengin raporlar üretilmesine olanak tanır. Ancak, bu sistemlerde ajanslar arası tutarlılık ve çakışma riskleri dikkatle yönetilmelidir.
- **Son Kullanıcılar İçin Önem:** Yalnızca YZ mühendisleri için değil, son kullanıcılar için de YZ araçlarından en iyi sonuçları elde etmek adına, **doğru miktarda ve kalitede bağlamı** bilinçli olarak yönetmek giderek daha önemli hale gelecektir. Bu, doğru model seçimi ve modele verilecek bağlam türü gibi kararları içerir.
- **Sürekli Gelişim:** Bağlam mühendisliği alanı hala gelişmekte olan bir disiplindir ve bağlam yazma, bağlam seçme, bağlam sıkıştırma ve bağlam izole etme gibi yeni stratejiler sürekli olarak ortaya çıkmaktadır. Bu beceriler, mevcut yazılım mühendisliği pozisyonlarının (YZ Mühendisi, YZ Çözüm Mimarı, YZ Ürün Yöneticisi) yapay zeka alanındaki bilgiyi içerecek şekilde evrilmesini tetiklemektedir.

1.9 Sonuç

Bağlam mühendisliği, LLM'lerden en iyi performansı elde etmek için prompt mühendisliğinin ötesine geçen, sistem düzeyinde bir yaklaşımdır. AI ajanları ve daha karmaşık AI uygulamaları geliştikçe, doğru, ilgili ve iyi yapılandırılmış Bağlami dinamik olarak sağlama yeteneği, ortalama çıktı ile olağanüstü performans arasındaki farkı yaratacaktır. Bu yeni alan, yazılım mühendisliği, veri mühendisliği ve sistem mimarisi becerilerini bir araya getiren kapsamlı bir disiplin olarak ortaya çıkmaktadır.

1.10 Anahtar Terimler Sözlüğü

- **Ajan (Agent):** Yapay zekayı kullanarak kullanıcılar adına hedefleri takip eden ve görevleri tamamlayan bir yazılım sistemi.
- **Bağlam (Context):** Büyük Dil Modelinin (LLM) bir yanıt üretmeden önce gördüğü tüm bilgi (istemler, belgeler, bellek, araç çıktıları, yapılandırılmış veri vb.).
- **Bağlam Penceresi (Context Window):** Bir LLM'nin belirli bir zamanda işleyebileceği veya "görebileceği" token sayısı sınırı, modelin çalışma belleği.
- **Bağlam Zehirlenmesi (Context Poisoning):** LLM'nin bağlamında halüsinasyon veya hataların tekrarlanarak referans alınması, yanıltıcı sonuçlara yol açması.
- **Bağlamda Dağınıklık (Context Distraction):** Bağlamın o kadar uzun olması ki modelin bağlama aşırı odaklanması ve yeni veya farklı eylemler üretme yeteneğini kaybetmesi.
- **Bağlam Karmaşası (Context Confusion):** Bağlamdaki gereksiz veya ilgisiz içeriğin model tarafından düşük kaliteli yanıtlar üretmek için kullanılması, özellikle çok sayıda aracın sunulduğu durumlarda.
- **Bağlam Çatışması (Context Clash):** Bağlama eklenen yeni bilgi veya araçların, bağlamdaki mevcut bilgilerle doğrudan çelişmesi.
- **Bağlam Mühendisliği (Context Engineering):** Bir LLM'e görevi yerine getirmesi için doğru bilgiyi, doğru formatta ve doğru zamanda sağlamak üzere dinamik sistemler tasarlama ve inşa etme sanatı ve bilimi.
- **Bağlam İzolasyonu (Context Isolation):** Ajanın performansını artırmak için farklı görevler veya bilgi alanlarını ayrı bağlam pencerelerine ayırma stratejisi.
- **Bağlam Sıkıştırma (Context Compression):** Bağlam penceresi sınırları içinde kalmak ve verimliliği artırmak için bilgiyi özetleme veya kırpma yoluyla yoğunlaştırma.
- **Bağlam Yazma (Writing Context):** Ajanın görevi yerine getirmesine yardımcı olmak için bilgiyi bağlam penceresinin dışına kalıcı olarak kaydetme.
- **Bağlam Seçimi (Selecting Context):** Bir ajanın görevi için en ilgili ve gerekli bilgiyi seçici olarak bağlam penceresine çekme.
- **Deterministik Bağlam (Deterministic Context):** Kullanıcının doğrudan kontrol edebildiği ve belirleyebildiği bağlam (örn. statik istemler, bilgi tabanları, belgeler).
- **Geri Alma Destekli Üretim (Retrieval Augmented Generation – RAG):** Büyük veri tabanlarından ilgili bilgiyi seçici olarak çekerek bir LLM'nin yanıtlarını zenginleştirme

tekniği.

- **Guadrialler (Guardrails):** Bir yapay zeka ajanının uygun davranışını sağlamak için tasarlanmış güvenlik mekanizmaları veya kısıtlamaları.
- **Halüsinasyon (Hallucination):** Bir LLM'nin gerçek olmayan veya yanıltıcı bilgi üretmesi.
- **JSON (JavaScript Object Notation):** Yapılandırılmış veriyi insanlar tarafından okunabilir ve makineler tarafından kolayca ayrıştırılabilir bir formatta temsil etmek için kullanılan hafif bir veri değişim formatı.
- **Kırpma (Pruning):** Bağlamdan ilgisiz veya gereksiz bilgi parçalarını seçici olarak kaldırma tekniği.
- **LangChain:** LLM destekli uygulamalar ve ajanlar geliştirmek için bir çerçeve.
- **LangGraph:** LangChain'in ajanlar oluşturmak için kullanılan düşük seviyeli bir orkestrasyon çerçevesi.
- **LLM (Large Language Model – Büyük Dil Modeli):** Geniş veri kümeleri üzerinde eğitilmiş, insan dilini anlama ve üretme yeteneğine sahip bir yapay zeka modeli.
- **Bellek (Memory):** Bir ajanın geçmiş etkileşimlerden veya harici kaynaklardan bilgiyi depolama ve geri alma yeteneği (kısa veya uzun vadeli olabilir).
- **Model Bağlam Protokolü (Model Context Protocol – MCP):** LLM'lerin harici sistemlere veya veri tabanlarına erişimini sağlayan protokoller.
- **Çoklu Ajan Sistemleri (Multi-Agent Systems):** Tek bir karmaşık görevi tamamlamak için işbirliği yapan birden fazla bağımsız veya birbiriyle ilişkili yapay zeka ajanı grubu.
- **Orkestrasyon (Orchestration):** Bir yapay zeka ajanının bileşenlerini (model, araçlar, bellek vb.) yönetme, dağıtma, izleme ve iyileştirme süreçleri.
- **Özetleme (Summarization):** Bağlamdaki bilgiyi daha kısa ve yoğunlaştırılmış bir özete dönüştürme tekniği.
- **Probabilistik Bağlam (Probabilistic Context):** LLM'nin web aramaları veya geniş dahili veri tabanları gibi dış kaynaklardan dinamik olarak elde ettiği, tam olarak kontrol edilemeyen bağlam.
- **Prompt (İstem):** Bir LLM'ye belirli bir görev veya yanıt almak için verilen metin tabanlı talimat veya soru.
- **Prompt Mühendisliği (Prompt Engineering):** Bir LLM'den istenen yanıtı almak için istemleri dikkatlice tasarlama ve optimize etme disiplini.
- **RAM (Random Access Memory):** Bilgisayarda kısa vadeli veri depolama için kullanılan bir tür bellek, LLM'nin bağlam penceresine benzetilir.
- **Karalama Defteri (Scratchpad):** Bir ajanın bir görevi yerine getirirken geçici notlar alması veya bilgiyi kalıcı hale getirmesi için kullanılan bir mekanizma.
- **Sistem İstemi (System Prompt):** Bir LLM'nin rolünü, davranışını ve kısıtlamalarını tanımlayan, genellikle kullanıcının isteminden önce gelen genel talimatlar.
- **Durum Nesnesi (State Object):** Bir ajanın oturumu boyunca kalıcı olan ve farklı bileşenlerin bilgi okuyup yazabileceği merkezi bir veri yapısı (LangGraph'ta kullanılır).
- **Token:** Bir LLM'nin metni işlerken kullandığı en küçük dil birimi (kelimelerin parçaları, harfler, noktalama işaretleri vb.).
- **Araçlar (Tools):** Bir LLM'nin dış dünyayla etkileşim kurmasını sağlayan işlevler veya

API’lar (örn. web arama, hesap makinesi, takvim).

- **Araç Yüklemesi (Tool Loadout):** LLM’ye sunulan araç setini göreve göre seçici olarak daraltma tekniği.
- **XML (Extensible Markup Language):** Bilgiyi yapılandırılmış bir formatta kodlamak için kullanılan bir işaretleme dili.