

Self-Adapting Language Models (Öz-Uyumlu Dil Modelleri) İncelemesi

Giriş: Makalenin Hedefi ve Önemi

Büyük dil modelleri (Large Language Models, **LLM**) günümüz yapay zeka uygulamalarında çok güçlüdür, ancak **statik** yapıda olmaları önemli bir sınırlamadır. Eğitimleri tamamlandıktan sonra ağırlıkları sabitlenir ve yeni görevler, bilgiler veya örneklerle karşılaştıklarında kendi kendilerine uyum sağlayamazlar ¹. Bu durum, bir modelin güncellenmesi gerektiğinde genellikle yeniden eğitime veya ek ince ayar (finetuning) gibi zahmetli işlemlere başvurmayı gerektirir. Alternatif bir yaklaşım olarak bağlamsal öğrenme (prompt'lar ile *in-context learning*) veya harici bilgi çağırma (Retrieval-Augmented Generation, RAG) kullanılabilir; ancak bunlar da model ağırlıklarını kalıcı olarak değiştirmez ve her yeni bağlam için yeniden uğraş ister ².

Self-Adapting Language Models (SEAL) adlı çalışma, bu soruna yenilikçi bir çözüm getiriyor. Makalenin hedefi, LLM'lere kendi ağırlıklarını **öz uyumlu (self-adapting)** şekilde güncelleyebilme yeteneği kazandırmaktır ¹. Yani model, karşılaştığı yeni bir görev veya bilgi karşısında **kendi ürettiği** ek eğitim verileri ve güncelleme talimatları ile kendi kendini ince ayar yapabilir. Bu sayede model, dağıtımdaki değişimlere, yeni bilgilere veya örneklerle **insan müdahalesi olmadan ve sürekli biçimde uyum sağlayabilecek** duruma gelir ³ ². Bu yaklaşım, her uyarılma için büyük miktarda etiketli veriye ve uzun süren insanlı eğitim süreçlerine olan ihtiyacı azaltmayı vaat eder ³. Sonuç olarak SEAL, LLM'lerin **dinamik ve otonom** bir şekilde kendilerini geliştirebilmeleri için atılmış önemli bir adımdır.

Model Mimarisi ve Varsayımlar

Makalenin önerdiği yaklaşımda kullanılan model mimarisi, halihazırda var olan bir **dönüştürücü (Transformer)** tabanlı dil modelidir. Yazarlar deneylerde açık kaynak bir LLM'yi (Alibaba'nın Qwen modeli, yaklaşık birkaç milyar parametre ölçeğinde bir model) temel almıştır ⁴. Önemle belirtmek gerekir ki, SEAL yöntemini uygulamak için modelin mimarisinde köklü bir değişiklik yapılmamıştır; modelin normal metin üretme ve eğitilebilir olma özellikleri üzerine kuruludur. Varsayımlar şunlardır:

- **Eğitilebilir Ağırlıklar:** Modelin ağırlıkları gerektiğinde güncellenebilir olmalıdır. SEAL yaklaşımı, modelin ağırlıklarını tamamen sabit tutan *prompt tuning* gibi yöntemlerin aksine, model parametrelerini değiştirmeye izin verir. Yani model, yeni veri ile **denetimli ince ayar (SFT)** yapıldığında performansını artıracak şekilde öğrenmeye devam edebilir ¹.
- **Metin Üretim Yeteneği:** Model, girdi olarak bir görev tanımı veya yeni bilgi verildiğinde, çıkış olarak anlaşılabilir metin (doğal dil) üretebilmelidir. Özellikle, SEAL'de modelin kendi kendine bir *düzenleme komutu* üretebilmesi kritik. Bu komutlar doğal dil formunda olacağından, modelin metin üretim kabiliyeti bu yaklaşımın temelini oluşturur ⁵.
- **Hızlı İnce Ayar İmkânı:** Modelin, ürettiği yeni verilere dayanarak hızlı bir şekilde eğitilebilmesi (ince ayar yapılabilmesi) gerekir. SEAL döngüsünde her *öz-düzenleme* sonrası modelde küçük bir fine-tuning adımı yapılır. Bu nedenle sistem, bu sık yinelemeli güncellemeleri kaldırabilecek bir altyapıya (örneğin uygun donanım, verimli eğitim kodu) sahip olmalıdır ⁶. Makalenin yazarları,

deneylerini 2 adet A100/H100 GPU ile gerçekleştirilebildiklerini belirtmişlerdir; farklı donanımlarda model boyutunu küçültmek veya ayarlamak gerekebilir ⁶ .

Bu varsayımlar altında, SEAL herhangi bir özel ek modüle ihtiyaç duymadan, standart bir LLM'nin kendi metin üretimini ve eğitilebilirliğini kullanarak çalışır. Önemli bir fark, önceki bazı uyarlamalı yöntemlerde görülen ayrı bir “uyarlama ağı” veya harici modül kullanımının burada olmamasıdır. SEAL, **harici bir yardımcı ağa ihtiyaç duymaksızın**, doğrudan modelin kendi ürettiği içerikle kendi ağırlıklarını güncelleyebileceği fikrine dayanır ⁷ . Bu, yaklaşımı mimari açıdan sade tutar ve modelin **içsel generatif kapasitesini** sonuna kadar kullanır ⁸ .

Öz-Uyum (Self-Adaptation) Kavramı ve SEAL Yöntemi

Öz-uyum, bir dil modelinin *kendi kendini adapte etmesi* anlamına gelir. Teknik olarak bu, modelin yeni bir girdi veya görev karşısında **kendi ürettiği talimatlar ve veriyle** kendi ağırlıklarını güncellemesi sürecidir. SEAL çerçevesinde bu amaçla tanımlanan temel birim, **self-edit (öz-düzenleme)** adı verilen yapıdır.

Bir **self-edit**, model tarafından doğal dil biçiminde üretilen ve modelin nasıl güncellenmesi gerektiğini tarif eden bir çıktıdır ⁹ . Bu çıktı şunları içerebilir:

- Yeni gelen bilginin modelde içselleşmesine yardımcı olacak şekilde bilginin **yeniden yapılandırılması veya çıkarımı** (örneğin, verilen metinden önemli bir olgusal cümlenin türetilmesi),
- Modelin kendini eğitirken kullanacağı **hiperparametreler** için talimatlar (örneğin öğrenme oranı, kaç epoch eğitileceği gibi ayarlar),
- Gerekli ise veri artırımı (*data augmentation*) için bazı araçların çağırılması veya yeni örnekler üretilmesi için yönlendirmeler.

Bu self-edit çıktısı, bir bakıma modelin **“kendi kendine yazdığı eğitim notu”** gibidir. Model bu notu ürettikten sonra, sistem bu notu alır ve modelin üzerinde küçük bir **denetimli ince ayar (SFT)** uygulayarak modelin ağırlıklarını günceller ¹⁰ . Böylece model, kendi ürettiği veriyle gerçekten **kalıcı bir değişikliğe** uğrar – bu değişiklik modelin davranışında kalıcı bir adaptasyon demektir ¹¹ .

Not: Bu süreçte yapılan ince ayar, orijinal modele göre oldukça küçük bir güncellemedir ve sadece ilgili yeni bilgi/görev etrafında gerçekleşir. Bu sayede model tam bir yeniden eğitimden geçmez, yalnızca gerek duyulan kısmi uyarlamaları yapar.

SEAL Yönteminin Adımları

SEAL yönteminin çalışma prensibi bir **takviyeli öğrenme (reinforcement learning, RL)** döngüsü ile açıklanır ¹² . Her bir döngüde model hem bir “öğrenci” hem de kendi “öğretmeni” gibi davranır. Aşağıda SEAL’in RL döngüsünün adımlarını özetliyoruz:

Şekil 1: SEAL yönteminin genel döngüsü (RL dış döngüsü). Her bir iterasyonda model, bir görev bağlamına dayanarak öz-düzenleme (self-edit) önerileri üretir, bu öneriye göre kendi ağırlıklarını günceller (denetimli ince ayar ile) ve güncellenmiş modelin performansını ölçer. Sonuçlara göre başarılı öz-düzenlemeler pekiştirilir,

başarısızlar göz ardı edilir. Bu sayede model, hangi kendi kendine düzenlemelerin işe yarar olduğunu zamanla öğrenir ¹³ ¹⁴ .

1. **Girdi & Bağlam:** Her döngü başlangıcında modele yeni bir girdi verilir. Bu bir görev tanımı, bir soru veya modelin henüz öğrenmediği yeni bir bilgi olabilir. Örneğin, modele “X konusundaki bilgiyi öğren ve soruları cevapla” şeklinde bir bağlam verilebilir.
2. **Self-Edit Üretimi:** Model, verilen bu bağlamı dikkate alarak bir **self-edit (öz-düzenleme talimatı)** üretir. Bu talimat, modelin kendi ağırlıklarını nasıl değiştireceğine dair bir plandır. Örneğin, model yeni bir bilgi içeren bir paragraf aldıysa, self-edit olarak “Bu paragraftaki önemli gerçekleri çıkar ve bunları bilgin olarak içselleştir” gibi bir cümle üretebilir; ya da birkaç örnekle yeni bir görev verildiyse, self-edit olarak “eldeki az sayıda örneği çoğaltmak için şu stratejileri kullan ve eğitim parametrelerini şu şekilde ayarla” şeklinde bir yönerge verebilir ⁹ ¹⁵ .
3. **Ağırlık Güncellemesi (İnce Ayar):** Ardından, modelin ürettiği bu self-edit talimatı bir eğitim verisi olarak ele alınır. Sistemin dışındaki bir ince ayar mekanizması, modelin ağırlıklarını bu talimat doğrultusunda **denetimli olarak günceller (SFT uyguları)** ¹⁰ . Örneğin, eğer self-edit yeni bir bilgi cümlesi ürettiyse, model o cümleyi doğru kabul edecek şekilde kendini eğitir; veya self-edit hiperparametre önerdiyse, belirtilen hiperparametrelerle birkaç adım eğitim yapılır. Bu adım sonucunda modelin parametreleri güncellenmiş (θ şeklinde düşünelim) ve model yeni bilgiye uyum sağlamış hale gelir ¹⁶ ¹⁷ .
4. **Değerlendirme:** Güncellenen model, asıl görev üzerinde test edilir. Yani eğer amaç yeni öğrendiği bilgiyi soruları yanıtlama şeklinde göstermekse, model bu bilgiyi kullanarak bir soruya cevap üretir; ya da yeni bir görevde başarı göstermesi bekleniyorsa, güncellenmiş model o görevi yapmayı dener. Bu performans bir **ödül sinyali (reward)** olarak değerlendirilir ¹² . Örneğin, model soruya doğru cevap verirse yüksek ödül, yanlışsa düşük ödül alır. Bu ödül sayısal bir geri bildirimdir ve modelin ürettiği self-edit’in ne kadar etkili olduğunu yansıtır.
5. **Pekiştirme (RL Güncellemesi):** Son adımda, modelin ürettiği self-edit’in başarısına göre modelin **self-edit üretme politikası** güncellenir. Burada devreye RL girmektedir. Yüksek ödül getiren öz-düzenlemeler, modelin gelecekte benzer durumlarda benzer self-edit’ler üretme eğilimini artıracak şekilde pekiştirilir; düşük ödül getirenler ise modelin politikasından zayıflatılır ¹³ ¹⁸ . Makalede bunun için özel olarak **ReST-EM** adlı hafif bir RL algoritması kullanılmıştır ¹⁴ . ReST-EM, her iterasyonda **reddetme örnekleme (rejection sampling)** ile en yüksek ödül getiren denemeleri seçip, modelin bunları üretme kabiliyetini denetimli bir şekilde artırarak politikayı günceller ¹⁴ . Bu aslında RL’nin bir çeşidi olan *beklenti-azamiye (EM)* yaklaşımı ile entegre edilmiştir; yüksek başarılı örnekler üzerinden modelin davranışı EM mantığıyla iyileştirilmektedir.
6. **Tekrar:** Bu dış döngü, model yeni verilerle karşılaştıkça veya aynı veri üzerinde birkaç tur iyileştirme yapmak istendiğinde tekrarlanabilir. Her döngü sonunda model biraz daha iyi bir **öz-öğretmen** haline gelir; hangi kendi kendine düzenlemelerin işe yaradığını öğrenir. Bir süre sonra model, neredeyse içgüdüsel olarak faydalı self-edit’ler üretmeyi öğrenmiş olacaktır.

Bu sürecin en önemli yanı, **modelin uyum sağlama mekanizmasının bizzat model tarafından üretilmesidir**. Yani model, “nasıl güncellenmesi gerektiğini” de bir çıktı olarak verir hale getirilmiştir ⁷ . Önceki yaklaşımlarda genellikle modelin yanına ek bir modül konur (örneğin bir *meta-öğrenci* veya ayrı bir ağı), bu modül modelin güncellemelerini hesaplar veya önerirdi. SEAL’da ise böyle bir ayrı modül

yoktur; model **kendi içinden gelen talimatla** güncellenir ⁸. Bu, mimariyi basitleştirirken modele daha fazla **özerklik** kazandırır.

Kullanılan Veri Türleri, Örnek Senaryolar ve Deneysel Düzenekler

Makale, SEAL yöntemini iki farklı senaryoda test ederek değerlendiriyor: **(1) Bilgi Entegrasyonu (Knowledge Incorporation)** ve **(2) Az-Örnekli Öğrenme (Few-Shot Learning)**. Bu senaryolar hem kullanılan veri tipleri hem de deney düzenekleri açısından farklılık göstererek yöntemin esnekliğini sergiliyor ¹⁵.

1. Bilgi Entegrasyonu Senaryosu

Bu senaryoda amaç, modele **yeni olgusal bilgiler** verip bunları modelin kalıcı bilgisinin parçası haline getirmektir. Model, daha önce eğitimi sırasında görmediği bir metin pasajını giriş olarak alır. Bu pasaj, örneğin bir ansiklopedi paragrafı veya haber metni olabilir ve çeşitli **faktöel (olgusal) bilgi** içerir. Görev, modelin bu yeni bilgiler ışığında ilgili soruları cevaplayabilir hale gelmesidir – üstelik soru sorulduğunda orijinal pasajı tekrar görmeden, yani bilgiyi **içselleştirmiş** olarak ¹⁹.

Deney düzeneği şöyle kurulmuştur: Öncelikle modele içinde yeni bilgiler bulunan bir metin verilir (ör. Apollo programı hakkında bir paragraf). Model bu paragrafı alır ve bir **self-edit** üretir. Bilgi entegrasyonu için self-edit formatı, verilen pasajdan çıkarılan önemli olgusal cümleler veya **mantıksal sonuçlar** şeklindedir ¹⁵. Örneğin aşağıdaki görselde görüldüğü gibi model, Apollo programıyla ilgili pasajı okuduktan sonra kendisine "Kennedy'nin bilim danışmanı Jerome Wiesner, insanlı uzay uçuşlarına karşı çıktı" gibi bir cümle üretmiştir. Bu cümle, pasajdaki bilginin özünü yakalar ve modelin öğrenmesi gereken kritik noktayı temsil eder.

Şekil 2: Bilgi entegrasyonu senaryosu örneği ²⁰ ²¹. Solda modelin henüz bilmediği yeni bilgi içeren pasaj görülüyor. Model bu pasajı okuyarak ortada bir öz-düzenleme (self-edit) üretmiştir. Bu örnekte self-edit, pasajdaki önemli gerçeği vurgulayan sentetik bir cümledir (Kennedy'nin bilim danışmanının adı ve tutumu). Ardından model bu cümleyi kendi ağırlıklarına işler (denetimli ince ayar). Son olarak, sağda gösterildiği gibi, model güncellenmiş haliyle sorulan soruya pasaj olmadan doğru cevabı verebilmektedir. Bu, bilginin modelin parametrelerine başarıyla entegre edildiğini gösterir.

Self-edit üretildikten sonra model bu cümle ile hızlıca eğitilir (örneğin birkaç gradient adımı atılır). Sonrasında modelden, pasajı kullanmadan bir soru cevaplama istenir. Örneğimizde, "Kennedy'nin insanlı uzay uçuşlarına karşı çıkan bilim danışmanı kimdi?" sorusu sorulur; model, artık pasajdaki bilgiyi öğrendiğinden **"Jerome Wiesner"** cevabını kendi bilgisine dayanarak verebilir duruma gelmiştir. Bu şekilde, model pasajdaki bilgiyi parametrelerine *kalıcı olarak entegre etmiş* olur.

Deneyler iki alt durumda gerçekleştirilmiştir:

- **Tek Pasaj Entegrasyonu:** Model tek seferde bir adet yeni pasaj alarak kendini günceller ve sonrasında o pasajla ilgili soruları cevaplar. Bu ortam, modelin **anlık** bir bilgi parçasını özümseme yeteneğini ölçer.
- **Devamlı Entegrasyon (Çok Pasaj):** Model ardışık olarak birden fazla pasaj alır ve her defasında kendini günceller. Örneğin model sırayla 200 farklı metinle beslendiğinde, bu bilgilerden ne kadarını koruyabildiği ve ölçeklendikçe performansın ne olduğu incelenir ²². Bu, yöntemin sürekli öğrenme altında ölçeklenebilirliğini test eder.

Bilgi entegrasyonu senaryosu, SEAL'in **bilgiyi sonradan öğrenip adaptasyon** yapma kabiliyetini gösterir. Bu, pratikte bir LLM'nin yayınlandıktan sonra yeni ortaya çıkan gerçekleri öğrenmesi (örneğin yeni bilimsel bulgular, güncel olaylar) ve soruları bu yeni bilgilere dayanarak cevaplayabilmesi açısından çok değerlidir.

2. Az-Örnekli Öğrenme (Few-Shot) Senaryosu

Bu senaryoda model, çözmesi gereken yeni bir görevle karşılaşılıyor ancak elinde o göreve dair yalnızca **birkaç örnek (few-shot examples)** var. Normalde bir LLM, böyle durumlarda *in-context learning* yaparak, yani verilen birkaç örneği giriş prompt'u ile alarak soruları cevaplamaya çalışabilir. Fakat bağlam içi öğrenme, özellikle soyut ve karmaşık görevlerde yetersiz kalabilir – modelin aslında ağırlıklarında bu göreve özel bir uyarılama yoktur, sadece geçici olarak örnekleri taklit etmeye çalışır.

SEAL burada farklı bir yaklaşım sunuyor: Model, elindeki az sayıda örneği nasıl **genişleteceğine** ve bu örneklerle kendini nasıl eğiteceğine dair bir **self-edit** üretiyor ¹⁵. Few-shot senaryoda self-edit formatı, modelin kullanacağı **veri artırma stratejileri** ve **eğitim hiperparametreleri** şeklinde tasarlanmıştır ¹⁵. Başka bir deyişle, model örnekleri analiz ederek “Bu görevi çözmek için elimdeki az veriyi şöyle çoğalt, şu tür dönüşümleri uygulayarak yeni eğitim örnekleri türet, ve eğitim yaparken şu öğrenme oranını kullan, şu kadar epoch eğit” tarzında bir talimat listesi verir.

Şekil 3: Az-örnekli öğrenme senaryosu örneği ²³ ²⁴. Solda modelin çözmesi gereken görev için verilen yeterli sayıda örnek (few-shot examples) görülüyor. Model bu örneklerden yola çıkarak ortada bir öz-düzenleme (self-edit) oluşturuyor. Örnekte self-edit içinde veri artırma seçenekleri (ör. “basic_augmentations: true, chain_augmentations: false, ...”) ve eğitim hiperparametreleri (öğrenme oranı, epoch sayısı vb.) bulunuyor. Model, bu talimatlara uyarak mevcut birkaç örneği çeşitlendirilmiş bir eğitim setine dönüştürüyor ve kendini eğitiyor. Sağda, güncellenmiş modelin artık soyut mantık görevini başarıyla çözebildiği görülüyor.

Örneğin, yazarlar modelin **ARC (Abstraction and Reasoning Challenge)** adlı bir zor soyut düşünme görevini çözmesini istemişlerdir ²³ ²⁴. ARC görevleri genellikle insan mantık ve örüntü tanıma becerisine yakın problem setleridir ve klasik derin öğrenme yöntemleriyle çözülmesi zordur. Modele bu görevden sadece birkaç örnek verildiğinde (**few-shot examples**), SEAL yöntemiyle model şu şekilde çalışır:

- Model, örnekleri inceler ve “veri artırma” için neler yapılabileceğini çıkarır. Örneğin belki basit dönüşümler (rotate, flip gibi) kullanarak ek örnekler oluşturulabilir, veya mevcut örneklerin farklı varyasyonları düşünülebilir. Self-edit içerisinde model bu stratejileri belirtir (yukarıdaki görselde turuncu kutu içinde bu yönergeler listelenmiş). Ayrıca, model kendi eğitim sürecini ayarlamak için öğrenme oranı, kaç tur eğitim yapılacağı gibi hiperparametre önerileri de self-edit'te verir.
- Daha sonra modelin belirttiği veri artırma işlemleri uygulanarak yapay bir eğitim veri seti oluşturulur (örneğin birkaç puzzle'ın varyasyonu otomatik üretilir). Model, kendini bu üretilen verilerle belirtilen ayarlar doğrultusunda eğitir.
- Sonuç olarak model, artık elindeki kısıtlı bilgilere rağmen bu yeni görevde başarılı olacak şekilde ağırlıklarını ayarlamış olur.

Deneyde, ARC görevlerinden bir alt küme kullanılmış ve modelin başarı oranı ölçülmüştür. Bu senaryo, SEAL'in modelin kendi kendine **yeni stratejiler geliştirerek** öğrenebileceğini gösterir. Özellikle insan müdahalesi olmadan, modelin “şu parametrelerle kendimi eğitirsem daha iyi olur” diyebilmesi oldukça

çarpıcıdır. Bu yaklaşım, gelecekte modellerin çok az insan rehberliğiyle yeni yetenekler kazanabileceğini ima etmektedir.

Sonuçlar ve Kıyaslamalar: Yöntemin Avantajları

Yapılan deneyler, SEAL yönteminin her iki senaryoda da dikkate değer iyileştirmeler sağladığını göstermiştir. Bu iyileştirmeleri ve yöntemin avantajlarını, **diğer yaklaşımlarla kıyaslayarak** ele alalım:

- **Kalıcı ve Yüksek Başarımli Adaptasyon:** SEAL, modeli kalıcı olarak uyarladığı için sadece anlık performans artışı değil, *sürekli* bir yetenek kazanımı sağlar. Bilgi entegrasyonu deneyinde, **iki RL döngüsü** sonunda modelin yeni bilgiyle ilgili sorulardaki doğruluk oranı %32.7'den %47.0'ye yükselmiştir ²⁵. Bu, yaklaşık 15 puanlık büyük bir artıştır. Üstelik bu sonuç, aynı veriyi kullanarak harici yöntemlerle yapılan ince ayarlardan bile daha iyidir. Karşılaştırma için, sadece ham pasaj üzerinde doğrudan ince ayar yapan model ~%33 civarında kalırken, GPT-4.1 modelinin ürettiği sentetik verilerle eğitilen model %46.3 doğruluk elde etmiştir ²⁵. **SEAL (öz-uyum)** ise %47.0 ile hepsini geride bırakmıştır ²⁵. Bu, modelin kendi ürettiği eğitim verisinin kalitesinin, hatta GPT-4 gibi güçlü bir modelin ürettiği veriden bile yüksek olabileceğini gösterir. Ayrıca çoklu pasaj entegre edilen senaryoda da SEAL en yüksek performansa ulaşmıştır (%43.8) ²⁶, ki bu yöntem tek pasaj üzerinden RL ile eğitildiği halde, politika daha geniş veri dağılımına başarılı şekilde ölçeklenebilmiştir. **Özetle, SEAL ile model kendi kendine en uygun güncellemeleri bularak, mümkün olan en yüksek göreve özel performansa yaklaşmaktadır.**

- **Az-Örnekli Görevlerde Çarpıcı İyileşme:** Few-shot öğrenme senaryosunda sonuçlar daha da dikkat çekicidir. Model, SEAL olmadan (sadece bağlam içi öğrenme ile) neredeyse **hiç genelleme yapamamıştır** – başarı oranı %0 olarak raporlanmıştır ²⁷. Bu, beklenen bir durumdur çünkü ARC gibi zor bir görevde sadece birkaç örnekle, modelin ağırlıkları değiştirilmediği sürece başarısız olması doğaldır. Öte yandan, SEAL yöntemi uygulandığında modelin başarı oranı **%72.5** gibi yüksek bir seviyeye çıkmıştır ²⁷. Bu iyileşme, insan tarafından sağlanmamış bir öğretimle, tamamen modelin kendi kararlaştırdığı augmentasyon ve eğitim stratejisi sayesinde elde edilmiştir. Kıyaslamak gerekirse, test zamanında modelin ağırlıklarını güncelleyen fakat *eğitilmemiş* bir self-edit politikası kullanıldığında (yani RL ile öğrenmeden, belki rastgele veya kural-tabanlı bir güncelleme denendiğinde) başarı ancak %20'ye çıkabilmiştir ²⁷. SEAL'in RL ile öğrendiği politika ise bunu katbekat aşarak %72.5'e ulaşıyor. Bu değer, hatta "oracle" seviyesinde bir test-zamanı eğitimle (mükemmel bilgilendirme ile) kıyaslandığında bile oldukça yüksektir. **Sonuç olarak, SEAL az örnekli durumlarda dahi modelin performansını sıfır düzeyinden kullanışlı bir seviyeye taşıyabilmektedir**, ki bu pratik uygulamalarda çok kritik olabilir.

- **İnsan Müdahalesi ve Veri İhtiyacının Azalması:** SEAL'in önemli avantajlarından biri, **asgari insan emeğiyle** modelin kendini iyileştirebilmesidir. Geleneksel olarak model uyarlamak için ya insan tarafından hazırlanmış yeni veri gerekir, ya da insan geri bildirimi (örneğin RLHF'de olduğu gibi) kullanılır. Bu çalışma ise modelin kendi **ürettiklerini** kullanarak öğrenmesini sağlıyor. Yani model hem öğrenci hem öğretmen rolünü üstleniyor. Emergent Mind özetinde vurgulandığı gibi, bu yaklaşım kapsamlı etiketli veri ihtiyacını modelin **içsel üretim kabiliyetini** kullanarak minimize ediyor ³. Özellikle hızla değişen veya kullanıcıya özel uyarlama gereken durumlarda SEAL, insan uzmanlardan veri toplamadan modeli güncel tutma potansiyeli sunuyor. Bu da ölçeklenebilirlik ve maliyet açısından büyük bir artı. Örneğin bir şirketin dil modeli, yeni çıkan her ürün için dokümantasyonu okuyup kendini güncelleyebilir; bunun için ayrı ayrı veri hazırlamaya gerek kalmaz.

- **Otonom ve Modüler Öğrenme:** SEAL ile bir model, kendi kendine yeni bir beceri kazanırken bunu **ayrık ve modüler** biçimde yapıyor. Yöntemin yapısı gereği, modelin ürettiği her self-edit aslında belirli bir amaca yönelik modüler bir güncelleme. Bu, potansiyel olarak modelin farklı yetenekleri ayrı ayrı öğrenip gerektiğinde kullanabilmesi demek. Bir self-edit bilgi eklemeye yönelik olabilirken bir başkası mantık çıkarımı yeteneğini iyileştirmeye yönelik olabilir. Model, her birini kendi döngüsünde pekiştirerek öğrendiği için bu beceriler modele ekleniyor. İleride istenirse, belirli bir beceriyi sağlayan güncelleme geri alınabilir veya değiştirilebilir (teoride). Bu esneklik, klasik tam model eğitime kıyasla çok daha ince taneli kontrol imkanı verir.
- **Model İçin “Deneme-Yanılma” Öğrenimi:** Takviyeli öğrenme döngüsü sayesinde model, yaptığı hatalardan kendi kendine ders çıkarır hale geliyor. SEAL, *hangi güncellenmenin işe yarayıp yaramadığını* doğrudan modelin performansı ile ölçtüğünden, model gereksiz veya zararlı güncellemeleri zamanla **unutmayı**, faydalıları ise **hatırda tutmayı** öğreniyor ¹⁸. Medium üzerinde yapılan bir yorumda bu, *“deneme-yanılma ama akıllı bir şekilde: model neyin işe yaradığını öğrenip neyin yaramadığını unutuyor”* şeklinde özetlenmiştir ¹⁸. Bu, adeta modelin kendi kendine araştırma yapıp doğru yöntemi bulmasına benzetilebilir.

Yukarıdaki avantajlar, SEAL’i dil modelleri alanında önceki yöntemlerden ayrı bir konuma oturtuyor. **Peki SEAL, daha önce bilinen diğer yaklaşımlarla karşılaştırıldığında tam olarak ne yönleriyle farklılaşmaktadır?** Aşağıdaki bölümde bunu ele alacağız.

Önceki Yöntemlerle Karşılaştırma (RLHF, Prompt Tuning, Sürekli Öğrenme vs.)

SEAL’in getirdiği yenilikleri daha iyi anlamak için, onu halihazırda LLM’lerin uyarlanması için kullanılan popüler yöntemlerle karşılaştırmak yararlı olacaktır:

- **RLHF (İnsan Geri Bildirimiyle Pekiştirmeli Öğrenme) ile Farkı:** *Reinforcement Learning from Human Feedback* yönteminde, bir dil modeli insan tercihleriyle hizalanmak üzere bir ödül modeli yardımıyla eğitilir. Yani modelin çıktıları, insanlar tarafından iyi-kötü diye puanlanır ve bu puanları taklit edecek şekilde model RL ile ince ayar yapılır. **RLHF’in temel farkı**, adaptasyon için gereken ödül sinyalinin insandan gelmesi ve modelin genellikle *çıktı davranışını* ayarlamaya odaklanmasıdır. SEAL’de ise ödül sinyali, modelin **asıl görevi ne kadar iyi başardığından** (ör. soruyu doğru cevaplayabildin mi?) otomatik olarak türetilir ¹². Yani insan etiketlemesi yerine, görevin kural tanımı ödülü belirler. Ayrıca RLHF, modeli hizalamak (ör. zararlı çıktıları önlemek veya daha tercih edilir yanıtlar vermek) için kullanılırken, SEAL modelin **bilgi ve yeteneklerini** genişletmek için kullanılır. Bir başka deyişle RLHF, modelin *nasıl cevap verdiğini* şekillendirirken; SEAL, modelin *neleri bildiğini ve nasıl öğrendiğini* geliştirir. Yine RLHF’de model tipik olarak tek bir sefer global olarak ince ayar görür (ör. ChatGPT’nin son aşama eğitimi gibi), SEAL ise her yeni durumda modelin kendi kendine ufak ince ayarlar yapması üzerine bir çerçeve sunar. Bu yönüyle SEAL, insan geri bildirimine bağımlı olmayıp **otonom bir pekiştirmeli öğrenme** yaklaşımı sergiler.
- **Prompt Tuning / In-Context Learning ile Farkı:** *Prompt tuning* veya *bağlam içi öğrenme* yöntemleri, modelin ağırlıklarını değiştirmeden uyum sağlamayı hedefler. Örneğin prompt tuning’de modele sabit bir ön ek (prompt vektörü) öğrenilir ve yeni görevler için bu ön ek kullanılarak model yönlendirilir. In-context learning’de ise modelin içine, çözülecek görevin birkaç örneği konur ve modelden genellemesi beklenir. Bu tekniklerde, model **statik kalır**, sadece girdi manipülasyonlarıyla davranışı değiştirilir. **SEAL’in farkı**, doğrudan modelin parametrelerini güncellemesidir ¹¹. Bu sayede elde edilen adaptasyon kalıcıdır ve modele gerçekten yeni bilgi

kazandırır. Örneğin, prompt tuning ile bir LLM belirli bir stil veya görevde daha iyi hale gelebilir ama bu ayar kaldırıldığında eski haline döner; SEAL'de ise model bizzat değiştiği için yeni beceri artık ayrılmaz bir parçasıdır. Öte yandan, prompt tuning'in avantajı olan *kolay tersine çevrilebilirlik* (prompt'u çıkarınca eski haline döner) SEAL'de yoktur – model güncellemeyi geri almak için tekrar eğitmek gerekebilir. Fakat bu aynı zamanda SEAL'in amacıdır: gerçekten **sürekli öğrenen** bir model elde etmek. Ayrıca, SEAL'de model kendi kendine talimat oluştururken dahi doğal dil kullanır; bir bakıma modeli *kendi prompt'unu yazıp uyguluyor* gibi düşünebiliriz, ancak bu prompt (self-edit) sadece geçici bir yönlendirme olmayıp eğitim datası işlevi görür.

- **Sürekli Öğrenme (Continual Learning) Yöntemleri ile Farkı:** Sürekli öğrenme, bir modelin zaman içinde ardışık olarak gelen farklı veri dağılımlarına uyum sağlarken önceki öğrendiklerini unutmamasını amaçlayan geniş bir alan. Klasik sürekli öğrenme yöntemleri genellikle **harici veriler** kullanır – model sırayla veri setleri görür ve çeşitli tekniklerle (replay, regularization vs.) eski bilgiyi koruyarak yenilerini öğrenmeye çalışır. SEAL de özünde bir sürekli öğrenme yaklaşımıdır, ancak en büyük fark **veri kaynağı ve kontrol mekanizmasıdır**. Sürekli öğrenmede veri dışarıdan gelir ve model nasıl öğreneceğine kendi karar veremez; SEAL'de ise model, yeni veriyi **kendisi sentezler veya yönlendirir**. Bu, modelin öğrenme sürecine meta-düzeyde katılımını sağlar. Örneğin klasik sürekli öğrenmede “önceki görevlerden örnekleri bellekte tut ve arada tekrar et” gibi bir stratejiyi insan belirler, SEAL yaklaşımında ise böyle bir stratejiyi modelin kendisinin önermesi prensipte mümkündür. Nitekim SEAL'in *limitations* kısmında, yazarlar **catastrophic forgetting (felaket unutma)** sorununa değinirken, olası çözümler arasında *replay (önceki verileri yeniden kullanma)* veya *kısıtlı güncelleme* gibi teknikleri modelin gelecekte kendisinin uygulayabileceğini ima etmişlerdir ²⁸. Yani SEAL ileride, klasik sürekli öğrenme tekniklerini bile otonom şekilde entegre edebilecek bir çerçeve sunabilir. Şu anki haliyle fark, SEAL'in bu mekanizmaları açıkça uygulamadığı, bu yüzden de çoklu ardışık uyarlamalarda unutma yaşayabildiğidir (aşağıda detaylı ele alınacak). Özetle, sürekli öğrenme “**nasıl unutmadan öğrenirim?**” sorusunu çözerken genelde modele bir şeyler **uygular**; SEAL ise “**öğrenmek için ne yapmalıyım?**” sorusunu modelin kendisine sordurtur ve cevabını uygular. Bu, paradigmatik bir farklılıktır.

- **Diğer Adaptasyon Teknikleri:** Literatürde ayrıca *meta-learning (öğrenmeyi öğrenme)*, *model editing (modellerde sonradan bilgi düzenleme)*, *adapter modülleriyle ince ayar*, vb. gibi yöntemler vardır. Meta-learning (ör. MAML gibi) genelde modele bir adaptasyon yapabilme yetisi kazandırır ama bunu ayrı bir eğitim aşamasında yapar; model, yeni bir veriye bakıp hızlıca adapte olmayı öğrenir. SEAL, meta-learning'in hedeflediği şeyi farklı yoldan yapmış oluyor: Model kendi adaptasyon verisini bulup uyguluyor, yani dolaylı bir meta-öğrenme biçimi denebilir. Model editing yaklaşımları (örneğin bir modele belirli bir bilgi enjekte etmek için küçük network'ler kullanmak gibi) genellikle tek seferlik düzenleme yapar; SEAL bunları tekrarlı ve model tarafından yönlendirilir hale getirir. Adapter veya LoRA gibi **parametre-etkin** ince ayar yöntemleri ise modelin tüm parametrelerini değiştirmek yerine küçük ekstra parametreler ekler. Bu, pratikte bir görev için özel modüller takmak gibidir. SEAL ise tüm parametreleri değiştirse de bunu modelin kararıyla yaptığından, gelecekte **hangi parametrenin veya modülün ne kadar değişeceğine** dahi model karar verebilir. Emergent Mind'da belirtildiği üzere, SEAL doğrudan gradient ile kendi ağırlık güncellemesini yaparken, adapter tabanlı yöntemler ağırlıkların ufak bir kısmını değiştirerek hızlı uyarlama yapar ²⁹. Bu iki yaklaşım birbiriyle rekabetten çok, birleşebilir de aslında: SEAL çerçevesi ileride “*hangi adapter'ı eklemeli?*” sorusunu bile kendi kendine sorabilen modelleri mümkün kılabilir.

Özetle, SEAL mevcut yöntemlerden **otonom veri üretimi, kalıcı ağırlık güncellemesi ve RL tabanlı politika öğrenimi** kombinasyonuyla ayrıışmaktadır. RLHF insan ödülüne dayalı tek seferlik ince ayar sağlarken, SEAL görev performansına dayalı çok seferlik öz-öğrenme döngüsü sunar. Prompt tuning ve

benzeri yöntemler ağırlıkları değiştirmezken, SEAL değiştirmek üzerine kuruludur. Sürekli öğrenme dış veriyle yapılırken, SEAL kendi verisini içeriden üretir. Bu farklılıklar, SEAL'i literatürde **kendine yeten model adaptasyonu** konusunda özgün bir noktaya yerleştiriyor.

Sınırlamalar (Limitations)

SEAL yaklaşımı her ne kadar umut verici olsa da, makalede ve deneylerde belirlenen bazı önemli sınırlamalar mevcut:

- **Catastrophic Forgetting (Felaket Unutma):** En önemli sınırlamalardan biri, modelin art arda birden çok kez kendi kendini güncellemesi durumunda önceki bilgileri unutabilmesidir ²⁸. Yazarların sürekli öğrenme deneyinde gözlemledikleri üzere, model farklı konularda ardışık self-edit'lerle güncellendiğinde, ilk öğrendiği konulardaki performansı zamanla düşüş göstermiştir. Bunu görselleştiren bir ısı haritasında, diyelim ki model 1. iterasyonda Konu A'yı öğrendi ve iyi performans sergiledi; 5 iterasyon sonra model Konu E'yi öğrendiğinde, Konu A ile ilgili başarısı başlangıça kıyasla belirgin ölçüde azalıyor. Bu klasik bir sürekli öğrenme sorunudur: *yeni bilgiler eskileri bozabilir*. SEAL çerçevesi şu an bu sorunu önlemek için özel bir mekanizma içermiyor. Yazarlar da **"modelin kendi kendini değiştirmesi, açıkça bir bilgi korunumu mekanizması olmadan, değerli önceki bilgileri üzerine yazabilir"** diyerek bunu kabul ediyorlar ²⁸. Çözüm için, makalede bahsi geçen olasılıklar arasında **replay (önceki verileri ara sıra yeniden kullanma)**, **kısıtlı güncellemeler (güncellemelerin boyutunu/sınıfını sınırlama)** veya **temsil süperpozisyonu (representational superposition)** gibi ileri teknikler var ²⁸. Ancak bunların SEAL'e nasıl entegre edileceği gelecekteki çalışmalara bırakılmış. Yani şu an için SEAL, birden fazla ardışık adaptasyon gerektiğinde dikkatli kullanılmalı veya ek mekanizmalarla desteklenmelidir.
- **Self-Edit Formatının Göreve Özel Olması:** Bir diğer pratik sınırlama, self-edit dediğimiz modelin ürettiği talimatların formatının her yeni görev türü için insan tarafından tanımlanması gerekmesidir. Makalede de görüldüğü gibi, bilgi entegrasyonu için self-edit'ler "olgusal cümleler" formatında, az-örnekli öğrenme için "ayar ve strateji listesi" formatında tasarlanmıştır ¹⁵. Bu formatları yazarlar belirlemiş ve model, bu format dahilinde çıktılar üretmiştir. Demek ki şimdilik model, keyfi bir konuda kendi kendine güncelleme talimatı oluşturacak *genel bir dil* bilmiyor; her yeni problem alanı için bizim onun ne tür bir self-edit üretmesi gerektiğini kabaca tarif etmemiz gerekiyor. Bu bir çeşit istem mühendisliği (prompt engineering) veya arayüz tasarımı ihtiyacı doğuruyor. İleride daha genel bir çerçeve geliştirilebilirse model gerçekten hiç yönlendirme olmadan da kendi güncelleme formatını keşfedebilir. Şu anki haliyle SEAL, **her uygulama alanında biraz elle ayar gerektirebilir** (örneğin görüntü işleme yapan bir modelde nasıl self-edit tanımlanacağı ayrı bir çalışma konusu olacaktır).
- **Hatalı veya Zararlı Güncellemeler:** Modelin kendi kendine güncelleyebilmesi konsepti beraberinde bazı riskler de getirir. Örneğin model yanlış bir çıkarım yapıp bunu knowledge base'ine entegre ederse, kendi kendine **yanlış bilgi** öğrenmiş olur. Makale, modelin performans odaklı ödül sinyali sayesinde genelde hatalı güncellemeleri eliyor olsa da (ödül düşük geleceği için o self-edit pekiştirilmiyor), yine de kusurlu bir ödül tanımı veya eksik test durumunda modelin istenmeyen yönleri adapte olması mümkün. Aynı şekilde, kötü niyetli kullanım senaryolarında (örneğin modelin kendini zararlı cevaplar vermeye doğru eğitmesi gibi) kontrol mekanizmaları gerekecektir. Bu çalışmanın kapsamında böyle güvenlik ve doğruluk konuları derinlemesine ele alınmamış, ancak pratikte dikkat edilmesi gereken noktalar olduğunu not etmek gerekir.

- **Hesaplama Maliyeti:** SEAL her ne kadar tam bir yeniden eğitimden daha hafif bir süreç olsa da, tamamen ücretsiz değildir. Özellikle RL döngüsü içermesi nedeniyle, birden fazla aday self-edit üretme, her biriyle ince ayar yapıp deneme, ödüle göre güncelleme gibi ekstra işlemler vardır. Bu, tek bir fine-tuning'e kıyasla daha yüksek zaman ve hesaplama maliyeti getirebilir. Yazarlar ReST-EM algoritmasının hafif olduğunu belirtmişlerdir ¹⁴, ancak gene de uygulamada bu yöntemin büyük modellerle uygulanması ciddi GPU zamanı gerektirebilir. Nitekim deneylerde kullanılan model boyutu (birkaç milyar parametre) ve donanım (A100/H100 GPU'lar) bunu ima etmektedir. Dolayısıyla SEAL'in *"her istekte bir RL döngüsü çalışsın, model anında kendini uyarlasın"* şeklinde gerçek zamanlı uygulanabilir hale gelmesi için optimizasyonlara ihtiyaç duyulabilir. Belki daha verimli öğrenme algoritmaları veya model mimarilerinde değişiklik gerekebilir.

Bu sınırlamalara rağmen, SEAL kavram kanıtı niteliğinde güçlü sonuçlar vermiştir. Unutma sorunu gibi açık problemlerin çözümü, aynı zamanda bu yöntemin gelişimi için bir yol haritası sunmaktadır. Nitekim yazarlar da sonuçları tartışırken gelecekte nelerin yapılabileceğine dair bazı fikirler ortaya atmışlardır.

Bulguların Yorumlanması ve İleriye Yönelik Çıkarımlar

SEAL ile ilgili deneysel bulgular, dil modellerinin gelecekte nasıl daha **esnek ve otonom** hale gelebileceğine dair önemli ipuçları veriyor. İlk olarak, **modelin kendi hatalarını ve eksiklerini giderebilmesi mümkündür**. Bu, yapay genel zekâ yolunda kritik bir kabiliyettir. SEAL, modelin performansındaki açığı kapatmak için kendi kendine ek veriler üretebileceğini ve parametrelerini iyileştirebileceğini gösterdi. Özellikle az örnekle öğrenmedeki büyük sıçrama, modellerin insan benzeri şekilde *deneme yaparak öğrenebileceğinin* kanıtı sayılabilir.

İkinci olarak, **insan bilgisi ve kontrolüyle model güncelleme** paradigmalarına alternatif bir yol belirdi. Şu anda büyük modelleri güncel tutmak için ya onları yeniden eğitiyoruz ya da harici bilgi kaynaklarına (örn. arama motorları, veritabanları) başvuruyoruz. SEAL ise *içsel bir güncelleme mekanizması* sunuyor. Bu, zamanla modellerin adeta **kendi kendini yazılım güncellemesi** yapması gibi bir vizyonu mümkün kılıyor. Örneğin, gelecekte bir LLM, internette karşılaştığı yeni bir kavramı otomatik olarak öğrenip kendi parametrelerine dahil edebilir; bu sırada geliştiriciye sadece minimal bir denetim rolü düşer.

Yazarların makaledeki gelecek işleri tartıştıkları bölümde çizdiği vizyon da oldukça ilgi çekici: **"Modellerin sadece ağırlıklarını uyarlaması değil, ne zaman ve nasıl uyarlayacaklarına da karar vermesi"** beklentisi dile getiriliyor ³⁰. Bu, modelin bir soruyu cevaplarken bile durup **"Ben bu soruyu daha iyi cevaplamak için kendimi güncelleyebilir miyim?"** diye düşünebilmesini ima ediyor. Örneğin, model bir problemi çözerken zorlanırsa anında bir self-edit üretecek, kendi üzerindeki bir dizi mikro-eğitimi o anda uygulayacak ve sonra soruya devam edecek. Bu tür **anlık adaptasyon (on-the-fly adaptation)**, insan öğrenmesinde de gördüğümüz bir şeydir – bir soruya bakıp *"önce ilgili formülü hatırlayayım/öğreneyim"* deyip sonra çözmek gibi. Bu gelecekte gerçekleşirse, LLM'ler *sadece akıllı metin üreticiler değil, aynı zamanda kendi kendinin eğitimcisi olan ajanlar* haline gelebilir.

Ayrıca **zincirleme düşünceyi (chain-of-thought)** ağırlıklara işleme fikri de ortaya atılıyor ³⁰. Günümüzde zincirleme düşünce, modelin bir soruyu yanıtlarken ara adımları metin olarak üretmesi demek. Ancak bu ara adımlar cevap verildikten sonra unutuluyor. Yazarlar, eğer model bu düşünce zincirlerinden faydalı şeyler öğrenip bunu kalıcı hale getirebilirse, geçici akıl yürütmeler **kalıcı kabiliyetlere** dönüşür diyorlar ³⁰. Bu, çok güçlü bir konsept: Yani model bir kez çözmeyi öğrendiği bir bulmacanın çözüm stratejisini bir daha unutmamak üzere kendi ağırlıklarına yazabilir. Böylelikle her çözüm, modelin bilgi birikimini artırır. Bu yaklaşımla modellerin sürekli kendi deneyimlerinden ders çıkarması ve hiç durmadan daha yetenekli hale gelmesi hayal edilebilir.

Son olarak, SEAL'in başarısı **yapay zekâ ajanları** konusunda da önemli bir adımdır. Bir otonom ajanın, ortamıyla etkileşimde bulunurken kendini geliştirmesi gerekebilir. SEAL, bu tür ajanlar için bir temel mekanizma sağlayabilir. Örneğin bir sohbet botu düşünün, kullanıcılarla konuştukça yeni konuları öğrenip kendini güncelliyor ve bir sonraki konuşmada daha bilgili hale geliyor. Bu, *insanlarla etkileşimden öğrenen* AI hayalini gerçekleştirebilir. Elbette, bunun kontrollü ve güvenli bir şekilde yapılması gerekecek; aksi halde istenmeyen bilgileri de öğrenebilir. Ancak doğru uygulandığında, **sürekli gelişen ve kendini iyileştiren AI sistemleri** ortaya çıkacaktır.

Sonuç ve Değerlendirme

"Self-Adapting Language Models" makalesi, büyük dil modellerinin statik doğasını kırma yönünde önemli bir kavramsal ilerleme sunuyor. Önerilen SEAL çerçevesi sayesinde bir model, kendi ürettiği verilerle kendi parametrelerini güncelleyerek yeni bilgilere ve görevlere adaptasyon gösterebiliyor. Bu çalışma, **orta düzey teknik ayrıntıları** ile incelediğimiz üzere, hem sağlam deneysel kazanımlar (daha yüksek doğruluklar, yeni görevlerde başarı vs.) sağlamakta hem de yapay zeka araştırmalarında yepyeni sorular doğurmaktadır. Öz-uyumlu modellerin daha güvenli, verimli ve genelleştirilebilir hale gelmesi için yapılacak çok iş var. Fakat bu yaklaşım şimdiden göstermiştir ki, gelecekte LLM'ler sadece pasif bilgi bankaları değil, **aktif öğrenen ve kendini geliştiren varlıklar** olarak karşımıza çıkabilirler ³⁰. Bu da yapay zekanın otonomisine giden yolda heyecan verici bir gelişmedir.

Kaynaklar:

- Zweiger, A. vd. (2025). *Self-Adapting Language Models*. arXiv:2506.10943 ¹ ⁷ ¹³ ¹⁵ ²⁵ ²⁷ ²⁸ ³⁰. (Bu raporda makalenin içeriği ve sonuçları belirtilen satırlardan özetlenmiş ve yorumlanmıştır.)

¹ ⁷ ¹¹ ¹² [2506.10943] Self-Adapting Language Models

<https://arxiv.org/abs/2506.10943>

² ⁴ ¹⁸ Self-Adapting Language Models. Weights are the magical stuff that... | by Rachit | CodeX | Jun, 2025 | Medium

<https://medium.com/codex/self-adapting-language-models-my-two-cents-f9b4806f14fc>

³ ⁵ ⁸ ⁹ ¹⁰ ²⁹ Self-Adapting Language Models

<https://www.emergentmind.com/papers/2506.10943>

⁶ GitHub - Continual-Intelligence/SEAL: Self-Adapting Language Models

<https://github.com/Continual-Intelligence/SEAL>

¹³ ¹⁴ ¹⁵ ¹⁶ ¹⁷ ¹⁹ ²⁰ ²¹ ²² ²³ ²⁴ ²⁵ ²⁶ ²⁷ ²⁸ ³⁰ Self-Adapting Language Models

<https://jyopari.github.io/posts/seal>