Transformer Modelleri

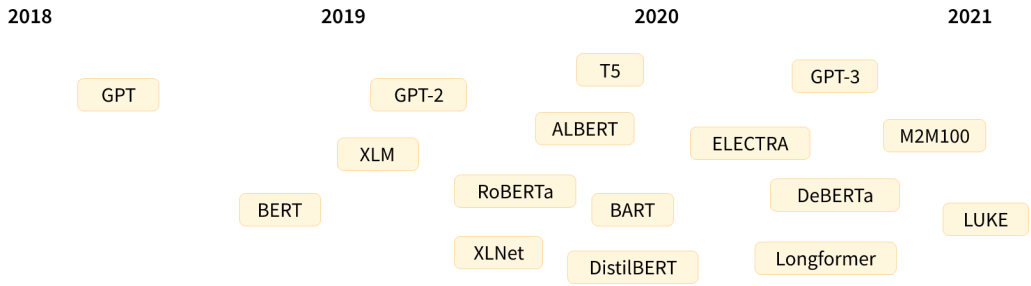
Neler Yapabilirler?

1. Pipeline Fonksiyonu
2. Zero-shot Pipeline
3. Text Generation
4. Mask Filling
5. Named Entity Recognition
6. Question Answering

Transformer: Nasıl Çalışır?

Bu kısımda **transformer** modeller için high-level bir incelemede bulunacağız.

Transformer Tarihi



Transformer mimarisi 2017'de tanıtıldı ve ana amacı çeviri yapmaktı. Aşağıdaki etkili modelleri etkilemiştir:

- **Haziran 2018**: GPT, İlk pretrained Transformer modelidir ve fine-tune edilebildiği için NLP task'larında başarı göstermiştir.
- **Ekim 2018**: BERT, Bir diğer pretrained modeldir. Cümleleri daha iyi özetlemek amacıyla geliştirilmiştir.
- **Şubat 2019**: GPT-2, GPT'nin geliştirilmiş ve daha büyük versiyonudur. Etik kaygılardan dolayı direkt yayına alınmamıştır.
- **Ekim 2019**: DistilBERT, BERT modelinin "damıtılmış" versiyonudur, BERT'e göre %60 daha hızlıdır ve %40 daha az bellek tüketir.
- **Ekim 2019**: BART ve T5, iki büyük pretrained model, orijinal Transformer mimarisini kullanır.
- **Mayıs 2020**: GPT-3, GPT-2'nin daha büyük olan versiyonudur. Birçok işte başarılı sonuçlar gösterir ve fine-tune ihtiyacı yoktur (*zero-shot* learning olarak bilinir).

Bu listeye bakarak aşağıdaki 3 kategoriyi oluşturabiliriz:

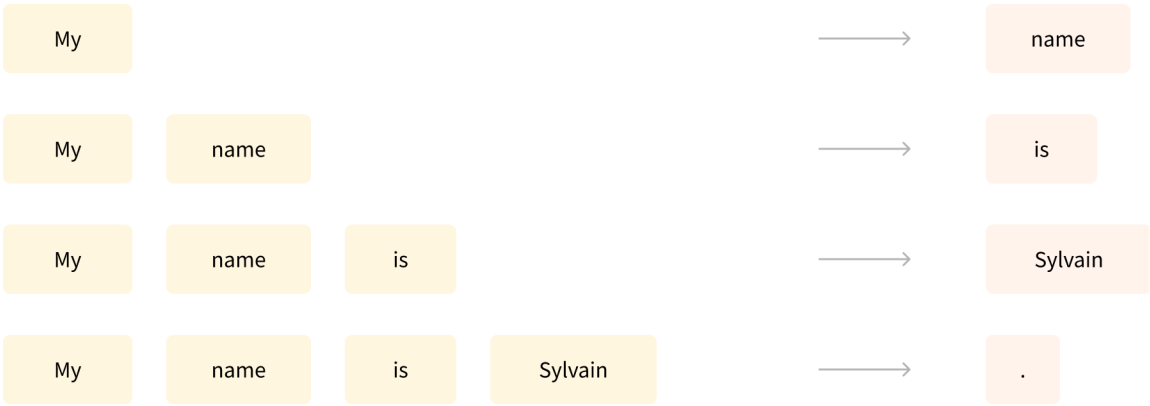
- GPT-like (auto-regressive Transformer modelleri olarak da bilinir.)
- BERT-like (auto-encoding Transformer modelleri olarak da bilinir.)
- BART/T5-like (sequence-to-sequence Transformer modelleri olarak da bilinir.)

Transformer'lar Dil Modelleridir

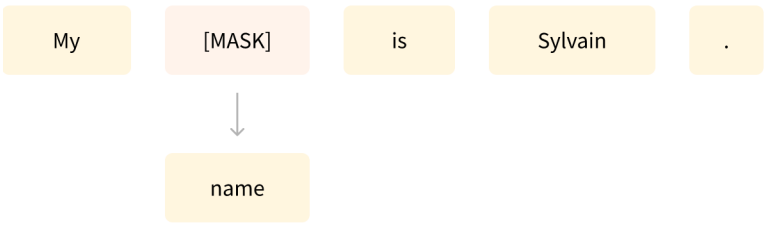
Yukarıda bahsedilen modellerin tamamı dil modeli olarak bilinir. Büyük miktarda ham metin ile self-supervised yöntemi ile eğitilir. Self-supervised öğreniminde objective, otomatik olarak input'lardan hesaplanır.

Bu tip modeller eğitildiği dil hakkında istatistiksel bir kavrayış oluşturur ancak bu kavrayış özel ve pratik işler için uygun değildir. Bu yüzden bu pretrained modeller sonradan bazı işlemlere tabi tutulur ve buna **transfer-learning** denir. Bu işlemler sırasında model supervised olarak fine-tune edilir. Yani model, verilen iş için insan tarafından etiketlenmiş verilere ihtiyaç duyar.

Buna bir örnek olarak şu iş verilebilir: Önceki n adet kelimeyi inceleyerek sonraki kelimeyi tahmin etmek. Bu *casual language modeling* olarak bilinir çünkü modelin çıktısı geçmiş ve şu anki girdilere bağlıdır.

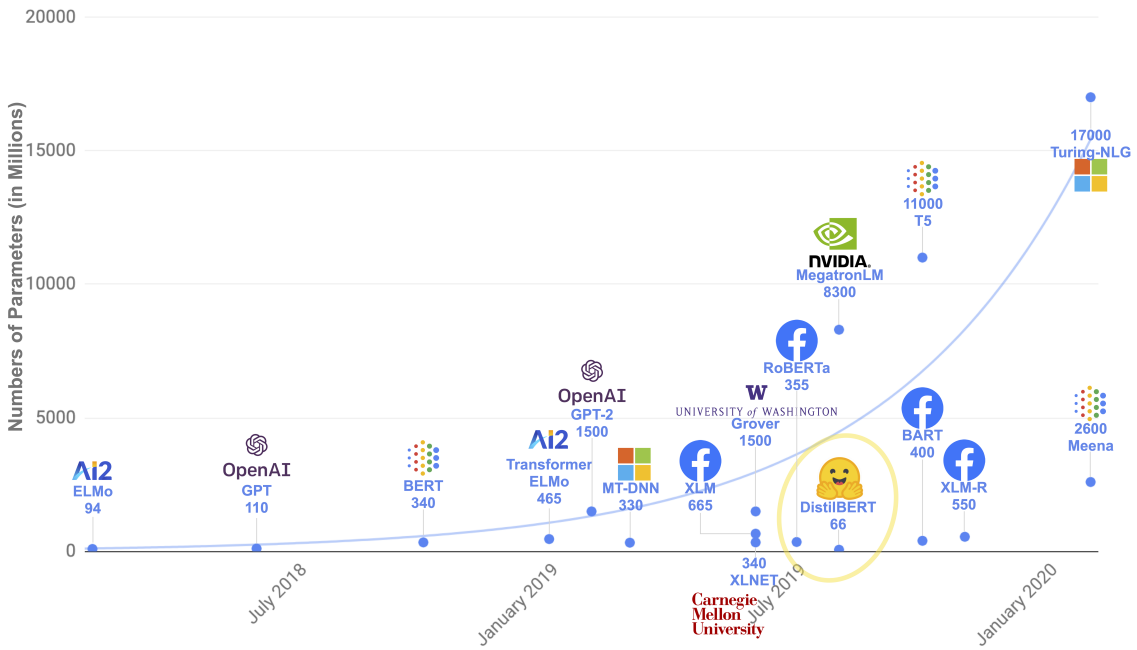


Bir başka iş örneği olarak *masked language modeling* verilebilir.

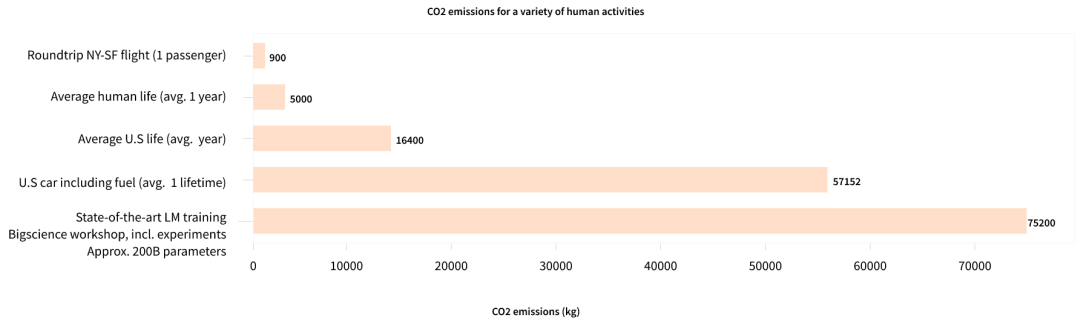


Transformer'lar Büyük Modellerdir

Birkaç istisna dışında(DistilBERT), modelin performansını iyileştirmenin yolu daha çok veri sağlamak ve modelin boyutlarını büyütmektir.



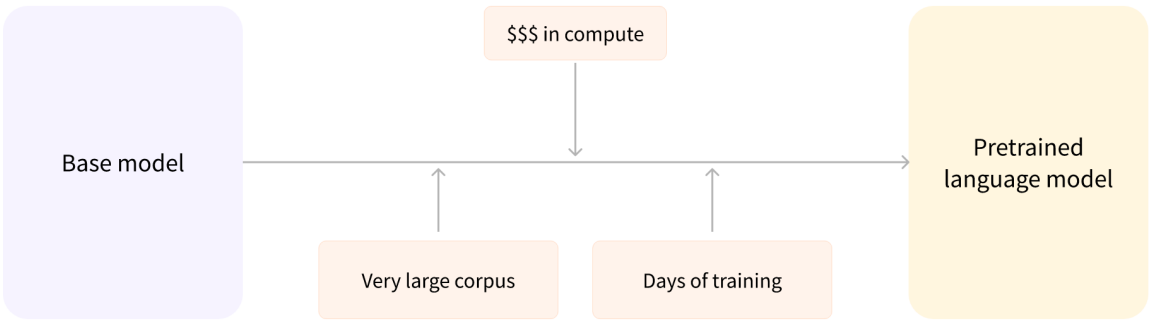
Ne yazık ki, büyük bir model eğitmek oldukça maliyetlidir, büyük işlem gücüne ihtiyaç duyar ve çevresel etkileri de azımsanacak boyutlarda değildir. İşin içine hiperparametre optimizasyonu ve denemeler de girince, modelin çevresel etkileri büyük problem haline gelebilir.



Bu yüzden eğitilmiş modellerin paylaşımı hesaplama maliyetlerini ve karbon ayak izini azaltıcı etkilere sahiptir. Modeli eğitim sürecinde oluşacak emisyonu hesaplamak için Python kütüphaneleri mevcuttur.

Transfer Learning

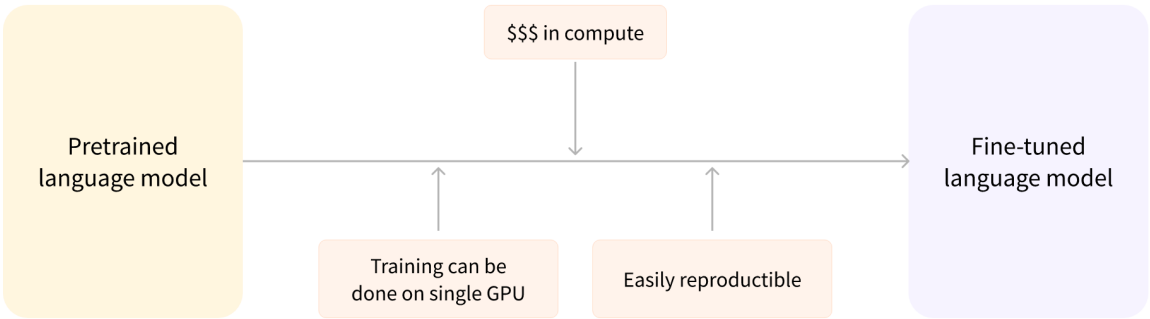
Pretraining, modeli sıfırdan eğitme işlemidir. Başlangıçta model parametreleri rastgele belirlendiği için eğitim sıfır bilgi ile başlar.



Pretraining çok büyük miktarda veri ile gerçekleştirilir ve eğitimin tamamlanması haftaları bulabilir. Öte yandan, *fine tuning* işlemi pretraining işleminden sonra gerçekleştirilir. Fine tune işlemi için önceden eğitilmiş bir modele ve modelin uzmanlaşması gereken iş ile ilgili bir verisetine ihtiyaç vardır. Bu final verisetini neden sıfırdan bir ağ eğitmek için kullanamazsınız?

- Eğitilmiş model büyük verisetleri ile eğitildiği için, final verisetinin büyük olmasına gerek yoktur.
- Eğitilmiş modeli fine-tune ederek çok daha iyi sonuçlara daha az kaynak ile ulaşabilir.

Örneğin, İngilizce için eğitilmiş bir modeli tıbbi makaleler ile fine-tune edersek tıp alanında daha uzman bir model elde edilmesi beklenir. Bu fine-tune işlemi için çok daha az veri ve işlem gücü gerekir. Bir bakıma bilgi **transferi** sağlanmış olur.

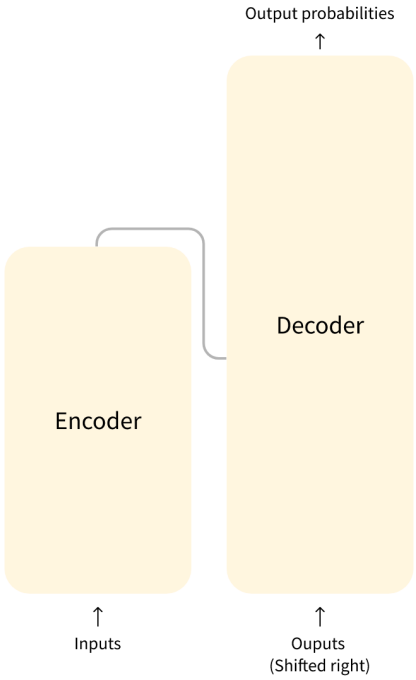


Bütün bu nedenlerden dolayı modeli sıfırdan eğitmek yerine **Transfer Learning** tercih etmek akıllıcadır.

Genel Mimari

Model genel olarak iki bloktan oluşur.

- **Encoder (sol)**: Encoder, girdileri toplar ve girdileri temsil eden bir "representation" oluşturur. Model, girdilerini kullanarak bir kavrayış geliştirmek için optimize edilir.
- **Decoder (sağ)**: Decoder, encoder'in ürettiği "representation'ları" ve diğer girdileri kullanarak hedeflenen sequence'i oluşturmaya çalışır. Model bir çıktı üretmek için optimize edilir.



İşe göre, bu iki blok birbirinden bağımsız olarak da kullanılabilir:

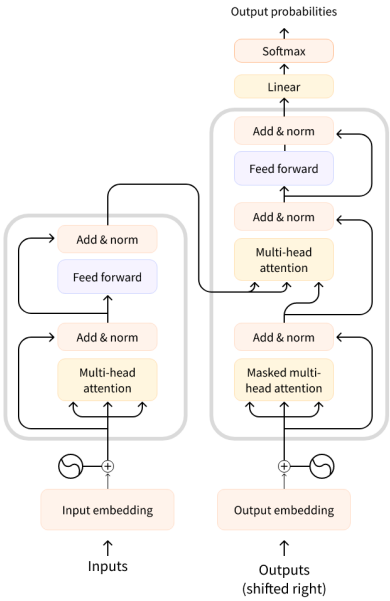
- **Encoder-only models**: Girdilerin anlamını kullanan uygulamalar (NER, Cümle Sınıflama) için kullanışlıdır.
- **Decoder-only models**: Generative görevler (Text Generation) için kullanışlıdır.
- **Encoder-decoder models** veya **Sequence-to-sequence models**: Girdi gerektiren generative görevler (Çeviri, Özetleme) için kullanışlıdır.

Attention Layers

Transformer'ları özel kılan özellik **Attention** mekanizması barındırmasıdır (*Attention Is All You Need* makalesinde anlatılmıştır). Detaylı incelemeden önce bir özet geçmek gerekirse bu mekanizma şu işe yarar: Kelimelerin temsilleri (representation'ları) ile uğraşırken, modelin hangi kelimeye özel ilgi göstermesi gerektiğini söylemek.

The Original Architecture

Aslen çeviri işleri için tasarlanmıştır. Training süresince encoder input'ları toplar, bu durumda inputlar cümledir, ve decoder de arzu edilen dildeki cümleyi input olarak alır. Encoder tarafında input olarak verilen cümlenin tamamı kullanılır. Decoder tarafında sadece önceden çevrilen ve şu an çevrilen kelimelere odaklanılır. Süreci hızlandırmak icin decoder'a bütün hedef cümle verilir ancak gelecekteki kelimeleri görmesi engellenir.



Decoder bloğundaki ilk attention katmanı decoder'a verilen tüm geçmiş input'ları kullanır. Ancak ikinci attention layer sadece ecnoder'dan gelen bilgilere erişebilir.

Attention Mask, encoder ve decoder'in özel kelimelere ilgi göstermesinin önüne geçer. Mesela bütün cümleleri aynı boyuta çekmek için eklenen PADDING kelimesinin burada ilgi çekmesi engellenir.

Architectures vs Checkpoints

Kursun ilerleyen kısımlarında modeller, checkpointler ve mimariler konuşulacak. Bazı anlam farklarına dikkat çekmek gerekir:

- **Mimari**: Modelin iskeletidir, modeldeki her bir layer ve operasyonun tanımıdır.
- **Checkpoint**: Verilen mimari için yüklenecek ağırlıklardır.
- **Model**: Bu birçok anlama gelen bir kelimedir ve yukarıdakiler gibi net bir tanımı yoktur. Mimari veya checkpoint anlamlarına gelebilir.

Örneğin, BERT bir mimaridir ama `bert-base-cased` Google tarafından eğitilen bir modelin ağırlık setinden oluşan bir checkpointtir.

Encoder Modelleri

Encoder modelleri, Transformer modellerinin sadece encoding bloğunu kullanırlar. Attention katmanları verilen cümlenin bütün kelimelerine ulaşabilir. Bu tip modeller *bi-directional* olarak karakterize edilir ve **auto-encoding** modeler olarak anılırlar.

Pretrain işlemi şuna benzer: Verilen cümleyi bir şekilde bozarak başlanır (mesela, rastgele kelimeler maskelenir) ve modele iş olarak bozulmamış cümlenin yeniden oluşturması verilir.

Bu tip modeller tüm cümlenin anlamlandırılması gereken durumlarda kullanılır. Örnek problemler cümle sınıflandırma, named entinty recognition ve extractive question answering olabilir.

Bu aile aşağıdaki modeller tarafından temsil edilir:

- ALBERT
- BERT
- DistilBERT
- ELECTRA

- RoBERTa

Decoder Modelleri

Decoder modelleri Transformer modellerinin sadece decoder bloğunu kullanır. Her aşamada, attention katmanları sadece verilen kelime ve öncesindeki kelimelere erişim sağlayabilir. Bu tip modellere **auto-regressive modeller** denir.

Pretrain işlemi genelde sonraki kelimenin tahmin edilmesi üzerine gerçekleşir. Bu tip modeller metin oluşturan task'lar için uygundur. Bu model ailesi aşağıdaki modeller ile temsil edilir:

- CTRL
- GPT
- GPT-2
- Transformer XL

Sequence to Sequence Modelleri

Enconder-decoder modelleri Transformer mimarisinin iki bileşenini de kullanır. Her adımda, encoder modelinin attention katmanları cümlenin bütün kelimelerine erişir. Decoder modelindeki attention katmanları sadece önceki kelimelere erişebilir.

Pretrain işlemi encoder ve decoder modellerini ayrı ayrı eğiterek sağlanabilir ancak farklı senaryolar da vardır. Örneğin T5 modeli bir kelime bloğunu tek bir maske özel kelimesi ile değiştirir ve maskelenen kelimeleri tahmin eder.

Bu modeller verilen cümleleri göz önünde bulundurarak metin üretebilirler, özetleme ve metin çevirisi yapabilirler. Bu model ailesi aşağıdaki modeller ile temsil edilir:

- BART
- mBART
- Marian
- T5