Nüfus ve Sosyal Araştırmalar Enstitüsü - Yapay Zeka Modülü : Makine Öğrenmesi - 3

Biz sadece bizi uykusuz bırakan hayallerin peşinden gittik, her şeyi onları inşa ederken öğrendik. *Dr. Oktay Altun*

1 - Web Kazıma (Web Scraping) Nedir?

Web Kazıma (Web Scraping), web sitelerinde yer alan, normalde insan okunabilirliğine sunulmuş yapılandırılmamış (unstructured) verilerin, yazılımlar (botlar/crawlerlar) aracılığıyla otomatik olarak çekilip yapılandırılmış (structured) formata dönüştürülmesi işlemidir.

Özünde yaptığımız şey, bir web tarayıcısının arka planda yaptığı işlemleri kod aracılığıyla çok yüksek hızlarda simüle etmektir.

1.1 Temel İşleyiş Adımları

Web scraping işlemi hiyerarşik olarak şu 4 ana fazdan oluşur:

- **1. İstek Gönderme (HTTP Request):** Hedef web sitesinin sunucusuna bir `GET` isteği atılır. Bu, tarayıcının adres çubuğuna bir URL girip Enter'a basmakla teknik olarak aynı şeydir.
- **2. Yanıt Alma (HTTP Response):** Sunucu isteğimizi kabul ederse, bize sayfanın ham kaynak kodunu (genellikle **HTML**, **XML** veya **JSON** formatında) döndürür.
- **3. Ayrıştırma (Parsing):** Gelen bu karmaşık metin bloğu, kod tarafında bir ağaç yapısına (**DOM Tree**) dönüştürülür. İstenen veriler spesifik etiketler (örn. `<div>`, `<span>`, `<a>`) veya CSS seçicileri kullanılarak bu ağaçtan süzülür.
- **4. Saklama (Storage):** Çekilen ve temizlenen veriler analiz edilmek üzere CSV, Excel veya bir veri tabanına (SQL/NoSQL) kaydedilir.

1.1.1 Basit Bir Matematiksel Gösterim

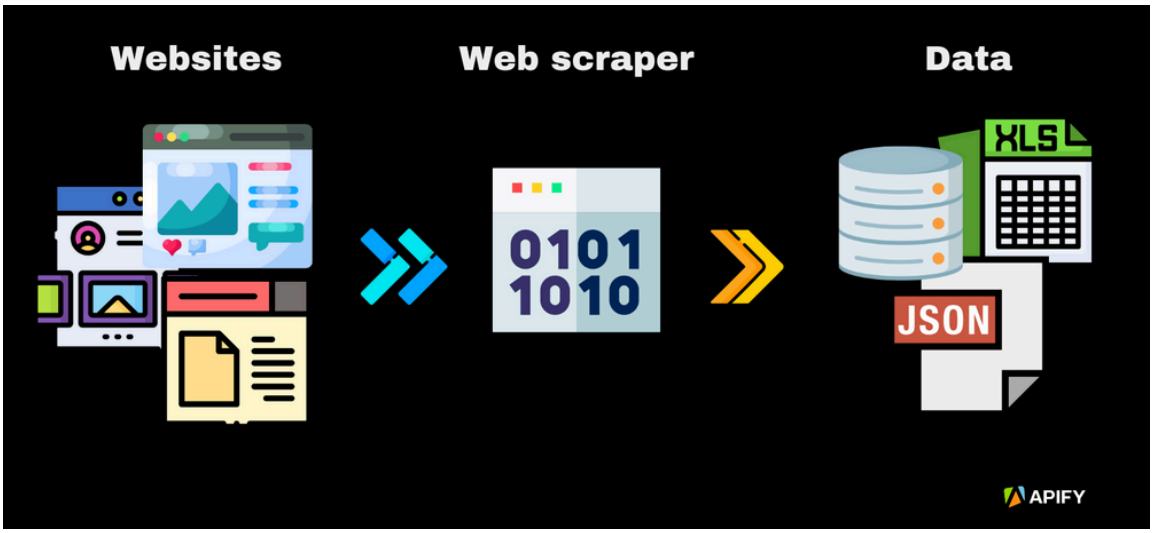
Bu süreci soyut bir fonksiyon olarak düşünürsek:

$$S(URL, C) = \{d_1, d_2, \ldots, d_n\}$$

Burada;

- S: Scraping (Kazıma) fonksiyonunu,
- URL: Hedef web sayfasının adresini,
- C: Hangi HTML etiketlerinin/sınıflarının çekileceğini belirten kural setini (CSS Selectors),

- d_i: Elde edilen anlamlı yapılandırılmış veri noktalarını (örneğin ürün fiyatları veya başlıklar) temsil eder.



1.2 Popüler Kütüphaneler ve Yaklaşımlar

Web sayfalarının mimarisine göre iki farklı yaklaşım izlenir:

- **Statik Kazıma (Static Scraping):** Sayfanın tüm verisi ilk HTTP yanıtında (HTML içinde) hazır geliyorsa kullanılır.
  - *Python Araçları:* **Requests** (istek atmak için) + **BeautifulSoup** (HTML'i ayrıştırmak için). Çok hızlı ve hafiftir.
- **Dinamik Kazıma (Dynamic Scraping):** Sayfa JavaScript ile sonradan yükleniyorsa (React, Vue vb. Single Page Application'lar) veya bir etkileşim (kaydırma, tıklama) gerekiyorsa gerçek bir tarayıcı motoru ayağa kaldırılmalıdır.
  - *Python/JS Araçları:* **Selenium**, **Playwright** veya **Puppeteer**.

1.3 Senaryolaştırma

Diyelim ki bir e-ticaret sitesinde belirli bir bilgisayar modelinin fiyatını rakip analizi için takip etmek istiyorsun.

- **Manuel Yaklaşım:** Her sabah siteye girip ürünü bulmak ve fiyatını bir Excel tablosuna elle kopyalamak. (Zaman maliyeti yüksek, ölçeklenemez).
- **Algoritmik Yaklaşım:** Python'da yazılmış bir bot her sabah 08:00'de tetiklenir. İlgili URL'ye gider, `class="product-price"` etiketini bulur, içindeki nümerik değeri (örn. 25000) çeker ve zaman damgasıyla (timestamp) birlikte doğrudan veri tabanına yazar.

1.4 Etik ve Yasal Sınırlar

Pekiştirmeli öğrenmede (Reinforcement Learning) bir ajanın çevreyi keşfederken cezalandırılmamak için dikkatli olması gerektiği gibi, scraping yaparken de sunucu tarafından

engellenmemek için dikkatli olunmalıdır:

- **Robots.txt Kontrolü:** Sitenin kök dizinindeki `robots.txt` dosyası (örn. `site.com/robots.txt`) kontrol edilmeli, site sahibinin hangi dizinlere bot erişimine izin verdiği (Allow/Disallow) teyit edilmelidir.
- **Rate Limiting (İstek Sınırlandırma):** Çok kısa sürede binlerce istek atıp hedef sunucuyu çökertmemek için döngüler arasına mutlaka bekleme süresi (**Delay / Sleep**) konulmalıdır. Aksi takdirde bu bir siber saldırı (DDoS) olarak algılanabilir.

Özetle: Web scraping, internet dünyasını devasa ve sınırsız bir veri tabanı gibi kullanmamızı sağlayan; Veri Bilimi ve Makine Öğrenmesi (ML) projeleri için veri seti (Dataset) oluştururken ihtiyaç duyulan en temel mühendislik yeteneklerinden biridir.

2 - Veri Çoğaltma (Data Augmentation) Nedir?

Veri Çoğaltma (Data Augmentation), makine öğrenmesi ve derin öğrenme modellerini eğitirken, elimizdeki mevcut veri setini (dataset) sentetik olarak büyüterek ve çeşitlendirerek modelin performansını artırma tekniğidir.

Özünde yaptığımız şey, mevcut veriler üzerinde mantıklı varyasyonlar ve dönüşümler uygulayarak, modelin **Aşırı Öğrenme (Overfitting)** yapmasını engellemek ve **Genelleştirme (Generalization)** yeteneğini güçlendirmektir.

2.1 Farklı Alanlarda Veri Çoğaltma Yaklaşımları

Kullanılan veri tipine göre çoğaltma stratejileri (transformation rules) farklılık gösterir:

- **1. Görüntü İşleme (Computer Vision):** En yaygın kullanıldığı alandır. Orijinal bir resme şu dönüşümler uygulanabilir:
  - *Geometrik Dönüşümler:* Döndürme (Rotation), Çevirme (Flipping - yatay/dikey), Yakınlaştırma (Zooming), Kırpma (Cropping).
  - *Renk Dönüşümleri:* Parlaklık (Brightness), Kontrast (Contrast), Renk Tonu (Hue) veya Doygunluk değişimleri.
- **2. Doğal Dil İşleme (NLP):** Metin verisini çoğaltmak daha risklidir çünkü cümlenin semantik anlamı bozulabilir.
  - *Eş Anlamlı Değişimi (Synonym Replacement):* Cümledeki kelimelerin yerine rastgele eş anlamlılarını koymak.
  - *Geri Çeviri (Back-translation):* Bir metni önce İngilizceden Fransızcaya, sonra o Fransızca metni tekrar İngilizceye çevirerek aynı anlamda farklı bir cümle yapısı elde etmek.
- **3. Ses İşleme (Audio Processing):**
  - Arka plana rastgele beyaz gürültü (White Noise) ekleme, sesin oynatma hızını (Time Stretching) veya perdesini (Pitch Shifting) değiştirme.

2.1.1 Basit Bir Matematiksel Gösterim

Veri çoğaltma işlemini matematiksel olarak ifade edersek; x girdi verisini, y ise bu verinin etiketini (label / ground truth) temsil etsin. Bir T dönüşüm (transformation) fonksiyonu uyguladığımızda:

$$x_{yeni} = T(x)$$

$$y_{yeni} = y$$

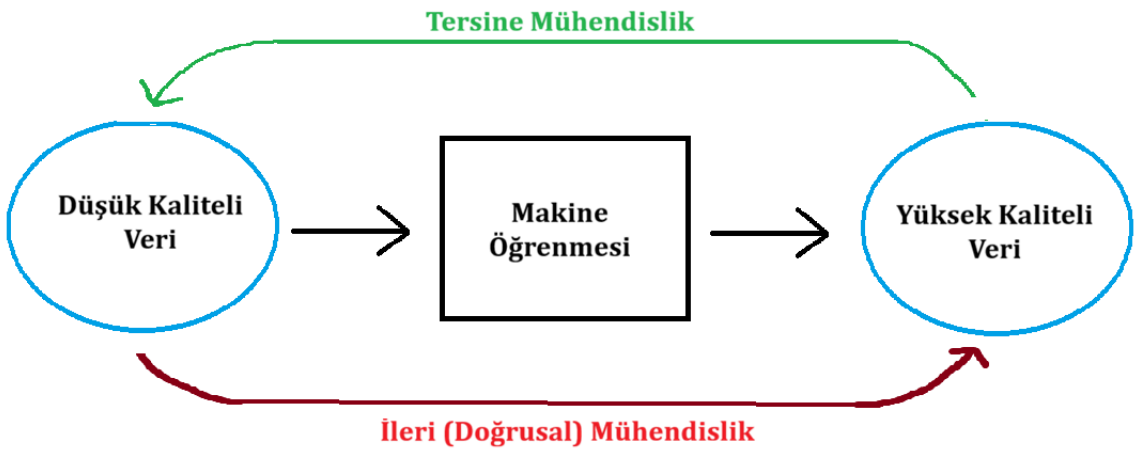
Buradaki en kritik nokta şudur: x_{yeni} orijinalinden oldukça farklı görünse bile, ona atanan etiket (y) **asla değişmemelidir**. (Baş aşağı döndürülmüş bir kedi fotoğrafı hala bir kedidir).

2.2 Senaryolaştırma

Diyelim ki bir otonom araç projesi için "Dur Tabelası" tanıyan bir **Konvolüsyonel Sinir Ağı (CNN)** modeli geliştiriyorsun ve elinde sadece 1.000 adet fotoğraf var.

- **Problemli Durum:** Elindeki fotoğrafların hepsi güneşli bir günde, tam karşıdan ve net çekilmiş. Model sadece bu koşulları öğrenir. Gerçek hayatta yağmurlu bir günde veya eğik duran bir tabelayı gördüğünde onu tanıyamaz.
- **Algoritmik Çözüm (Data Augmentation):** Kod aracılığıyla orijinal fotoğraflara rastgele yağmur damlası efektleri (gürültü) ekler, parlaklığı kısar ve fotoğrafları 15 derece sağa/sola döndürürsün. Veri setin bir anda 10.000 fotoğrafa çıkar. Model artık "karanlık ve eğik" duran kırmızı sekizgen objelerin de dur tabelası olduğunu öğrenir.

3 - Veri Üretiminde "Tersten Bakmak" (Reverse Engineering)



Makine öğrenmesinde bir modeli eğitmek için genellikle **Girdi $\rightarrow$ Çıktı (Input $\rightarrow$ Output)** eşleşmelerine ihtiyaç duyarız. Ancak gerçek hayatta bu eşleşmeleri hazır bulmak her zaman mümkün değildir. Görseldeki MR (Manyetik Rezonans) örneği, bu veri krizini "tersten düşünerek" nasıl çözdüğümüzü harika bir şekilde özetler.

3.1 Hedeflenen Sistem (Mavi Oklar)

- **Amaç:** Bir makine öğrenmesi (ML) modeline **Düşük Kaliteli MR** (Girdi) verip, modelin bunu netleştirerek **Yüksek Kaliteli MR** (Çıktı) üretmesini sağlamaktır. (Buna *Super-Resolution* denir).
- **Sorun (Veri?):** Bu modeli eğitmek için (Denetimli Öğrenme), elimizde kusursuz eşleşen "Düşük Kalite - Yüksek Kalite" fotoğraf çiftleri olmalıdır. Ancak hastanelerden aynı anın hem çok kötü hem de çok iyi çekilmiş MR görüntülerini bulmak imkansıza yakındır.

3.2 Çözüm: Veriyi Tersten Üretmek (Kırmızı Ok)

Eğitim verisi bulamadığımızda sistemi tersten çalıştırarak kendi verimizi kendimiz üretiriz:

- **Adım 1 (Kaliteyi Al):** Önce hastanelerden sadece en iyi, en net **Yüksek Kaliteli MR** görüntülerini toplarız (Örneğin 10.000 x 10.000 çözünürlüğünde devasa ve net görüntüler).
- **Adım 2 (Kasten Boz):** Bu mükemmel görüntüleri bir yazılım aracılığıyla kasten bozarız; çözünürlüklerini örneğin 1000 x 1000 piksele düşürür, üzerlerine bulanıklık ve gürültü (noise) ekleriz.
- **Adım 3 (Eşleştir):** Artık elimizde modeli eğitmek için gereken kusursuz veri çifti vardır: Kendi elimizle bozduğumuz "Sentetik Düşük Kaliteli MR" ve onun ilk hali olan "Gerçek Yüksek Kaliteli MR".

> **Özetle:** İstediğimiz girdiyi (düşük kalite) bulup onu çıktıya (yüksek kaliteye) eşleştirmek için kıvranmak yerine; elimizdeki kesin çıktıları (yüksek kalite) kasten bozarak kendi eğitim girdilerimizi (düşük kalite) yaratırız. Bu tersine mühendislik yaklaşımı, yapay zekada veri darboğazını aşmanın en etkili yollarından biridir.

4 - Gözetimsiz Öğrenme (Unsupervised Learning) ve K-Means Kümeleme

Gözetimsiz öğrenme, modele verilen veri setinde hedef değişkenlerin (etiketlerin) bulunmadığı durumdur. Modelin amacı, veriler arasındaki gizli örüntüleri, benzerlikleri ve yapıları kendi kendine keşfetmektir.

Görseldeki notlar, bu konuyu bir "Çiftlik" senaryosu üzerinden adım adım anlatmaktadır:

4.1 Problem Tanımı ve Girdi (Input)

- **Senaryo:** Bir çiftlikteki hayvanlardan (sensörler veya ölçümler aracılığıyla) veri toplanıyor. Elimizde 100.000 adet veri kaydı var.
- **Etiketsiz Veri:** Sistem, gelen ölçümlerin hangi hayvana ait olduğunu başlangıçta **bilmiyor**.

- **Hedef:** Çiftlikte 3 çeşit (Tavuk, Koyun, İnek) hayvan olduğunu biliyoruz. Amacımız bu büyük veri yığınını, özelliklerindeki benzerliklere göre **3 farklı gruba (çeşide)** ayırmaktır.

4.2 Veri Uzayı ve Özellikler (Features)

Veriler, hayvanların iki temel fiziksel özelliğine göre bir koordinat sistemine yerleştirilir:

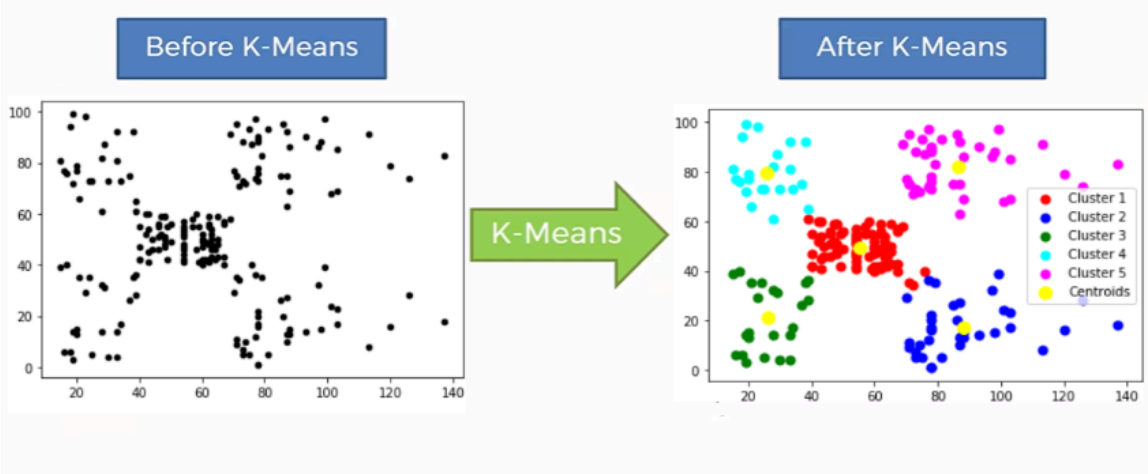
- **X Ekseni:** Boy
- **Y Ekseni:** Ağırlık
- Grafikteki her bir çarpı işareti (x), bir hayvanın boy ve ağırlık ölçümünü temsil eder.

4.3 Sonuçların Anlamlandırılması (Mapping)

Görselde yeşil renkle belirtilen **K-Means**, en popüler gözetimsiz öğrenme algoritmalarından biridir. Algoritma çalışmayı bitirdiğinde, grafiği sarı çizgilerle görüldüğü gibi 3 farklı bölgeye ayırır.

Gözetimsiz öğrenme sadece veriyi gruplar; bu grupların ne anlama geldiğini model bilmez. Kümeler oluşturtuktan sonra, fiziksel gerçekliğe (boy/ağırlık ilişkisine) dayanarak sonradan şu etiketleme/anlamlandırma yapılır:

- **B Grubu (Kırmızı - Sağ Üst):** Hem ağırlığı hem de boyu en yüksek olan grup. $\rightarrow$ **İnek**
- **A Grubu (Sarı - Sol Üst/Orta):** Ağırlık ve boy olarak orta seviyede olan grup. $\rightarrow$ **Koyun**
- **C Grubu (Yeşil - Sol Alt):** Boyu en kısa ve ağırlığı en düşük olan grup. $\rightarrow$ **Tavuk**



4.4 K-Means Algoritmasının İteratif Çalışma Mantığı

K-Means algoritması, verileri kümelere ayırmak için arka planda sürekli kendini güncelleyen (iteratif) bir deneme-yanılma süreci izler. Bu matematiksel süreç şu adımlardan oluşur:

1. **K Değerinin Belirlenmesi:** İlk olarak verinin kaç küme (grup) ayrılacağına karar verilir. Çiftlik senaryomuzda bu değer $K = 3$'tür.

2. **Rastgele Merkezlerin (Centroids) Atanması:** Koordinat sistemine (boy ve ağırlık grafiğine) rastgele konumda K adet (3 adet) merkez noktası fırlatılır.
3. **Mesafelerin Ölçülmesi ve Kümeye Atama:** Grafikteki her bir veri noktası (her bir hayvan ölçümü), bu 3 merkezden hangisine en yakınsa, o merkezin grubuna dahil edilir.
4. **Yeni Merkezlerin Hesaplanması (Güncelleme):** Kümeler geçici olarak oluştuktan sonra, her kümenin içindeki noktaların ortalaması (geometrik ağırlık merkezi) hesaplanır. Görselin alt kısmındaki formül bu durumu iki nokta için özetler:

$$Merkez = \left(\frac{x_1 + x_2 + \ldots + x_n}{n}, \frac{y_1 + y_2 + \ldots + y_n}{n} \right)$$

   Hesaplanan bu yeni ortalama değer, o kümenin **yeni merkezi** olur. Merkezler, başlangıçtaki rastgele konumlarından bu yeni "gerçek" merkeze doğru kayar.
5. **Tekrar (İterasyon):** Merkezler yer değiştirdiği için, bazı veri noktaları artık komşu kümelerin merkezine daha yakın hale gelmiş olabilir. Bu nedenle 3. ve 4. adımlar tekrar edilir; noktalar yeniden en yakın merkeze atanır ve merkezler tekrar hesaplanır.
6. **Durma Koşulu (Convergence):** Merkez noktaları artık hareket etmiyorsa (veya çok çok az hareket ediyorsa) ve hiçbir veri noktası küme değiştirmiyorsa algoritma durur. Algoritma optimum noktaya ulaşmıştır ve kümeler nihai halini almıştır.

Not : Bu konuya ilişkin farklı tarzda notlar 2. derse ilişkin pdfte mevcut.

5 - K-Means Kümeleme (Clustering) Algoritması: Görsel ve Matematiksel Açıklama

5.1 Kümeler (Clusters) Nedir?

- Kümeler, benzer özelliklere sahip veri noktalarının oluşturduğu gruplardır [00:00:00].
- Bir küme içindeki veri noktaları, birbirlerine diğer kümelerdeki noktalardan daha çok benzerler [00:00:08].

5.2 K-Means Algoritmasının Adım Adım Çalışma Mantığı

K-Means algoritması, veri setindeki kümeleri bulmak için iteratif (tekrarlayan) bir süreç izler:

- **Başlatma:** İlk adımda, oluşturulmak istenen küme sayısı kadar (K) merkez noktası (centroid) rastgele yerleştirilir [00:00:36].
- **Atama Adımı:** Her bir veri noktasının merkezlere olan uzaklığı hesaplanır ve her veri noktası kendisine en yakın olan merkeze atanır [00:00:57].
- **Güncelleme Adımı:** Tüm noktalar atandıktan sonra, her kümenin merkezi, o kümeye ait noktaların ortalama konumuna (geometrik merkezine) kaydırılır [00:01:16].
- **Döngü:** Merkezlerin anlamlı bir şekilde hareket etmesi durana kadar atama ve güncelleme işlemleri sürekli tekrarlanır [00:01:23].

5.3 Matematiksel Arka Plan ve Maliyet Fonksiyonu

- **Mesafe Hesaplaması:** Algoritma uzaklıkları ölçerken ham mesafeyi almak yerine, matematiksel kolaylık sağlaması açısından mesafenin karesini alır (Squared Distance) [00:02:13].
- **Maliyet Fonksiyonu (Cost Function - J):** K-Means aslında bir maliyet fonksiyonunu (küme içi kareler toplamı) minimize etmeye çalışır [00:03:06]. Bu fonksiyon, noktaların atandıkları merkeze ne kadar uzak olduğunu ölçer. Algoritma her adımda bu maliyeti azaltmayı hedefler [00:03:30].

5.4 K Değerini (Küme Sayısını) Belirleme Stratejileri

Veriyi kaç kümeye ayıracağımıza (K değerine) karar vermek için iki temel yaklaşım kullanılır:

- **Elbow (Dirsek) Metodu:** Farklı K değerleri için maliyet fonksiyonu hesaplanıp grafiğe dökülür. Maliyet düşüşünün belirgin bir şekilde azaldığı ve grafiğin "dirsek" yaptığı nokta, optimal K değeri olarak seçilir [00:04:03].
- **Alan Bilgisi (Domain Knowledge):** Sadece matematiksel yöntemlere güvenmek yerine, verinin kullanım amacı ve bağlamı da dikkate alınmalıdır. Örneğin; şehirleri gruplarken analizinizin detay seviyesine göre 3 veya 5 küme seçmek tamamen sizin probleminize bağlıdır [00:04:47].

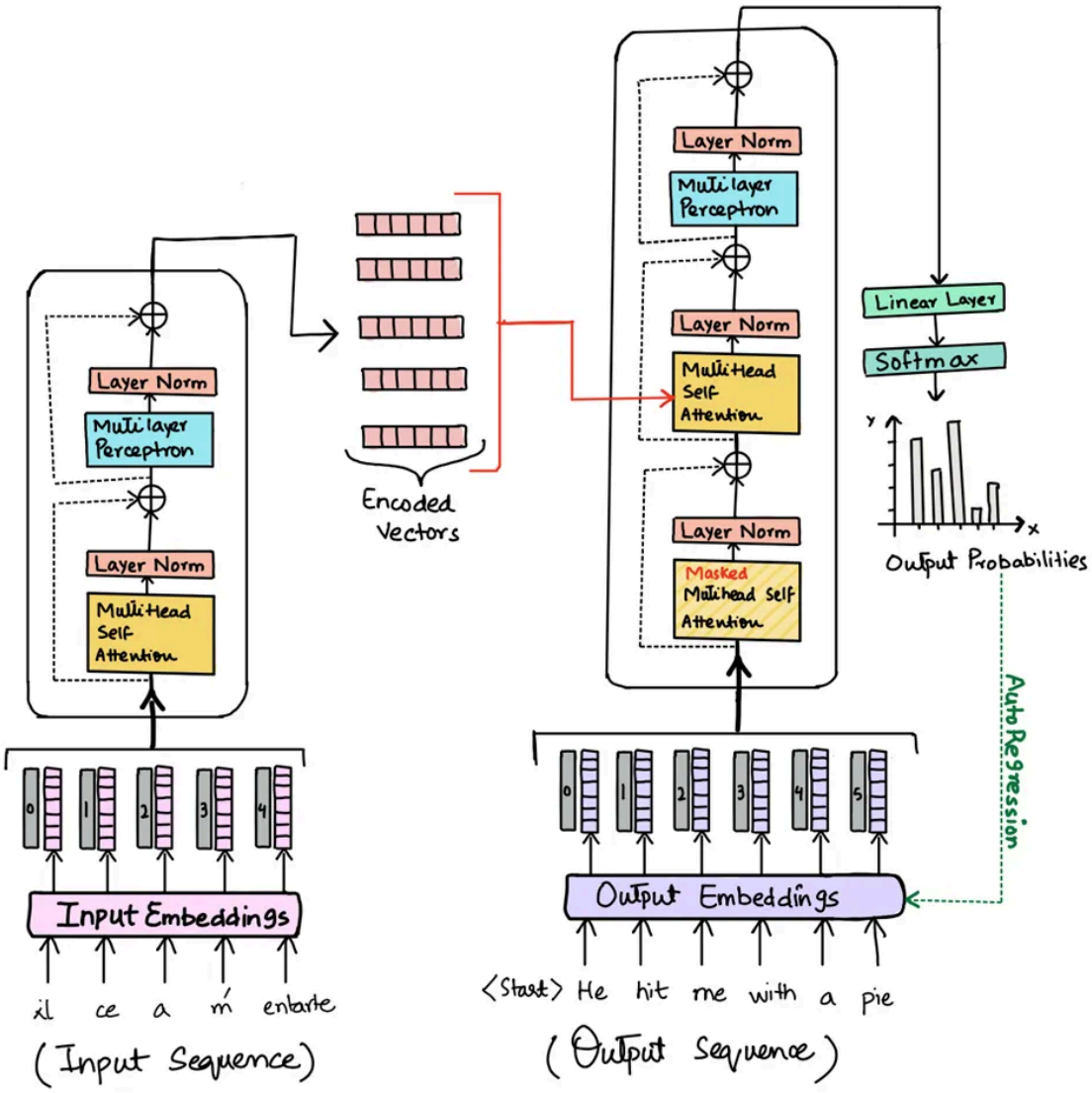
5.5 Merkezleri Başlatma (Centroid Initialization) Optimizasyonu

- **Daha İyi Başlatma:** Merkezleri uzayda tamamen rastgele yerleştirmek yerine, veri seti içinden rastgele seçilen K adet veri noktasını ilk merkezler olarak atamak, algoritmanın çok daha hızlı ve mantıklı sonuçlara ulaşmasını (convergence) sağlar [00:05:35].
- **Farklı Sonuçlar Sorunu:** İlk merkezlerin rastgele seçilmesi, algoritma her çalıştığında farklı kümelenme sonuçları (Local Optima) üretmesine yol açabilir [00:06:43].
- **Çözüm:** Bu sorunu aşmak için K-Means yüzlerce farklı rastgele başlatma ile tekrar tekrar çalıştırılır ve en sonunda maliyet fonksiyonu (J) en düşük olan kümelenme sonucu nihai olarak seçilir [00:07:06].

Not 2 : Bu nota ilişkin youtube videosunu izlemek isterseniz link : https://www.youtube.com/watch?v=3RAEwPiZkkw

6 - LLM'lerin Arkasındaki Teknoloji "Transformer" Mimarisi

Transformer, 2017 yılında Google tarafından icat edilmiş ve günümüzde ChatGPT gibi Büyük Dil Modellerinin (Large Language Models - LLM) temelini oluşturan spesifik bir yapay sinir ağı mimarisidir. Temel amacı, verilen bir metin parçasını alıp, **bir sonraki kelimenin ne olacağına dair bir olasılık dağılımı** üretmektir.



6.1 Tokenizasyon ve Kelime Gömülmeleri (Word Embeddings)

- **Tokenizasyon:** Girdi metni "token" adı verilen küçük parçalara (kelimeler, heceler veya karakter grupları) bölünür.
- **Gömülme Matrisi (Embedding Matrix):** Modelin kelime dağarcığındaki (vocabulary) her bir token, çok yüksek boyutlu bir uzayda bir vektöre (sayı listesine) dönüştürülür. Örneğin GPT-3'te bu uzay 12.288 boyutludur.
- **Anlamsal Uzay:** Bu vektör uzayında yönler ve mesafeler kelimelerin **anlamlarını** taşır. Birbirine anlamsal olarak yakın kelimelerin vektörleri uzayda birbirine yakındır. (Örn: `Kral - Erkek + Kadın = Kraliçe` veya `İtalya - Almanya + Hitler = Mussolini` gibi matematiksel işlemler anlamsal karşılık bulur).

6.2 Transformer'ın Kalbi: Bloklar

Kelime vektörleri oluşturulduktan sonra, bağlamı (context) anlamak üzere sırayla iki temel yapıdan defalarca geçer:

- **Dikkat Blokları (Attention Blocks):** Vektörlerin (kelimelerin) birbirleriyle iletişim kurmasını, birbirlerine bilgi aktarmasını ve cümlede hangi kelimelerin birbirini etkilediğini bulmasını sağlar.
- **İleri Beslemeli Ağlar (Multi-Layer Perceptron / Feed Forward):** Vektörlerin birbirleriyle konuşmadan, her birinin paralel olarak kendi başına matematiksel işlemlerden geçip anlamsal olarak daha da rafine hale gelmesini sağlar.
- *Not: Tüm bu işlemler devasa matris çarpımlarından ibarettir ve bu matrislerin içindeki sayılar (ağırlıklar / weights) model eğitilirken öğrenilir.*

6.3 Çıktı Üretimi: Gömülmeyi Geri Alma (Unembedding)

Metnin sonundaki kelime vektörü, tüm cümlenin bağlamını içine çekmiş, çok zengin bir anlama sahip devasa bir vektör haline gelir.

- **Unembedding Matrisi:** Bu son vektör, modelin kelime dağarcığındaki (örn. 50.000 kelime) her bir kelime için bir "skor" (Logit) üretecek şekilde başka bir matrisle çarpılır.

6.4 Softmax ve Sıcaklık (Temperature) Parametresi

- **Softmax Fonksiyonu:** Unembedding matrisinden çıkan ham skorlar (Logits), Softmax fonksiyonundan geçirilerek toplamları 1 olan, 0 ile 1 arasında değişen **olasılık değerlerine** dönüştürülür. Böylece "bir sonraki kelime %80 ihtimalle X, %15 ihtimalle Y olacak" gibi bir dağılım elde edilir.
- **Sıcaklık (Temperature - T):** Olasılık dağılımı hesaplanırken matematiksel formüle bir "T" sabiti eklenebilir.
  - **Düşük Sıcaklık (T → 0):** Model her zaman en yüksek olasılıklı kelimeyi seçer. Çıktı çok mantıklı, garantici ama sıkıcı/tekdüze olur.
  - **Yüksek Sıcaklık:** Modelin daha düşük olasılıklı kelimeleri seçme şansı artar. Çıktı daha "yaratıcı" olur, ancak çok yükseltilirse saçmalamaya (halüsinasyon) başlar.

7 - Transformer Mimarisinde "Attention" (Dikkat) Mekanizması

Attention mekanizmasının temel amacı, kelimelerin vektörlerini etrafındaki diğer kelimelerden (bağlamdan) aldığı bilgilerle güncelleyerek onlara çok daha zengin bir anlam kazandırmaktır. Notun çıkartıldığı videonun linki : https://www.youtube.com/watch?v=eMlx5fFNoYc

7.1 Vektör Uzayı ve Kelime Gömülmeleri (Embeddings)

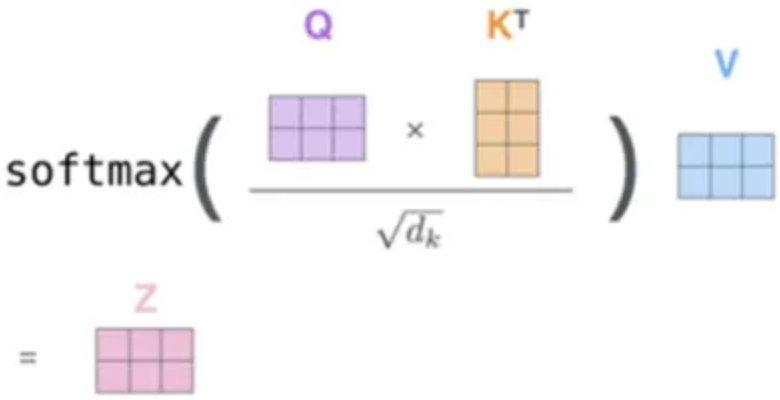
- **Bağlam Eksikliği:** İlk aşamada oluşturulan vektörler (Bkz 6.1) sadece kelimenin yalın halini temsil eder, bağlamı (context) bilmezler. Örneğin; "yüz" kelimesinin çehre mi, sayı mı yoksa yüzmek eylemi mi olduğu ilk vektörde belli değildir.

7.2 Attention (Dikkat) Mekanizmasının Amacı

7.2.1 "Query, Key ve Value" (Sorgu, Anahtar ve Değer) Mantığı

Attention süreci, her kelime için hesaplanan 3 farklı vektör üzerinden işler:

- **Query (Sorgu - Q):** Bir kelimenin "Benim ne tür bir bilgiye/kelimeye ihtiyacım var?" diye sormasını temsil eder.
- **Key (Anahtar - K):** Diğer kelimelerin "Bende ne tür bir bilgi var, ben neyim?" diye cevap vermesini temsil eder.
- **Dot Product (Nokta Çarpımı) ve Skorlama:** Query ve Key vektörleri birbiriyle çarpılarak bir uyum skoru (Attention Pattern) hesaplanır. Puan ne kadar yüksekse, o kelimeler birbirleriyle o kadar alakalı demektir.
- **Softmax Adımı:** Hesaplanan bu skorlar, Softmax fonksiyonundan geçirilerek 0 ile 1 arasında olasılık değerlerine dönüştürülür (Normalize edilir).
- **Value (Değer - V):** Eğer iki kelime eşleştiyse (skor yüksekse), "Peki bu kelimenin anlamını tam olarak nasıl güncellemeliyim?" sorusunun cevabıdır. Value vektörleri, Softmax'ten çıkan ağırlıklarla çarpılarak toplanır ve orijinal kelime vektörüne eklenerek kelimenin anlamı zenginleştirilir.



7.3 Maskeleme (Masking)

- Eğitim sırasında modelin geleceği görüp hile yapmasını engellemek için, bir kelimenin kendinden sonra gelen kelimelerden etkilenmesi engellenir.
- Bunu yapmak için, henüz gelmemiş kelimelerin Query-Key eşleşme skorları Softmax işleminden önce -∞ (eksi sonsuz) yapılır, böylece olasılıkları 0'a eşitlenir.

7.4 Çok Başlı Dikkat (Multi-Headed Attention)

- Model sadece tek bir Attention işlemi yapmaz. Örneğin GPT-3'te yan yana çalışan 96 farklı "Attention Head" (Dikkat Başı) bulunur.
- Her bir baş, bağlamın farklı bir yönüne odaklanır. Biri sıfat-isim ilişkisine bakarken, diğeri gramer yapısına, bir başkası ise duygu durumuna (sentiment) dikkat edebilir.

- Tüm bu başlardan çıkan güncellemeler birleştirilerek kelime vektörüne son ve çok zengin hali verilir.

7.5 Kritik Transformer Kavramları ve Mimari Yapı Taşları

7.5.1 Ölçeklendirilmiş Nokta Çarpımı (Scaled Dot-Product) ve Sayısal Kararlılık

- **Sorun:** Query ve Key vektörlerinin nokta çarpımı yapıldığında elde edilen skorlar çok büyük sayılara ulaşabilir. Bu durum, Softmax fonksiyonuna girdiğinde gradyanların kaybolmasına ve modelin öğrenememesine yol açar.
- **Çözüm:** Matematiksel kararlılığı sağlamak için, elde edilen nokta çarpımı skorları, Key/Query uzayının boyutunun karekköüne bölünerek normalize edilir.

7.5.2 Bağlam Penceresi Darboğazı (Context Size Bottleneck)

- **Kavram:** Dikkat matrisinin boyutu, bağlamdaki kelime sayısının karesine eşittir (N x N).
- **Etkisi:** Modelin girdi metni uzadıkça hesaplama maliyetinin (işlem gücü ve bellek ihtiyacı) katlanarak artması anlamına gelir.

7.5.3 Konum Kodlama (Positional Encoding)

- **Kavram:** Transformer modelleri veriyi kelime kelime sırayla okumaz; tüm kelimeleri aynı anda, paralel olarak işler.
- **İşlevi:** Model kelimelerin sırasını doğal olarak bilemeyeceği için, ilk adımdaki kelime vektörlerine (embedding) kelimenin anlamsal bilgisinin yanı sıra cümlede **hangi sırada/konumda** bulunduğunu belirten matematiksel bir bilgi (sinüs/kosinüs dalgaları ile) eklenir.

7.5.4 Value Matrisinin Boyut İndirgemesi (Value Down / Value Up)

- **Kavram:** Verimlilik sağlamak adına, tek bir devasa "Value" matrisi kullanmak yerine bu işlem iki küçük matrise bölünür.
- **İşleyiş:** Önce büyük embedding vektörü `Value Down` matrisi ile daha küçük bir boyuta indirgenir, güncellemeler hesaplanır ve ardından `Value Up` matrisi ile tekrar orijinal boyuta genişletilir.

7.5.5 Öz-Dikkat (Self-Attention) vs. Çapraz-Dikkat (Cross-Attention)

- **Self-Attention:** Bir metnin kendi içindeki kelimelerin birbirleriyle olan ilişkisini inceler. Query, Key ve Value aynı veri kaynağından gelir.
- **Cross-Attention:** Çeviri veya sesten metne gibi görevlerde kullanılır. Örneğin; Query'ler üretilen yeni dildeki kelimelerden gelirken, Key ve Value'lar kaynak dildeki kelimelerden gelir.

7.5.6 İleri Beslemeli Ağlar (Multi-Layer Perceptron - MLP)

- **İşlevi:** Attention mekanizması kelimeler "arasındaki" ilişkiyi kurarken; ardından gelen İleri Beslemeli Sinir Ağları (MLP), her bir kelimenin kendi anlamsal temsilini bağımsız olarak daha soyut ve karmaşık bir seviyeye taşır.

7.5.7 Artık Bağlantılar (Residual / Skip Connections)

- **Sorun:** Modelin katmanları derinleştikçe ilk başlardaki kelime bilgisi silikleşebilir veya veri kaybolabilir.
- **Çözüm/İşlev:** Veri bir katmana girmeden önceki orijinal hali, o katmandan çıkan işlenmiş sonuca doğrudan eklenir. Ağın, ihtiyaç duymadığı işlemleri es geçmesini ve bilginin derinlere kayıpsız akmasını sağlar.

7.5.8 Katman Normalizasyonu (Layer Normalization)

- **Sorun:** Milyarlarca işlemin sonucunda verilerin sayısal değerleri çok büyüyebilir/küçülebilir.
- **Çözüm:** Veri her bir Attention veya MLP bloğundan çıktığında normalize edilir. Vektördeki değerlerin ortalaması 0, standart sapması 1 olacak şekilde ayarlanır. Modelin stabil eğitilmesini sağlar.

7.5.9 Doğrusal Olmayan Aktivasyonlar (GELU / ReLU)

- **İşleyiş:** İleri Beslemeli Ağların (MLP) kalbinde, veriyi belirli bir eşik değerinden geçiren ve modele doğrusal olmayan özellikler katan aktivasyon fonksiyonları bulunur.
- **Modern Standart:** Günümüzdeki gelişmiş LLM'ler olasılıksal bir düzleştirme sunan **GELU (Gaussian Error Linear Unit)** aktivasyonunu kullanır.

7.5.10 Ailenin Diğer Üyeleri: Encoder vs. Decoder Mantığı

- **Sadece Decoder (GPT Ailesi):** Maskeleme kullanır, sadece geçmiş kelimelere bakar ve bir sonraki kelimeyi tahmin eder. Üretken görevlerde kullanılır.
- **Sadece Encoder (BERT Ailesi):** Maskeleme kullanmaz. Cümleyi hem soldan sağa hem de sağdan sola okuyarak bağlamı anlar. Metin sınıflandırma veya duygu analizi gibi görevlerde kullanılır.
- **Encoder-Decoder (Orijinal Transformer / T5):** Hem metni bütünüyle anlama hem de yeni dilde metin üretme kapasitesine sahip olduğu için çeviri (Translation) görevlerinde kullanılır.

Güzel -> https://medium.com/machine-intelligence-and-deep-learning-lab/transformer-the-self-attention-mechanism-d7d853c2c621

-Memreu