Sorunun Cevabı Nerede?

Yavuz Selim Doğdu
Fen Bilimleri Enstitüsü
Yıldız Teknik Üniversitesi
Davutpaşa 34220, İstanbul, Türkiye
yavuzselimdogdu@hotmail.com

***Özet*—Bu çalışma, kendi hazırladığımız bir veri kümesi üzerinde doğal dil işleme (NLP) alanındaki anlamsal analiz tekniklerini değerlendirmeyi amaçlamaktadır. Veri kümemiz, Türkçe metinlerden ve bu metinlerle ilgili sorulardan oluşmaktadır. Ana hedefimiz, TF-IDF, Türkçe BERT ve mGPT gibi farklı sistemleri kullanarak veri kümemizde bulunan soruların hangi metinlerde cevaplandığını, sorular ile metinler arasındaki kosinüs benzerliklerini hesaplayarak bulmak ve bu sistemlerin performanslarını karşılaştırmaktır.**

I. GİRİŞ

Gelişen teknoloji, bilgiye erişim şeklimizi de köklü bir şekilde değiştirmiştir. Bu çalışmada modern bilgi işleme alanında önemli bir yere sahip olan vektör temsilleri ve bunların kosinüs benzerlikleri üzerinden metinler ile sorular arasındaki ilişkileri inceleyerek cevapların hangi metinlerde bulunduğunu tahmin eden sistemler geliştireceğiz. Gündelik hayatta sıkça karşılaştığımız kullanıcıların sorularına en uygun içeriği sunan arama motorları, bu teknolojinin somut bir örneğidir. Çalışmamızda, bu benzerlik hesaplamalarını gerçekleştirmek için üç farklı sistemden metinlerin vektör temsillerini alacağız: Term Frequency-Inverse Document Frequency (TF-IDF), Bidirectional Encoder Representations from Transformers (BERT) ve modified Generative Pre-trained Transformer (mGPT).

Yazının ilerleyen bölümlerinde, her bir sistemin teorik temellerini ve metinlerin sayısal temsillerinin nasıl alındığını derinlemesine inceleyeceğiz. Ayrıca bu sistemlerin pratik uygulamalarındaki avantajları ve sınırları üzerinde duracağız. Böylece, sistemlerin görevimiz üzerindeki performansları için genel bir izlenim oluşturacağız.

II. VERİ KÜMESİ

Doğal dil işleme alanındaki araştırmalar, özellikle farklı dillerdeki veri kümesi çeşitliliğinin artmasıyla son yıllarda büyük ilerlemeler kaydetti. Ancak Türkçe gibi bazı dillerde herkese açık ve kaliteli soru-cevap metin veri kümelerinin eksikliği dilimizde NLP uygulamalarının gelişimini sınırlamakta. Bu durum, araştırmamızda kullandığımız veri kümesini oluşturma sürecinde temel motivasyon kaynağımız oldu.

Projemiz için gerekli Türkçe soru-cevap veri kümesini oluştururken bazı popüler kaynaklardan yararlandık. Bunlardan ilki Türkçe Wikipedia sayfaları oldu. Wikipedia, geniş kapsamlı ve çoğu zaman güvenilir bilgi içeren, çeşitli konularda yazılmış metinleri barındıran bir kaynaktır. Bu platform, doğal dil işleme uygulamaları için zengin bir metin kaynağı sunar. Wikipedia'dan seçilen metinleri işleyerek projemizin temel veri kümesini oluşturduk. Projenin gereksinimlerindeki her metnin 2048 token'dan uzun olma şartını, metinleri GPT 4 Token Counter aracı ile ölçerek karşıladık.

İkinci olarak, Stanford Üniversitesi'nin İngilizce olarak hazırlamış olduğu Stanford Question Answering Dataset (SQuAD) veri kümesinde bulunan bazı metinleri Türkçe diline çevirerek veri kümemize dahil ettik. Çeviri sürecinde, metinlerin doğallığını ve anlam bütünlüğünü korumaya özen gösterdik.

Sonuç olarak, bu iki farklı kaynaktan elde edilen metinleri birleştirerek ufak çaplı Türkçe NLP uygulamaları için değerli bir kaynak oluşturduk. Bu veri kümesi, Türkçe metin işleme teknolojilerinin gelişimi için ufak bir adım oluşturmakta ve gelecekteki NLP çalışmalarına katkı sağlamayı hedeflemektedir.

III. VEKTÖRLEŞTİRME

Vektörleştirme, verileri ham formatından bilgisayarların anlayabileceği gerçek sayılardan oluşan vektörlere dönüştürme işlemidir. Projemizde vektörleştirme işlemi için TF-IDF, BERT ve mGPT gibi sistemleri kullanacağız fakat bu işlemi gerçekleştirebilmek için daha pek çok yöntem bulunuyor. Bunlardan bazıları:

A. Bag of Words

Vektörleştirme yöntemleri arasında en basitlerinden biridir. Üç aşamadan oluşur. Bunlar:

1- Tokenleştirme: Her cümle kendisini oluşturan kelimelerin listesiyle temsil edilir.

2- Sözlük oluşturma: Toplanan tüm kelimelerden eşsiz olanları ile bir sözlük oluşturulur ve alfabetik olarak sıralanır.

3- Vektör oluşturma: Sonuç olarak cümledeki kelimelerin frekansını temsil eden matrisler oluşturulur.

B. Word2Vec

BoW ve TF-IDF gibi sistemlerde her kelime bireysel bir girdi olarak kabul ediliyordu ve anlamları göz ardı ediliyordu. Word2Vec, kelimelerin anlamlarını ve birbirleriyle olan ilişkilerini de sayısal vektörlerde temsil edebilir. Bu yöntemde, bir kelimenin anlamı, onu çevreleyen kelimelerle belirlenir. Bağlamdan gelen bu anlamları öğrenebilmek için bir tür yapay sinir ağı kullanılır.

Bu yöntem, yüksek boyutlu seyrek vektörler yerine düşük boyutlu yoğun vektörler üretir ve teorik olarak TF-IDF, BoW gibi yöntemlerden daha başarılı sonuçlar vermesi beklenir.

C. *GloVe (Global Vectors for Word Representation)*

Bu yöntem, Word2Vec gibi kelimelerin anlamlarını bağlama göre belirlerken aynı zamanda kelimelerin bütün veri kümesi içerisindeki eş zamanlılığını da dikkate alır.

Bir kelimenin başka kelimelerle olan eş zamanlılığı, o kelimenin diğer kelimelerle ne kadar sık ve hangi bağlamlarda kullanıldığını gösterir. Örneğin "kahve" ve "fincan" kelimeleri sıklıkla birlikte kullanılır çünkü bu kelimeler genellikle aynı bağlamda yer alır. Dolayısıyla bu iki kelimenin eş zamanlılık değeri, bağlam dışındaki herhangi bir kelimeye göre daha yüksek olacaktır.

D. *FastText*

Word2Vec ve GloVe gibi modeller milyarlarca kelime ile eğitilmiş olsa da yeni girdilerimiz için her zaman bir sınırlama bulunuyordu. Bu da, modelin eğitildiği kelimeler sözlüğüydü. FastText ise bilinmeyen kelimelerde de yüksek genelleme yeteneği göstererek diğer modellerin bir adım önüne geçmiştir. FastText, bunu başarmak için embedding'leri kelimelerden oluşturmak yerine bir seviye daha derine inerek harfleri kullanmıştır.

IV. SİSTEMLER

Projede kullanılan vektörleştirme yöntemleri olan TF-IDF, BERT ve mGPT sistemlerini bu bölümde detaylı bir şekilde inceleyeceğiz.

A. *Term Frequency-Inverse Document Frequency (TF-IDF)*

TF-IDF, bir kelimenin metindeki önemini hem metindeki sıklığına (TF) hem de tüm metin setindeki nadirliğine (IDF) göre değerlendirir. Özellikle arama motorları, doküman sınıflandırma ve içerik önerme sistemlerinde kullanılır.

Terim Frekansı (TF): Bir kelimenin metindeki tekrar sayısını, metindeki toplam kelime sayısına bölerek hesaplanır. Böylece metindeki kelimenin göreceli önemi ortaya çıkar.

Ters Doküman Frekansı (IDF): Bir kelimenin kaç farklı metinde geçtiğini tespit edilir ve bu sayı, toplam metin sayısına bölünerek belge frekansı hesaplanır. Kelime çok kullanılmışsa önemsiz olacağı için bu belge frekansının tersi (IDF) bulunur. Daha sonra elde edilen bu değerlerin logaritması alınır. Böylece yaygın olarak kullanılan kelimelerin ağırlığı azalır ve nadir kelimelerin ağırlığı artar.

Matematiksel olarak ifade etmek gerekirse, TF-IDF puanı 't' kelimesi için 'd' dokümanında ve 'D' doküman setinde şu şekilde hesaplanır:

$$tf\ idf(t,d,D) = tf(t,d).idf(t,D)$$

B. *Türkçe BERT*

BERT, Google tarafından geliştirilmiş bir doğal dil işleme modelidir. Projede kullandığımız model, BERT modelinin Türkçe diline özelleştirilmiş bir sürümüdür. Çeşitli Türkçe veri kümeleri kullanılarak toplam 4 milyar token ile eğitilmiştir.

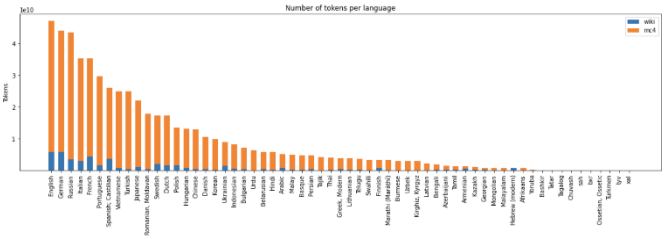
BERT, kelimelerin bağlamını anlamada çift yönlü bir yaklaşım benimser. Bu, bir kelimenin hem önceki hem de sonraki kelimelerle olan ilişkisini anlamlandırma kapesitesi demektir. Metin sınıflandırma, duygu analizi ve soru-cevap sistemleri gibi çeşitli NLP görevlerinde kullanılabilir.

C. *mGPT*

mGPT, 1,3 milyar parametreye sahip, 61 dilde ve 25 dil ailesinden oluşan, Wikipedia ve C4 veri kümeleriyle eğitilmiş GPT benzeri bir modeldir. GPT 2 kaynaklarını ve seyrek dikkat mekanizmasını kullanarak GPT 3 mimarisini yeniden üretir.

Seyrek dikkat mekanizması, tam dikkat mekanizmasına göre modelin çok uzun dizilere daha verimli ölçeklenmesine olanak tanır. Tam dikkat mekanizmasında, dizideki her eleman diğer elemanlara da bağlanır ve bu durum dizinin uzunluğuna bağlı ikinci dereceden bir karmaşıklık getirir. Seyrek dikkat mekanizmasında ise her elemanın bağlanabildiği bir diğer eleman sayısını kısıtlayarak karmaşıklığı azaltır.

Model, 600 GB metin verisi üzerinde eğitilmiş ve tüm diller için toplam 440 milyar token görmüştür. Modelin eğitildiği Türkçe veri kümesinin net büyüklüğü hakkında, makalesinde veya herhangi bir harici kaynakta belirtilmediğinden net bir bilgimiz yok. Ancak model için oluşturulan aşağıdaki görselde, farklı dillerin token dağılımlarına baktığımızda yaklaşık 25 milyar token'ın sadece Türkçe eğitiminde kullanıldığını varsayabiliriz:



V. DENEYLER

Projede, bu üç sistemi kullanarak metinlerin ve ilgili soruların vektör temsillerini alıyoruz. Sonrasında bu temsilleri kullanarak sorular ile metinler arasındaki benzerlikleri hesaplıyoruz. Elde ettiğimiz değerlere göre, soruların hangi metinlerde cevaplandığını bulup farklı doğruluk kriterlerine göre sistemlerin performansını karşılaştırıyoruz.

Metinlerin birbirine ne kadar benzer olduğunu hesaplamak için birçok yöntem bulunuyor. Bunlardan bazıları; kosinüs benzerliği, öklid uzaklığı ve jaccard benzerliğidir. Bu yöntemlerin her birinin kendine has spesifikasyonları bulunmaktadır. Projemizde kullanacağımız benzerlik ölçme yöntemi ise kosinüs benzerliğidir.

A. *Kosinüs Benzerliği*

Kosinüs benzerliği, iki vektör arasındaki benzerliği veya iki nesnenin ne kadar benzer olduğunu ölçen bir metriktir. Genellikle metin analizi, öneri sistemleri ve öğe sınıflandırma gibi alanlarda kullanılır. Temel olarak, iki vektör arasındaki kosinüs açısının değerini hesaplayarak bu iki vektörün birbirine olan benzerliğini belirler.

A ve B vektörlerinin kosinüsü, bu vektörlerin skaler çarpımının, vektör normlarının çarpımına bölümüyle bulunur. Kosinüsü 1'e yakın vektörler yüksek derecede benzerlik olduğunu gösterir, 0'a yakın vektörler çok az veya hiç benzerlik olmadığını gösterir ve -1'e yakın değerler ise iki vektörün birbirine tamamen zıt olduğunu gösterir.

$$Kosinüs\ Benzerliği = \frac{A.B}{|A||B|}$$

B. Deney 1: TF-IDF ile Kosinüs Benzerliği

Projemizde, metin içeriklerini sayısal vektör temsillerine dönüştürmek için Scikit-learn kütüphanesinden yararlandık. Veri kümesindeki dokümanlar ve ilgili sorular üzerinde çalışmak için öncelikle "TfidfVectorizer" sınıfını kullanarak bir vektörleştirme işlemi gerçekleştirdik. Bir örnek üzerinden göstermek gerekirse:

Üç farklı cümlemiz olduğunu varsayalım. Bunlar:

"Gökyüzü mavi.",

"Deniz mavi ve uçsuz.",

"Yeşil dağlar mavi gökyüzü ile buluşuyor."

Kullandığımız sınıf, bu cümlelerde geçen tüm kelimeleri kullanarak bir sözlük oluşturdu:

[

'buluşuyor', 'dağlar', 'deniz', 'uçsuz',

'gökyüzü', 'ile', 'mavi', 've', 'yeşil'

]

Sonrasında, her cümle için kelimelerin ağrılıklarını belirten değerler ile vektörler oluşturulur. Örneğin, "Gökyüzü mavi." cümlesinin vektör temsili:

[0, 0, 0, 0, 0.7898, 0, 0.6134, 0, 0]

Sözlükte bulunan tüm kelimeler kontrol edilir ve olmayan kelimeler için 0 değeri atanır. Görüldüğü üzere, ilk cümle için "Gökyüzü" kelimesinin ağırlığı "0.7898" ve "mavi" kelimesinin ağırlığı ise "0.6134".

Dokümanların ve soruların vektör temsillerini elde ettikten sonra yine Scikit-learn kütüphanesinden "cosine_similarity" sınıfıyla bu vektörler arasındaki kosinüs benzerliklerini hesapladık. Her soru için veri kümemizdeki tüm metinlerle olan benzerliklerini karşılaştırarak en alakalı metni belirledik.

Daha sonrasında, her bir soru-metin eşleşmesi için elde edilen tahminlerin doğruluğunu değerlendirdik. Bu süreçte her tahminin doğruluğu ve doğru cevapların ne kadar öncelikli bulunduğu, sorular ve cevaplarının bulunduğu asıl metinlerle karşılaştırılarak, Mean Reciprocal Rank (MRR) ve Precision metrikleri kullanılarak ölçüldü.

Mean Reciprocal Rank, modelin cevabın bulunduğu doğru metni ne kadar önce bulduğunu gösteren bir metriktir.

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

Örnek üzerinden anlatmak gerekirse, aşağıdaki üç sorgu örneğini ele alalım:

Sorgu Kelimeleri	Önerilen Cevaplar	Doğru Cevap	Sıra	Karşıt Sıra
Kedi	Köpek, Yavur, **Kediler**	Kediler	3	$\frac{1}{3}$
Virüs	Bakteri, **Virüsler**, Covid	Virüsler	2	$\frac{1}{2}$
Simit	**Simit**, Poğaça, Açma	Simit	1	1

Bu örneklere bakarak MRR değerini şu şekilde hesaplayabiliriz:

$$\frac{(\frac{1}{3}+\frac{1}{2}+\frac{1}{1})}{3} = \frac{11}{18} \approx 0{,}61$$

MRR değerinin yüksek olması, sistemin doğru cevapları üst sıralarda bulduğunu gösterir.

Precision değeri ise modelin yaptığı soru-metin tahminlerinden kaçını doğru yaptığını gösterir. Örneğin, modelimiz 90 sorudan 70'ini doğru metinle eşleştirmişse Precision değerimiz:

$$\frac{70}{90} \approx \%77{,}78$$

Veri kümemizdeki 100 metinle ilgili 300 soruyu ve sorularla ilişkisi olmayan ek 1000 metni TF-IDF sistemine verdiğimizde elde ettiğimiz performans kriterleri şu şekilde:

TF-IDF	
Metrik	Değer
Mean Reciprocal Rank	0,83
Precision	%76,8

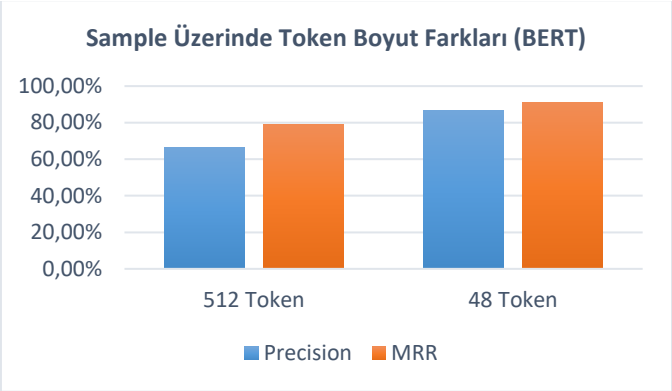
Sonuçlara baktığımızda, sistemimiz doğru metinleri %76,7 oranında bulabiliyor. Ancak elde ettiğimiz düşük MRR değerinin sebebi, doğru cevapların sadece 100 dokümanda bulunması ve modele ilgisiz 1000 dokümanı daha ilave etmemiz olarak gözüküyor.

C. Deney 2: BERT Temsilleri ile Kosinüs Benzerliği

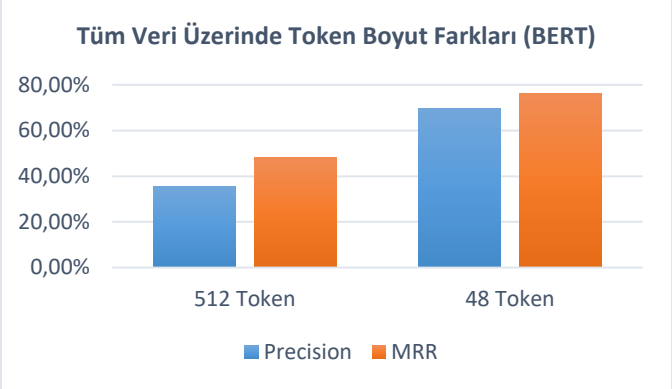
İkinci sistemde Türkçe BERT modelini kullanarak metinlerin vektör temsillerini elde ettik. Bu süreçte karşılaştığımız bir problem, BERT modelinin en fazla 512 token işleyebilmesi ve metinlerimizin 2048 token'dan uzun olmasıydı.

Bu sınırlamayı aşmak için uzun metinlerimizi 512 token'lık parçalara ayırdık ve her parçayı ayrı ayrı BERT modeline yolladık. Bu parçalara ayırma işleminde, metinleri farklı token

boyutlarına ayırmanın performans üzerinde etkisi olduğunu gözlemledik. Test etmek adına veri kümemizin küçük bir kısmı üzerinde farklı token boyutlarıyla gözlemler gerçekleştirdik. Örneğin, metinleri 32, 128, 256 ve 512 token'lık parçalara ayırmak ufak bir sample üzerinde büyük performans farkları ortaya koydu:



Aynı farklar, veri kümesinin tamamı kullanıldığında da gözlemlendi:

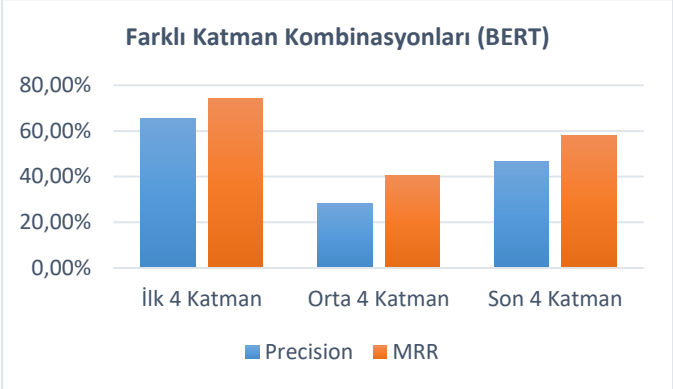


Token boyutunun başarı üzerinde bu kadar etkili olmasının nedeni, sorular ile daha ufak parçalara ayrılmış metinlerin daha yüksek bir kosinüs benzerliği sonucu vermesi ve yüksek token boyutunda soruyla alakasız metinlerin de işin içine girerek model tahmininde güçlük çıkarmaları olarak düşünüyorum. Ayırma işlemini cümle bazlı yapmak yerine token bazlı yaptığımız için muhtemel cevaplar da bir noktada bölünebiliyor. Ayrıca metnin son 512 token'lık parçası birkaç kelimeden oluşuyorsa ve potansiyel bir cevap içeriyorsa, çıktı vektöründe padding kaynaklı fazlaca boşluk olacağından ilgili soruyla ilişkilendirilmesi zor olabilir.

Model, her parça için bağlama duyarlı embedding'ler üretti ve sonuç olarak metinleri temsil eden birden çok vektörümüz oldu.

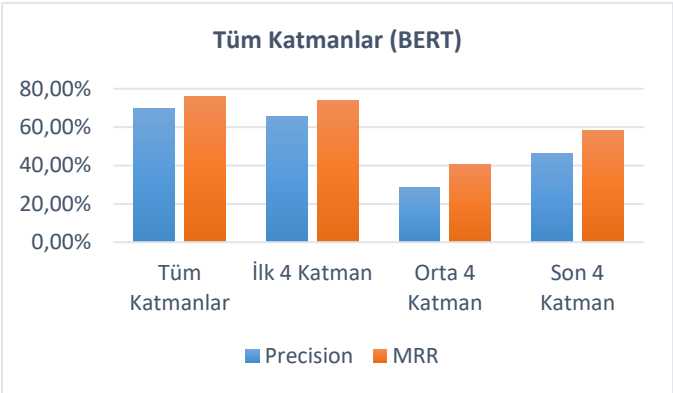
Metinlerin tek bir vektörde genel temsillerini elde etmek için modellerin katmanlarından çıktı olarak aldığımız temsil vektörlerinin ilkini, maksimumunu veya ortalamasını almak gibi yöntemleri test ettik. Kullandığımız Base BERT modeli toplamda 12, mGPT modeliyse 24 katmandan oluşuyordu. Genellikle son katmanlar daha soyut ve genel özellikleri temsil ederken ilk katmanlar ayrıntılı ve somut (örneğin, dil bilgisi vb.) özellikleri temsil eder.

Daha yüksek bir performans almak adına sadece bazı katmanların kombinasyonlarını da denedik. Örneğin sadece ilk 4 katmanı veya son 4 katmanı, hatta farklı yerlerden seçilen 4 katmanı seçerek performans değerlerini inceledik. Alacağımız sonuçlar veri kümemizin doğasına göre değişiklik gösterebileceği için bir süre deneme/yanılma yaparak sonuçları gözlemledik:



Pratikte, tüm katmanlardan gelen vektörlerin ortalamasını alıp kosinüs benzerliklerini hesaplamak en yüksek performansı verdi:

Türkçe BERT	
Metrik	Değer
Mean Reciprocal Rank	0,76
Precision	%69,7



D. Deney 3: mGPT Temsilleri ile Kosinüs Benzerliği

Üçüncü yaklaşımda ise mGPT modelini ele aldık. BERT modelinde olduğu gibi, metinler için üretilen vektör temsillerinin ortalamasını alarak daha yüksek bir performans elde ettik:

mGPT	
Metrik	Değer
Mean Reciprocal Rank	0,63
Precision	%54,5

BERT modeli, metinlerin temsili için 768 boyutlu vektörler üretirken mGPT modeli, 2048 boyutlu vektörler üretti. Bu boyut artışı ve modelin sahip olduğu yüksek parametre sayısı

hesaplama maliyetini de ciddi oranda artırdı. Dolayısıyla mGPT ile metinlerin temsillerini alma süresi BERT'e göre çok daha uzun sürdü. Öte yandan, yüksek boyutlu vektör temsillerine rağmen BERT modelinin daha yüksek performans göstermesi, vektörlerdeki boyut artışının metinlerin daha iyi temsili anlamına gelmediğini de gösterdi. mGPT modeli BERT modeline kıyasla yaklaşık %15 daha az bir Precision değerine sahip oldu. Ayrıca mGPT modelinin 61 dilde eğitilmiş olması Türkçe gibi bireysel dillerdeki performansına da pek katkı sağlamış gibi durmuyor.

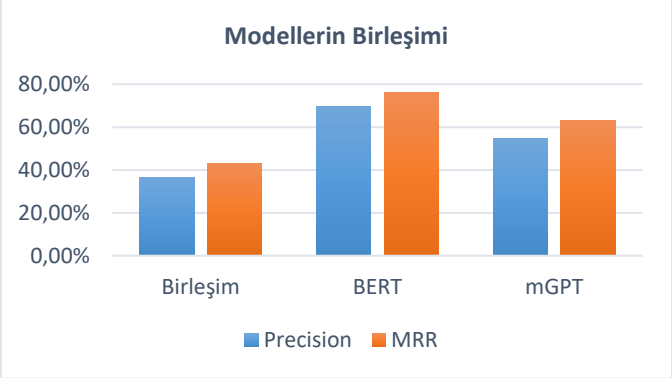
E. Deney 4: BERT ve mGPT Sonuçlarının Birleştirilmesi

Son deneyimizde BERT ve mGPT modellerinin sonuçlarını birleştirerek daha iyi performans kriterlerine ulaşmaya çalışacağız. İki modeli birleştirmek için birçok yöntem bulunuyor. Örneğin, metinlerimizin temsillerini iki modelden de aldıktan sonra bu vektör temsillerini birleştirebiliriz. Burada ortaya çıkan bir sorun, BERT modelinin 768 boyutlu çıktılar verirken mGPT modelinin 2048 boyutlu çıktılar veriyor olması. mGPT modelinden aldığımız çıktıları PCA gibi yöntemler kullanarak BERT modelinin çıktısına denk olacak şekilde indirgeyebiliriz ancak bu yöntem de, PCA için veri kümesi üzerinde bir dönüşüm matrisi hesaplama sürecini şart kılıyor.

Modelleri birleştirmenin bir diğer yolu ise, her sorunun, metin parçaları ile kosinüs benzerliklerini iki model tarafından da hesaplandıktan sonra, bu iki model tarafından elde edilen benzerlik sonuçlarının direkt olarak toplanmasıdır. Bu yöntem, teorik olarak iki modelin de bilgilerini birleştirerek daha iyi bir sonuç almamızı sağlayabilir.

Her iki modelden aldığımız kosinüs benzerliklerinin toplanması sonrası sistem performansında bireysel modellere göre büyük bir düşüş gözlemliyoruz:

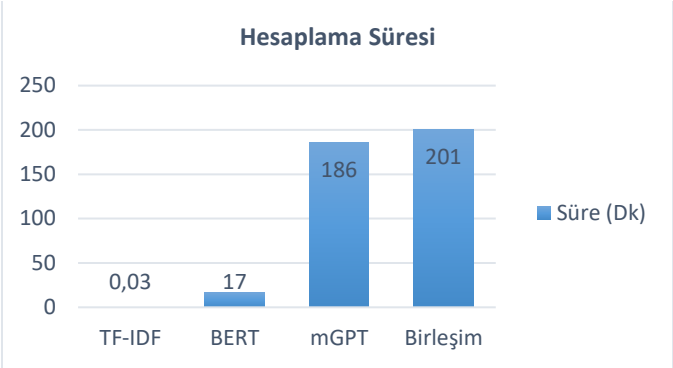
Birleşim	
Metrik	Değer
Mean Reciprocal Rank	0,43
Precision	%36,4



VI. Sonuçların Analizi

Dört sistemin de MRR ve Precision değerlerine baktığımızda en iyi performansı TF-IDF modelinin sergilediğini görüyoruz. BERT de iyi bir MRR değerine sahip fakat TF-IDF modeline göre biraz daha geride kalıyor.

Öte yandan TF-IDF, ilgili dokümanları en kısa sürede bularak maliyet konusunda da büyük bir avantaj sağladı:

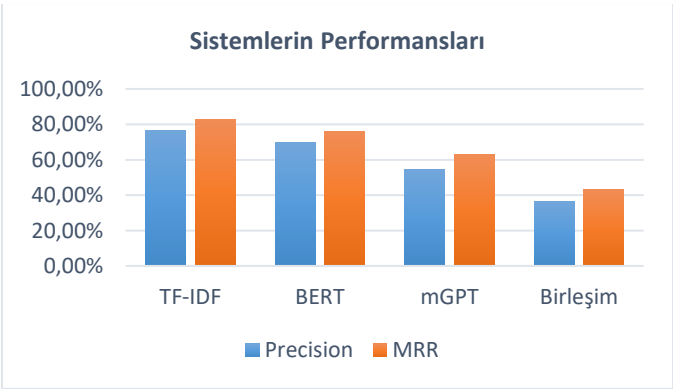


Bu iki kriterdeki farklara bakılarak en makul sistemin TF-IDF olduğunu gözlemliyoruz. Örneğin, bu sistemler bir arama motorunda kullanılacaksa doğru cevapların üst sıralarda yer alması hayati bir önem taşıyor. Dolayısıyla yüksek bir MRR değerine sahip olan TF-IDF modeli, bu gibi işlemler için daha uygun. BERT gibi diğer modeller de her ne kadar yakın performans gösterse de zaman ve işlem maliyeti açısından değerlendirildiğinde TF-IDF modeli, BERT'ten yaklaşık 580 kat ve mGPT modelinden 6200 kat daha hızlı sonuç vererek diğer iki sistemin ciddi oranda önüne geçiyor.

Zaman, verimlilik, doğruluk ve doğru cevapların üst sıralarda bulunması önemlidir. Özellikle büyük veri kümeleriyle çalışırken veya hızlı yanıt gerektiren durumlarda doğru cevabı ilk denemelerde bulmak gerekir.

Sonuç olarak TF-IDF, gözlemlediğimiz her kulvarda öne geçerek dört sistem arasından en makulu olarak gözüküyor.

Sistem/Performans	MRR	Precision
TF-IDF	0,83	%76,8
mGPT	0,63	%54,5
BERT	0,76	%69,7
Birleşim	0,43	%36,4



VII. Kaynaklar

Projenin GitHub linki: Link

[1] DevHunter (2020), Sinir Ağlarında Dikkat Mekanizması Link

[2] J. Shliazhko, O., Fenogenova, A., Tikhonova, M., Kozlova, A., Mikhailov, V., & Shavrina, T. (2015). mGPT: Few-Shot Learners Go Multilingual. SaluteDevices, Russia, HSE University, Russia, AIRI, Russia, AI Center, NUST MISiS, Russia, Institute of Linguistics RAS, Russia.

[3] AI Forever: mGPT model Link

[4] MDZ Turkish BERT model Link

[5] Abhishek Jha (2023), Vectorization Techniques in NLP Link

[6] Chirag Goyal (2021), Word Embedding and Text Vectorization Link

[7] Cosine Similarity – Text Similarity Metric Link

Yavuz Selim Doğdu
23574119